Master's thesis
Master's Programme in Data Science

# A Supervised Learning Approach to Predicting Loan Applicant's Expenditures in Loan Origination

Jaakko Paavola

March 6, 2024

Supervisor(s): Professor Kai Puolamäki
Dr. Matti Koivu

Examiner(s): Professor Kai Puolamäki
Dr. Matti Koivu

Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Koulutusohjelma — Utbildningsprogram — Degree programme |
|---|---|
| Faculty of Science | Master's Programme in Data Science |

Tekijä — Författare — Author

Jaakko Paavola

Työn nimi — Arbetets titel — Title

A Supervised Learning Approach to Predicting Loan Applicant's Expenditures in Loan Origination

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidantal — Number of pages |
|---|---|---|
| Master's thesis | March 6, 2024 | 99 |

Tiivistelmä — Referat — Abstract

Lenders assess the credit risk of loan applicants from both affordability and indebtedness perspective. The affordability perspective involves assessing the applicant's disposable income after accounting for regular household expenditures and existing credit commitments, a measure called money-at-disposal or MaD. Having an estimate of the applicant's expenditures is crucial, but simply asking applicants for their expenditures could lead to inaccuracies. Thus, lenders must produce their own estimates based on statistical or survey data about household expenditures, which are then passed to the MaD framework as input parameters or used as control limits to ascertain expenditure information reported by the applicant is truthful or at least adequately conservative.

More accurate expenditure estimates in the loan origination would enable lenders to quantify mortgage credit risk more precisely, tailor loan terms more aptly, and protect customers against over-indebtedness better. Consequently, this would facilitate the lenders to be more profitable in their lending business as well as serve their customers better. But there is also a need for interpretability of the estimates stemming from compliance and trustworthiness motives. In this study, we examine the accuracy and interpretability of expenditure predictions of supervised models fitted to a micro-dataset of household consumption expenditures. To our knowledge, this is the first study to use such a granular and broad dataset to create predictive models of loan applicants' expenditures.

The virtually uninterpretable "black box" models we used, aiming at maximizing predictive power, rarely did better accuracy-wise than interpretable linear regression ones. Even when they did, the gain was marginal or in predicting minor expenditure categories that contributed only a low share of the total expenditures. Thus, ordinary linear regression is what we suggest generally provides the best combination of predictive power and interpretability. After careful feature selection, the best predictive power was attained with 20-54 predictor variables, the number depending on the expenditure category. If a very simple interpretation is needed, we suggest either a linear regression model of three predictor variables representing the number of household members, or a model based on the means within 12 "common sense groups" that we divided the households in. An alternative solution with a predictive power somewhere between the full linear regression model and the two simpler models is to use decision trees providing easy interpretation in the form of a set of rules.

ACM Computing Classification System (CCS):
Computing methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by regression
Applied computing → Enterprise computing → Business rules

Avainsanat — Nyckelord — Keywords

supervised learning, expenditure estimation, expenditure prediction, mortgage loan origination

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

*To Tilda and Tien*

# Contents

# 1. Introduction

When assessing consumer credit applications, lenders must look at the associated credit risk from the point of view of *affordability* of the expected repayments of the loan for the loan applicant, and from the point of view of the applicant's *indebtedness*. This helps the lenders to assess a) the probability that the applicant would fail to repay the debt, i.e., *default*, and b) the potential for loan loss should the applicant default. The affordability of a loan is typically evaluated using information about the amount of disposable income the applicant has left after regular household expenditures and existing credit commitments have been taken into account, a measure called *money-at-disposal* (MaD) [21]. Indebtedness refers to the ratio between all the expenditures and the net income of the loan applicant. In simple terms, to be able to make good credit decisions, the lender must know the money-at-disposal and the level of indebtedness of the applicant, and to know them, the lender must know how much money the applicant makes and spends, i.e., their income and expenditures [21]. In this research, we focus on the expenditures.

Using the data typically required on a loan application form together with credit data about the applicant that is available from credit bureaus it is possible to build good models for assessing affordability and indebtedness. The models help lenders to detect and reject applications from too indebted applicants and, in the case of approval, decide the credit limit to an amount the applicant can afford to pay back. The incentives for making these decisions in an informed manner are twofold: good decisions, on the one hand, aid the lender in managing their credit risk prudently, and on the other hand, allow the lender to protect their customers against over-indebtedness.

If the lender only has information about the applicant's income and credit obligations and not much about other expenditures, it is hard for them to judge if, and under what loan terms, the applicant can afford a new credit product. Thus, estimating the applicant's expenditures is an important part of the loan application and granting process, the *loan origination* [21]. The more accurate the estimates of expenditures, the better the credit assessment models are at serving their purpose of helping the lender to assess credit risk accurately. Therefore, it is of great interest to lenders to have accurate estimates of the applicant's expenditures.

But how to get accurate estimates of expenditures? Merely asking for the applicant's expenditures on the loan application form will likely lead to inaccurate estimates, since it is a difficult task for most loan applicants to quantify their regular expenses very precisely. For this reason, it is important for the lender to be able to produce their own estimate of the applicant's expenditures. This estimate can be based on statistical or survey information about household expenditures in the population, and perhaps be provided to the applicant as a baseline, which they can accept as is, or amend to better match their own view of their expenses. Furthermore, it is useful for the lender to be able to determine some kind of control limits, especially a lower limit or a benchmark for minimum plausible expenditures, to ensure that the expenditures reported by the applicant are truthful or at least adequately conservative.

Intuitively, estimating in statistics refers to the act of using a sample of observations as a proxy for the entire population by making *inferences* from the sample about the values of directly *unobservable* parameters of the population. In other words, estimation is about "uncovering" the most *likely* values, the *estimates*, of the population distribution parameters based on the sample [41]. Estimation is done using *estimators*, "rules" or functions for computing an estimate of a given quantity from the observed data [17]. Estimation can be understood as a special kind of *statistical inference*, which in turn refers to "uncovering" knowledge about unobservable aspects of a phenomenon of interest, or its *data-generating process* (DGP), more generally than just parameter estimation; statistical inference is characterized as "using mathematics to draw conclusions in the presence of uncertainty" [4].

In contrast, determining the most *probable* values of *observable* random variables – such as the expenditures of loan applicants – when we do not have access to the true values, is called *prediction*. Intuitively, predicting in statistics refers to the act of using predictor variables as a proxy for a response variable to provide us values to represent the response variable in the absence of its true values. This value – our prediction – is, e.g., the expected value given the values of the predictor variables (i.e. the conditional mean typical as the prediction in regression tasks), or the value with the highest conditional probability given the values of the predictor variables (i.e. the conditional mode typical as the prediction in classification tasks) based on our knowledge about the relationship between the predictors and the response. Prediction is done using *predictors*, "rules" or functions for computing a prediction for a response variable from the observed values of predictor variables.

Prediction can be considered a type of estimation (e.g. [43, 31]) where we estimate the value of an observable random variable. This is in distinction to "conventional estimation", where we estimate the value of an unobservable population parameter, which in frequentist statistics (in fact, also in Bayesian [24]) is considered a fixed

but unknown quantity, not a random variable. On the other hand, there are also interpretations such as "one can think of estimation as the prediction of future mean values" [19], which would consider estimation as a kind of prediction. Perhaps due to interpretations such as these, the term "to estimate" is often used loosely in a similar sense as "to predict", as it is when we talk about estimating expenditures of a new loan applicant. We will also be guilty of loose usage of these two terms in this thesis, at times calling the problem estimation and other times prediction of expenditures.

Statistical modeling is sometimes dichotomized into what can be called *data modeling* and *algorithmic modeling* [9], or *explanatory modeling* and *predictive modeling* [56], or modelling for *inference* and modelling for *prediction* [33]. These pairs of modeling approaches are often presented juxtaposed due to their different motives for the modeling, and even characterized as different cultures [9] or mindsets [41] of modeling. Our approach to the research problem of this thesis, the estimation of expenditures of a loan applicant, is essentially a predictive one, but we also want to be able to explain and interpret the predictions that our models give at a sufficient level.

The variables we can make use of as predictors in predicting a loan applicant's expenditures must consist of information that we a) have available in our training dataset consisting of observations of household expenditures, and b) can enquire from all new loan applicants whose expenditures we wish to predict. Point a) is limited by the extensiveness of information in the datasets about household consumption expenditures that we have available, and point b) by what information is deemed to be possible and reasonable to request from the loan applicants. There is a limit to how many details a lender wants to require in the loan application process, as too heavy a process might deter potential credit customers [58].

It is a standard requirement in a mortgage application process that the loan applicant must provide some basic information about their household and the property they want the mortgage for. Numeric variables that usually are available to the lender and can make highly informative predictor variables in predicting expenditures include at least the number and age or age bracket of the loan applicant's household members. Moreover, there often are related categorical variables, such as the gender and socioeconomic status of the household members, available to the lender. Let us refer to the information about the household members described by the above and any other relevant variables collectively as the *household composition*. Furthermore, things such as the number of cars owned, recreational dwellings owned, and the size of and other relevant information about the property to be mortgaged are pieces of information relevant to predicting expenditures and commonly enquired in the loan application process. We will use information such as the aforementioned and any other that we deem relevant and have available as predictor variables in our prediction model.

To be able to predict, the relationship between the predictors and the response variable must first be modeled from a sample of observations. Predictive modeling can be seen as a process of first estimating the underlying multivariate distribution of the response variable and the predictors, and then, using this estimated distribution, predicting the response variable value for a new observation given the values of its predictor variables. In such a case, we would view our model as a statistical estimator of multivariate distributions $P(X, Y)$, where $Y$ is the response variable and $X$ is a vector of predictor variables. Having obtained an estimate of the multivariate distribution, we can get an estimate of any of its "cross-sections" $P(Y \mid X)$ to use for prediction. The family of probabilistic models for this kind of modeling is referred to as *generative models*. In practical prediction tasks, we normally do not try to estimate the full, usually very complex, underlying multivariate distribution, because it suffices to estimate only the conditional distribution $P(Y \mid X)$ to be able to predict [41]. The family of probabilistic models for this kind of modeling is referred to as *discriminative models* [42, 40].

In fact, most often we do not even need to estimate a distribution, since estimating just, e.g., the conditional mean of the response variable [41], or using any other way to be able to produce point predictions as a function of predictor variables, is all that is needed for most prediction tasks. This is the angle from which we look at statistical learning in Chapter three, thinking of it as approximation of a function rather than estimation of a distribution (or a model thereof). Regardless of the viewpoint, we call the act of learning from observations of predictor and response variables *supervised learning*, and the act of using what was learned in predicting the value of the response variable *supervised prediction*, particularly in machine learning context.

The purpose of this research was to use supervised learning methods to develop an accurate yet interpretable prediction model for expenditures, which lenders could use in their mortgage origination processes. The data the modeling was based on was the *statistical microdataset* of Statistics Finland's household consumption expenditures survey. To the best of our knowledge, a dataset of similar granularity and breadth has not been used for predictive modeling of loan applicants' expenditures before.

In the bigger picture, a more accurate expenditure prediction model would help a lender to improve their MaD framework, the system for assessing loan applicants' loan servicing capability and determining appropriate loan terms for them. Having accurate predictions of expenditures in the MaD framework allows lenders to quantify the mortgage credit risk associated with each loan applicant better, enabling them to differentiate between the risks of each applicant better and, as a result, determine the loan terms for each of them more aptly. However, no matter how high the predictive power, expenditure predictions in loan origination providing poor *interpretability*, i.e.,

little insight into the reasons why a prediction turned out the way it did, would be problematic for the lenders from at least compliance and trustworthiness points-of-view. Therefore, the interpretability aspect of predictions must also be taken into account in, e.g., choosing the *model families* included in our prediction pipeline as well as in making modeling choices within those families.

In our workflow of model development, we attempted to follow the best practices and "empirical wisdom" (e.g. [38, 11, 13, 45, 47]) that practitioners have discovered over the past couple of decades of proliferation of supervised machine learning techniques. The best practices we employed were related to, e.g., putting sufficient thought and effort in feature engineering and selection, model selection, and model evaluation.

Lenders typically want to predict the applicant's expenditures not only as a total sum but broken down into some number of expenditure categories – including everyday expenditures, other regular expenditures, housing expenditures, and transportation expenditures – to have a granular and accurate assessment of the expenditures. Therefore, we also built our prediction pipeline to make separate predictions for different expenditure categories. Moreover, lenders often want to have two different kinds of predictions for a given loan applicant: 1) the average expenditures, which can be used as suggestions of typical expenses on the application form, and 2) the minimum expenditures, which are used for detection of abnormally low expenditures reported by the applicant on the application form, i.e., lower control limits. In this research, we predict only the average expenditures, but make use of the concept of minimum expenditures as well.

The research question of this thesis can be formulated as: using the statistical microdataset of household consumption expenditures, how good solutions can we find for estimating or predicting household expenditures (used as a proxy for loan applicants' expenditures) when considered in terms of a) predictive power, and b) a combination of or a suitable compromise between predictive power and interpretability?

The structure of this thesis is as follows. Chapter two defines and introduces central risk concepts, particularly around credit risk. It then proceeds to talk about credit risk in mortgage loans and models used in assessing it, all the while referencing related literature. Chapter three provides a cursory introduction to supervised machine learning through defining some key concepts. Chapter four starts the description of our workflow for addressing the research problem with an account of our analysis of the data and how we framed the problem based on this analysis. Chapter five describes the methodology we used in pre-processing and preparing the data for modeling as well as our methodology in the actual modeling. Chapter six illustrates the various outputs from the modeling and prediction pipeline, and sums up the results of model evaluation and comparison. Finally, Chapter seven presents the conclusions from this whole endeavor and discusses thoughts on improvement and further research directions.

# 2. Credit Risk: Modeling and the Role of Expenditure Estimates in It

Risk can be defined as the potential effect that uncertainty attached to a particular phenomenon has on one's objectives [36]. Here, effect refers to a deviation from the expected outcome, while uncertainty refers to stochasticity, or randomness of the outcome. Put more simply, risk is the potential for or possibility of deviating from our expectation due to uncertainty.

A usual connotation attached to the word risk is that the focus is on the negative or undesired direction of the outcome from the expectation, the *downside risk*, as against the positive or desired direction, the *upside risk*. Indeed, risk is usually conceived as a combination of the impact and probability of the realization of an adverse event [36]. So when we say "there is a risk", we usually refer to some undesired outcomes that may realize. After the realization of an adverse event, we can say the risk representing the event's impact and probability has materialized. *Risk exposure* usually refers to the impact associated with downside risk, in financial context often quantified in monetary terms, that one consciously takes, i.e., the undesired effect they deem and accept has the potential to come true, also called the *downside potential*.

## 2.1 Credit Risk

Following the definition of risk given above, *credit risk* or *payment default risk* is the potential effect stemming from the uncertainty about a borrower's timely paying ability on the lender's objectives (i.e., generally, to make maximum profit within certain boundary conditions). The phenomenon under scrutiny is thus the borrower's timely paying ability (quantified into the amount of payments they make over time) with a range of different outcomes as its possible realizations, spanning from failing to meet their obligations to paying everything as per agreement over the lifetime of the loan.

A lender's knowledge of the borrower's paying ability is ultimately quite limited, not least due to *information asymmetry* between the lender and the borrower. The term refers to a common setting where the borrower knows and can anticipate their

own finances better than the lender, and does not, or is not even willing to, disclose this information to the lender with the intention of gaining benefit in loan negotiations. On the other hand, even the borrowers themselves may have a very limited knowledge of their own paying ability, especially the further into the future we look.

The lender's expectation is to gain a certain *expected return* from the credit relationship with the borrower, and by the above definition credit risk is the potential deviation from this expectation. This expected return incorporates a *risk premium* that is determined based on the level of credit risk assessed for the credit relationship. The downside potential is the amount of loss from the possibility of the borrower failing to pay their loan obligations as scheduled, resulting in the lender falling short of the expected return – at least in the agreed time frame, and possibly permanently, too.

In lending institutions, credit risk is assessed from the point of view of the borrower's *repayment capacity* and *recovery position* in order to make a *total credit assessment*, a combined risk conclusion about the two points of view. This is the decisive step in making the final credit decision. Let us talk next more in detail about these two components of the total credit assessment.

### 2.1.1 Repayment Capacity

Repayment capacity of a borrower refers to the borrower's ability to service their commitments out of their future cash flow with sufficient monthly affordability, usually quantified as MaD, i.e., money left after all outgoings. The system that lenders use for assessing a loan applicant's repayment capacity is thus called the *MaD framework*. Repayment capacity encompasses the borrower's *repayment ability*, or *debt-servicing ability*, and *financial flexibility*.

In the context of mortgages, repayment ability refers to the borrower's overall financial situation and their ability to fulfil all their debt obligations with the assessed stable and recurring income over the lifetime of their debt obligations. Here, relevant metrics are the (gross) *debt-servicing ratio* and *housing expense ratio*, which indicate the ratio between the borrower's housing-related debt servicing expenditures, or alternatively a wider array of housing-related expenditures, and (gross) income. These metrics help lenders to see how big a portion of the borrower's income goes to mortgage repayments and other regular obligations, and thus to quantify the borrower's repayment ability.

Financial flexibility is a point of view considering the customer's wealth, or *equity*, which is an important aspect in assessing credit risk since wealth serves as a buffer in events of loss of income or similar unexpected changes.

### 2.1.2 Recovery Position

Recovery position of a borrower refers to the potential amount the lender can recover from the borrower in the event of default. The recovery position primarily depends on loan terms and conditions, collaterals, and guarantees. Collaterals, guarantees, and other credit risk mitigation measures are important for securing a sufficient recovery position, but a good recovery position should not be the main driver behind a positive credit decision. *Recovery ratio* refers to the proportion of the receivables that the lender is able to secure should the borrower default.

## 2.2 Credit Loss

*Delinquency* refers to a payment of an outstanding debt having become overdue, a situation also called *falling into arrears*, or *mortgage arrears* in the case of a mortgage loan. Technically, delinquency occurs at the instant the borrower fails to make a scheduled payment on a loan [2], but there are also differing definitions for specific cases, such as "mortgages are considered delinquent when payments are thirty days or more past due" [35]. Delinquent loans are typically classified based on the "overdueness" of the oldest unpaid payment, commonly as 30, 60, 90, or 120 or more days past due [2, 35].

*Insolvency* is a legal term to describe a borrower's inability to meet their financial obligations as a more permanent status than in the case of delinquency. While many borrowers experience delinquency due to temporary financial setbacks like unexpected expenses, income disruptions, or plain negligence, most manage to recover by catching up on missed payments or renegotiating the loan terms, such as by agreeing on repayment plans with lenders, without ever going insolvent [2].

Default is another legal term to describe a borrower's status. Sometimes the term is used loosely to mean something similar to our above definition of delinquency, but the correct use refers to a more permanent state of inability or unwillingness to pay. In fact, defaults represent a much smaller proportion of cases relative to delinquencies. Insolvency typically leads to default, but in general, insolvency or even delinquent payments are not a pre-requisite for default. This is because in some jurisdictions where defaulting rids the borrower of all liabilities regardless of the current market value of the loan's collateral, default can be viewed as a legal option for the borrower even when they are not facing insolvency, using which they can get exempt from their liability for debts by forfeiting the collateral to the lender. As such, defaulting can be a rational choice for the borrower when, e.g., the housing market and along with it the value of the collateral of their loan, their property, has plummeted [46].

Typically, however, default is considered a situation where the borrower has exceeded a certain amount of delinquency in scheduled payments or is otherwise unlikely to pay their due payments in full, and the lender thus wants the remaining outstanding to all come due at once. By the European Capital Requirements Regulation (CRR) definition for default the borrower is considered to have defaulted when either 1) the lender considers the borrower unlikely to pay their credit obligations in full without the lender resorting to *recovery proceedings*, such as demanding realization of collateral or starting *foreclosure proceedings* on it, or 2) the borrower is past due in their scheduled payments, i.e. delinquent, for more than 90 days [20].

By the Finnish legislation, if the realization of the collateral does not cover all the outstanding debt, the residual amount remains as the borrower's liability. Due to this, retail borrowers in Finland are reluctant to default in market conditions where the value of their collateral has decreased, and thus they typically default only once they have become insolvent [46].

Credit risk materializes into *credit loss*, or *loan loss*, in the case of such a default of the borrower wherein the lender fails to recover the entire sum comprised of the outstanding debt, remaining principal and accrued interest payments after the collateral has been realized, possibly through foreclosure proceedings. Foreclosure is a legal process for selling real estate pledged as security or collateral for a loan through a foreclosure sale to repay a debt.

## 2.3 Creditworthiness and Loan Terms

A lender's credit granting process typically involves two distinct decisions made in sequence. First, there is the decision to either accept or decline the applicant based on their *creditworthiness* [2]. A creditworthy customer can be defined as one whose debt-servicing performance is such that the lender would accept them again, if faced with the decision of accepting or declining the customer's application now [3]. The second decision involves establishing the "terms of business" of the credit relationship, or, determining the loan terms, such as the credit limit and interest rate [2].

### 2.3.1 Credit Scoring

In simple terms, a customer's creditworthiness is determined by assessing the probability of their future default against a predefined threshold, the cutoff value. Applicants below this threshold are approved and offered credit, while those above it rejected [2]. In practice, models analyzing various aspects of customer behavior, such as attrition, revenue, and response, are often used alongside models of probability of default to cre-

ate what is called a *credit scoring system*, a statistical model that maps the applicant's characteristics with the probability of them defaulting the loan and assigns them a *credit score* [8, 3, 2]. Thus, determining creditworthiness becomes a matter of whether the credit customer's credit score is above a certain *cutoff score* or not. The input variables to these models typically include data on the applicant's financial behavior such as their present and past finances, the type and value of the collateral for the loan, and demographic data, at times complemented by loan-specific details [39].

Thomas et al. define credit scoring as "the set of decision models and their underlying techniques that aid lenders in the granting of consumer credit" [58]. Note that besides for deciding whether to accept or decline the loan applicant, lenders also use credit scoring systems for determining the loan terms for them. How a lender evaluates credit risk and subsequently makes decisions about credit granting and loan terms directly impacts the occurrence of delinquency and default in their credit portfolio, the profitability of those loans that do not incur credit loss, as well as the share of potentially profitable loan applicants that get rejected.

The ever increasing availability and decreasing cost of computing and data storage over the past half a century enabled credit institutions to first increasingly and then routinely start calculating credit scores for loan applicants. Largely thanks to credit scoring becoming an industry standard, the lending sector started to offer differential pricing, where borrowers deemed less risky are offered loans at lower interest rates, and consequently different types of consumer credit, such as mortgages, unsecured personal loans, and credit card debt became available [25]. As the developments in the modern digital world lead to increasingly more data and potential to extract information from them, lenders are constantly seeking ways to improve their methods for evaluating credit risk, usually by introducing new models or by incorporating additional variables and characteristics to the old ones [8, 2].

## 2.4 Credit Risk in Mortgage Lending

In the European banking sector, it is estimated that mortgage loans make up approximately one third of all the banking sector's assets, around seven trillion euros in total [39, 57]. Mortgages are the most significant type of loan for many lenders, particularly the traditional commercial banks, and as such, a major source of credit risk for them.

### 2.4.1 Credit Loss

As with credit risk in general, credit risk of mortgages materializes into credit loss in the case of such a default of the borrower wherein the lender fails to recover the entire

outstanding loan after realization of the collateral, possibly through foreclosure proceedings. In the case of severe delinquencies, the lender has an incentive to try to avoid foreclosure proceedings on the mortgage by encouraging the borrower to voluntarily sell the property, or by resorting to various *forbearance measures*. These forbearance measures can be, e.g., accepting a partial payment of the delinquent principal and interest payments, or agreeing on loan modifications, such as a lower interest rate on the loan, an extension to the length of the loan, or a cut to the amount of principal.

Even severe mortgage delinquencies where the borrower cannot recover by catching up on missed payments can often be resolved without credit loss to the lender simply by the borrower voluntarily selling the property to pay off the loan. The lender's incentive to avoid foreclosure stems from the fact that foreclosure is typically an expensive process due to, e.g., loss of accrued interest between the time from delinquency to foreclosure, legal fees, property maintenance expenses, costs related to the property's sale, and the loss incurred if the foreclosed property sells for less than the remaining loan balance [2].

A large number of concurrent mortgage defaults and foreclosures can have significant, in the worst case even catastrophical, consequences to any lending organization. Moreover, a mortgage default often has grave consequences to the borrower's household, and in a greater number, to the society as a whole. Borrowers experiencing default typically face consequences such as diminished credit scores, limited access to future credit, loss of assets, and the expenses associated with relocating. Additionally, when defaults concentrate in specific areas, they can have substantial social impacts due to decreasing local property values, rising credit risks, and reduction of available credit [2].

Since there is a risk of credit loss in the form of the expenses associated with foreclosures and defaults, lenders include a risk premium in the interest rates to compensate for it. When the reasons behind defaults are not well-known, lenders might increase the average price of mortgage credit or grant credit to only highly creditworthy applicants to compensate for this uncertainty. From the point of view of efficient mortgage lending markets and subsequently the availability and affordability of funding for all who by their finances should be eligible for an aptly priced mortgage, this is a grievance. By more accurate identification of applicants who are likely to repay their loans successfully and differentiating them from those who might struggle, lenders can make mortgages more accessible to a broader range of borrowers at rates that accurately reflect the true level of underlying risks [2].

### 2.4.2 Creditworthiness and Credit Scoring

An applicant's creditworthiness in mortgage lending is increasingly determined using credit scoring models, which rely on the applicant's credit history and other relevant data [2]. Although it is said that these credit scoring models are not seen as "real credit risk models" like those used with bonds and wholesale loans based on more sophisticated modeling of the probability of default [46], in mortgage lending these credit scoring models are generally deemed sufficient models of credit risk.

In traditional mortgage credit scoring models, applicants are primarily either approved or denied based on whether they meet the lender's *underwriting criteria* for eligibility for a mortgage, although the credit decision is not entirely binary since, e.g., differential pricing is applied as well [2, 49]. The underwriting criteria are conventionally based on metrics like the *loan-to-value ratio*, *debt-to-income ratio*, *loss-given-default*, and various documentation elements confirming the borrower's income and assets from external sources like employers, tax records, and bank statements [49]. To extend on this traditional approach to mortgage credit scoring, models that are better at capturing the applicant's outgoings have been suggested, and at the center of these approaches is the estimation of expenditures of the applicant [21].

#### 2.4.2.1 Estimation of Expenditures

In the mortgage credit granting process, the lender collects various basic and financial information about the loan applicant and the property, analyses the applicant's repayment capacity and recovery position based on them, and then forms a credit decision. MaD framework, the system for assessing a loan applicant's repayment capacity, is a focal part of the decision engine in the credit granting process. In turn, expenditure estimates for loan applicants are a key input to the MaD framework and thus an important piece of information in making credit decisions in loan origination. The output from the MaD framework is the amount of money the household is left with after paying their stress tested financing costs and other monthly expenses based on the data given as inputs. If the outcome from the MaD calculation is negative, it leads to an automatic rejection of the loan application.

In estimating expenditures in the MaD framework, capital and one-off expenditures are typically left out with the argument that people usually fund these from savings or through credit instead of running income. Moreover, their infrequent occurrence might lead to challenges in getting accurate data on them. This category includes items like cars, vacations, household appliances, home renovations, and moving costs. Additionally, credit repayments are typically left out since they can be directly calculated from credit bureau data and taken into account elsewhere in the credit granting

process [21].

Mortgage applications can be made by an individual applicant or joint applicants, but in either case the debt affects the expenditures and affordability of all concerned parties on both personal and household levels. As such, should, e.g., expenditures be assessed individually or at the household level? Traditionally, households have been treated as unitary by dividing overall expenditures based on proportional incomes within the household, thereby assuming that all family members share identical preferences. However, research suggests that the income-to-expenditures relationship is more intricate than this, influenced by the type of expenditure as well as the household's composition and its members' personalities. In particular, allocating children's expenditures against the adults' incomes presents a challenge. Due to these difficulties, the focus has shifted to models considering expenditures and over-indebtedness only at the household level, but at the same time taking into account both the household characteristics and personal characteristics of its members [21].

# 3. Supervised Machine Learning

Say we have a dataset $\{(x_i, y_i)\}_{i=1}^{N}$ with $x_i$ denoting a vector of predictor variables – typically called *features* in machine learning – and $y_i$ denoting the response variable – typically called *target variable* (or *label* or *supervisory signal*) in machine learning – for each observation $i = 1...N$ in the dataset. Incidentally, observations (i.e. examples or instances) are often called *samples* in machine learning context, which is a bit of an abuse of terminology, since sample traditionally refers to a set of observations sampled from the population, not individual observations. In this thesis, we will hold on to this distinction between the terms. In statistical learning theory – a framework for machine learning – *inductive learning*, or just *learning*, is a process where the goal is to utilize such a dataset to find a *hypothesis function* $f^*$ among all hypothesis functions or *mapping functions* (or just *functions*) $f$ that approximates the output $y_i$ for each input $x_i$ well enough [30]. Thus, we can formulate learning as trying to find $f^*$ such that

$$y_i = f^*(x_i) + \varepsilon, \tag{3.1}$$

where the error term $\varepsilon$ encodes all the stochasticity in the DGP that makes the relationship between $x$ and $y$ non-deterministic [29]. The true underlying mapping function, which is some fixed but unknown function we call the *target function*, encodes the systematic information that $x$ provides about $y$, representing the deterministic aspects of the DGP [33].

Since what we learn about the mapping from the input to the output is recorded and represented with the help of a *model class* – a "blank template", if you will, for a (learning) model from a specific model family with a specific hyperparameter configuration – we say we *learn a model* of the DGP, or rather of the target function representing the DGP [16]. If we had a more traditional statistical approach, we might talk about estimators and their estimate of a model of the DGP, or rather of a distribution representing the DGP. It is worth noting that in the domain of (statistical) modeling and learning theory, there is plenty of more or less overlapping terminology originating from different viewpoints to the same subject matter, such as from traditional statistics, the early schools of pattern recognition and signal processing, and the more modern school of machine learning. The difference between many of these terms

is subtle at most, although they may bear pertinent connotations to their particular point-of-view. For example, learning has the connotation of a more general approach to discovering knowledge about the DGP than estimation, since learning covers "whatever works", i.e., methods that are distributional or distribution-free, based on estimators (i.e. $f$ has a parametric form) or are estimator-free (i.e. $f$ has a non-parametric form) etc.

Inductive learning from observations of features and target variables is essentially what supervised machine learning is all about. Inductive learning can be formulated as the minimization problem

$$f^* = \arg\min_{f \in \Omega} L(f), \tag{3.2}$$

where $L(f)$ is the *loss function*, which quantifies how well each function $f$ fits the data, and $\Omega$ is the *function search space*, wherein we search for our hypothesis function $f^*$.

The process of solving the minimization problem in the equation, or more generally, optimizing the *objective function*, is called *training* [30]. The means to do training is by using a *machine learning algorithm* (i.e. a *learner* or *function approximator*). In other words, training is the act of making our model or learner to learn the hypothesis function, or, put in another way, making our learner to learn a model (of the target function [16]). Several ways of expressing the same thing are in common use.

In statistics parlance, we might rather say we estimate a model of the underlying distribution. However, with non-parametric models, such as decision trees, talking about estimating would sound inaccurate, because estimating implies estimating model parameters, which non-parametric models do not have (at least not in the same sense as parametric models do). Thus, the terms learning (from data), training (with data), and *fitting* (to data) are probably used more often with non-parametric models.

Albon [1] defines fitting as "applying a learning algorithm to data using analytical approaches", contrasting it with training, which he defines as "applying a learning algorithm to data using numerical approaches like gradient descent". Going with these definitions, fitting can be considered analogous to training – the process of optimizing the objective function – but particularly for the more traditional methods where the optimization problem can be solved analytically, such as classical approaches to estimation (e.g. maximum likelihood estimation). On the other hand, Molnar [41] contrasts fitting with estimation, associating fitting with what he calls the "likelihoodist mindset" of modeling: "adapting" model parameters until the observed data "seem likely under the statistical model" (an approach classically called "curve fitting"). This, he says, is a different angle to modeling than parameter estimation in the focus of the "frequentist mindset" and "Bayesian mindset" of modeling. Be that as it may, in practice the terms to learn, to estimate, to fit, and to train are all used quite loosely and interchangeably,

and we may also do so in this thesis.

So learning involves using a training set $\{(x_i, y_i)\}_{i=1}^{N}$ to explore $\Omega$ and find the function $f^*$ that minimizes the loss function. The learner is supposed to learn to approximate $y_i$ even for instances it did not see during training. Although infinitely many functions can perfectly fit any given dataset, most of them would merely memorize the data and perform poorly on new data, i.e., they would just overfit to the training data and generalize poorly to unseen data. To remedy this, we want to constrain the search for $f^*$ to specific parts of the function space $\Omega$ that contain functions that generalize better in the particular problem at hand. *Inductive bias*, or *learning bias*, refers to imposing such constraints to obtain better generalization. Here, the term bias can be thought of as referring to the difference between the $f^*$ we can find under these constraints and the "global optimum" $f^*$ without the constraints [30, 32].

Inductive learning always requires some form of inductive bias to be able to generalize, and the choice of $\Omega$ is the critical design parameter in establishing it [30, 32]. Too low an inductive bias (large search space $\Omega$) may lead to overfitting, while too high an inductive bias (small search space $\Omega$) may result in underfitting. However, we cannot directly choose the value of $\Omega$, but in practice we choose the assumptions that the machine learning algorithm explicitly or implicitly makes about the nature of the target function, e.g., its functional form, determining the $\Omega$ in the process [5]. These assumptions enable the model to predict the outputs from unprecedented inputs, i.e., to generalize, well enough. Thus, inductive bias is a bias with a potentially very positive effect as opposed to how we usually conceive the effect of biases in modeling, that is, as negative.

We said inductive bias enables machine learning models to generalize "well enough", and we used "well enough" also in the first paragraph when characterizing the desired ability of the hypothesis function to approximate $y_i$ for each $x_i$. This "well enough" in machine learning context is typically defined in terms of predictive power, implying that the emphasis is heavily on prediction instead of explanatory or causality aspects of modeling. How predictive power is quantified is through prediction error.

In statistics, the term bias, or *statistical bias*, generally refers to *estimation bias*, i.e., the difference of the expected value of an estimator $E[\hat{\beta}]$ and the true value of the parameter $\beta$ being estimated:

$$\text{Bias} = E[\hat{\beta}] - \beta \tag{3.3}$$

[30, 29]. An unbiased estimator has no difference between its expected value and the true value, while a biased estimator does. Practically, even with an unbiased estimator, we might rarely be close to zero difference to the true value, because the variance of the estimator may be high with the amount of data we base our estimation on.

This definition of bias extends seamlessly to situations where we are not esti-

mating just a parameter but an entire model (consisting of a number of parameters). Similarly, we could talk about the bias of our learning process searching for the hypothesis function $f^*$. Here, bias would be the difference between the $f^*$ we would arrive at with this particular learning process if we had all the data and all the computation time in the world (the expected value) and the target function.

We can decompose the prediction error of our statistical or machine learning model to its bias, variance and irreducible constituent components. This bias component can be called the *model bias* [30, 29]. Why is it called a bias and how is it connected to the statistical bias of the model that exhibits it? The statistical bias of a model estimator or a learning process "propagates through" the machine learning pipeline and gives rise to a bias in the predictions, the model bias [30]. The model bias can be interpreted as the learner's tendency to consistently learn the same wrong thing [18]. It is the error stemming from approximating a complex real-world DGP using the rough simplification of our model [33].

The variance component can be interpreted as the learner's tendency to learn random things irrespective of the real signal [18]. Perhaps a more concrete interpretation is that variance is a measure of the variability of the model's predictions if we repeat the learning process with small fluctuations in the training set, i.e., altering or using different data for training between the repetitions. The more sensitive the model-building process is to these fluctuations or to changing the training data between the repetitions, the higher the variance [47, 33]. In other words, variance represents our predictions' stochasticity with respect to the samples we use as training data, contributing to the reducible error. This error is subsumed in the error term $\varepsilon$ together with irreducible error stemming from our predictions' stochasticity with respect to individual observations [33]. Formulated in yet another way, variance describes our predictions' sensitivity to observable idiosyncrasies between different training data, while irreducible error describes our predictions' sensitivity to unobservable idiosyncrasies between observations.

Apart from inductive bias, statistical bias and model bias, the learning process can involve also other biases. One focal group of biases is related to the data in our sample. We will touch on data-related biases in Chapter four. Another focal group of biases comes from model selection. Many machine learning algorithms have numerous hyperparameters the values of which cannot be learned from the data by solving a similar kind of an optimization problem as in the training, but must be chosen either manually by the modeler or by using a meta-optimization process. The choice of the values of these meta-parameters may impact the model significantly. The difference between the best hyperparameter configuration that we can ideally find with the process we are using for choosing it and the "global optimum" we can refer to as the

*hyperparameter bias* [30]. Finding good hyperparameter values is usually not trivial and thus a prudent choice requires its own meta-optimization task called *hyperparameter search*, *optimization* or *tuning*, a process for evaluating hyperparameter configurations. All the different hyperparameter configurations we evaluate define a set of models, among which we choose the best-performing one, a focal part of a step called *model selection* [47].

As evident from above, the term bias is a bit overloaded in the domain of statistics and machine learning. We can think that the "least common denominator" of the different uses is that they all refer to the ideal configuration or selection "foundable" from the search space we have specified having some sort of a difference to the underlying ground truth or the "global optimum".

# 4. Analysis

This chapter describes the steps taken in analyzing the microdataset on household consumption expenditures before starting the actual implementation of the modeling and prediction pipeline based on it. The purpose of this analysis was to be able to understand the problem at hand better and to frame it as a supervised prediction task, and to make a preliminary choice for the set of variables to proceed to data pre-processing and preparation for the actual modeling with.

## 4.1 Overview of the Data

The main dataset used in this study was the statistical microdataset underlying Statistics Finland's statistics "001 – Household consumption expenditure by type of household" [44] using only the data points from the year 2016, which were the latest data points available at the start of this work in 2023. We shall often refer to this dataset simply as the microdataset. The statistics are publicly available on Statistics Finland's website, but the microdataset is made available only for research purposes and is subject to a fee.

The work was based on the assumption that the household consumption expenditure survey by Statistics Finland is the *gold standard* for obtaining data on not only households' but also potential loan applicants' expenditures, i.e., the method providing the data closest to the ground truth, the (directly unobservable) true expenditures in the population. In this thesis, we use the term ground truth in the context of the expenditures loosely to refer to the true values as per the microdataset. However, note that the population of potential loan applicants is in fact a subset of the population of households, because only some households are potential loan applicants, and thus the household consumption expenditure survey has a different underlying population than what we are interested in. We effectively use the population of all households in Finland, represented by the microdataset, as a proxy for the population of potential loan applicants in Finland, because there is no data available directly on the expenditures of just those households that are potential loan applicants. We assume no significant error results from this simplification.

A secondary dataset used in this study was from a research on minimum reference budgets of households by University of Helsinki [37]. Using this data we determined expenditure estimates that we used as one of our benchmarks for our predictive models, as well as minimum references for expenditures in each category, which we used in outlier handling. We shall not delve deeper into the details of how we derived the minimum references with the help of this minimum reference budget study or what the values of those minimum references that we arrived at were exactly. We will touch more on these minimum references in the sections on outliers and benchmarks.

### 4.1.1 Target Variables

In the microdataset, there were 3,673 households, and for each household there were 997 rows each holding a value for a particular expense classified by a Classification of Individual Consumption by Purpose (COICOP) expenditure code. We specified a mapping of these 997 COICOP expenditure codes to the following 12 expenditure categories:

1) food expenses,

2) freetime expenses,

3) car expenses (incl. car-related insurance, maintenance and management fees),

4) heating, electricity and water expenses,

5) housing expenses without utilities and energy (incl. housing-related insurance, maintenance and management fees),

6) other regular expenses including recreational dwelling,

7) health and hygiene expenses,

8) communication, IT and media expenses,

9) clothing expenses,

10) childcare expenses,

11) insurance expenses other than those related to car or home,

12) local public transportation expenses,

but many of the 997 COICOP codes were not mapped to any of the above, because they were deemed not to be within the scope of expenditure estimation in the MaD framework. The logic behind determining which of the COICOP codes were not within the scope has to do with the nature of the expenditures: expenditure estimation in the MaD framework is only supposed to account for expenditures of a more running nature, not for, e.g., acquisition or financing costs of property or vehicles.

We summed up the expenses classified by the COICOP codes based on this mapping. As a result, we got 12 different expenditures aggregated from these expenses, the 12 target variables for our predictive modeling.

### 4.1.2 Explanatory/Predictor Variables

Apart from the variable holding the values of the expenses and the variable holding the classification of those expenses using the 997 COICOP codes, the microdataset had 172 other variables, which represented the *master data*, or *registry data*, characterizing the households. 172 variables would make a very high-dimensional dataset for any modeling purposes, so before proceeding to modeling it was important to determine which variables to keep and which ones to drop because they are not relevant, available or otherwise not providing enough value to outweigh the cost incurred through the added dimensionality they introduce.

Before looking into the dataset's contents and commencing *feature elimination* using more analytical and technical methods, we started by simply studying the dataset's variable description sheet and, using common sense and domain knowledge, determining which variables to keep out of these 172. For each variable, we considered whether that piece of information about a loan applicant would realistically be available for a lender. As mentioned before, it is also not in the lender's interest to make the loan application process too heavy with a lot of information required from the applicant.

After screening out variables, or features, based on such assessment of availability and feasibility, the starting point for our set of variables to proceed to the data preprocessing and preparation with was as shown in Table A.1 in the appendices.

## 4.2 Data Exploration and Exploratory Visualization

Statistics Finland provides the background information for the household consumption statistics and microdataset shown in Table 4.1, describing summary statistics about both the sample and the population it aims to represent. The categorization of households in the dataset breaks them down to households of: 1) one-person, aged under 65; 2) a couple without children, aged under 65; 3) single-parent; 4) two-parent; 5) persons aged over 64; and 6) other households. Among the 3,673 households partaking in the survey, the majority, about 77%, were households without children, which sounds high and calls into question the dataset's representativeness of the whole population. Misrepresentation of the population by a sample is one type of a data-related bias, referred to as the *sampling bias*, *selection bias*, or *population bias* [30].

From Table 4.1, we can calculate that the relative frequencies of the above-mentioned household categories in the dataset are roughly 0.20, 0.20, 0.03, 0.20, 0.27, and 0.10, whereas in the whole population they are 0.26, 0.17, 0.04, 0.18, 0.28, and 0.07. Now we can see that the relative frequencies of one-person households under 65

**Table 4.1:** Background information of Statistics Finland's household consumption expenditures statistics and microdataset.

| | Households in sample | Households in population | Average size of household |
|---|---|---|---|
| One-person, aged under 65 | 723 | 699,558 | 1.00 |
| Couple without children, aged under 65 | 747 | 446,409 | 2.00 |
| Single-parent households | 108 | 107,565 | 2.68 |
| Two-parent households | 735 | 475,979 | 4.07 |
| Households of persons aged over 64 | 1,008 | 752,274 | 1.40 |
| Other households | 352 | 195,314 | 2.79 |
| Total | 3,673 | 2,677,100 | 2.02 |

(0.20 vs. 0.26, i.e. underrepresentation), that of couples without children under 65 (0.20 vs. 0.17, i.e. overrepresentation), and that of other households (0.10 vs. 0.07, i.e. overrepresentation) differ the most between the sample and the population. The relative frequencies between households with and without children (we assume "other households" have no children) are quite equal between the sample and the population (0.23 and 0.77 vs. 0.22 and 0.78). With this in mind, the sample's representativeness of the whole population of households in Finland can probably be deemed acceptable.

It is yet another question how well this sample represents the population that lenders are actually interested in, the population of potential mortgage loan applicants. But as mentioned, our underlying assumption is that no significant error results from using this dataset as the basis for our modeling.

### 4.2.1 Expenditures

Let us scrutinize the expenditures in more detail. Figure 4.1 shows the means of expenditures per household by category as well as the mean of the sum of those expenditures, denoted as "total expenses". Moreover, there is also the bar "grand total", which represents the mean of the sum of the expenses of all 997 COICOP codes per

household. The difference between the grand total and the total expenses stems from those expenses that are not within the scope of the MaD framework.

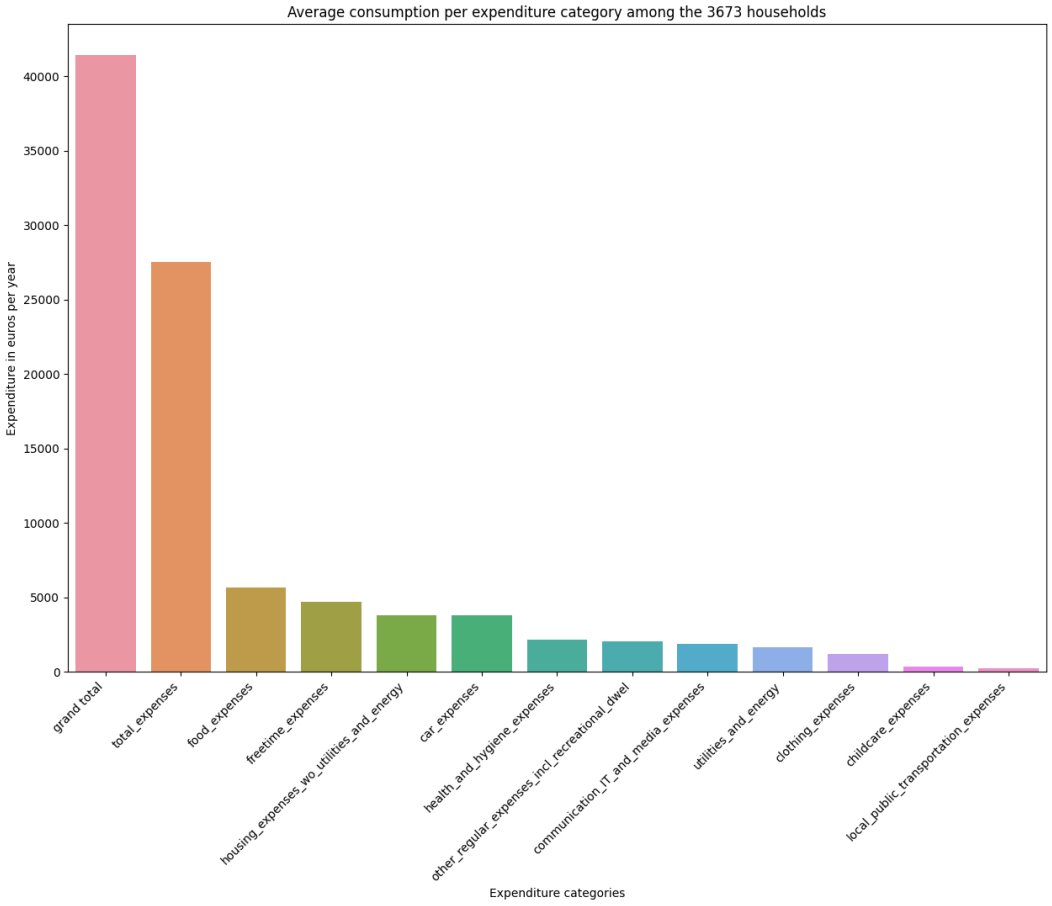


**Figure 4.1:** Mean expenditures per household in each category, the total expenses over the categories, and the grand total, which includes also expenses that are not within the scope of the MaD framework.

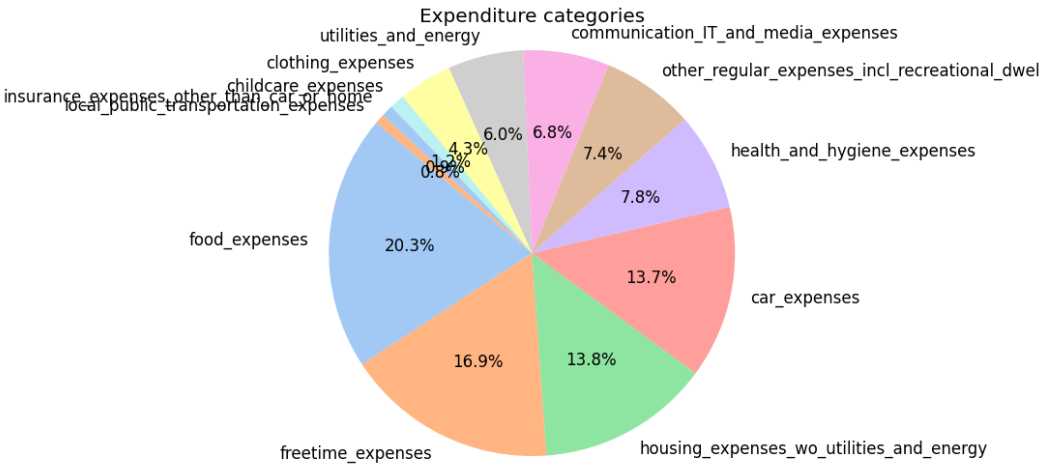


**Figure 4.2:** Share of the total expenditures by expenditure category.

Figure 4.2 shows the share of each expenditure category out of the total expenses.

We can see that among the 12 expenditure categories the four largest – food expenses, freetime expenses, housing expenses without utilities and energy, and car expenses – make up well over half of the total. Note that housing expenses without utilities and energy also contain all housing-related insurance, maintenance, and management fees, and car expenses all car-related insurance, maintenance and management fees. Freetime expenses contain meals in restaurants outside of working hours (workday lunch expenses are included in food costs). The next three largest categories – health and hygiene, other regular expenses including recreational dwelling, and communication, IT and media expenses – make up just over a fifth of the total. The remaining categories – utilities and energy, clothing, childcare, insurance other than car or home, and local public transportation expenses – make up approximately one eighth of the total.

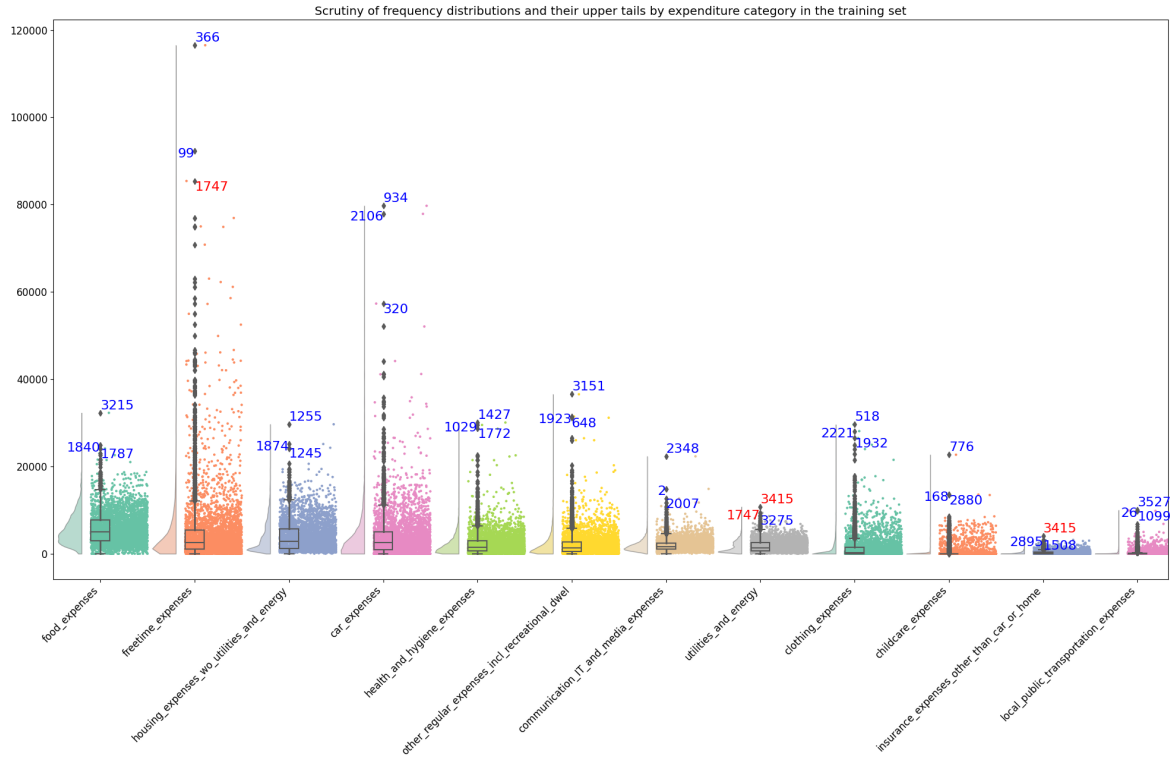


**Figure 4.3:** Distributions of observations by expenditure category and the indices of the households that had the top three largest values in each category in the training set.

Figure 4.3 shows the frequency distributions by expenditure categories in the training set as a *raincloud plot*. The annotated numbers show the index, or ID numbers, of those households that had the top three largest values in each expenditure category. If the annotated number is in red, it means that the same household is among the top three also in another expenditure category. We can make three notable observations: 1) all categories have data points at or very close to zero, which is surprising especially for expenditure categories such as food expenses, 2) especially freetime expenses and car

expenses have very long upper tails, and 3) there are no households that systematically have expenditures among the top three largest across the categories.

#### 4.2.1.1 Outliers

Let us address aforementioned observation 1. The fact that there are observations at or close to zero in every expenditure category is problematic, because those households will also be below the levels of our minimum references for expenditures – the levels that we determined for each category based on the minimum reference budget study by University of Helsinki. We want to assume the minimum references give us strict "floor" levels for each expenditure category, and that almost no loan applicant's true expenditures should fall below these control limits. The more we use observations with values lower than these minimum references to train our model, the higher the chance that the model predicts a value lower than a minimum reference for a new loan applicant. Since the model is meant to predict typical expenditures for a household akin to the loan applicant's, it would not make sense these predictions were ever below the minimum references.

Figure 4.4 shows a closer scrutiny of the lower ends of the frequency distributions of expenditures by category. In this box plot, the horizontal line in each box depicts the median, the cross the mean, the lower end of the box the first quartile, the upper end of the box the third quartile, and the whiskers plus the diamond-shaped markers the remaining observations at the tails. Moreover, for each category, there is an annotated number inside a red box that tells us the number of observations below the minimum reference for one-person households in that expenditure category (denoted by a red horizontal line) – except for the categories 'housing expenses without utilities and energy' and 'childcare expenses', for which we determined no minimum references due to some ambiguity in the data. Note that this is the number of observations below the minimum reference of just one-person households; the number of observations with expenditures below their particular household composition's corresponding minimum reference would be higher. For each category, there is also an annotated number inside a blue box that tells us the number of observations below one standard deviation from the mean (denoted by a blue horizontal line), unless it would be below zero.

Let us then address observation 2. In Figure 4.5, we again depict the frequency distributions of expenditures by category, but this time with a focus on the upper tails. The blue horizontal lines denote the levels of multiples of standard deviations above the mean, from one up to six. The annotated numbers inside the blue boxes are the number of observations above two (lower box) and three standard deviations (upper box) from the mean. We can see that in some of the categories there are quite

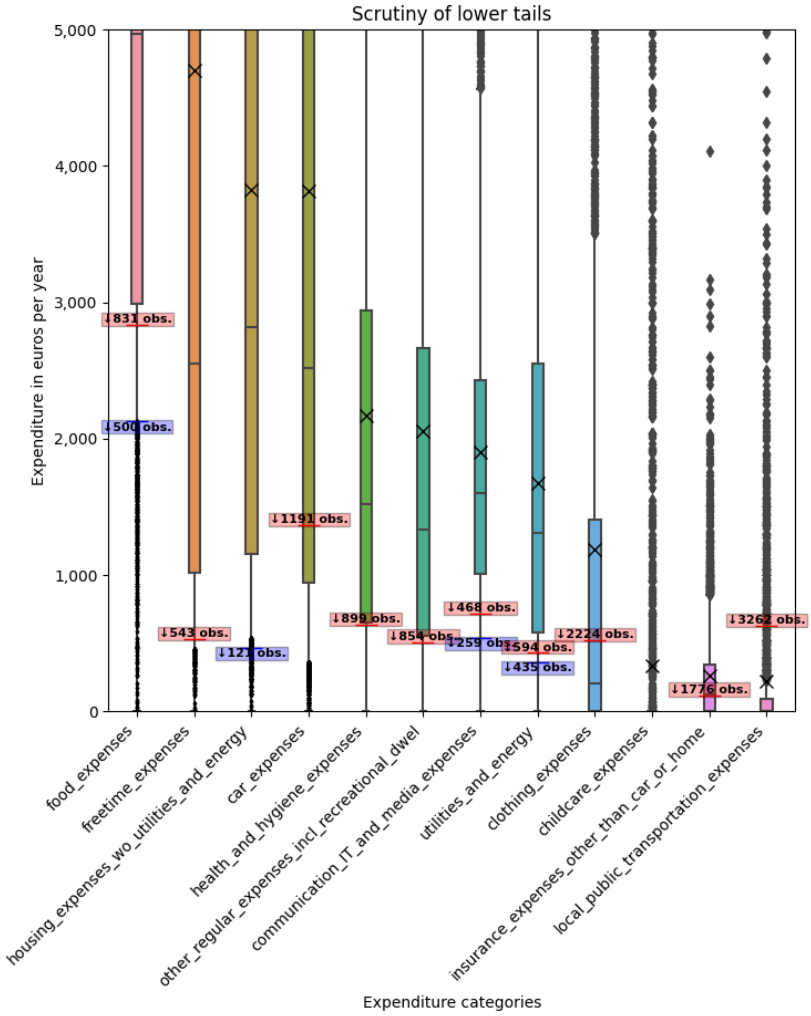


**Figure 4.4:** Box plot of expenditures by category with numbers in the red boxes showing the number of observations below the minimum reference for one-person households and those in the blue boxes showing the number of observations below one standard deviation from the mean.

many observations even more than six standard deviations from the mean. For similar reasons as if we trained our model using observations with very low values, training it with observations with such high values might be problematic. The upper ends of the expenditure ranges do not have natural "cap" levels similarly as the minimum references are natural floors, but it is not desired that there is a chance the model would predict an expenditure that is far above the true value for some new loan applicant. To prevent training our model using observations with too low or too high values, we must identify these values as outliers.

How to draw the line between an outlier and a non-outlier at the tail ends of the frequency distributions? The minimum reference gives us a natural lower limit for non-outliers, below which to consider an observation an outlier. As for the upper tail observations, we can use the conventional practice of choosing a number of standard

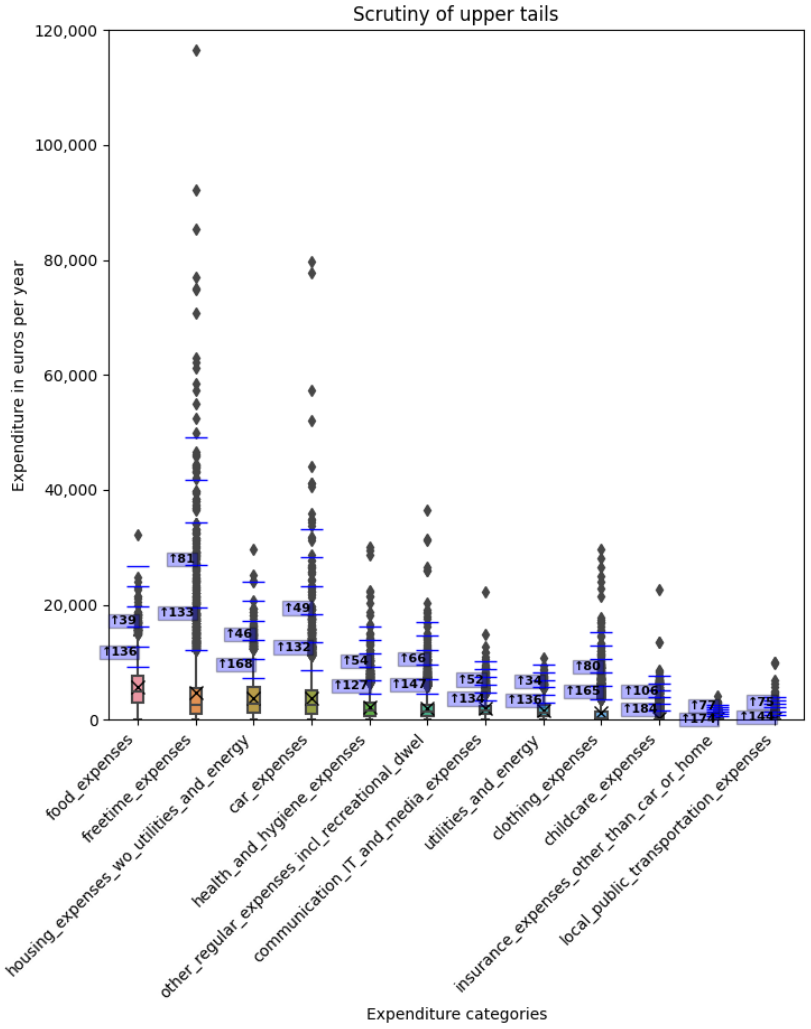


**Figure 4.5:** Box plot of expenditures by category with numbers in the blue boxes showing the number of observations above two (lower) and three (upper) standard deviations from the mean.

deviations above the mean to mark the upper limit for non-outliers [13]. Judging by Figure 4.5, either two or three standard deviations above the mean seems like a reasonable upper threshold, letting us detect most of the observations denoted by the diamond-shaped markers (which the box plot plotting function determines with some threshold in a similar way), while not throwing away excessively many observations. We shall describe the handling of these outliers in the section on pre-processing.

We are yet to address the third observation made earlier: no households have expenditures systematically among the top three largest across the categories. This observation is connected to an essential question: what to make of the outliers? Are they due to errors in data recording or input, such as typing errors or accidentally using wrong units or perhaps data from a wrong tracking period? Are they due to misunderstanding, inaccuracy or carelessness, or perhaps outright dishonesty of the respondent or the data collector? Or are the outliers simply truthful observations of

exceptional households? We have no way of knowing the answer, and there are of course likely to be multiple different reasons behind the outliers. The fact that no single household was systematically among the top three across the categories suggests that there is no household that was clearly misreporting its numbers much higher than the actual, say, by reporting its expenditures in the unit of cents instead of euros.

### 4.2.2 Household Compositions

Let us then scrutinize the household compositions in more detail. Figure 4.6 shows the frequencies of different household compositions in the microdataset. Here, the household compositions are differentiated by their number of adults from the variable 'aikuisia' and that of children under 25 living as dependants of their parents or guardians, meaning that they are e.g. non-working and unmarried, from the variable 'lapsia24'. The value of 'lapsia24' is effectively the sum of the other variables representing the number of children in different age brackets, namely 'muku, 'peru', 'murk' and 'isot'. As such, 'lapsia24' was not included in Table A.1 in the appendices as a variable to proceed to modeling with, because the information in it was deemed needlessly overlapping with that in those four variables. In a similar way, an entry in 'aikuisia' may be overlapping with 'lapsia24', because aikuisia encompasses all persons that turned 18 in the year of the survey (2016) or older. Therefore, the same person may be counted in both the 'aikuisia' and 'lapsia24' element of the tuple ('aikuisia', 'lapsia24') in Figure 4.6.

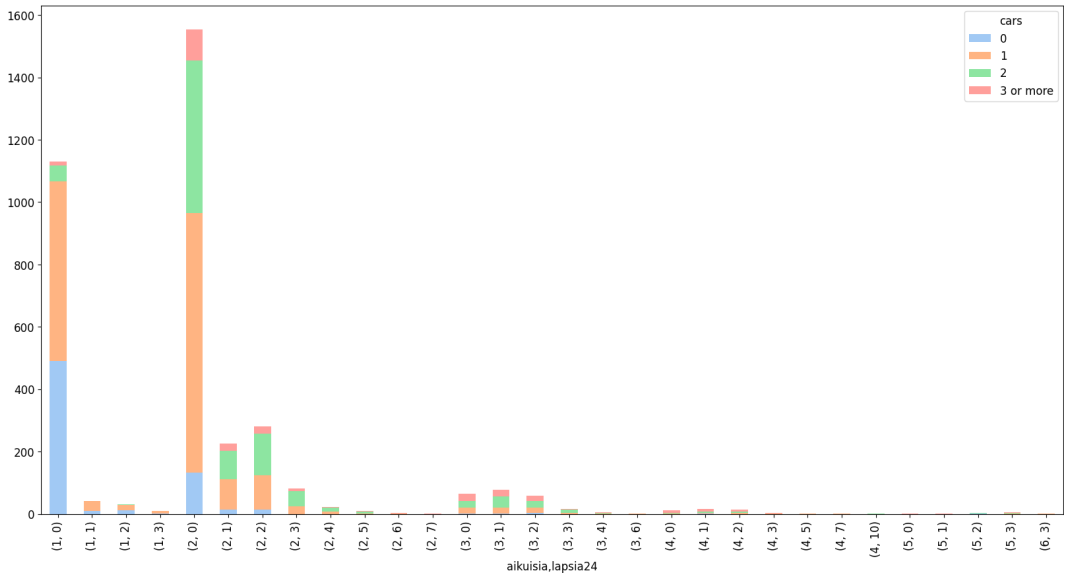


**Figure 4.6:** Frequency distribution of household compositions by the number of adults and that of children under 25 living as dependants of their parents or guardians (i.e. non-working, unmarried) as well as by the number of cars.

The above picture does not change significantly if the definition of child is changed from under 25 to under 17 years, as evident from Figure 4.7. The value of 'lapsia17' is effectively the sum of 'muku, 'peru', and 'murk', and for the same reason as for 'lapsia24' it is not included in Table A.

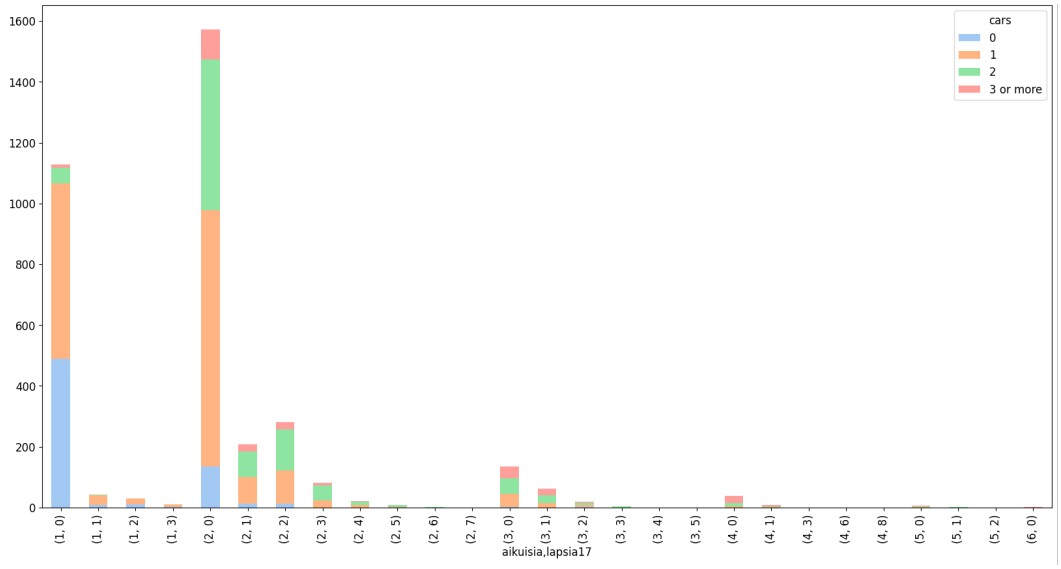


**Figure 4.7:** Frequency distribution of household compositions by the number of adults, children under 17, and number of cars.

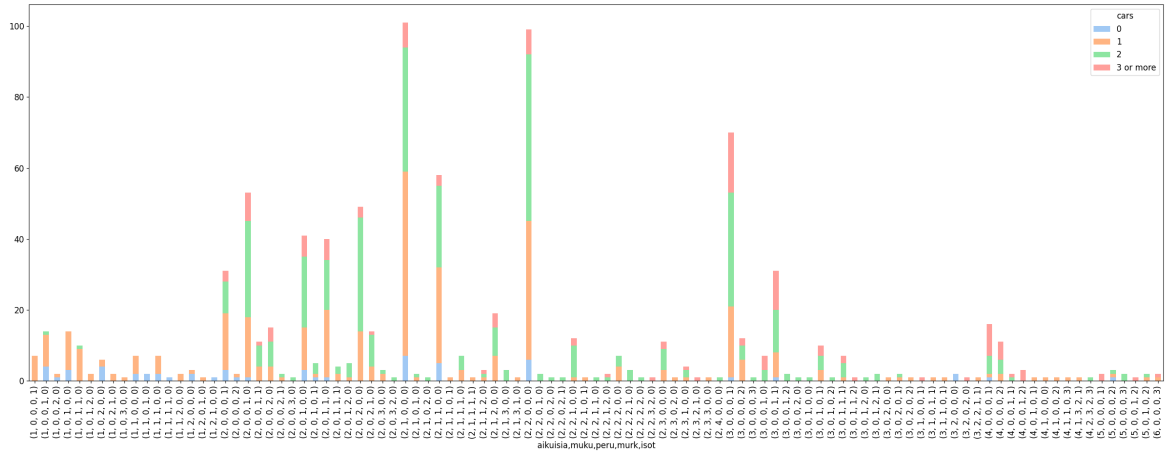


**Figure 4.8:** Frequency distribution of household compositions on the most granular level: by the number of adults, children under school age, children from 7 to 12, children from 13 to 16, and children from 17 to 24 living as dependants of their parents or guardians, as well as by the number of cars.

Figure 4.8 gives us a more granular look into the household compositions as specified by 'aikuisia', 'muku', 'peru', 'murk', and 'isot'. Note that by the definition of 'aikuisia', 'isot', i.e., children between 17-24 living as dependants of their parents or guardians, should have an entry in both 'aikuisia' and 'isot' if the child was 18 by the

end of the year of the survey or older. This is likely to explain most of the household compositions that have 3-6 adults in Figure 4.8.

### 4.2.3 Data Profiling of the Predictor Variables

We carried out basic profiling on the predictor variables in the microdataset. Before profiling, however, we split a 15-20% test set off the dataset, and conducted the profiling only on the remaining 80-85%, of the data, the training set. This is a best practice to prevent *data leakage*, also called *information leakage*. Data leakage effectively makes the model more likely to overfit to the test set due to the model inadvertently learning aspects about the test set in the pre-processing, training, or model selection stages, resulting in poorer generalization [34, 13, 38]. To top it all, the poor ability to generalize is not revealed by evaluating against the test set, but only becomes evident when evaluating against entirely unseen, truly out-of-sample observations, which can be done only once the ground truth of such observations is attainable, often very much in retrospect.

To prevent data leakage, we keep the test set entirely unseen in process steps where either some algorithm or the modeler might make or affect any modeling decisions based on the data [38]. Even exploratory data analysis with its typical exploratory visualization and data profiling activities might be such a process step, because, e.g., the appearance of the distributions of the variables may affect the modeler's modeling decisions. For example, say the modeler looks at a data profile done on the dataset before splitting it into a training and test set and decides to drop a particular variable because its distribution does not match the modeler's pre-conception. The intent behind this may be an innocuous attempt to screen out irrelevant or noisy variables, but it can introduce a harmful amount of data leakage from the test set into the model, because the modeler utilizes information that should not be available at training time.

We used the Python library PandasProfiler to create the profiling. A summary of the profile of the training set is shown in Figure 4.9. Let us next look at different aspects that the profiling charted about our training set.

Dataset statistics

| | |
|---|---|
| Number of variables | 36 |
| Number of observations | 2938 |
| Missing cells | 4324 |
| Missing cells (%) | 4.1% |
| Duplicate rows | 0 |
| Duplicate rows (%) | 0.0% |

Variable types

| | |
|---|---|
| Numeric | 35 |
| Text | 1 |

**Figure 4.9:** Summary from the profiling of the training set.

#### 4.2.3.1 Missing Values

One aspect of the profiling was the analysis of missing values, illustrated in Figure 4.10. Value 1.0 in the figure means that 100% of the values were non-missing, and a lower value that its difference to 1.0 was the relative share of missing values for that particular variable. The variables with the majority of the missing values, 'pamto', 'ppika', 'ppsup', and 'psoss90', all had to do with the spouse of the reference person. The most likely explanation for the missing values in these variables is that these respondents did not have a spouse, and thus a missing value was recorded. Apart from these spouse-related variables, the only variable with missing values was 'paalammi', the type of the main heating source in the household, with four missing values in the training set.

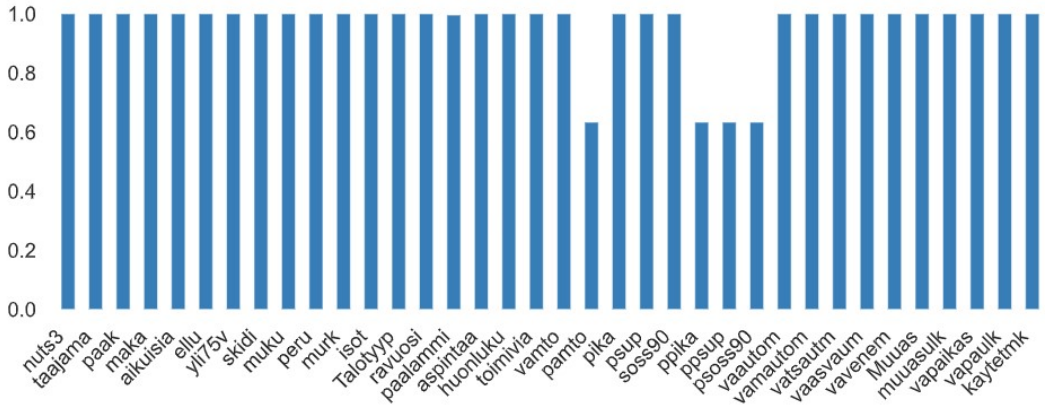


**Figure 4.10:** Relative shares of missing values in the training set.

#### 4.2.3.2 Correlations

The profiling also provided analysis of correlations between the variables by producing a (Pearson) correlation matrix shown in Figure 4.11. In it, we can see some variable pairs that correlate strongly, such as 'vamto' (status in employment of the reference person of the household) and 'soss90' (socioeconomic status of the reference person).

Another look into the correlations in the dataset was provided by the cluster_columns function from the Fastbook library. It computes the Spearmann correlation between the variables in a hierarchical fashion and displays the results as a dendogram, as show in Figure 4.12. In the dendogram, the closer a split between variables or groups of variables is to zero, the stronger the correlation between them. For example, we can see that the variables 'huonluku' (number of rooms) and 'aspintaa' (dwelling area in square meters) correlate the most, which makes intuitive sense. The colored branches of the dendogram indicate groups of variables that had significant correlation with one another. In the following chapter we will see whether what our

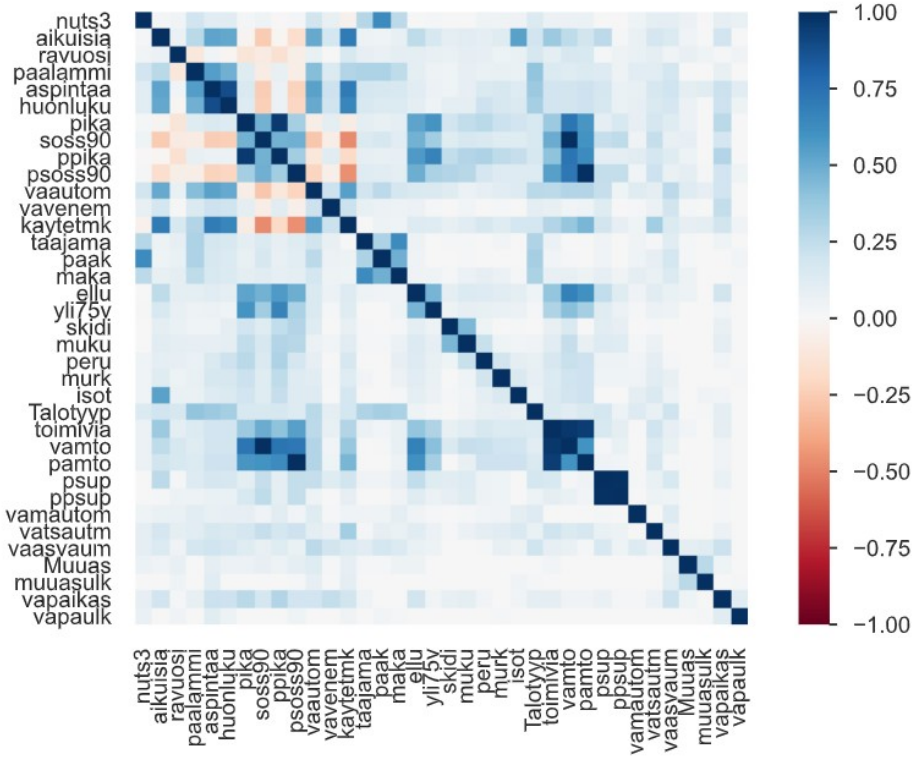


**Figure 4.11:** Correlation matrix of the training set.

feature elimination methods suggest as variables to be removed are in line with the correlations found here.

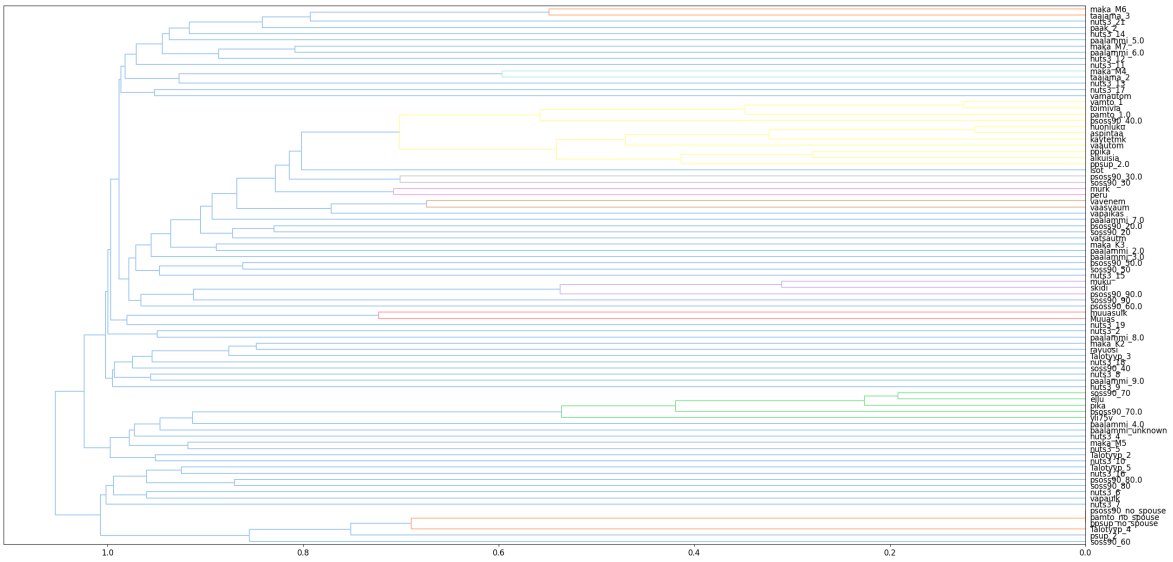


**Figure 4.12:** Correlation dendogram of the training set.

#### 4.2.3.3 Univariate Profiles

Moreover, the profiling provided us with univariate profiles and summaries, e.g., frequency distributions, of the variables. Figure 4.13 shows the profile of one of the numeric variables with a "wide" distribution (i.e. a distribution with high cardinality), 'aspintaa'. We can see that the frequency distribution is somewhat skewed to the side of higher values, while ideally for some types of modeling we would like to see it resemble a normal distribution. We shall talk about transformation as a means to try to render variable distributions more towards Gaussianity in the section on pre-processing.

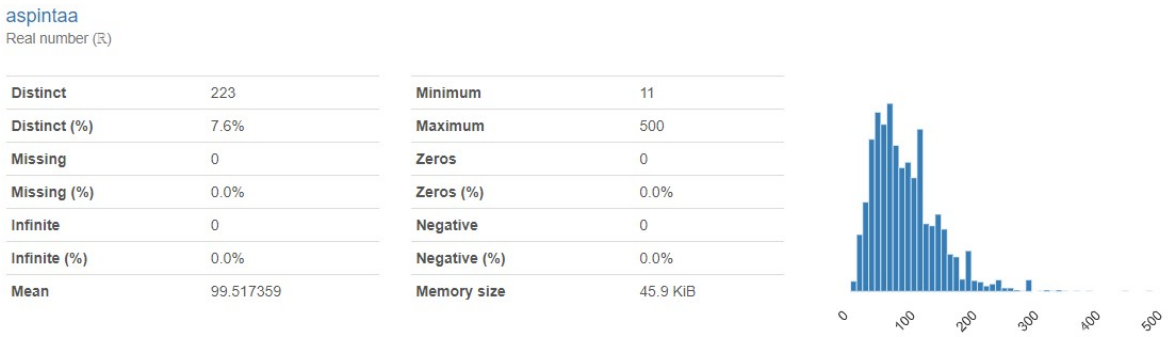


**Figure 4.13:** Initial profile of 'aspintaa'.

Figure 4.14 shows a typical profile of a numerical variable with a "narrow" distribution (i.e. a distribution with low cardinality), such as variables representing low discrete quantities like the number of household members. This particular distribution is that of 'muku', children under the age of 7. The narrower the distribution, the more difficult it is to transform the distribution towards Gaussianity. Moreover, variables like 'muku' with the frequency for the "zero-point" dominating are far from normally-distributed by nature, and no transformation can make them so.

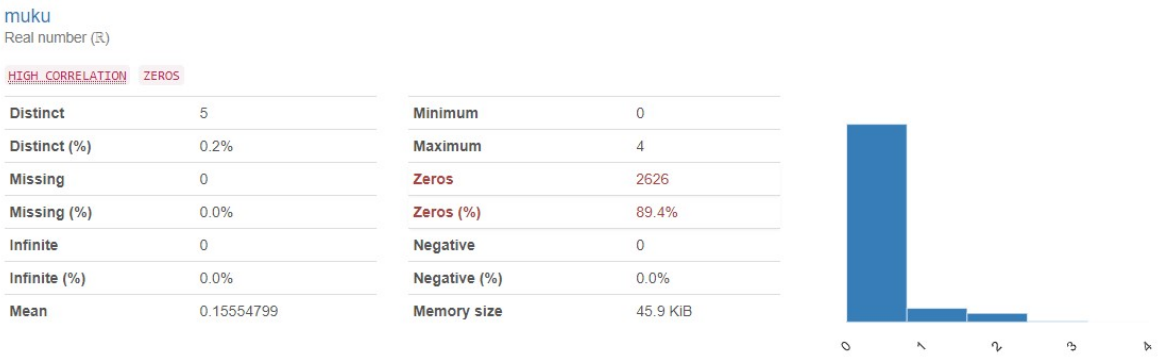


**Figure 4.14:** Initial profile of 'muku'.

## 4.3 How to Quantify our Models' Quality?

Once the nature of the modeling problem is clear, it is a good practice to take a moment to define "what success looks like". This should include deciding the metrics to be used for model evaluation and the baselines for benchmarking the models. The metrics and benchmarks do not have to be set in stone at this stage, but if one decides them early on and sticks to them, they can at least be more confident that they did not cherry-pick the metrics and benchmarks to be favorable for their models.

### 4.3.1 Metrics for Model Evaluation

The *risk* or *scoring metrics* we chose for evaluation were *coefficient of determination*, or *R squared* ($R^2$), *explained variance* (EV), *mean absolute error* (MAE), and *root mean squared error* (RMSE), which are commonly used evaluation metrics for regression tasks [11, 10]. In the context of machine learning, the term risk refers to the expected value of the loss function, which in turn is some function that quantifies and aggregates the error, a random variable, in a particular way. To be precise, error refers to the difference between the underlying ground truth – the true values from the DGP – and the predictions, while *residual* refers to the difference between the values according to our sample and the predictions. Residuals can be considered our estimates of errors. However, the term error is often used quite loosely to mean the same as residual, and this is how we use it in this thesis as well.

$R^2$ and EV are similar metrics with an upper bound of one in their range of values. $R^2$ is a commonly used measure of the *goodness-of-fit* of a model [10, 11]. It is defined as

$$R^2 = 1 - \frac{u}{v}, \tag{4.1}$$

where $u$ is the *residual sum of squares*, or *sum of squares of residuals*,

$$u = \sum_{i}^{n} (y_{true,i} - y_{pred,i})^2, \tag{4.2}$$

and $v$ is the *total sum of squares* – or *regression sum of squares* or *explained sum of squares* particularly in the context of regression analysis – defined as

$$v = \sum_{i}^{n} (y_{true,i} - \bar{y}_{true})^2, \tag{4.3}$$

where $y_{true,i}$ is the observed value of the $i$-th observation in our sample (i.e. a true value as per our sample), $y_{pred,i}$ is our modeled or predicted value for the observation, $\bar{y}_{true}$ is the mean of the observations in the sample (or in the split of the sample we are scoring against), and $n$ is the number of observations over which we estimate $R^2$ [55].

Looking at the above definitions of $R^2$ and residual sum of squares, it is obvious that if the modeled or predicted value from our model matched the observed values $y_{true}$ perfectly, then the term $\frac{u}{v}$ would be zero and the value of $R^2$ one. Thus, the $R^2$ of a model matching the true values perfectly with no error is one. In contrast, if the modeled or predicted value from our model was just the constant $\bar{y}_{true}$, then the term $\frac{u}{v}$ would be one, and the value of $R^2$ zero. Thus, the $R^2$ of a model constantly "predicting" the mean of the true values, effectively disregarding the predictor variables, is zero.

Furthermore, $R^2$ can be negative when the predictions of the model are arbitrarily worse compared with "predicting" just the mean of the true values. This normally happens only if the model is not based on a fit to the true values we are scoring its predictions against (e.g. fitted to a training split and scoring against a test split), and the predictions are not generalizing well to these true values. Thus, normally the training $R^2$ should always be at least zero, because we have fitted the model to the same data we are scoring it against. So at the very least the model can just predict the mean of the training data, yielding zero as the $R^2$ score. In scoring against the test set, the model does not know the mean, so it cannot resort to "predicting" that. If the model does a lousier job at predicting the true values compared with if it knew the mean of the test set and just "predicted" that, the $R^2$ becomes negative.

A common notation for the definition of $R^2$ is

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}, \tag{4.4}$$

where the term $\frac{SS_{res}}{SS_{tot}}$ is called the *unexplained variance*. $R^2$ is thus the complement function of the unexplained variance, so logically it is the *explained variance*. Indeed, the interpretation of $R^2$ when the modeled values or predictions have been obtained from a linear regression model is that it expresses the proportion of the variance in the target variable that is explained or predicted by the predictors. In other words, the value of $R^2$ expresses the percentage of the variability of the target variable that the predictors have accounted for, while its complement function, the unexplained variance, is the variability that is still unaccounted for [23].

How is then the EV metric different from $R^2$, since the latter is already understood as explained variance? The definition of EV is

$$\mathrm{EV} = 1 - \frac{\mathrm{Var}(Y_{true} - Y_{pred})}{\mathrm{Var}(Y_{true})}, \tag{4.5}$$

where $Y_{true}$ is a vector of observed values in our sample (i.e. a vector of true values as per our sample) and $Y_{pred}$ is a vector of our modeled or predicted values for the observations. EV is thus the complement function of the ratio between the variance of the residuals and the variance of the observations. The definition seems very similar to

that of $R^2$, but what is not so obvious is that this definition of explained variance fails to account for systematic offset in the predictions. The failure to do so means that EV gives a value close to 1.0 if the error just remains close to constant from prediction to prediction, even if the predictions' difference to the true value is high, i.e., there is a systematic offset. Instead, $R^2$ for the same predictions would not give a good score, and is therefore usually the preferred metric for explained variance, because it is thus more truthful as a measure of accuracy of the predictions. The use for EV as a performance metric is that if it gave a good score while $R^2$ did not, that would hint us of the presence of a systematic offset in the predictions, which we could probably easily rectify [54].

The two other metrics, RMSE and MAE, are ones with an arbitrary range of values unlike $R^2$ and EV with their upper bound of 1.0. RMSE and MAE describe the amount of error in our modeled values or predictions. The scale of the error and consequently the values of RMSE and MAE depends on the order of magnitude of our target variable, the units in which the target variable is expressed, and – naturally – the accuracy of our modeled values or predictions.

RMSE corresponds to the square root of the *mean squared error*, or square root of the *expected value of the squared loss* (i.e. a risk function, as implied earlier):

$$\mathrm{RMSE}(y, y_{pred,i}) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_{true,i} - y_{pred,i})^2}, \tag{4.6}$$

where $n$ is the number of observations over which we estimate RMSE. MAE corresponds to the *expected value of the absolute error loss* or *L1-norm loss*:

$$\mathrm{MAE}(y, y_{pred,i}) = \frac{1}{n}\sum_{i=0}^{n-1}\left|y_{true,i} - y_{pred,i}\right|, \tag{4.7}$$

where $n$ is the number of observations over which we estimate MAE [54].

The essential difference between RMSE and MAE is that due to the quadratic nature of the loss, or the squaring of the errors, RMSE gives proportionally more weight to, or "penalizes" for, predictions with a large error, since that is the effect squaring has on large numbers. In contrast, MAE only penalizes for errors in the same proportion as the error itself, thus making it more "tolerant" of outliers. Which one of these two metrics to pay more attention to should be determined by the nature of the prediction problem. If the cost (i.e. "harm inflicted") of individual predictions with large errors, i.e., outliers in our predictions, is high, it might be wise to stress RMSE more in the evaluation, but if the cost of occasional predictions with large errors is low and one does not want them to make an otherwise well-performing model look bad, then MAE might be the metric to look at.

### 4.3.2 Baselines for Benchmarking

The purpose of benchmarking is that it functions as a "sanity check" for the use of more complex modeling solutions. If a given solution does not perform clearly better than benchmarks, then its use might not be justified [10]. We defined four different baselines for benchmarking to compare the performance of our prediction models to.

To serve as our first benchmark, *benchmark 1*, we used data from the University of Helsinki study on minimum reference budgets of households and Statistics Finland's household consumption statistics (i.e. the statistics which our microdataset is the underlying data of) to develop a system based on a table of numbers, or tabulated estimates, representing the contribution of different household members to the loan applicant's household's expenditures. In this system, for each expenditure category, the first adult of the family is given an individual number – their contribution to the given expenditure category – and the adults after the first are given another, smaller number. The number being smaller stems from the fact that in most household expenditures there are "synergy gains" as the adults of the household share the fixed costs, and thus, the costs that come along with, e.g., the "second adult of the household" are only some fraction of the first adult's costs. As for the contribution of children to the expenditures, there are two different numbers in this system, one for two different age brackets. One bracket is children aged 0-7 and the other one 7-17. Generally, these numbers are also smaller than that for the first adult, except that for the older age group the number is larger in a couple of expenditure categories.

What these tabulated numbers effectively define is a linear regression model with four predictor variables: the first adult, the number of adults after the first, the number of children aged 0-7, and the number of children aged 7-17. In fact, since households always have at least one adult, the first of these variables is always one, and can be ignored. The tabulated number of the first adult then represents the intercept of the regression fit of the resulting three-variable linear regression model.

Note, however, that these tabulated estimates were not determined by fitting a linear regression model, but otherwise derived from data from two datasets: the minimum reference budgets and the statistics. The data in these datasets is aggregated data, namely means and numbers based on a consensus of the survey participants. This was the approach used with benchmark 1, because we wanted to have one benchmark that is derived just from publicly available data. Deriving the tabulated estimates from these two aggregate datasets involved plenty of manual work in an Excel spreadsheet, making a number of simplifying assumptions and consequent transformations to harmonize and adjust the data from the two datasets to be comparable. We shall not delve deeper into what the exact procedure of arriving at these tabulated estimates

was, but just take them as is to serve as a benchmark.

Fitting a linear regression model to the microdataset, i.e., determining a fit that has minimal errors in the microdataset, could provide estimates of regression coefficients and an intercept that can be interpreted as the contributions of the household members to the expenditures much like the tabulated estimates in benchmark 1. *Benchmark 2* is such a linear regression model fitted to the microdataset. It consists of estimates of three regression coefficients and an intercept with an identical interpretation as the tabulated estimates of benchmark 1. These estimates can thus be thought of as "updated" values for the tabulated estimates of benchmark 1 based on a wider dataset on household consumption expenditures, the microdataset.

*Benchmark 3* is based on a division of the households in the microdataset into 12 "common sense groups": different compositions of under retirement-age adults-only households, retirement-age adults-only households, single-parent households with children, and two-parent households with children, as well as the rest of the households that did not belong to any of the former. In this benchmark, the estimates for the expenditures of an unseen household are then just the means of the different expenditure categories of the group wherein the given household falls. Effectively, this defines a "hand-crafted" decision tree, a kind of, or a representation of, an *expert system*.

*Benchmark 4* is based on clustering the observations in the microdataset by their predictor variables, and then assigning unseen observations to these clusters. It is a recommended practice to perform dimensionality reduction before clustering if the feature space is very high-dimensional, because in high-dimensional spaces we suffer from various manifestations of the *curse of dimensionality*, such as the fact that the commonly used distance measures, e.g. Euclidean, no longer work as expected [18]. Therefore, we first did a *principal components analysis* (PCA) transformation on the training set, dropped the rest of the principal components after a set cutoff level for the cumulative explained variance, and only then learned a set number of clusters from the remaining principal components using the *k-means clustering* algorithm. We then computed the means within the groups determined by the clusters, and so we could assign unseen observations to these clusters (effectively just *one-nearest neighbor* prediction using the cluster centroids that were learned by the k-means algorithm) and give the respective means of each expenditure category in the cluster as our predictions.

# 5. Methodology

Until now, we have discussed only observations made about the problem and framing the problem in data science or machine learning context, but, apart from describing the actions taken to derive the benchmarks, have not yet covered any "manipulative" actions done with the data. In this chapter, we describe the methodology used in the more "actionable" parts of the work, namely the algorithms and techniques used in data pre-processing for preparing the data for modeling and in the modeling and prediction pipeline, as well as the "technology stack" used in the implementation.

## 5.1 Data Pre-Processing and Preparation

The pre-processing steps we took consisted of handling outliers, handling missing values, and various feature engineering actions. Furthermore, our preparation for the modeling also included running various diagnostics and feature elimination steps.

### 5.1.1 Outlier Handling

As discussed in the exploratory visualization section, we have a good case to consider observations both at the lower end and at the upper end of the frequency distributions as outliers. The minimum reference values are an obvious choice to use as a threshold for outliers at the lower end. At the upper end, we set a number of standard deviations from the mean as the threshold. Then, once we have a way to identify outliers, we must yet decide what to do with them. There are two options: 1) to "clip", i.e., to floor the values below the lower threshold or to cap them above the upper threshold by assigning them the value of the threshold, or 2) to drop the observations below the lower threshold or above the upper threshold.

In the section on data visualization, we pondered about possible causes for the outliers. If we had a reason to suspect a particular cause for them, it would help us to choose an appropriate way of handling them in our pre-processing. This is because we could get a hunch on whether the outliers' values are in the right ballpark, i.e., in the correct tail end of the distribution, or whether the outliers' error to the actual value

is arbitrary. In the former case we could better justify choosing to floor or cap the outliers, while in the latter case we would probably be better off removing the outliers before our modeling. However, in this case we are left without any strong suspects of the causes for the outliers, so we simply must try different combinations of handling them and see and compare the outcomes.

Clipping the outliers is simple in terms of how the data is manipulated, as we simply assign the floor or the cap value to the outliers. In contrast, the removal of outliers is a bit more complex. If we removed the entire observation because the value for one of the 12 expenditure categories in that observation crossed our threshold for outliers, we would throw away the values for the other 11 expenditure categories as well (most or all of which likely to be non-outliers). Therefore, we must handle the outlier so that we remove the value only for the corresponding prediction problem, i.e., only the prediction for that target variable, the expenditure category, in which the outlier was. Besides adding complexity to the handling of the data by way of needing each prediction problem have its own dataframe retaining its specific non-outlier observations, this caused some challenges with multi-target models, as our dataframe for them was now somewhat sparse with missing values. Table 5.1 shows the numbers of observations we were left with for each prediction problem after dropping outliers.

### 5.1.2 Feature Engineering

The feature engineering stage of our data pre-processing consisted of transforming some of the numeric predictor variables, scaling the numeric predictor variables, handling the categorical predictor variables, doing manipulation and extraction on the raw input features, and doing two different steps of feature elimination on the features. From here on, let us call our explanatory/predictor variables *features* in line with the convention in data science and machine learning context.

#### 5.1.2.1 One-hot encoding

We call features whose values are on the (non-numeric) ordinal or nominal scale *categorical features*. With most learning algorithms, categorical features must be explicitly handled in some way to make them into *numeric features* before feeding them to the algorithm. For categorical features in the ordinal scale, a simple *integer encoding* that encodes the levels, or values, of the feature with integer values in an appropriate order is usually all that is needed. But integer encoding is often recommended against when dealing with categorical features in the nominal scale, because different integer values assigned to the levels imply ordinality between the levels. Thus, a learning algorithm taking in an integer-encoded categorical feature on the nominal scale would

**Table 5.1:** Number of observations left in the training set by prediction problem after 1) dropping upper tails at mean + 2 standard deviations, 2) dropping lower tails at minimum reference, 3) dropping upper tails at mean + 2 standard deviations and lower tails at minimum reference.

| Observations left in training set | After dropping upper tails at mean + 2 std | After dropping lower tails at minimum reference | After dropping upper tails at mean + 2 std and lower tails at minimum reference |
|---|---|---|---|
| total | 2938 | 2938 | 2938 |
| food | 2836 | 1361 | 1267 |
| freetime | 2827 | 2288 | 2177 |
| housing w/o... | 2803 | 2938 | 2803 |
| car | 2843 | 1597 | 1502 |
| health and hygiene | 2826 | 1818 | 1706 |
| other regular incl... | 2821 | 2095 | 1978 |
| communication, IT... | 2829 | 2330 | 2221 |
| utilities and energy | 2822 | 2059 | 1943 |
| clothing | 2803 | 823 | 688 |
| childcare | 2793 | 2938 | 2793 |
| insurance... | 2799 | 1417 | 1278 |
| local public transp. | 2825 | 188 | 82 |

learn there is an order among its levels, when there is not, possibly degrading the model performance [13].

A recommended way to handle categorical features in the nominal scale is *one-hot encoding* them to dummy features. This means creating a new feature for each level of the categorical feature and encoding the presence of that level on a given row with value one and absence with value zero. The biggest problem with one-hot encoding is that if there are many categorical features on the nominal scale and if they have high cardinality, i.e., a lot of levels, the end result will be a lot of dummy features and thus a very high-dimensional feature space [13]. Some learning models boast *native handling* of categorical features, meaning that the conversion to numeric form happens under the hood in some proprietary way on an as-needed-basis. The manuals of these models often claim their native handling provides superior performance compared with feeding the model one-hot encoded categorical features [62].

#### 5.1.2.2 Transformations

Contrary to common belief, it is not a strict requirement for any model, even linear regression, to have the frequency distributions of features resemble normal distribution (see e.g. [6, 50]). However, many sources assert that having more Gaussian-like features can be beneficial for better fulfilling assumptions underlying our model, leading to better results (see e.g. [13, 23, 45]).

We can perform various transformations on a feature to try and make its frequency distribution more closely normally distributed. Four common transformations for this purpose are the logarithmic, power, root and exponential transformation. For example, the logarithmic transformation means just taking the logarithm of, or applying the logarithmic function on, the values of the feature. One may think of the transformations alternatively as "warping" the scale from the arithmetic or linear scale to the logarithmic, power, root, or exponential one. Looking at the frequency distribution against this warped scale, we would ideally see a shape resembling normal distribution.

The warping effect of logarithmic and root transformation on the scale can be thought of as "stretching" the scale at the lower end, in a sense making the lower end more granular or having a higher resolution, while "compressing" the higher end to be more coarse or having a lower resolution. In contrast, the warping effect of exponential and power transformation can be thought of as "stretching" the scale at the higher end, while "compressing" the lower end. As for determining whether to use logarithmic or root transformation – and which base or root, respectively – is much a matter of trial and error. The same goes for exponential and power transformation and their respective base and exponent [13, 23, 45].

In this work, it was deemed sufficient to try a few different transformations on the numeric features that had a wide distribution, namely 'aspintaa', 'ravuosi', 'pika', 'ppika', and 'kaytetmk', and to simply assess visually if the resulting distribution looked more Gaussian than the starting point, and whether the result looked Gaussian enough or if a different transformation should be tried instead. Thus, transformations were not applied to numeric features with narrow distributions, such as the features representing the number of household members.

Figure 4.13 showed the skewed frequency distribution of 'aspintaa' and Figure 5.1 shows another case of a skewed frequency distribution, that of the variable 'kaytetmk'. Let us use these two as examples of transforming skewed features towards Gaussianity. The resulting distribution of 'aspintaa' after a base-2 log-transformation is shown in Figure 5.2, and that of 'kaytetmk' after a square root transformation in Figure 5.3. Assessing just visually, it seems like these transformations were useful in rendering the

kaytetmk
Real number (ℝ)

HIGH CORRELATION

| | | | |
|---|---|---|---|
| Distinct | 2895 | Minimum | 0 |
| Distinct (%) | 98.5% | Maximum | 543014 |
| Missing | 0 | Zeros | 1 |
| Missing (%) | 0.0% | Zeros (%) | < 0.1% |
| Infinite | 0 | Negative | 0 |
| Infinite (%) | 0.0% | Negative (%) | 0.0% |
| Mean | 50349.782 | Memory size | 45.9 KiB |

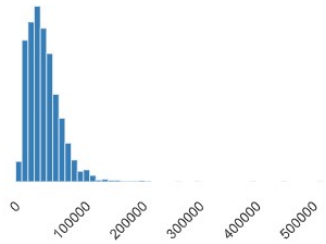


**Figure 5.1:** Initial profile of 'kaytetmk'.

frequency distributions more towards Gaussian. The benefit of possibly getting them look even more Gaussian with more trial and error using more complex transformations was not deemed worth the extra effort, and thus we settled for these outcomes.

aspintaa
Real number (ℝ)

| | | | |
|---|---|---|---|
| Distinct | 223 | Minimum | 3.4594316 |
| Distinct (%) | 7.6% | Maximum | 8.9657843 |
| Missing | 0 | Zeros | 0 |
| Missing (%) | 0.0% | Zeros (%) | 0.0% |
| Infinite | 0 | Negative | 0 |
| Infinite (%) | 0.0% | Negative (%) | 0.0% |
| Mean | 6.441338 | Memory size | 48.8 KiB |

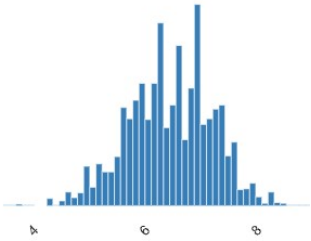


**Figure 5.2:** Transformed profile of 'aspintaa'.

kaytetmk
Real number (ℝ)

| | | | |
|---|---|---|---|
| Distinct | 2900 | Minimum | 0 |
| Distinct (%) | 98.7% | Maximum | 736.89484 |
| Missing | 0 | Zeros | 1 |
| Missing (%) | 0.0% | Zeros (%) | < 0.1% |
| Infinite | 0 | Negative | 0 |
| Infinite (%) | 0.0% | Negative (%) | 0.0% |
| Mean | 214.4514 | Memory size | 48.8 KiB |

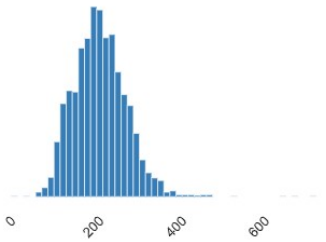


**Figure 5.3:** Transformed profile of 'kaytetmk'.

#### 5.1.2.3 Scaling

Scaling of features is another pre-processing step, and one that is likely much more crucial for many models than transformation of features towards Gaussian. The reason for doing scaling is to make the features not have unequal effect on the model just due

to differences in the scale in which they are presented. Without scaling, e.g., whether the disposable income, 'kaytetmk', was reported in euros or in cents would likely make a big difference to the model. Also, since the values of 'kaytetmk', even in euros, are many orders of magnitude larger than those of features representing small quantities, such as the number of household members, the latter might easily get overpowered by the former in the way the model represents the effect of the features. Therefore, it is important for models to have their features scaled, although there are exceptions; e.g., tree-based models should normally not be affected by either transformation or scaling of features [13].

We considered two common methods of scaling. One was min-max scaling, or normalization, that scales the values of a feature to a range between 0 and 1, while keeping the "distances" between the values in the same proportion within that range as they were in the original scale. Min-max scaling seemed like a more reasonable scaling for features representing low quantities, such as the number of household members. Another type of scaling we considered was standardization scaling, or Z-score normalization, in which the mean of the frequency distribution is shifted to zero, and the distribution is scaled so that the width of one standard deviation becomes one. Standardization scaling is said to benefit especially models that involve finding linear combinations of features to explain the variance in the target variable.

As there were different models with differing needs for scaling used in this work, three dataframes with different scaling were produced to cater for these models on top of keeping one dataframe unscaled (and untransformed). One of the dataframes was with all numeric features min-max scaled, another one was with all features (including dummy variables) standardized, and a third one was a mixture with the features with wider distributions standardized and the rest of the numeric features min-max scaled.

#### 5.1.2.4 Feature Manipulation and Extraction

With some of the features in Table A.1, using them as such was deemed unideal for two reasons: "sparseness" and "overlapping". Some of the features were "sparse", i.e., rarely containing a value other than zero (or some equivalent "zero point value"). The features 'vamautom' representing the number of recreational vehicles, 'vatsautm' representing the number of company-owned cars, and 'vaasvaum' representing the number of trailers and sleeping caravans were deemed particularly sparse. To address this issue, a new variable 'autoja' (cars) was created by summing up 'vamautom' and 'vatsautm' with 'vaautom' representing the number of cars and vans, whereas 'vaasvaum' was decided to be dropped.

Besides these, 'muuasulk' representing possession of a second home or a buy-to-

let home abroad, 'vapaulk' representing possession of recreational dwelling abroad, and 'vavenem' representing the number of motor or sailing boats were deemed so sparse as to simply be dropped. Moreover, 'muuas' representing possession of a second home or a buy-to-let home and 'vapaikas' representing possession of a recreational dwelling in Finland were combined together under a new variable 'vapaa_ajan_tai_kakkosasunto' (recreational or second dwelling).

Some of the features contained "overlapping" information. The overlapping of the information contained by 'aikuisia' and 'lapsia24' was already mentioned. To address this, the value of 'lapsia24' was deducted from 'aikuisia' on every row. Also, the information in 'aikuisia' overlapped with the information in the variable 'ellu' representing the number of over 64-year-olds in the household. To address this, also the value of 'ellu' was deducted from 'aikuisia' on every row. The result from these two deductions was assigned to a new variable 'aikuisia_alle65v' representing adults under the age 65, excluding those under 25-year-old adults that were living at home as dependants of their parents or guardians. Lastly, the variable 'aikuisia' was dropped. In the same vein, the information in the variable 'muku' representing the number of under seven-year-olds overlapped with the information in the variable 'skidi' representing the number of under three-year-olds. In this case, a new variable 'alle7v_yli2v' representing children over two but under seven was created and variable 'muku' dropped.

Furthermore, a number of small manipulative actions were carried out involving making sure the datatype of each feature is appropriate, handling missing values, e.g., filling in the value "no_spouse" to the spouse-related features where missing value most likely meant that the respondent had no spouse, and one-hot encoding categorical features. As the outcome of the pre-processing part of the workflow, we obtained a number of different versions of the dataframe containing the features so that we could cater for each model's particular needs. For example, one version of the dataframe was without transformations, scaling and categorical features one-hot encoded, because the gradient boosted tree models we used did not need transformation and scaling of features, or even one-hot encoding of the categorical features, because these models come with a native handling of categorical features. In contrast, one version of the dataframe was with all features standardized, because e.g. Gaussian processes and PCA work best with standardized features, and several versions were with categorical features one-hot encoded, because most models will not accept categorical features with nominal values. Moreover, for some of the benchmarks the features 'ellu', 'muku', and 'aikuisia' were needed, although for all our actual models they were not. Thus, there was also a version of the dataframe with these features included.

### 5.1.3 Feature Selection

After the feature manipulation and extraction steps, with some features combined together, some dropped, and others left as they were, we were still left with 27 features when categorical features were not one-hot encoded, and over 80 when they were. Next, we wanted to assess the importance of each feature in explaining the target variable in each expenditure category, based on which we hoped to be able to make informed decisions about dropping some more features.

#### 5.1.3.1 Measures of Feature Importance

In the data exploration section we saw that some of the variables correlated strongly. When we talk about correlation of features with each other, this is called *collinearity* or *multicollinearity*. High (multi)collinearity implies similarity or overlapping of the information content of the features (possibly as a *spurious relationship* through a *lurking variable*). But if we computed correlations between features and the target variable, we would obtain a measure of how each feature and the target vary together linearly, giving us insight into the strength of the linear relationship between them. This would give us a measure for comparing how important each of the features is in terms of strength of its linear relationship with the target.

Another way of obtaining a measure of importance would be by fitting a *simple linear regression model* with only a given feature against the target variable, and then computing the explained variance, or, the amount of variance of the target variable explained by the variance of the predictors, of this one-regressor regression model. Moreover, the estimates of regression coefficients of a linear regression model with all predictors included would give us an interpretation of the relative importances of the predictors in explaining the target variable.

A very different measure of the importance of features, often dubbed literally *feature importance*, is obtained by fitting any decision tree-based model, including tree ensemble models, to our dataset. This feature importance score is computed based on the reduction in the *splitting criterion* for each of the features in the decision tree or trees [13].

We computed the above-mentioned scores of feature importance for all the features. With categorical features, we also wanted to compute these metrics for their one-hot encoded dummy variables, so if the metrics indicated that any of the levels of the categorical features, i.e., any of the dummy variables, was of little importance we could just eliminate that one dummy variable and proceed to modeling and prediction without it in those models that took in categorical features as dummy variables.

Linear regression for *scaled and transformed* training set against total_expenses

| | | | | | | |
|---|---|---|---|---|---|---|
| Dep. Variable: | y | R-squared: | 0.511 | | | |
| Model: | OLS | Adj. R-squared: | 0.502 | | | |
| No. Observations: | 2938 | F-statistic: | 50.96 | | | |
| Covariance Type: | HC0 | Prob (F-statistic): | 0.00 | | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| kaytetmk | 68010.0 | 1.3e+04 | 5.242 | 0.000 | 4.26e+04 | 9.34e+04 |
| aspintaa | 16970.0 | 4294.825 | 3.951 | 0.000 | 8551.340 | 2.54e+04 |
| autoja | 15430.0 | 4523.499 | 3.410 | 0.001 | 6560.312 | 2.43e+04 |
| murk | 11610.0 | 2555.256 | 4.543 | 0.000 | 6599.195 | 1.66e+04 |
| skidi | 10840.0 | 2840.487 | 3.816 | 0.000 | 5270.952 | 1.64e+04 |
| isot | 10100.0 | 2360.994 | 4.277 | 0.000 | 5470.550 | 1.47e+04 |
| peru | 9306.3687 | 1853.162 | 5.022 | 0.000 | 5674.238 | 1.29e+04 |
| const | 8568.6736 | 1869.627 | 4.583 | 0.000 | 4904.272 | 1.22e+04 |
| toimivia | 8515.8572 | 3807.209 | 2.237 | 0.025 | 1053.865 | 1.6e+04 |
| alle64v_aikuisia | 8095.0677 | 3130.549 | 2.586 | 0.010 | 1959.304 | 1.42e+04 |
| pika | -6749.9284 | 1582.494 | -4.265 | 0.000 | -9851.560 | -3648.297 |
| psoss90_30 | 5366.1586 | 1662.802 | 3.227 | 0.001 | 2107.127 | 8625.190 |
| paak_2 | -4181.8506 | 875.181 | -4.778 | 0.000 | -5897.175 | -2466.527 |
| paalammi_5 | -4103.1203 | 1422.667 | -2.884 | 0.004 | -6891.497 | -1314.744 |
| alle7v_yli3v | 3876.6483 | 2182.172 | 1.777 | 0.076 | -400.331 | 8153.628 |
| soss90_30 | 3827.8494 | 1160.554 | 3.298 | 0.001 | 1553.205 | 6102.494 |
| soss90_20 | 3770.2451 | 1356.202 | 2.780 | 0.005 | 1112.138 | 6428.352 |
| soss90_70 | 3684.7522 | 958.100 | 3.846 | 0.000 | 1806.911 | 5562.593 |
| ppsup_2 | 3251.7751 | 809.389 | 4.018 | 0.000 | 1665.401 | 4838.149 |
| yli64v_alle75v | 3222.5798 | 1886.971 | 1.708 | 0.088 | -475.815 | 6920.975 |
| soss90_40 | 2994.5954 | 1060.598 | 2.823 | 0.005 | 915.861 | 5073.330 |
| Talotyyp_3 | 2798.2128 | 931.101 | 3.005 | 0.003 | 973.287 | 4623.138 |
| nuts3_10 | -2603.902 | 1134.578 | -2.295 | 0.022 | -4827.634 | -380.170 |
| nuts3_11 | -2496.3458 | 960.376 | -2.599 | 0.009 | -4378.648 | -614.043 |
| Talotyyp_4 | 2384.0806 | 1011.543 | 2.357 | 0.018 | 401.492 | 4366.669 |
| nuts3_4 | -2270.3336 | 898.626 | -2.526 | 0.012 | -4031.609 | -509.058 |
| ravuosi | 2264.7741 | 984.141 | 2.301 | 0.021 | 335.893 | 4193.656 |
| psoss90_20 | 2142.8108 | 1775.033 | 1.207 | 0.227 | -1336.190 | 5621.812 |
| paalammi_8 | -2106.1193 | 2037.749 | -1.034 | 0.301 | -6100.035 | 1887.796 |
| paalammi_7 | -2048.1782 | 1303.232 | -1.572 | 0.116 | -4602.467 | 506.110 |
| nuts3_21 | -1866.1774 | 1195.924 | -1.560 | 0.119 | -4210.144 | 477.790 |
| psoss90_70 | 1737.8265 | 1356.181 | 1.281 | 0.200 | -920.240 | 4395.893 |
| vapaa_ajan_tai_kakkosasunto_1 | 1692.4758 | 565.292 | 2.994 | 0.003 | 584.525 | 2800.427 |
| Talotyyp_2 | 1670.649 | 1414.032 | 1.181 | 0.237 | -1100.803 | 4442.101 |
| psoss90_40 | 1471.533 | 1258.995 | 1.169 | 0.242 | -996.051 | 3939.117 |
| nuts3_7 | 1383.5556 | 1101.108 | 1.257 | 0.209 | -774.577 | 3541.688 |
| nuts3_12 | -1341.5776 | 915.837 | -1.465 | 0.143 | -3136.585 | 453.429 |
| psup_2 | 1292.9619 | 546.270 | 2.367 | 0.018 | 222.293 | 2363.631 |
| paalammi_6 | -1281.8069 | 1029.838 | -1.245 | 0.213 | -3300.252 | 736.638 |
| nuts3_9 | 1260.3095 | 1334.112 | 0.945 | 0.345 | -1354.501 | 3875.120 |
| paalammi_3 | 1155.8234 | 1642.279 | 0.704 | 0.482 | -2062.984 | 4374.631 |
| nuts3_5 | -1105.7842 | 1188.933 | -0.930 | 0.352 | -3436.050 | 1224.481 |
| soss90_60 | 1050.8545 | 1054.488 | 0.997 | 0.319 | -1015.905 | 3117.614 |
| nuts3_14 | -1046.9179 | 1119.643 | -0.935 | 0.350 | -3241.378 | 1147.543 |
| nuts3_13 | -858.9818 | 892.892 | -0.962 | 0.336 | -2609.018 | 891.054 |
| psoss90_50 | 843.7322 | 1358.332 | 0.621 | 0.534 | -1818.550 | 3506.014 |
| paalammi_2 | -770.0207 | 783.395 | -0.983 | 0.326 | -2305.448 | 765.406 |
| psoss90_80 | 580.5902 | 1210.797 | 0.480 | 0.632 | -1792.529 | 2953.710 |
| ppika | -346.8783 | 1938.163 | -0.179 | 0.858 | -4145.608 | 3451.852 |
| nuts3_17 | 343.9114 | 909.341 | 0.378 | 0.705 | -1438.363 | 2126.186 |
| paalammi_4 | 297.9538 | 1081.292 | 0.276 | 0.783 | -1821.340 | 2417.247 |
| nuts3_6 | 203.3397 | 820.527 | 0.248 | 0.804 | -1404.863 | 1811.542 |
| yli75v | 173.6431 | 1918.746 | 0.090 | 0.928 | -3587.030 | 3934.316 |
| soss90_50 | -167.7145 | 1010.069 | -0.166 | 0.868 | -2147.412 | 1811.984 |
| nuts3_2 | 121.1025 | 795.935 | 0.152 | 0.879 | -1438.902 | 1681.107 |

Notes:
[1] Standard Errors are heteroscedasticity robust (HC0)

**Figure 5.4:** Linear regression diagnostics for scaled and transformed training set with total expenses as the target variable.

#### 5.1.3.2 Linear Regression Diagnostics

As mentioned, the estimates of regression coefficients from regression analysis give one way to assess the (relative) importances of the features in predicting the value of the target or explaining its variance. Apart from this, fitting a linear regression model to data and running various statistical diagnoses based on it can give plenty of insight into the ability to model the linear relationships in the data and possible problems therein. In other words, these diagnostics are a way to check whether our modeling assumptions are reasonable [41]. We used the Statsmodels library to run such diagnostics on the training set.

In the diagnostics report of a linear regression model fitted to our training set shown in Figure 5.4, we can see:

1) the point estimates of coefficients sorted by their absolute value,

2) the standard error computed for the coefficient estimates,

3) the z-statistic for testing the null hypothesis that a given coefficient is zero together with the corresponding p-value, and

4) the 95% confidence interval, or interval estimate, for the coefficients.

Other interesting results in the report include the $R^2$ describing the fit of the regression model to the data, $R^2$ adjusted, which describes the fit when accounting for the "inflation of $R^2$", or, penalizes for excessive features, and the F-statistic for testing the null hypothesis that the features are not significant in explaining the variance of the target variable, i.e., that the regression line's (hyperplane's) slope is equal to zero, and the corresponding p-value ("Prob").

Moreover, the report would show a notification at the "Notes:" section at the bottom if there were problems in the data with, e.g., multicollinearity. We got a notification about multicollinearity when running this diagnostics before feature elimination. But since in the diagnostics shown in the figure we had done feature elimination based on multicollinearity, we did not get this notification. For example, the variable 'huonluku' (number of rooms) was dropped since it correlated strongly with 'aspintaa' (dwelling area in square meters), as we saw in section Data Exploration and Exploratory Visualization. The feature elimination steps will be described in the following section.

Figure 5.5 shows how the fit of the linear regression model looks like when plotted against one numeric feature as a free variable at a time while keeping the rest of the features fixed. Note that the features here are transformed and scaled, as described earlier, which somewhat obscures the interpretation. But at least the features representing quantities are easy to interpret, as they start from zero (except for alle64v_aikuisia, which starts from 1) and every "pillar" in the plot is just an increment of one in the quantity.

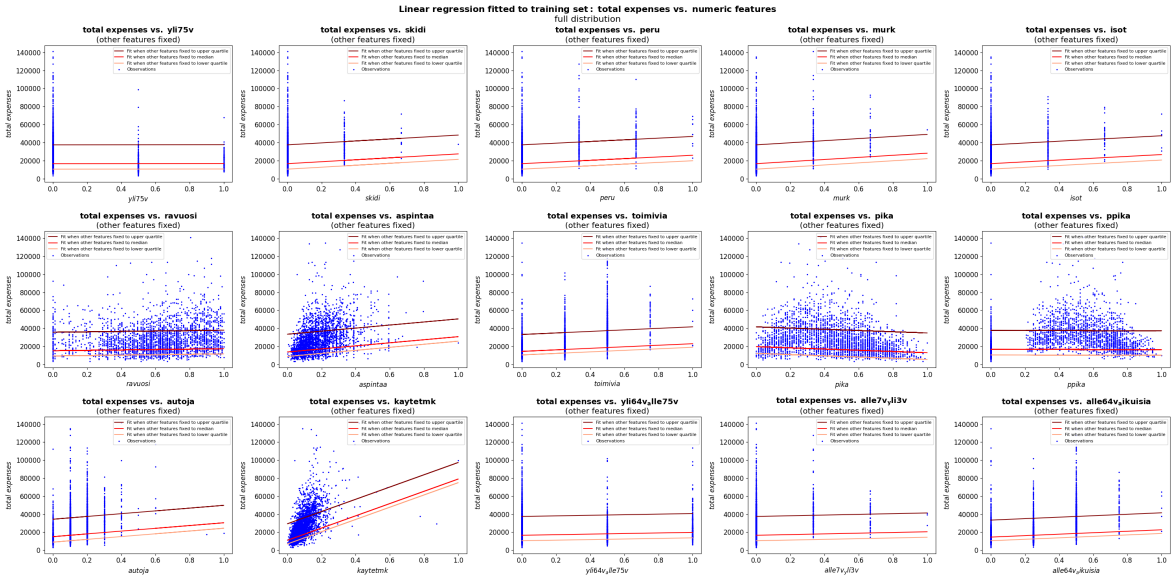


**Figure 5.5:** Plot of the fit of linear regression model in scaled and transformed training set by each numeric feature at a time while keeping the other features fixed to a) lower quartile (orange), b) median (red), and c) upper quartile (brown).

#### 5.1.3.3 Feature Elimination

After the actions described above, we ended up with a dataframe with four different measures that can be interpreted as telling us about the importance of features – correlation, $R^2$, (tree-based) feature importance, and estimates of regression coefficients – for each of the dozens of features we still kept onboard. We could sort the dataframe by any of the metrics to scrutinize how each of them would "rank" the features by that specific measure of importance. The rankings by the metrics were partially in line with each other, but had also many discrepancies, so it was not an easy task to decide which features could be dropped just by looking at this dataframe. Therefore, we decided to fully rely on libraries purpose-built for picking features to be eliminated.

Firstly, we used a library called FeatureSelector for feature selection and elimination. The library fits a LightGBM regressor, a gradient boosted trees-based ensemble model, into the dataset to obtain feature importances. It also has functionality to identify (multi-)collinearity of features, and convenience functions to drop features that 1) have a zero feature importance, 2) have a contribution to the cumulative feature importance beyond a set threshold, and 3) exhibit collinearity higher than a set threshold. Moreover, in 3), the convenience function smartly attempts to drop first those features that correlate strongly with the largest number of other features, so that collinearity is eliminated while the maximum amount of information is retained. These feature elimination steps were carried out against each of the 12 target variables separately, resulting in differing sets of features for each of the 12 prediction problems.

The features dropped varied depending on, e.g., the chosen values for the thresholds and hyperparameters of LightGBM, and even from run to run due to stochasticity in LightGBM. Figure 5.6 shows the features (categorical ones one-hot encoded) dropped for some of the prediction problems when the collinearity threshold was set to 0.8 and the cumulative importance threshold to 0.99.

| total_expenses | food_expenses | freetime_expenses | housing_expenses_wo_... | car_expenses | health_and_hygiene_expenses | other_regular_expenses_... |
|---|---|---|---|---|---|---|
| huonluku | huonluku | huonluku | skidi | huonluku | huonluku | huonluku |
| Talotyyp_5 | Talotyyp_5 | Talotyyp_5 | huonluku | Talotyyp_5 | Talotyyp_5 | Talotyyp_2 |
| psoss90_60 | psoss90_20 | psoss90_50 | Talotyyp_5 | psoss90_60 | psoss90_60 | Talotyyp_5 |
| psoss90_90 | psoss90_60 | psoss90_90 | psoss90_60 | psoss90_80 | psoss90_80 | psoss90_60 |
| psoss90_no_spouse | psoss90_90 | psoss90_no_spouse | psoss90_70 | psoss90_90 | psoss90_no_spouse | psoss90_90 |
| nuts3_8 | psoss90_no_spouse | nuts3_7 | psoss90_80 | psoss90_no_spouse | nuts3_9 | psoss90_no_spouse |
| nuts3_15 | nuts3_9 | nuts3_10 | psoss90_90 | nuts3_5 | nuts3_19 | nuts3_9 |
| nuts3_16 | nuts3_11 | nuts3_15 | psoss90_no_spouse | nuts3_8 | soss90_60 | soss90_60 |
| nuts3_18 | nuts3_16 | nuts3_16 | nuts3_5 | nuts3_16 | soss90_80 | soss90_80 |
| nuts3_19 | nuts3_18 | soss90_90 | nuts3_9 | nuts3_18 | soss90_90 | soss90_90 |
| soss90_80 | soss90_60 | ppsup_no_spouse | nuts3_18 | soss90_90 | ppsup_no_spouse | ppsup_no_spouse |
| soss90_90 | soss90_80 | paalammi_5 | ppsup_no_spouse | ppsup_no_spouse | paalammi_5 | paalammi_3 |
| ppsup_no_spouse | soss90_90 | paalammi_8 | paalammi_3 | paalammi_3 | paalammi_9 | paalammi_9 |
| paalammi_9 | ppsup_no_spouse | paalammi_9 | paalammi_5 | paalammi_8 | paalammi_unknown | paalammi_unknown |
| paalammi_unknown | paalammi_3 | paalammi_unknown | paalammi_9 | paalammi_9 | NaN | NaN |

**Figure 5.6:** Features dropped by prediction problem in the first feature elimination stage for some of the prediction problems.

We did another feature elimination step using scikit-learn's Recursive Feature Elimination with Cross-Validation (RFECV) module using randomized search cross-validation with XGBoost regressor fitted to the training set. RFECV evaluates the importances of features by recursively removing a specified number or fraction of the features at each iteration [13]. Again, the exact features dropped vary depending on, e.g., the configuration of RFECV set before the run, besides also varying from run to run due to stochasticity in XGBoost, the random search, and the cross-validation. Figure 5.7 shows the features (categorical ones one-hot encoded) dropped with a particular configuration of RFECV for some of the prediction problems. As a result of both of these feature elimination steps, the numbers of columns we were left with for each prediction problem are shown in Table 5.2.

| food_expenses | freetime_expenses | housing_expenses_wo_... | other_regular_expenses_... | communication_IT_and_media_... | utilities_and_energy | clothing_expenses | childcare_expenses |
|---|---|---|---|---|---|---|---|
| paalammi_7 | nuts3_2 | murk | yli75v | skidi | nuts3_4 | psoss90_80 | nuts3_4 |
| Talotyyp_2 | paalammi_7 | nuts3_12 | nuts3_4 | paalammi_7 | soss90_50 | psoss90_30 | ppsup_2 |
| psoss90_80 | nuts3_18 | psoss90_20 | nuts3_16 | psoss90_70 | soss90_90 | nuts3_9 | nuts3_15 |
| paalammi_6 | aspintaa | NaN | NaN | nuts3_18 | psoss90_80 | NaN | murk |
| psoss90_50 | nuts3_4 | NaN | NaN | nuts3_14 | yli75v | NaN | soss90_40 |
| nuts3_8 | peru | NaN | NaN | aspintaa | psoss90_30 | NaN | Talotyyp_3 |

**Figure 5.7:** Features dropped by prediction problem in the second feature elimination stage for some of the prediction problems.

## 5.2 Algorithms and Techniques

The learning models we used for modeling the dataset can be divided into three groups. Firstly, there are two linear regression or hyperplane representation-based models, "ordinary" linear regression and Bayesian ridge regression, which we refer to as the interpretable solutions 1.1 and 1.2, respectively. Secondly, there are three decision tree

**Table 5.2:** Number of columns left by prediction problem after both feature elimination stages for a dataframe with 1) categorical features not one-hot encoded, and 2) categorical features one-hot encoded.

| Columns left after feature elimination | Non-one-hot encoded | One-hot encoded |
|---|---|---|
| total | 24 | 50 |
| food | 20 | 21 |
| freetime | 20 | 36 |
| housing wo... | 22 | 50 |
| car | 23 | 53 |
| health and hygiene | 24 | 54 |
| other regular incl... | 27 | 49 |
| communication, IT... | 19 | 20 |
| utilities and energy | 23 | 46 |
| clothing | 24 | 50 |
| childcare | 23 | 35 |
| insurance... | 23 | 42 |
| local public transp. | 20 | 23 |

models with differing approaches, which we refer to as interpretable solutions 2.1, 2.2, and 2.3. Thirdly, there are various complex models, most of which are effectively not interpretable, and which we therefore refer to as the black box models.

The approach was to run a machine learning workflow as a loop, where in every iteration we have selected one black box model to learn from the data at a time, and in the process, also learn all five interpretable solutions, training which takes usually only a small fraction of the time compared with the black box models. The idea is that the interpretable models provide us with a reference on what kind of predictive power we can achieve while retaining a decent level of interpretability as opposed to the black box models, which are meant to give us a reference on what the highest possible predictive power we can achieve might be. Comparing these two we can get a sense of the trade-off between interpretability and predictive power in these prediction problems.

Furthermore, in the same evaluation step with these models, we also evaluate the benchmarks described earlier, the purpose of which is to act as baselines for predictive power. The predictive performance of any solution proposal must exceed these baselines with a clear enough margin, or otherwise we can conclude that either the data pre-processing and modeling work we have done is not good enough or the prediction

problem is too difficult for the models we have chosen given the data we have available.

### 5.2.1 Black Box Models

The black box models, or model families, we included in our modeling pipeline were:

1) XGBRegressor (XGBoost), a gradient boosted trees ensemble model through scikit-learn wrapper API,
2) CatBoostRegressor (CatBoost), a gradient boosted trees ensemble model,
3) XGBRFRegressor (XGBoostRF), a random forest implementation using the XGBoost "engine" through scikit-learn wrapper API,
4) RandomForestRegressor (Random Forest), a scikit-learn implementation of the random forest model,
5) GaussianProcessRegressor (Gaussian Processes), a scikit-learn implementation of the stochastic process-based Bayesian non-parametric model known as Gaussian processes,
6) ElasticNetRegressor (Elastic-net), a scikit-learn implementation of the Elastic-net, a L1 and L2 regularized linear regression model,
7) SVR, a scikit-learn implementation of the support vector regressor.

Our preconception was that XGBoost and CatBoost are likely to produce the best predictive power, as they are often dubbed state-of-the-art for prediction problems with tabular data, although they are also known to take a lot of effort with tuning, namely hyperparameter search, to reach their full potential [22]. In contrast, Random Forest, although already a rather dated model from the first decade of the millennium, has a reputation of providing good performance out-of-the-box or with only little tuning and tweaking [7]. XGBoostRF is a random forest implementation using the XGBoost engine and thus inheriting the hyperparameters of XGBoost, although fixing some of them to specific values or interpreting them in a different way from the usual to make the model behave like a random forest instead of an ensemble of boosted trees. We had no preconception as to how XGBoostRF would fare in comparison with Random Forest, but speculated that, e.g., the ability of the XGBoost engine to natively take in and handle categorical features in what is said to be a more performant way would provide benefits over Random Forest.

Gaussian Processes was included as one of the black box models largely due to it being a Bayesian model that produces not only point estimates, but also the uncertainty for its predictions, i.e., effectively an entire predictive distribution. A cost that comes along with using Gaussian Processes is that finding the best model requires a different approach from the rest, because Gaussian Processes' ability to converge to a solution depends mostly on the choice of *kernel*, or the covariance function, and the

kernel's parameters, which cannot really be treated as hyperparameters in a hyperparameter searching scheme due to computational heaviness of fitting just one choice of kernel. Choosing the kernel and its parameters, especially for a high-dimensional feature space, is a bit akin to a black art, where the modeler should have insight about the mathematical nature of their dataset and the underlying DGP, and what kind of a kernel would suit them best.

Lastly, Elastic-net and SVR, more classical models, were also included for curiosity. Our thinking was that Elastic-net's performance reveals whether the L1 and L2 regularization adds any benefit compared with the ordinary linear regression in these prediction problems, while SVR's performance reveals if an approach mixing instance representation [18] and hyperplane representation would provide good results.

Next, let us have a more detailed look into these models and their learning algorithms.

#### 5.2.1.1 XGBoost and CatBoost

XGBoost is an ensemble model using gradient boosted trees. Boosting is an ensemble method that makes use of the concept of *weak learner* – which can be any base learner but is expressly the decision tree in the case of boosted trees – that produces predictions that are at least slightly better than just random guessing. Boosting introduces more estimators based on the base learner in an iterative fashion, focusing on improving predictions that previous estimators got wrong. The final prediction is an ensemble of all the estimators, the number of which may very well be in the thousands [12, 7]. As mentioned, XGBoost can take in and handle categorical features natively, which is said to be done in a smart way that can result in better efficiency compared with feeding one-hot encoded features to the model [59].

XGBoost has dozens of hyperparameters that determine the way the algorithm works. One hyperparameter is 'booster', which determines the type of the base learner. It takes the value 'gblinear' for setting the base learner to be a L1 and L2 regularized linear model, making the model a boosted linear model; 'gbtree' for making it the default boosted trees model; and 'dart', which stands for "dropouts meet multiple additive regression trees", for making it use a more novel approach introducing regularization in boosted trees in the form of dropouts, a concept familiar from deep neural networks. Any base learner can be boosted, but typically the benefit is significant only with models that partition the dataset in some way, and thus decision trees, i.e. 'gbtree' or 'dart' set as the value of 'booster', are a typical choice for the base learner instead of linear models, or 'gblinear'. The use of 'dart' is reported to yield better results than 'gbtree' in some situations where the dropout functionality reduces overfitting [61].

XGBoost is considered a state-of-the-art model for tabular, or structured, data. It has earned its reputation, e.g., by being the model used in a number of winning solutions of machine learning prediction competitions.

CatBoost is another state-of-the-art boosted trees ensemble model known for faring well in prediction competitions. Its name comes from the word category, hinting that CatBoost is meant to be particularly good with using categorical features. Needless to say, CatBoost supports taking in and handling categorical features natively.

Both XGBoost and CatBoost support *early stopping*, a functionality that keeps track of the model performance against an early stopping validation set during the iterative boosting process. The idea is that early stopping stops the training once the model performance against the validation set has not improved more than a given tolerance limit for a specified number of boosting iterations. The main benefit of this is avoiding overfitting, since early stopping does not allow the training to proceed with an improving model performance against the training set while the generalization performance against the validation set is no longer improving. In fact, the generalization performance is bound to start degrading after some number of iterations when the maximum number of estimators has been set to a very high number as the model effectively starts to memorize the training set. A secondary benefit of early stopping is saving computation time as early stopping stops the training when no significant benefit in generalization performance is achieved anymore.

It is noteworthy that tree-based models, including boosted trees, are not well-suited for problems where predictions should extrapolate, i.e., generalize beyond the edges of our feature space. This is due to the nature of doing regression with trees: the prediction is an aggregate of the values of the examples encompassed by a leaf node or an ensemble of leaf nodes from multiple trees. Naturally, aggregates do not extrapolate beyond the elements from which the aggregate was computed.

#### 5.2.1.2 Random Forest and XGBoostRF

Random forest implementations, such as Random Forest and XGBoostRF, are based on the simplest ensemble method, *bagging*, which stands for *bootstrap aggregation*, combined with the *random subspace method*. What this roughly means is that random forests "grow" a *decision forest* consisting of a number of decision trees trained on random samples bootstrapped from the training set and with a random subset of the features selected at each candidate split (hence "random" in random forest). Effectively, the decision trees in the forest get trained on different subsets of the training set, reducing correlation between them, which consequently makes them provide better ensembles, such as a majority vote or an average, of their individual predictions [29,

14, 7]. As mentioned, Random Forest is considered an easy-to-use model that can often yield good results out-of-the-box or with little tuning. Although perhaps not the likely winner of prediction competitions these days anymore, with this ease-of-use, Random Forest is still relevant, at least for providing a rough idea of the achievable level of performance in a prediction problem.

As mentioned, XGBoostRF uses the XGBoost engine with some of the hyperparameters fixed to particular values and interpreted in ways that make it work like a random forest instead of an ensemble of boosted trees. For example, the 'n_estimators' hyperparameter gets interpreted as the size of the forest to be trained instead of as the number of boosting rounds [60].

#### 5.2.1.3 Gaussian Processes

Gaussian Processes is a non-parametric, Bayesian, and kernel-based supervised learning model using Gaussian processes, stochastic processes defined by random variables with a joint Gaussian distribution. Being a Bayesian approach, it provides a probabilistic approach to prediction; for instance, it not only gives point predictions but also the associated uncertainty. But it also adds complexity to the modeling by, e.g., requiring a prior distribution encoding the modeler's a priori knowledge to be passed to it. The prior is given to the learning algorithm by a) specifying the prior mean either as a constant, zero, or the mean of the training data, and b) passing the covariance function of the prior, i.e. the kernel, to the learning algorithm [48, 53].

Gaussian Processes uses the so-called *kernel trick* to map the feature space to a higher dimension for making the model more expressive to fit to the training data better. A kernel can be thought of as an *embedding* from the original feature space to another one, i.e., it is used to map our original data to a newer and more complex feature space, which is explicitly defined by the choice of the kernel. The kernel can also be thought of as encoding our assumptions about the target function in that it "defines the 'similarity' of two datapoints" under the assumption that "similar datapoints should have similar target values" [48, 53]. Apart from the Gaussian Processes learning algorithm having a few different hyperparameters itself, most kernels have their own set of hyperparameters that are optimized when fitting the Gaussian Processes. As mentioned, doing a hyperparameter search on the algorithm's hyperparameters was not deemed feasible due to the long computation time that the fitting of one kernel often takes.

As was also mentioned earlier, choosing a fitting kernel is anything but trivial, preferably requiring in-depth understanding of the mathematical nature of the DGP underlying the dataset. When the dimensionality of the dataset is higher than just a few

dimensions, such as with our dataset of dozens of features, gaining such understanding becomes difficult or outright impossible. Moreover, the intricacies of kernels and the components that constitute them are far outside this author's expertize. Therefore, our approach to choosing a kernel was simply to look up some kernels used in various works found online and try them out.

#### 5.2.1.4 Elastic-Net

Elastic-net is a linear regression model with regularization using *L1* and *L2-norm* penalty terms on the estimates of the regression coefficients, or weights, of the regression model. These penalty or regularization terms introduce additional constraints to the minimization problem of the loss function, which the optimization algorithm must respect, in contrast to the regular OLS estimator used with ordinary linear regression, which simply minimizes for the error without such constraints. Since L1 and L2 regularization are central techniques used heavily also in many other models, such as tree ensembles, let us discuss them here in a bit more detail.

The intuition behind L1 penalty, or *lasso* (stands for *least absolute shrinkage and selection operator*), estimation of the regression coefficients is that it regularizes for sparsity of the model, or its coefficients, and it effectively does it by performing both feature selection and regularization. This often results in both better prediction accuracy and interpretability [27].

L1 penalty introduces a "force" that subtracts some constant (hence "shrinkage") from a given weight at every optimization step. This is a force that drives weights quickly to zero, which can be of a great benefit if we have a lot of redundant features in our data, because the weight zero given to a coefficient of a feature in a regression model effectively removes the feature from the model. For such a linear regression modeling problem where the true coefficients are sparse, L1 penalty essentially performs dimensionality reduction at training time by providing a sparse solution or learning a sparse model [27, 29].

The intuition behind L2 penalty, or *ridge*, estimation of the regression coefficients is that it regularizes for simplicity of the model, or its coefficients, and it effectively does it by viewing model complexity as a function of the weights of all the features in the model and performing regularization accordingly. The regularization term of L2 penalty – the sum of squares of all the feature weights – can be interpreted as a measure of model complexity to which weights close to zero have little and outliers a large effect. Thus, L2 regularization views a feature with a large weight more complex than a feature with a small one [26].

The term *inflation of* $R^2$ refers to the well-known phenomenon that adding fea-

tures to a linear regression model will at least weakly increase its training $R^2$, driving the model towards overfitting the training set. L2 addresses this problem by introducing a penalty on the sum of the weights, which favors distributing most of the total weight to simple features that explain most of the variance of the target variable, and little to zero weight to complex features that explain only a little. The means through which L2 penalty affects the weights is, similarly to L1 penalty, by introducing a force that subtracts something ("shrinkage") from a given weight at every optimization step, but unlike with L1, with L2 this something is not a constant but proportional to the weight [29].

L2 is a regularization method for ill-posed problems, particularly used in mitigating problems brought about by multicollinearity in high-dimensional feature spaces and combatting overfitting resulting from the sheer number of features in high-dimensional problems. L2 regularization can be likened to doing PCA dimensionality reduction, but instead of "discretely" dropping dimensions with smallest explained variance like with principal components, L2 regularization can be thought of as reducing dimensionality in a "continuous" way by shrinking least important coefficient estimates gradually towards zero. The use of L2 penalty often provides more statistically efficient estimation, effectively a smaller variance, for a reasonable trade-off with an added bias [28, 26].

Elastic-net itself is said to be particularly useful when there are multiple collinear features and when the preconception is that the learned model should be more sparse and simpler than the feature space would imply. Through lasso and ridge, Elastic-net provides both regularization for sparsity and regularization for simplicity, but practically the modeler must make a suitable trade-off between lasso and ridge regression for each problem at hand with the choice of hyperparameter values [51].

#### 5.2.1.5 Support Vector Regressor

Support vector machines (SVM) are a well-known class of learning algorithms, but mostly known only in the context of classification problems, in which case they are often specifically called support vector classifiers (SVC). In contrast, support vector regressors (SVR), the variety of SVMs for regression problems, are not as widely known a tool.

Like Gaussian Processes, also SVMs famously use the kernel trick to map the feature space to a higher dimension where the data is possible to be linearly separated with a hyperplane which can look highly non-linear observed from lower dimensions (although also linear kernels can be used with SVMs). The kernel used in the kernel trick is a set of mathematical functions for transforming the input data for mapping it to the higher dimension. In this higher dimension, SVR attempts to find the best fit,

which is a hyperplane that contains the most data points. It then draws two boundary lines around the hyperplane at a distance determined by the *maximum error* $\varepsilon$, defining a margin, which we can tune to gain a desired accuracy for our predictions [52].

As a special feature of SVR, its objective function is not to minimize the error between the fitted values of the hyperplane and the true values, but to find the best fit within a given threshold value, i.e., the margin or the distance between the hyperplane and the boundary lines. In fact, the model depends only on a subset of the training data, because it does not care about training points whose prediction lies closer to their target than the threshold value. In short, SVR lets us define how much error is acceptable in the predictions (the margin) and with that in mind, find the best fit (regression line or hyperplane) to the data.

### 5.2.2 Interpretable Models

As opposed to the black box models, interpretable models provide easily understandable information about the way a particular prediction was formed. In our selection of interpretable models, this information is either in the form of decision rules or regression coefficients.

#### 5.2.2.1 Linear Regression Model

The first interpretable model is the ordinary linear regression model without regularization. The interpretability of linear regression models comes from the fact that the estimates of regression coefficients, when in the original scale, tell you how many units an addition of one unit in a given feature will increase or decrease the value of the target variable. Secondly, the estimates of coefficients, when normalized, tell you about the relative importance of each feature [10].

#### 5.2.2.2 Bayesian Ridge Regression

The ordinary ridge regression imposes an L2-norm penalty on the sizes of the estimates of regression coefficients, because the sizes may be misleading if having become inflated by collinearity in the dataset. If inflated, the interpretation of the estimates of coefficients as presented above is not truthful. Ridge regression has one hyperparameter, the *complexity parameter*, or *shrinkage amount parameter*, alpha.

As the name implies, Bayesian ridge regression does ridge regression in a Bayesian fashion. It is a Bayesian regression technique that treats the regularization hyperparameter alpha as a parameter in the estimation procedure. Therefore, alpha gets estimated from or tuned to the data. As any Bayesian model, it regards its parameters, i.e. alpha, probabilistic, and introduces a prior distribution over it, but with Bayesian

ridge the prior is often chosen to be non-informative. The benefits of Bayesian ridge are that it is more robust to "ill-posed problems" than ordinary ridge or linear regression models and that as a Bayesian method it gives not only point predictions but also the prediction uncertainty [51].

#### 5.2.2.3 Pre-Pruned Decision Tree

Decision trees are known to have a tendency to overfit to data, as allowing their size to be arbitrarily large leads them towards learning the specific rules of each and every example in the training set. Pruning is a way to limit the size of the tree so that the rules it forms would remain at a level that generalizes beyond the training set [10]. Pruning can be divided into pre-pruning and post-pruning techniques.

Pre-pruning is controlled with hyperparameters that limit the tree ex ante, or as constraints to its growth. The hyperparameters used for this are predominantly 'max_leaf_nodes' for setting the maximum number of leaves, and thus also the maximum number of decision paths or rules, and 'max_depth' for setting the maximum depth of the tree, i.e., the number of nodes in the tree or splits of the feature space, and thus also the maximum number of conditions in each decision rule.

We wanted to fix our pre-pruned decision trees to have exactly twenty decision rules that each have at most four conditions.

#### 5.2.2.4 Post-Pruned Decision Tree

Post-pruning techniques are based on limiting the size of the tree ex post, or, by letting the tree first grow to its full size, and then making decisions about which nodes of the tree should be kept and which should constitute a new leaf node, i.e., a termination of a decision path. In contrast to pre-pruning, post-pruning can result in "unevenly pruned" trees, as the pruning decisions are made node by node [33].

Post-pruning is done using some measure of information gain of the nodes of the tree. A popular method is *cost complexity pruning*, which one controls with the cost complexity hyperparameter. The greater the value of the parameter, the more nodes of the tree get pruned. Post-pruning is often the way to get the best predictive performance out of a decision tree against a holdout test set, i.e., generalization performance. What is essential in it is finding the best value for the cost complexity hyperparameter [33].

#### 5.2.2.5 Decision Tree Regressor to K-means Clusters' Means

The third interpretable decision tree solution extends on benchmark 4, which clustered the training set based on the features, assigned unseen observations to these clusters

using the learned k-means estimator, and gave the means of the expenditures in the respective cluster as the prediction for each unseen observation.

In this solution we make further use of this clustered training set by learning a decision tree from it to predict the clusters, or rather the means of the expenditures in these clusters. The decision tree is then given an assortment of the usual hyperparameters, including both pre-pruning and post-pruning ones, specified as following certain random distributions, and then the model is searched from this search space using scikit-learn's RandomizedSearchCV. This solution is a somewhat different approach to partitioning the training set compared with the aforementioned two decision tree approaches, because it starts with partitioning using the unsupervised clustering algorithm (after dimensionality reduction), whereas decision trees do their partitioning as a "side product" of the supervised learning process. We had no preconception as to the pros and cons of or expectations for the differences in performance between the two approaches. Nor did we find literature comparing or paralleling these two approaches to partitioning a dataset into groups for making predictions from the group means. However, our intuition was that there is a certain kind of dualism between the two, i.e., that in some edge cases the two will result in equivalent partitions, but in practice there will be differences.

We also attempted to cluster the dataset based on the 12 target variables, the expenditure categories, instead of the features, and then train a decision tree to try and learn these clusters from the features. However, the results were poor: the decision tree learned the clusters with a very low accuracy, and giving the means of these clusters as the predictions of expenditures for unseen households gave such huge prediction errors that we decided to drop this approach.

### 5.2.3 Hyperparameter Search

To learn the best model for a given problem, apart from model training or fitting where we minimize the loss function by letting our learning algorithm determine the best model parameters based on the training data, we must also carry out a meta-optimization process called model selection [47]. A central part of model selection is the search for the best hyperparameter configuration for the task at hand. For this hyperparameter search process, we used the Optuna library. Optuna is a state-of-the-art hyperparameter search library, which uses Bayesian optimization, or Bayesian search, for searching for the optimal hyperparameters in the hyperparameter search space in a smarter way than the classical grid search or random search. Optuna also provides various advanced features, such as integrations with many popular libraries for various functionality. An example of such integrations are the *pruning callback func-*

*tions* one can use with XGBoost and CatBoost to catch unpromising hyperparameter search trials that should be pruned, or aborted, early on, saving a significant amount of computation time in the process.

### 5.2.4 Dimensionality Reduction Using PCA

In our prediction pipeline, we included a flag for opting to do dimensionality reduction with PCA on the training and test sets used by the black box models. In it, we drop principal components contributing to the cumulative explained variance the least, and then fit the black box on the reduced training set and do predictions on the reduced test set. The idea behind this was to mitigate overfitting, the presence of which could be seen manifested by a largish gap between the training error and the test error in tentative results from the black box models. Why it is justified to speculate that dimensionality reduction might reduce overfitting is due to two mechanisms: 1) the curse of dimensionality is alleviated as the feature space becomes less sparse and the fit in the feature space thus generalizes better to unseen data, and 2) the signal-to-noise ratio is improved as features contributing little to nothing to the cumulative explained variance, but possibly coming with noise swamping some of the signal, are dropped.

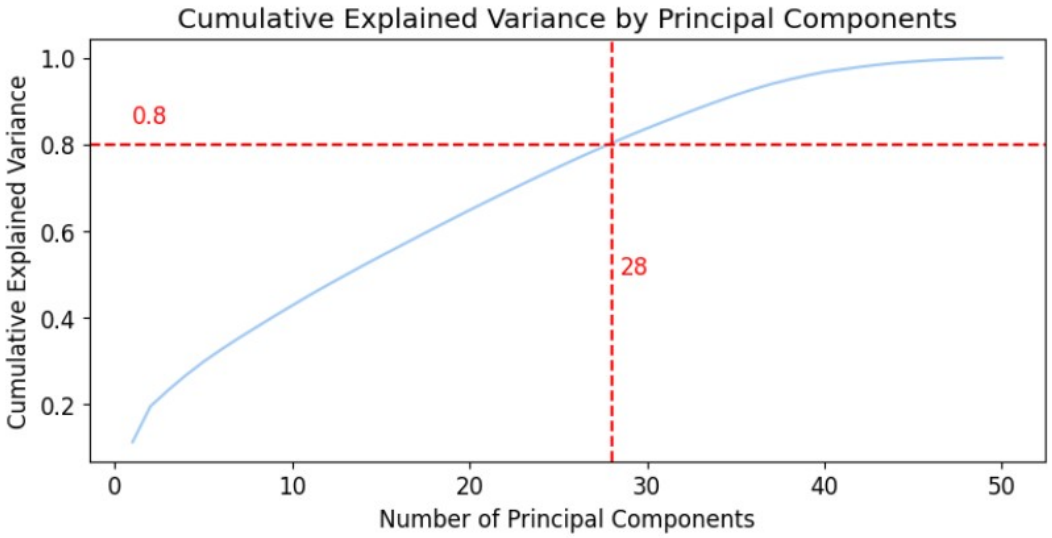


**Figure 5.8:** Cumulative explained variance by the number of principal components, and the number of principal components kept (28) with the cutoff level of 0.8 of the total cumulative explained variance for the clothing expenses prediction problem.

Figure 5.8 illustrates the effect of dimensionality reduction using a cumulative PCA cutoff level of 0.8. We can see that in this case the number of features is almost halved when we settle for 80% of explained variance in our data. Can dropping these principal components in fact prevent overfitting? That can only be found out by training the model without them and testing against the holdout test set, and comparing

against results with all features or principal components included. One case against using PCA dimensionality reduction in our problem is that some of the regularization functionality baked into the learning models we use probably already reap the same benefits. For example, as we already mentioned, what L2 regularization does can be likened to PCA dimensionality reduction. However, our opinion was that giving PCA dimensionality reduction a try among other things does not hurt, mostly just takes some computation time.

## 5.3 Implementation

The implementation of the whole data science or machine learning workflow – the data exploration, data pre-processing, modeling and model evaluation etc. – was done with the "Python data science stack", i.e., the typical toolbox of data handling-related libraries in Python, such as Numpy, Pandas and scikit-learn, and using various open source libraries and modules available on Github and PyPi repositories for more special needs.

One important aspect of the implementation was the use of MLFlow, an open source software for machine learning operations, or MLOps, providing automatic logging of the results from the numerous different ways, or configurations, of running our machine learning pipeline. The different configurations to run the pipeline included: 1) different ways of handling the outliers, 2) different sets of features kept after the feature elimination stage, 3) different choices for the black box model, and 4) with dimensionality reduction using PCA disabled or enabled (and if enabled, with different cutoff levels of cumulative explained variance). Altogether, these factors can make up dozens of different combinations, i.e. configurations for running the pipeline.

The logs from each configuration are placed under their own "folder", an *experiment* in MLFlow terminology, so an experiment with its logs correspond to a particular configuration. Since every run of the pipeline with a given configuration produces dozens of pages of output, it would be a challenge to keep track of the results manually for mutual comparison and interpretation of the results obtained from different models and configurations. This is where MLFlow steps in by conveniently logging the specified outputs from the runs, such as the performance metrics of the models and any plots created, as well as the inputs to the modeling, such as the parameter values and input data.

# 6. Results

As the output from our machine learning pipeline, we obtain an assortment of metrics of predictive performance of the different models and benchmarks, and plots visualizing their predictions against one another as well as the ground truth. Needless to say, the results in each of the 12 prediction problems are different. Also, as mentioned in section Implementation, there are dozens of possible configurations for running the pipeline, each leading to distinct results. On top of that, there are elements of stochasticity in most of the learning algorithms, as well as in their hyperparameter search processes. Therefore, we even get somewhat differing results for a given prediction problem at every run of the same configuration. This all makes it a rather challenging job to sum all the results up into some sort of an executive summary of "actionable insights".

In our best attempt to do this, we make some general observations from the results, such as identify solutions that were systematically top-of-the-line and a rough ranking between the models across different prediction problems. We also compute aggregated performance metrics, which distil the performance from all 12 prediction problems into one figure for each model.

## 6.1 Example Outputs from the Pipeline

To demonstrate what the various outputs from the pipeline look like, let us take an experiment where we had dropped the lower tails using the minimum reference for each expenditure as the threshold and capped the upper tails at mean + 2 standard deviations as an example. The MLFlow logs from the experiment look like as shown in Figure 6.1. These records can be, e.g., sorted and filtered to help analyzing them and distilling them into concise findings, such as one telling us which model had the best predictive power in a given prediction problem and by how much margin to the next best one. Such a crystallization of results into simple findings is a necessity for being able to formulate actionable insights, based on which concrete actions to make changes to, e.g., existing business processes is possible.

In the pipeline of an experiment, we create a number of plots visualizing the predictions from the models. Let us use plots of the problem of predicting total expenses,

XGBRegressor uni-target, distribution with lower tails dropped at minimum reference and upper tails capped

Provide Feedback

metrics.rmse < 1 and params.model = "tree" | Time created | State: Active | Sort: Created | Columns | Expand rows

Table | Chart | Evaluation | Experimental

| | | | | Metrics | | | | | | | | | | Parameters | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Run Name | Created | Duration | best trial R2 | best trial expvar | best trial rmse | test MAE | test R2 | test RMSE | test explained variance | trial R2 | trial rmse | train R2 | booster | learning_rate | n_estimators |
| XGBRegressor total_expenses | 16 hours ago | 58.5s | 0.177 | 0.177 | 11824.8 | 9206.6 | 0.172 | 11590.6 | 0.173 | - | - | - | gbtree | 0.0002215... | 1639 |
| Post-pruned decision tree | 16 hours ago | 2.6min | 0.322 | - | - | - | - | - | - | - | - | - | - | - | - |
| Pre-pruned decision tree | 16 hours ago | 2.5min | 0.347 | - | - | - | - | - | - | - | - | - | - | - | - |
| All predictions of all approaches summed up ... | 19 hours ago | 0.8s | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Bayesian ridge local_public_transportation_ex... | 19 hours ago | 0.6s | - | - | - | 185.7 | 0.284 | 239.8 | 0.291 | - | - | 0.275 | - | - | - |
| Linear regression local_public_transportation_... | 19 hours ago | 188ms | - | - | - | 177.5 | 0.331 | 231.9 | 0.335 | - | - | 0.416 | - | - | - |
| Pre-pruned decision tree | 2 days ago | 1.3d | 0.347 | - | - | - | - | - | - | - | - | - | - | - | - |
| XGBRegressor local_public_transportation_ex... | 19 hours ago | 20.0min | 0.174 | 0.178 | 255.5 | 201.8 | 0.143 | 262.3 | 0.155 | - | - | - | dart | 0.0239285... | 2817 |
| Post-pruned decision tree | 2 days ago | 1.3d | 0.359 | - | - | - | - | - | - | - | - | - | - | - | - |
| Bayesian ridge insurance_expenses_other_tha... | 19 hours ago | 6.6s | - | - | - | 192.1 | 0.201 | 241.1 | 0.202 | - | - | 0.235 | - | - | - |
| Linear regression insurance_expenses_other_t... | 19 hours ago | 1.2s | - | - | - | 193 | 0.182 | 243.8 | 0.183 | - | - | 0.273 | - | - | - |
| XGBRegressor insurance_expenses_other_tha... | 21 hours ago | 1.8h | 0.369 | 0.369 | 251.9 | 194.5 | 0.167 | 246.1 | 0.168 | - | - | - | dart | 0.0899421... | 2329 |
| Bayesian ridge childcare_expenses | 21 hours ago | 1.7s | - | - | - | 203.3 | 0.598 | 447.1 | 0.599 | - | - | 0.622 | - | - | - |
| Linear regression childcare_expenses | 21 hours ago | 496ms | - | - | - | 205.3 | 0.596 | 448.6 | 0.596 | - | - | 0.647 | - | - | - |
| XGBRegressor childcare_expenses | 1 day ago | 5.5h | 0.816 | 0.816 | 353.5 | 166.5 | 0.698 | 387.8 | 0.698 | - | - | - | dart | 0.0520944... | 2735 |
| Trial_4 | 21 hours ago | 37.0s | - | - | - | - | - | - | - | 0 | 682.9 | - | dart | 0.0395786... | 2648 |
| Trial_3 | 23 hours ago | 1.9h | - | - | - | - | - | - | - | 0.4 | 542.2 | - | dart | 0.0207668... | 2885 |
| Trial_2 | 1 day ago | 2.0h | - | - | - | - | - | - | - | 0.4 | 514.3 | - | dart | 0.0073916... | 2954 |
| Trial_1 | 1 day ago | 27.3s | - | - | - | - | - | - | - | 0 | 682.9 | - | dart | 0.0021419... | 659 |

**Figure 6.1:** MLFLow logs of runs with XGBoost as the black box model.

i.e., all the expenditure categories summed up, as examples of the visualizations we obtain from the pipeline. Figure 6.2 shows a plot where we parallel the predictions of what were usually the three strongest models in our experiments by their predictive power: the black box, the linear regression, and the Bayesian ridge. The colored range at the background adds another piece of information to the visualization, the 95% prediction interval of the prediction of Bayesian ridge. The predictions of the decision tree-based interpretable solutions in the same experiment and prediction problem are shown in Figure 6.3, and those of benchmarks 1 and 2 in Figure 6.4.

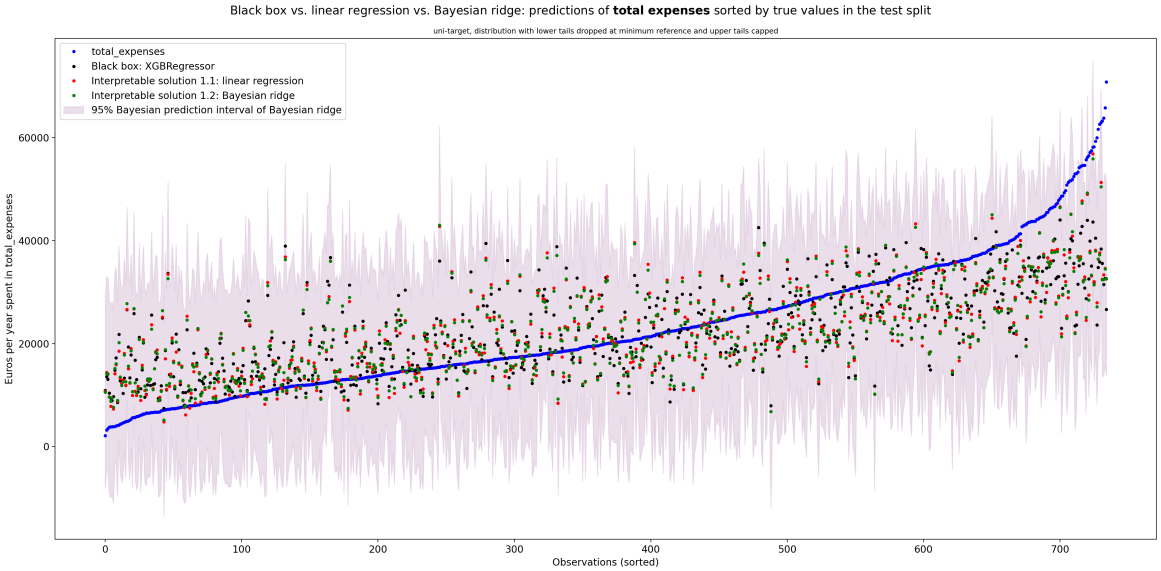


**Figure 6.2:** Predictions of XGBoost, linear regression and Bayesian ridge (together with its prediction uncertainty) plotted against the sorted true values in the test set with total expenses as the target.

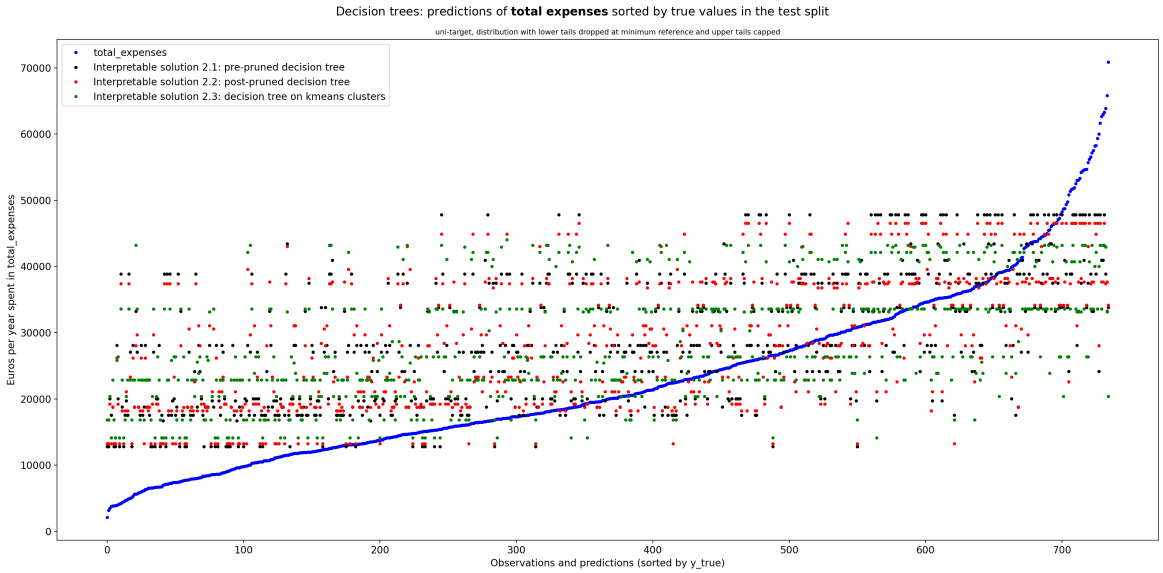


**Figure 6.3:** Predictions of the decision tree-based interpretable solutions plotted against sorted true values in the test set with total expenses as the target with total expenses as the target.

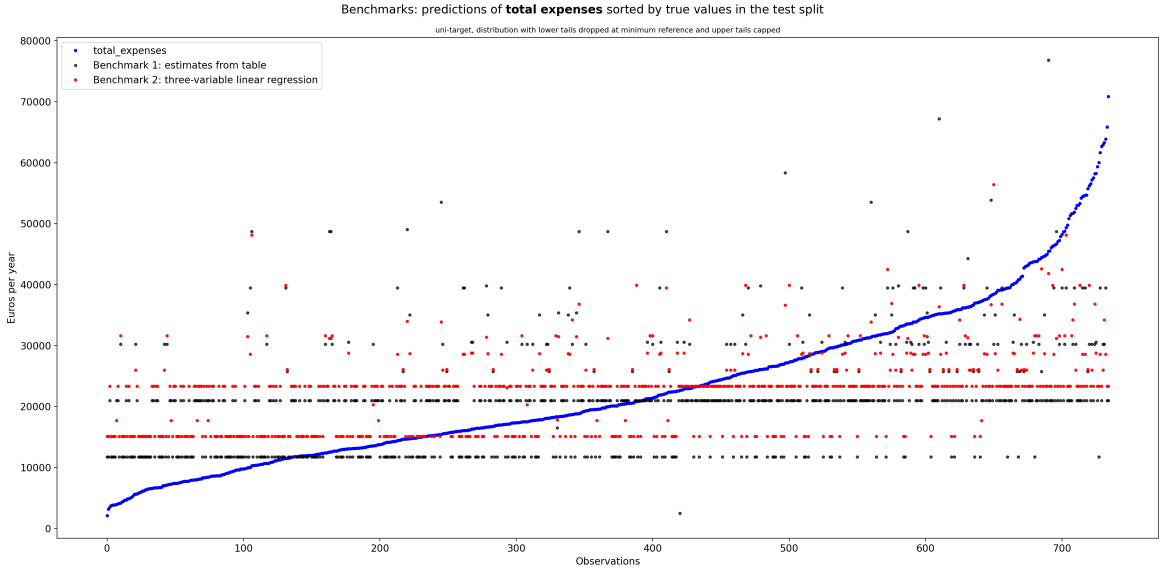


**Figure 6.4:** Predictions of benchmarks 1 and 2 plotted against sorted true values in the test set with total expenses as the target.

The way the performance metrics of the predictions look like when tabulated and ranked by their RMSE is shown in Table 6.1. Something akin to this was a typical-looking result in many of the experiments and prediction problems: the black box model is approximately equally good or at most slightly better than linear regression and Bayesian ridge. The order of the rest of the models and benchmarks varies, but most often benchmark 1 is the last. The results we will showcase in the next section will also include the predictions and performance metrics of benchmarks 3 and 4, which

**Table 6.1:** Performance metrics of an experiment with XGBoost as the black box, and the distribution with lower tails dropped and upper tails capped at mean + 2 standard deviations. Sorted by RMSE.

| | $R^2$ | EV | MAE | RMSE |
|---|---|---|---|---|
| Black box: XGBRegressor | 0.48 | 0.48 | 6890 | 9197 |
| Interpr. solution 1.2. Bayesian ridge | 0.48 | 0.48 | 6949 | 9206 |
| Interpr. solution 1.1: Linear regression | 0.48 | 0.48 | 6961 | 9212 |
| Benchmark 2: 3-predictor linear regression | 0.20 | 0.20 | 8829 | 11427 |
| Interpr. solution 2.2.: Post-pruned decision tree | 0.13 | 0.35 | 9452 | 11848 |
| Interpr. solution 2.1.: Pre-pruned decision tree | 0.12 | 0.33 | 9625 | 11976 |
| Interpr. solution 2.3.: K-means cluster decision tree | 0.03 | 0.22 | 10145 | 12515 |
| Benchmark 1: tabulated estimates | 0.00 | 0.03 | 9530 | 12731 |

are not included here.

Moreover, we create some additional plots showcasing how the black box model behaves. For all the black box models we plot its fit against all the numeric features. This provides a particularly interesting view into the fit in the case of Gaussian Processes as the black box model, since its fit can be highly non-linear, and since it also produces an uncertainty estimate for its predictions, as seen in Figure 6.5.

Furthermore, we also create other additional plots with Gaussian Processes making use of its uncertainty estimate, such as one where its predictions are sorted by the uncertainty estimate, as shown in Figure 6.6. These two plots and the like can give us insight into how the uncertainty varies as a function of a particular feature, or how it varies when we are predicting values at the low-end or high-end of the output space.

Apart from the visualizations, we also tabulate and print various information that provides interpretability and explainability about the fit and behavior of the models. Figure 6.7 shows such information for the linear regression model. On the left-hand side we have, among other information, the coefficient estimates when the model has been fitted to scaled features. These estimates can be interpreted as telling us about the importance of the features with respect to one another, and thus we call the information on the left-hand side "comparison diagnostics". On the right-hand side we have the coefficient estimates when the model has been fitted to unscaled features. These estimates tell us about the effect that adding one unit of a given feature has on the value of the target feature, and thus we call the information on the right-hand side "interpretation diagnostics".

With all the decision tree-based models, we print the importances of the features as measured by the tree-based feature importance metric, and with the plain decision tree models we print the decision rules. The output from fitting and predicting with

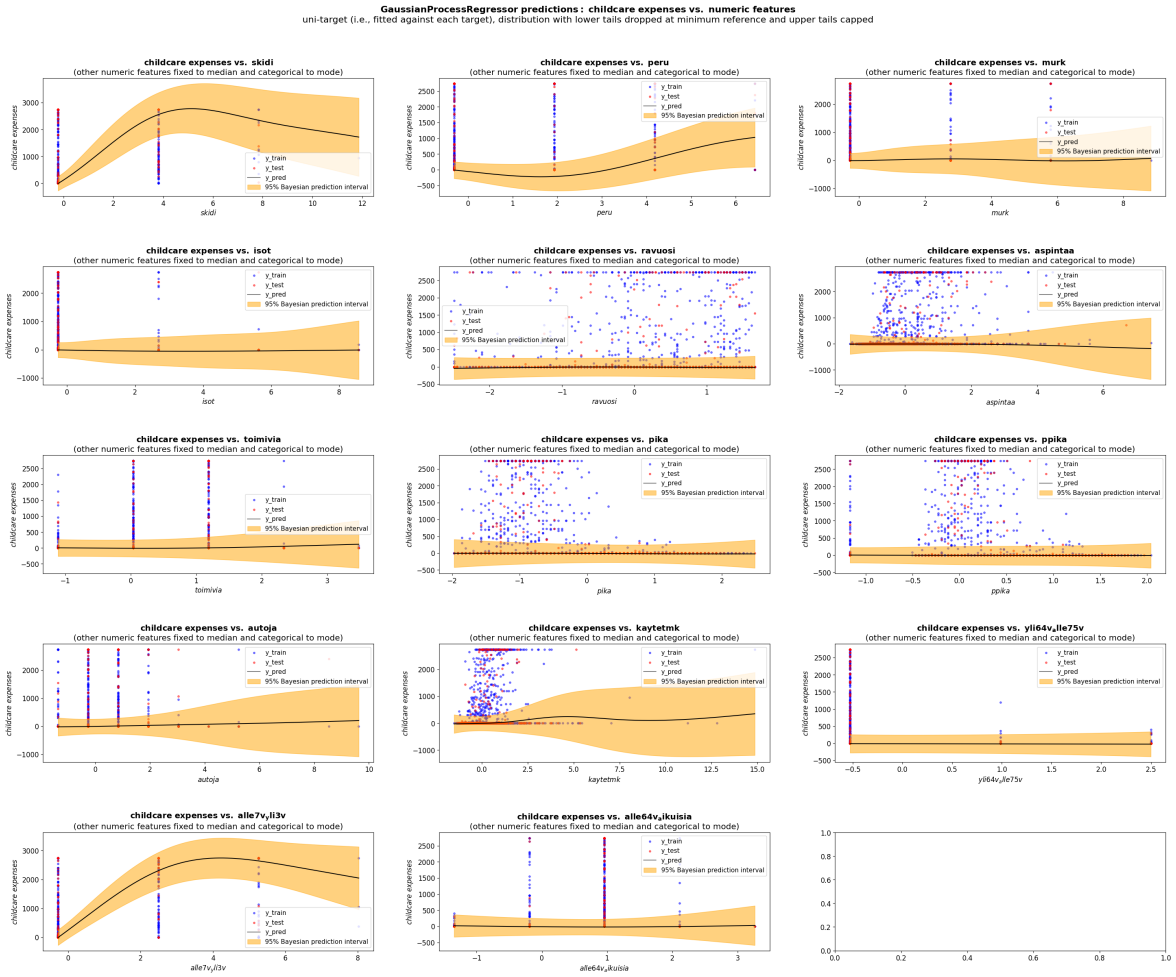


**Figure 6.5:** The fit and 95% prediction interval of Gaussian Processes against each numeric feature (standardized) when other features are kept fixed with childcare expenses as the target.

the pre-pruned decision tree including these two pieces of information is shown in Figure 6.8. Note that the decision rules from the plain decision tree models are the same for all the prediction problems, since the fitting is done in a multi-target fashion to get just one set of rules. We have set the maximum number of leaf nodes, and as such the maximum number of rules, for the pre-pruned decision tree to 20. In contrast, for the post-pruned decision tree there is no such maximum number of rules, and thus the number of rules may vary widely between experiments and even fits, as this is the way cost complexity pruning works.

Lastly, it is worth highlighting the output from benchmarks 1 and 2, since it is of a particular interest to parallel the results of the two. We are interested in not only comparing their performance metrics, but also their interpretable information, namely, the tabulated estimates of benchmark 1 and the regression coefficient estimates of benchmark 2. Figure 6.9 shows the former on the left-hand side, while the latter is shown on the right-hand side. Remember that the tabulated estimates of benchmark

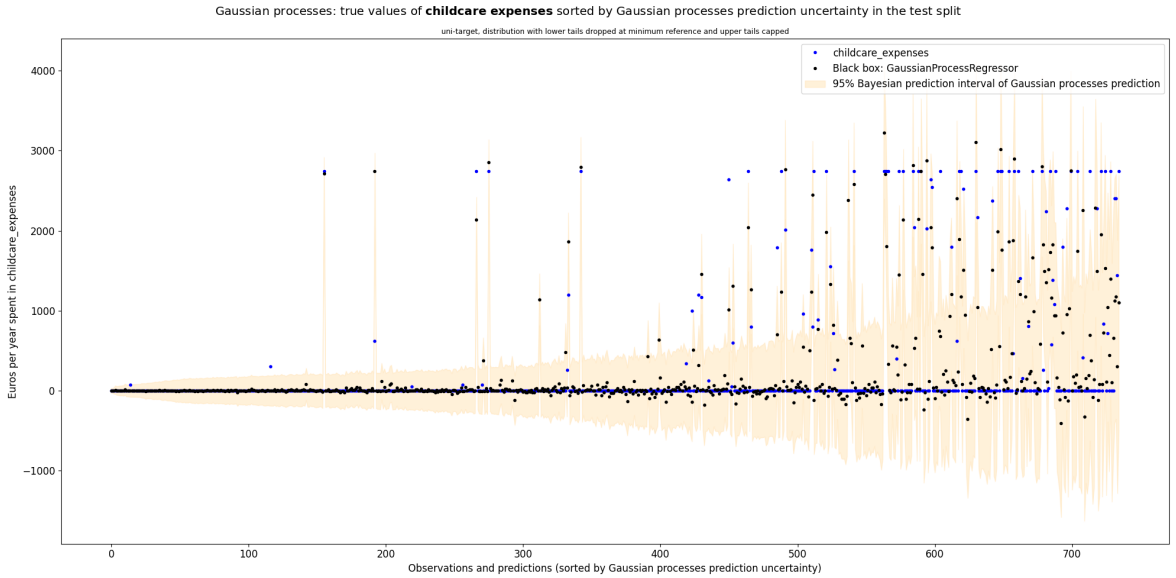


**Figure 6.6:** Predictions of Gaussian Processes sorted by the estimated prediction uncertainty, or the length of the 95% prediction interval, and the true values in the test set with childcare expenses as the target.

Comparison diagnostics: linear regression of *scaled* data against train set

| | |
|---|---|
| Dep. Variable: | total_expenses |
| Model: | OLS |
| No. Observations: | 2938 |
| Covariance Type: | HC0 |
| R-squared: | 0.510 |
| Adj. R-squared: | 0.502 |
| F-statistic: | 54.35 |
| Prob (F-statistic): | 0.00 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| kaytetmk | 67930.0 | 1.32e+04 | 5.142 | 0.000 | 4.2e+04 | 9.38e+04 |
| aspintaa | 17110.0 | 4298.204 | 3.982 | 0.000 | 8690.060 | 2.55e+04 |
| autoja | 15760.0 | 4548.519 | 3.465 | 0.001 | 6847.377 | 2.47e+04 |
| murk | 11510.0 | 2554.561 | 4.506 | 0.000 | 6504.002 | 1.65e+04 |
| toimivia | 10440.0 | 3345.096 | 3.120 | 0.002 | 3881.867 | 1.7e+04 |
| skidi | 10310.0 | 2824.504 | 3.650 | 0.000 | 4773.376 | 1.58e+04 |
| isot | 9837.9468 | 2373.595 | 4.145 | 0.000 | 5185.786 | 1.45e+04 |
| const | 9450.4043 | 1700.730 | 5.557 | 0.000 | 6117.034 | 1.28e+04 |
| peru | 9294.2576 | 1855.532 | 5.009 | 0.000 | 5657.481 | 1.29e+04 |
| pika | -7661.9191 | 1506.161 | -5.087 | 0.000 | -1.06e+04 | -4709.898 |
| alle64v_aikuisia | 7233.0389 | 3113.114 | 2.323 | 0.020 | 1131.447 | 1.33e+04 |
| paalammi_5 | -4752.0548 | 1396.724 | -3.402 | 0.001 | -7489.583 | -2014.527 |
| psoss90_30 | 4252.2916 | 1519.821 | 2.798 | 0.005 | 1273.496 | 7231.087 |
| paak_2 | -4123.3684 | 877.035 | -4.701 | 0.000 | -5842.326 | -2404.411 |
| soss90_70 | 3801.7379 | 961.269 | 3.955 | 0.000 | 1917.685 | 5685.791 |
| alle7v_yli3v | 3742.0353 | 2164.492 | 1.729 | 0.084 | -500.291 | 7984.362 |
| ppsup_2 | 3298.903 | 811.647 | 4.064 | 0.000 | 1708.104 | 4889.702 |
| soss90_20 | 3290.8737 | 1317.706 | 2.497 | 0.013 | 708.217 | 5873.531 |
| soss90_30 | 3278.3406 | 1073.896 | 3.053 | 0.002 | 1173.544 | 5383.137 |
| yli64v_alle75v | 2821.4163 | 1884.947 | 1.497 | 0.134 | -873.012 | 6515.845 |
| nuts3_11 | -2637.7192 | 941.246 | -2.802 | 0.005 | -4482.527 | -792.911 |
| Talotyyp_3 | 2530.777 | 891.631 | 2.838 | 0.005 | 783.212 | 4278.342 |
| soss90_40 | 2453.0706 | 980.705 | 2.501 | 0.012 | 530.925 | 4375.216 |
| nuts3_4 | -2325.1382 | 890.698 | -2.610 | 0.009 | -4070.875 | -579.402 |
| ravuosi | 2300.2251 | 984.372 | 2.337 | 0.019 | 370.892 | 4229.558 |
| paalammi_8 | -2270.3794 | 2035.973 | -1.115 | 0.265 | -6260.814 | 1720.055 |
| paalammi_7 | -2258.9944 | 1251.192 | -1.805 | 0.071 | -4711.285 | 193.296 |
| nuts3_15 | -2237.1621 | 1213.506 | -1.844 | 0.065 | -4615.590 | 141.266 |
| Talotyyp_4 | 2164.3941 | 952.428 | 2.273 | 0.023 | 297.669 | 4031.119 |
| nuts3_21 | -2035.9297 | 1195.042 | -1.704 | 0.088 | -4378.168 | 306.309 |
| paalammi_6 | -1706.0886 | 972.950 | -1.754 | 0.080 | -3613.035 | 200.858 |
| vapaa_ajan_tai_kakkosasunto_1 | 1671.088 | 564.840 | 2.959 | 0.003 | 564.022 | 2778.154 |

Interpretation diagnostics: linear regression of *unscaled* data against train set

| | |
|---|---|
| Dep. Variable: | total_expenses |
| Model: | OLS |
| No. Observations: | 2938 |
| Covariance Type: | HC0 |
| R-squared: | 0.510 |
| Adj. R-squared: | 0.502 |
| F-statistic: | 54.35 |
| Prob (F-statistic): | 0.00 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| alle64v_aikuisia | 1808.2597 | 778.279 | 2.323 | 0.020 | 282.862 | 3333.658 |
| alle7v_yli3v | 1247.3451 | 721.497 | 1.729 | 0.084 | -166.764 | 2661.454 |
| aspintaa | 34.9273 | 8.772 | 3.982 | 0.000 | 17.735 | 52.120 |
| autoja | 1576.231 | 454.852 | 3.465 | 0.001 | 684.738 | 2467.724 |
| const | -34510.0 | 2e+04 | -1.723 | 0.085 | -7.38e+04 | 4744.811 |
| isot | 3279.3156 | 791.198 | 4.145 | 0.000 | 1728.595 | 4830.036 |
| kaytetmk | 0.1251 | 0.024 | 5.142 | 0.000 | 0.077 | 0.173 |
| murk | 3836.95 | 851.520 | 4.506 | 0.000 | 2168.001 | 5505.899 |
| nuts3_11 | -2637.7192 | 941.246 | -2.802 | 0.005 | -4482.527 | -792.911 |
| nuts3_12 | -1323.672 | 905.057 | -1.463 | 0.144 | -3097.551 | 450.207 |
| nuts3_13 | -962.1808 | 890.327 | -1.081 | 0.280 | -2707.189 | 782.827 |
| nuts3_14 | -1175.0462 | 1104.880 | -1.064 | 0.288 | -3340.572 | 990.479 |
| nuts3_15 | -2237.1621 | 1213.506 | -1.844 | 0.065 | -4615.590 | 141.266 |
| nuts3_17 | 238.7485 | 888.191 | 0.269 | 0.788 | -1502.073 | 1979.570 |
| nuts3_2 | 92.116 | 787.026 | 0.117 | 0.907 | -1450.427 | 1634.658 |
| nuts3_21 | -2035.9297 | 1195.042 | -1.704 | 0.088 | -4378.168 | 306.309 |
| nuts3_4 | -2325.1382 | 890.698 | -2.610 | 0.009 | -4070.875 | -579.402 |
| nuts3_5 | -1218.7359 | 1193.297 | -1.021 | 0.307 | -3557.556 | 1120.084 |
| nuts3_6 | 166.7639 | 800.972 | 0.208 | 0.835 | -1403.113 | 1736.641 |
| nuts3_7 | 1305.9042 | 1088.211 | 1.200 | 0.230 | -826.951 | 3438.759 |
| paak_2 | -4123.3684 | 877.035 | -4.701 | 0.000 | -5842.326 | -2404.411 |
| paalammi_2 | -1000.4554 | 731.658 | -1.367 | 0.172 | -2434.478 | 433.568 |
| paalammi_4 | 215.8704 | 1033.668 | 0.209 | 0.835 | -1810.081 | 2241.822 |
| paalammi_5 | -4752.0548 | 1396.724 | -3.402 | 0.001 | -7489.583 | -2014.527 |
| paalammi_6 | -1706.0886 | 972.950 | -1.754 | 0.080 | -3613.035 | 200.858 |
| paalammi_7 | -2258.9944 | 1251.192 | -1.805 | 0.071 | -4711.285 | 193.296 |
| paalammi_8 | -2270.3794 | 2035.973 | -1.115 | 0.265 | -6260.814 | 1720.055 |
| peru | 3098.0859 | 618.511 | 5.009 | 0.000 | 1885.827 | 4310.345 |
| pika | -99.5054 | 19.561 | -5.087 | 0.000 | -137.843 | -61.168 |
| ppika | 11.8541 | 19.825 | 0.598 | 0.550 | -27.003 | 50.711 |
| ppsup_2 | 3298.903 | 811.647 | 4.064 | 0.000 | 1708.104 | 4889.702 |
| psoss90_30 | 4252.2916 | 1519.821 | 2.798 | 0.005 | 1273.496 | 7231.087 |

**Figure 6.7:** A report of the fit of a linear regression model with scaled features (left) for comparing coefficient estimates against each other and unscaled features (right) to facilitate interpretation.

1 can also be interpreted as regression coefficient estimates, although not obtained by fitting a linear regression model, and thus dubbed "regression coefficients" with quotation marks in the printout. Looking at these printouts like in Figure 6.9 from each prediction problem, we can compare how consistent these coefficient estimates are between the ones derived from the aggregate datasets used with benchmark 1 and the ones derived from the microdataset used with benchmark 2. In Figure 6.9, we can

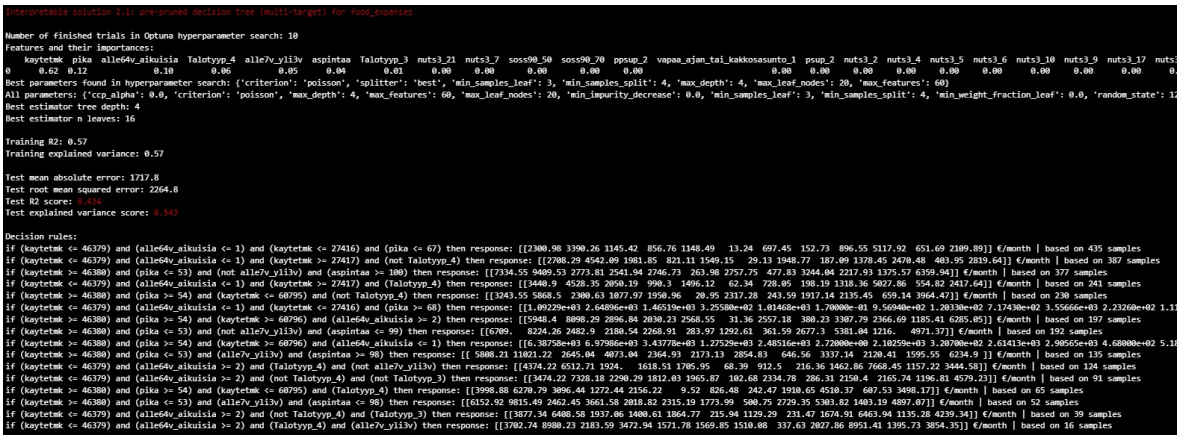


**Figure 6.8:** Output from fitting and predicting with the pre-pruned decision tree.

also see the 95% confidence interval, i.e., the interval estimate for the coefficients. The interval can be used for determining if the difference between the coefficient estimates of benchmark 1 and benchmark 2 is statistically significant. Looking at the figure, we notice that the coefficient estimates of benchmark 1 are outside the confidence intervals of benchmark 2, meaning that assessed at the commonly-used level of two standard deviations from the mean the difference is statistically significant with each of these coefficients.

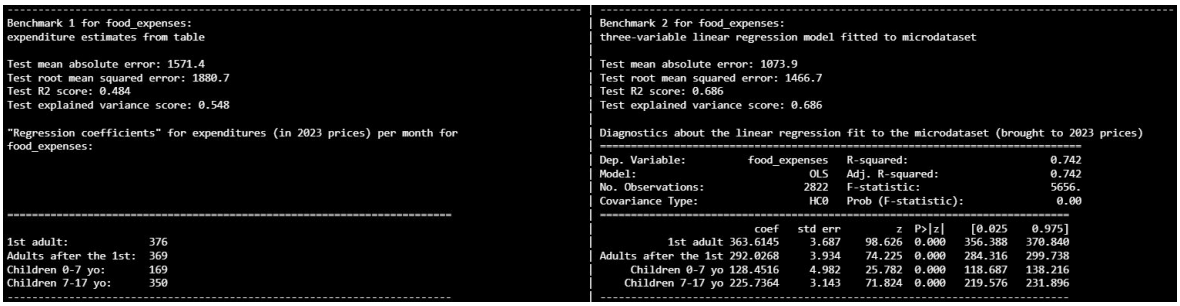


**Figure 6.9:** Contribution of each household member to food expenses: tabulated estimates of benchmark 1 vs. regression coefficient estimates of the three-variable linear regression model fitted to the microdataset of benchmark 2.

## 6.2 Model Evaluation and Comparison

In this section, we sum up the results from the model performance evaluation. As implied earlier, this was not a simple or unambiguous task, since there were the 12 different prediction problems, one for predicting each expenditure category, and in fact also a prediction problem where we predicted the total expenses, the sum of the 12 expenditure categories. Moreover, there were dozens of ways to run the pipeline for those predictions. For example, we could choose to deal with outliers in nine different

ways, because we can regard observations as outliers on both the lower tail and the upper tail, and we can handle the outliers in three different ways: leave them be, drop them, or clip them. The minimum references make a good lower threshold for flagging an outlier, but for the upper threshold we must choose a number of standard deviations above the mean, for which we used either two or three. Furthermore, we could opt to run the pipeline with PCA dimensionality reduction, and if so, we must choose the cutoff level for cumulative explained variance, typically at least 0.8. Lastly, we must choose the black box model out of the seven options to run the pipeline with. One can also run the pipeline with several black box models selected, which then runs the prediction loop one-by-one for each of them. Depending on the settings, such as the number of hyperparameter search trials, running the pipeline once with one configuration (i.e. specific settings for handling of outliers, dimensionality reduction on or off) and with each black box model included could take even around the clock with a regular laptop, so the model evaluation stage was rather time consuming.

A run of the pipeline with one configuration, or experiment in MLFlow terminology, for one black box model produces dozens of values of performance metrics and a couple of dozen different plots depicting the predictions and the training. Needless to say, the results vary between the different configurations, but they also vary between different runs within a given configuration due to stochastic elements in the pipeline. To be able to summarize the performance of our pipeline, we chose three configurations, the results of which we focused on. Our choice was: 1) the full distribution with and without PCA, 2) the distribution with lower tails floored and upper tails as is, with and without PCA, and 3) the distribution with lower tails floored and upper tails dropped at mean + 2 standard deviations, with and without PCA. We ran these three configurations with all seven black box models in our selection. We chose these configurations, because it was deemed to be good to 1) get the performance against the full distribution for reference, 2) only floor and not drop the lower tails as dropping would discard too many observations, and 3) also get to see and compare the effect of handling upper outliers, for which dropping was deemed an appropriate choice as the number of outliers was not so large.

Before delving into the results, let us look at our practices and procedures for model evaluation and validation. As mentioned, we split a 20% holdout test set at the very beginning of our workflow, before we even took a cursory look inside the dataset, to prevent the possibility of any modeling decisions to be influenced by idiosyncrasies in the test set. The remaining 80% of observations in our training set we used for hyperparameter search with a 10-fold cross-validation. In the 10-fold cross-validation, the model is trained with 9 folds, i.e., 90% of the observations in the training set, and validated against the fold left out, and this is repeated 9 more times so that each of

the folds is the validation fold in their turn. The final cross-validation score is the mean of all 10 validation scores. This, in turn, is repeated with every hyperparameter configuration, and the highest cross-validation score marks the best configuration to be chosen. The benefit of k-fold cross-validation is that we get to use all the observations in the training set for training the model, while the effect of data leakage from the observations used for validating the hyperparameter configurations to the training of the model is minimized by the averaging of the score over all k iterations. Secondly, the averaging also diminishes the effect of an exceptionally good "chance fit" of a particular hyperparameter configuration to a particular validation split [47].

With learning models where the objective function is optimized through an iterative process of an arbitrary number of iterations, such as the boosted tree models XGBoost and CatBoost, there is an additional optimization aspect to learning the best model, namely stopping the learning at the optimal number of iterations. This is where the early stopping functionality, already discussed earlier, comes in. Same as with validating the best hyperparameter configuration, validating the best iteration to stop the training at must be done against a validation split. Also this validation split should ideally consist of observations that have not influenced modeling decisions in previous stages, because that could result in overfitting to the idiosyncrasies of that particular validation split. Therefore, the observations used for validation of the best early stopping iteration should not have been used for model training, nor validating the best hyperparameter configuration.

The opinions of practitioners seem to vary as to how strict a requirement it is to keep the early stopping validation split unseen, but in our pipeline we decided to keep the split isolated from the prior stages of learning and model selection. To do this, we split our initial training set containing the 80% of the observations of the dataset further, splitting 10% off of the 80% to get what we call the evaluation set to use for validating the best early stopping iteration, and using the remaining 90% of the 80% as the "new training set" for the 10-fold cross-validation.

Figure 6.10 shows the *learning curves* from fitting the XGBoost to predict car expenses with one configuration. The dashed red line indicates the best early stopping iteration, the model corresponding to which was then the final outcome from model selection. Here, the parameter specifying the number of early stopping rounds, i.e., the number of iterations with no gains in minimizing the loss function after which the early stopping gets triggered, was set to 200. We can see that after the dashed red line the training carries on for another 200 rounds, during which the cross-validation loss keeps decreasing, but the evaluation loss does not. By contrast, the evaluation loss turns to worse after the dashed red line, because the model starts to overfit and memorize the data used in the cross-validation. Thus, early stopping can be very important in

avoiding overfitting, and even if the evaluation error would not start rising after some point, at the very least early stopping saves computation time by calling the training to a halt after there are no gains in generalization error anymore.

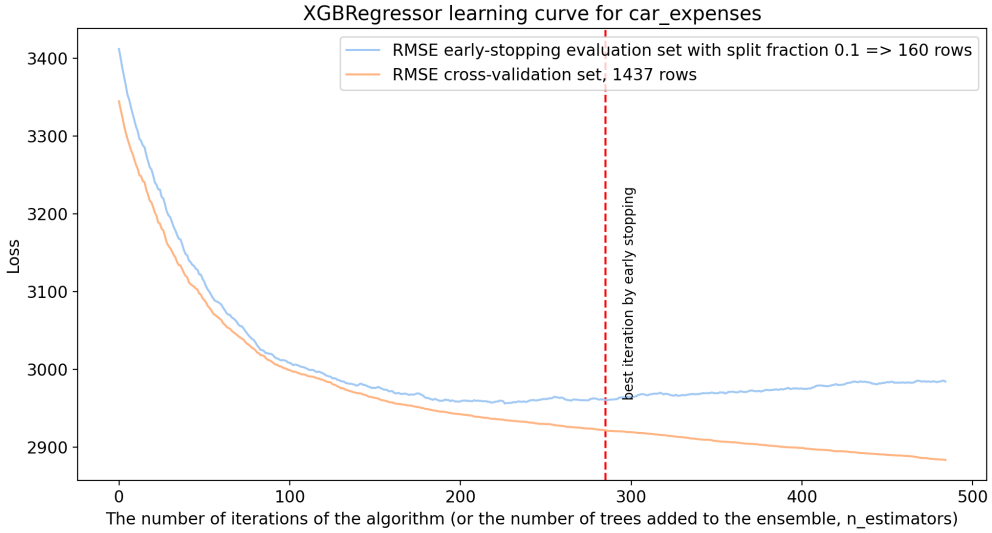


**Figure 6.10:** Learning curves for XGBoost with car expenses as the target.

Once a model has been selected, the ultimate yard stick for evaluating its performance is its performance against the holdout test set, the 20% of the observations of the whole dataset that we have kept unseen by both the training and searching algorithms and the human modeler. This evaluation is effectively inference of our model's ability generalize, or estimation of the generalization error, against truly out-of-sample data, or data that has not yet been observed. Earlier we introduced the four metrics we have chosen for the performance evaluation – $R^2$, EV, MAE, and RMSE. The way the four metrics rank the models is roughly coherent, but sometimes not, which adds another layer of complexity to "analysing the results of the analysis", and to our attempts to sort the models by their "goodness".

Next, let us take a look at what we deemed were the results most revealing of the models' performance in absolute terms and in relation to each other in the three configurations we mentioned we limit our scrutiny to. To better understand the bigger picture, we focus on 1) the performance metrics of predicting the total expenses, or the expenditures from all 12 expenditure categories summed up, and 2) the performance metrics computed from the predictions of the 12 expenditure categories summed up (effectively a weighted average of the performance metrics from these 12 different prediction problems), which we call "aggregated performance metrics". We look at the metrics from any of the 12 prediction problems individually only in a couple of cases of particular interest.

### 6.2.1 Full Distribution (no outlier handling)

Table 6.2 shows the aggregated performance metrics sorted by MAE, from best to worst, for all models and benchmarks using the full distribution a) without PCA dimensionality reduction, b) with PCA dimensionality reduction at 0.9 cutoff level, and c) with PCA dimensionality reduction at 0.8 cutoff level. Table 6.3 shows the performance metrics of predicting total expenses. Moreover, the tables also show the time it took to learn and select the model or models to make these predictions. This time includes both the training time and the model selection time using Optuna hyperparameter tuning with 10 trials for those models that need hyperparameter tuning. The time in Table 6.2 is effectively the total time needed for the training and selection of the models for predicting each of the 12 expenditures categories, except for the interpretable decision tree-based solutions, which were fitted targeting all of the expenditure categories at once as a multi-target problem. The time in Table 6.3 represents the time needed for the training and selection of the models for predicting just the total expenses.

Scrutinizing Table 6.2 and Table 6.3, we can see that the black box models did not really beat linear regression or Bayesian ridge. Another observation is that dimensionality reduction using PCA most often seems to have a negative effect on the performance rather than a positive one. Exceptions to this are Elastic-net, which scores better with both the 90% and 80% PCA dimensionality reduction than without, and SVR, which scores better with the 90% PCA dimensionality reduction than without in the total expenses prediction problem. Note that we could not get an aggregate of the performance metrics for SVR with dimensionality reduction, because the training from the principal components for predicting child care expenses did not converge for some reason.

Of the interpretable solution proposals, the decision tree-based models exhibit significantly poorer performance than linear regression and Bayesian ridge, but they clearly outperform the benchmarks, and even the Elastic-net, and are surprisingly close to the random forest solutions. After all, the decision tree-based models were only fitted once to all the targets as a multi-target problem in order to get just one set of rules for enhanced interpretability, so one might have expected a larger drawback in prediction power than what we see here. Thus, for the level of interpretability that the trees are offering, the trade-off in predictive power can be seen as acceptable, even a bargain. As for the approach of first clustering the feature space and then learning it with a decision tree, it did not seem to provide any benefits over just learning a decision tree from the start, as we see in the tables.

Among the benchmarks, it is no surprise that benchmark 2 does significantly better than benchmark 1. This was expected, because the "coefficient estimates" of

**Table 6.2:** Aggregated prediction performance metrics of all prediction problems using the full distribution (no outlier handling). Sorted by MAE, from best to worst.

| **Aggregated predictions by:** | $R^2$ | EV | MAE | RMSE | Time (h) |
|---|---|---|---|---|---|
| Linear regression | 0.49 | 0.49 | 7397 | 11788 | ∼0.0 |
| CatBoostRegressor (no PCA) | 0.49 | 0.49 | 7415 | 11807 | 4.3 |
| Bayesian ridge | 0.49 | 0.49 | 7430 | 11786 | ∼0.0 |
| XGBRegressor (no PCA) | 0.47 | 0.48 | 7448 | 11942 | 0.3 |
| GaussianProcessRegressor (no PCA) | 0.49 | 0.49 | 7505 | 11754 | 0.8 |
| GaussianProcessRegressor (PCA 0.9) | 0.47 | 0.47 | 7661 | 11977 | 0.8 |
| XGBRFRegressor (no PCA) | 0.46 | 0.46 | 7673 | 12139 | 0.1 |
| ElasticNet (PCA 0.9) | 0.46 | 0.46 | 7675 | 12153 | 0.1 |
| GaussianProcessRegressor (PCA 0.8) | 0.47 | 0.47 | 7699 | 12011 | 1.0 |
| RandomForestRegressor (no PCA) | 0.48 | 0.48 | 7720 | 11922 | 5.5 |
| ElasticNet (PCA 0.8) | 0.45 | 0.45 | 7721 | 12198 | 0.1 |
| CatBoostRegressor (PCA 0.9) | 0.43 | 0.43 | 7857 | 12428 | 2.5 |
| CatBoostRegressor (PCA 0.8) | 0.43 | 0.43 | 7929 | 12439 | 2.4 |
| XGBRegressor (PCA 0.8) | 0.43 | 0.43 | 7947 | 12492 | 0.3 |
| RandomForestRegressor (PCA 0.8) | 0.45 | 0.45 | 8011 | 12224 | 30.3 |
| RandomForestRegressor (PCA 0.9) | 0.45 | 0.46 | 8078 | 12177 | 37 |
| XGBRegressor (PCA 0.9) | 0.41 | 0.41 | 8103 | 12671 | 0.3 |
| XGBRFRegressor (PCA 0.9) | 0.40 | 0.40 | 8262 | 12801 | 0.3 |
| XGBRFRegressor (PCA 0.8) | 0.39 | 0.39 | 8340 | 12872 | 0.2 |
| Post-pruned decision tree | 0.42 | 0.42 | 8358 | 12564 | $< 0.1$ |
| Pre-pruned decision tree | 0.39 | 0.39 | 8463 | 12851 | $< 0.1$ |
| ElasticNet (no PCA) | 0.34 | 0.35 | 8638 | 13374 | 0.1 |
| Benchmark 4: K-means clusters | 0.33 | 0.33 | 8690 | 13499 | ∼0.0 |
| K-means cluster decision tree | 0.30 | 0.31 | 8940 | 13739 | $< 0.1$ |
| SVR (no PCA) | 0.24 | 0.42 | 8979 | 14371 | 0.2 |
| Benchmark 3: common sense groups | 0.29 | 0.29 | 9191 | 13912 | ∼0.0 |
| Benchmark 2: 3-predictor lin. reg. | 0.28 | 0.28 | 9400 | 14000 | ∼0.0 |
| Benchmark 1: estimates from table | 0.05 | 0.25 | 10358 | 16023 | ∼0.0 |
| SVR (PCA 0.9) | Fails to converge in childcare expenses | | | | |
| SVR (PCA 0.8) | Fails to converge in childcare expenses | | | | |

benchmark 1 were not obtained by fitting to the microdataset, so it was foreseeable that testing against observations from the microdataset was going yield worse results for benchmark 1. What was not expected was that benchmark 3 giving the means

**Table 6.3:** Prediction performance metrics when predicting total expenses using the full distribution (no outlier handling). Sorted by MAE, from best to worst.

| **Predictions for total expenses by:** | $R^2$ | EV | MAE | RMSE | Time (h) |
|---|---|---|---|---|---|
| Bayesian ridge | 0.49 | 0.49 | 7482 | 11824 | ∼0.0 |
| GaussianProcessRegressor (no PCA) | 0.48 | 0.48 | 7510 | 11847 | ∼0.0 |
| Linear regression | 0.48 | 0.48 | 7519 | 11858 | ∼0.0 |
| XGBRegressor (no PCA) | 0.47 | 0.47 | 7528 | 11964 | $\ll 0.1$ |
| CatBoostRegressor (no PCA) | 0.47 | 0.47 | 7556 | 11983 | 0.5 |
| RandomForestRegressor (no PCA) | 0.48 | 0.48 | 7630 | 11840 | 0.5 |
| XGBRFRegressor (no PCA) | 0.46 | 0.46 | 7663 | 12088 | $\ll 0.1$ |
| GaussianProcessRegressor (PCA 0.9) | 0.46 | 0.46 | 7715 | 12138 | $< 0.1$ |
| GaussianProcessRegressor (PCA 0.8) | 0.45 | 0.46 | 7729 | 12173 | $< 0.1$ |
| ElasticNet (PCA 0.9) | 0.44 | 0.44 | 7873 | 12363 | $\ll 0.1$ |
| XGBRegressor (PCA 0.8) | 0.42 | 0.42 | 7941 | 12573 | $\ll 0.1$ |
| CatBoostRegressor (PCA 0.8) | 0.43 | 0.43 | 7963 | 12494 | 0.3 |
| XGBRegressor (PCA 0.9) | 0.42 | 0.42 | 8015 | 12514 | $\ll 0.1$ |
| CatBoostRegressor (PCA 0.9) | 0.41 | 0.41 | 8043 | 12668 | 0.2 |
| ElasticNet (PCA 0.8) | 0.42 | 0.42 | 8051 | 12547 | ∼0.0 |
| XGBRFRegressor (PCA 0.8) | 0.41 | 0.41 | 8116 | 12711 | $\ll 0.1$ |
| RandomForestRegressor (PCA 0.9) | 0.42 | 0.42 | 8193 | 12570 | 3.5 |
| RandomForestRegressor (PCA 0.8) | 0.42 | 0.42 | 8221 | 12585 | 2.2 |
| XGBRFRegressor (PCA 0.9) | 0.38 | 0.39 | 8327 | 12926 | $\ll 0.1$ |
| Post-pruned decision tree | 0.42 | 0.42 | 8416 | 12526 | ∼0.0 |
| Pre-pruned decision tree | 0.40 | 0.40 | 8443 | 12742 | ∼0.0 |
| SVR (PCA 0.9) | 0.33 | 0.37 | 8550 | 13493 | $\ll 0.1$ |
| ElasticNet (no PCA) | 0.36 | 0.36 | 8595 | 13198 | ∼0.0 |
| K-means cluster decision tree | 0.31 | 0.31 | 8899 | 13720 | ∼0.0 |
| SVR (no PCA) | 0.33 | 0.34 | 8961 | 13498 | $\ll 0.1$ |
| Benchmark 3: common sense groups | 0.29 | 0.29 | 9191 | 13912 | ∼0.0 |
| Benchmark 2: 3-predictor lin. reg. | 0.28 | 0.28 | 9400 | 14000 | ∼0.0 |
| Benchmark 1: estimates from table | 0.05 | 0.25 | 10361 | 16026 | ∼0.0 |
| SVR (PCA 0.8) | 0.14 | 0.16 | 10672 | 15306 | $\ll 0.1$ |
| Benchmark 4: K-means clusters | -2.57 | 0.00 | 27556 | 31159 | ∼0.0 |

within the "common sense groups", simply 12 common household compositions, as its predictions manages to beat benchmark 2 in both of the tables.

For some reason Benchmark 4 scores much worse than all the rest in predicting

total expenses, although being the best among the benchmarks according to the aggregated prediction performance metrics. Its poor performance with total expenses is likely some glitch due to the prediction problem being a bit of a special case where the target is a summation of all 12 expenditure categories. An underlying problem of a systematic nature with it is evident from the difference between the values of $R^2$ and EV. As mentioned, if EV is higher than $R^2$, it implies of a systematic offset in the predictions. When looking at the plot of the predictions of the benchmarks in Figure 6.11, we can see that the predictions of benchmark 4 indeed systematically overshoot. However, even the $R^2$ score that removes the effect of such a systematic bias is not good for benchmark 4 when predicting total expenses, only reaching the zero level, i.e., matching the performance of a predictor predicting just a constant mean of the test set.

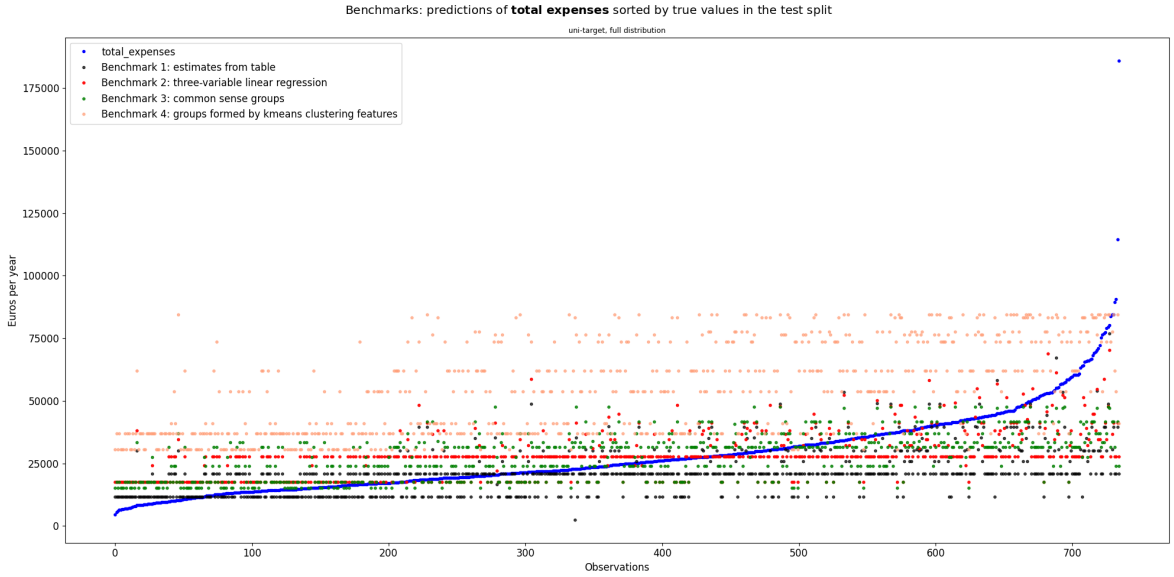


**Figure 6.11:** Predictions of the benchmarks in predicting total expenses using the full distribution (no outlier handling).

Noteworthy observations about the learning times include that CatBoost took roughly an order of magnitude longer to learn than XGBoost, and that for some reason Random Forest took roughly two orders of magnitude longer; the learning time for Random Forest was over 30 hours when using PCA-transformed and reduced data. Note that although the number of Optuna trials was set to 10 for them all, the specified hyperparameter search spaces differed, which may have had some effect on the differences between the learning times. However, these observations of learning times probably give a correct general impression about how fast these three models learn. For Gaussian Processes, the only black box model with which we did not do hyperparameter search, the learning time was greatly affected by the choice of the kernel. We

**Table 6.4:** Prediction performance metrics using the full distribution (no outlier handling) with CatBoost as the black box model and childcare expenses as the target. Sorted by RMSE, from best to worst.

| | $R^2$ | EV | MAE | RMSE |
|---|---|---|---|---|
| Black box: CatBoostRegressor (no PCA) | 0.63 | 0.63 | 259 | 727 |
| Interpr. solution 1.2.: Bayesian ridge | 0.57 | 0.57 | 325 | 776 |
| Interpr. solution 1.1.: Linear regression | 0.57 | 0.57 | 329 | 779 |
| Interpr. solution 2.2.: Post-pruned decision tree | 0.55 | 0.55 | 290 | 798 |
| Interpr. solution 2.1.: Pre-pruned decision tree | 0.54 | 0.54 | 298 | 806 |
| Benchmark 2: 3-predictor linear regression | 0.49 | 0.49 | 279 | 846 |
| Benchmark 4: K-means clusters | 0.44 | 0.44 | 304 | 888 |
| Interpr. solution 2.3.: K-means cluster decision tree | 0.34 | 0.34 | 376 | 965 |
| Benchmark 3: common sense groups | 0.34 | 0.34 | 349 | 1236 |
| Benchmark 1: estimates from table | -0.08 | 0.00 | 336 | 1236 |

saw the learning time vary from less than an hour with one kernel, like in Table 6.2, up to nearly six hours with another one.

One takeaway from these results with the full distribution is that in the following experiments where we run the pipeline on the distribution with its outliers handled we might as well save computation time by having XGBoostRF as the only random forest model since it is up to a couple of orders of magnitude faster to learn than Random Forest while its performance being roughly on par with it. Secondly, since PCA dimensionality reduction was generally not beneficial with the full distribution, it is likely to not be so using the distribution with its outliers handled either. However, we will still do PCA dimensionality reduction with 80% cutoff level of cumulative explained variance in the following experiments just to be sure of this.

Although the interpretable linear regression models scored better than our black box models by both the aggregated performance metrics and the performance metrics of predicting total expenses, there were individual prediction problems where the black box models yielded better performance by a notable margin, such as CatBoost when predicting childcare expenses, as shown in Table 6.4. Granted, all such cases were in predicting expenditures that constitute only a marginal share of the total expenses, childcare expenses being one of them.

The highest $R^2$ of any of the models was 0.67 obtained using XGBoost in predicting utilities and energy expenses, although with only a slight margin to the next best model, linear regression, as shown in Table 6.5. Figure 6.12 shows the predictions of XGBoost in comparison with those of linear regression and Bayesian ridge.

**Table 6.5:** Prediction performance metrics using the full distribution (no outlier handling) with XGBoost as the black box model and utilities and energy expenses as the target. Sorted by RMSE, from best to worst.

| | $R^2$ | EV | MAE | RMSE |
|---|---|---|---|---|
| Black box: XGBRegressor (no PCA) | 0.67 | 0.67 | 491 | 724 |
| Interpr. solution 1.2. Bayesian ridge | 0.65 | 0.65 | 503 | 745 |
| Interpr. solution 1.1: Linear regression | 0.65 | 0.65 | 504 | 747 |
| Interpr. solution 2.1.: Pre-pruned decision tree | 0.48 | 0.48 | 641 | 906 |
| Interpr. solution 2.2.: Post-pruned decision tree | 0.47 | 0.47 | 657 | 911 |
| Benchmark 4: K-means clusters | 0.27 | 0.27 | 796 | 1074 |
| Interpr. solution 2.3.: K-means cluster decision tree | 0.27 | 0.27 | 817 | 1075 |
| Benchmark 2: 3-predictor linear regression | 0.18 | 0.18 | 877 | 1135 |
| Benchmark 3: common sense groups | 0.18 | 0.18 | 874 | 1137 |
| Benchmark 1: estimates from table | -0.06 | 0.15 | 900 | 1294 |

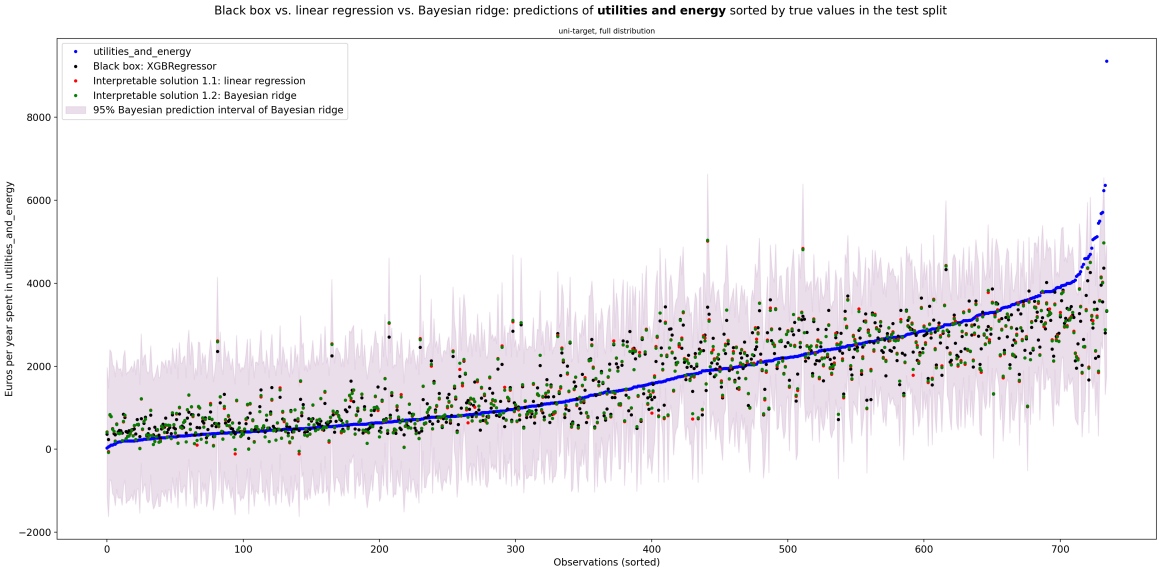


**Figure 6.12:** Predictions of XGBoost, linear regression, and Bayesian ridge against sorted true values of utilities and energy expenses using the full distribution (no outlier handling).

### 6.2.2 Lower Tails Floored

In Table 6.6 and Table 6.7 we again see the aggregated prediction performance metrics and the metrics for predicting total expenses, respectively, but this time when using data with lower tail outliers floored at the minimum references of each expenditure category. Comparing Tables 6.6-6.7 with Tables 6.2-6.3, we can see that flooring improved the metrics for most of the models significantly. Curiously, the metrics for benchmark 1 became significantly worse, however. Other things in Tables 6.6-6.7 seem to be in

line with the earlier observations, such as PCA dimensionality reduction resulting in worse scores, except for Elastic-net and perhaps SVR when predicting total expenses.

**Table 6.6:** Aggregated prediction performance metrics of all prediction problems using the distribution with lower tails floored at minimum references. Sorted by MAE, from best to worst.

| **Aggregated predictions by:** | $R^2$ | EV | MAE | RMSE | Time (h) |
|---|---|---|---|---|---|
| Linear regression | 0.55 | 0.55 | 6694 | 11014 | ∼0.0 |
| Bayesian ridge | 0.55 | 0.55 | 6720 | 11007 | ∼0.0 |
| CatBoostRegressor (no PCA) | 0.54 | 0.54 | 6784 | 11078 | 4.3 |
| XGBRegressor (no PCA) | 0.53 | 0.53 | 6806 | 11272 | 0.3 |
| GaussianProcessRegressor (no PCA) | 0.55 | 0.55 | 6818 | 11029 | 0.9 |
| GaussianProcessRegressor (PCA 0.9) | 0.52 | 0.53 | 7001 | 11316 | 1.0 |
| XGBRFRegressor (no PCA) | 0.50 | 0.50 | 7155 | 11633 | 0.2 |
| ElasticNet (PCA 0.8) | 0.49 | 0.49 | 7271 | 11679 | 0.1 |
| CatBoostRegressor (PCA 0.8) | 0.49 | 0.49 | 7324 | 11682 | 2.9 |
| XGBRegressor (PCA 0.8) | 0.45 | 0.46 | 7514 | 12098 | 0.3 |
| XGBRFRegressor (PCA 0.8) | 0.45 | 0.45 | 7635 | 12122 | 0.2 |
| Post-pruned decision tree | 0.42 | 0.42 | 7923 | 12452 | $< 0.1$ |
| Benchmark 4: K-means clusters | 0.40 | 0.41 | 7982 | 12662 | ∼0.0 |
| Pre-pruned decision tree | 0.42 | 0.42 | 8047 | 12465 | $< 0.1$ |
| K-means cluster decision tree | 0.39 | 0.39 | 8128 | 12791 | ∼0.0 |
| SVR (no PCA) | 0.31 | 0.48 | 8225 | 13619 | 0.2 |
| Benchmark 3: common sense groups | 0.38 | 0.38 | 8285 | 12934 | ∼0.0 |
| Benchmark 2: 3-predictor lin. reg. | 0.39 | 0.39 | 8354 | 12826 | ∼0.0 |
| ElasticNet (no PCA) | 0.34 | 0.34 | 8669 | 13314 | 0.1 |
| Benchmark 1: estimates from table | -0.12 | 0.35 | 11619 | 17330 | ∼0.0 |
| SVR (PCA 0.8) | Fails to converge in childcare expenses | | | | |

### 6.2.3 Lower Tails Floored and Upper Tails Dropped

Lastly, in Tables 6.8-6.9 we have the aggregated prediction performance metrics and the metrics for predicting total expenses when using data with lower tail outliers floored at the minimum references and upper tail outliers dropped at two standard deviations above the mean in each expenditure category. Comparing Tables 6.8-6.9 to the earlier, there is again a significant improvement in the metrics, especially in MAE and even more so in RMSE, which is nearly halving compared with just flooring the lower tails.

**Table 6.7:** Prediction performance metrics when predicting total expenses using the distribution with lower tails floored at minimum references. Sorted by MAE, from best to worst.

| **Predictions for total expenses by:** | $R^2$ | EV | MAE | RMSE | Time (h) |
|---|---|---|---|---|---|
| Bayesian ridge | 0.55 | 0.55 | 6771 | 11051 | ∼0.0 |
| CatBoostRegressor (no PCA) | 0.54 | 0.54 | 6784 | 11078 | 4.3 |
| Linear regression | 0.54 | 0.54 | 6806 | 11083 | ∼0.0 |
| GaussianProcessRegressor (no PCA) | 0.54 | 0.54 | 6837 | 11092 | 0.1 |
| XGBRegressor (no PCA) | 0.52 | 0.52 | 6912 | 11372 | $< 0.1$ |
| GaussianProcessRegressor (PCA 0.8) | 0.51 | 0.51 | 7092 | 11478 | 0.1 |
| ElasticNet (PCA 0.8) | 0.50 | 0.51 | 7188 | 11533 | $\ll 0.1$ |
| XGBRFRegressor (no PCA) | 0.49 | 0.49 | 7251 | 11706 | 0.1 |
| CatBoostRegressor (PCA 0.8) | 0.48 | 0.49 | 7279 | 11773 | 0.3 |
| XGBRegressor (PCA 0.8) | 0.47 | 0.47 | 7405 | 11913 | $< 0.1$ |
| ElasticNet (no PCA) | 0.46 | 0.46 | 7557 | 12054 | $\ll 0.1$ |
| XGBRFRegressor (PCA 0.8) | 0.45 | 0.45 | 7615 | 12125 | 0.1 |
| Post-pruned decision tree | 0.42 | 0.42 | 7923 | 12452 | ∼0.0 |
| Pre-pruned decision tree | 0.42 | 0.42 | 8047 | 12465 | ∼0.0 |
| K-means cluster decision tree | 0.38 | 0.38 | 8257 | 12937 | ∼0.0 |
| Benchmark 3: common sense groups | 0.38 | 0.38 | 8285 | 12934 | ∼0.0 |
| Benchmark 2: 3-predictor lin. reg. | 0.39 | 0.39 | 8354 | 12826 | ∼0.0 |
| SVR (PCA 0.8) | 0.19 | 0.25 | 9237 | 14728 | $< 0.1$ |
| SVR (no PCA) | 0.29 | 0.30 | 9387 | 13849 | $< 0.1$ |
| Benchmark 1: estimates from table | -0.12 | 0.35 | 11626 | 17336 | ∼0.0 |
| Benchmark 4: K-means clusters | -3.40 | 0.02 | 31100 | 34360 | ∼0.0 |

**Table 6.8:** Aggregated prediction performance metrics of all prediction problems using the distribution with upper tails dropped at mean + 2 std and lower tails floored at minimum references. Sorted by MAE, from best to worst.

| **Aggregated predictions by:** | $R^2$ | EV | MAE | RMSE | Time (h) |
|---|---|---|---|---|---|
| GaussianProcessRegressor (no PCA) | 0.57 | 0.57 | 4413 | 5809 | 0.9 |
| Linear regression | 0.57 | 0.57 | 4459 | 5833 | ∼0.0 |
| Bayesian ridge | 0.57 | 0.57 | 4463 | 5846 | ∼0.0 |
| CatBoostRegressor (no PCA) | 0.57 | 0.57 | 4467 | 5834 | 4.3 |
| GaussianProcessRegressor (PCA 0.8) | 0.55 | 0.55 | 4526 | 5971 | 0.8 |
| XGBRegressor (no PCA) | 0.54 | 0.55 | 4553 | 5982 | 0.2 |
| CatBoostRegressor (PCA 0.8) | 0.52 | 0.53 | 4647 | 6112 | 3.4 |
| ElasticNet (PCA 0.8) | 0.52 | 0.52 | 4696 | 6138 | < 0.1 |
| XGBRFRegressor (no PCA) | 0.52 | 0.52 | 4715 | 6148 | 0.2 |
| XGBRegressor (PCA 0.8) | 0.50 | 0.50 | 4757 | 6274 | 0.3 |
| XGBRFRegressor (PCA 0.8) | 0.46 | 0.47 | 4976 | 6494 | 0.2 |
| Benchmark 4: K-means clusters | 0.44 | 0.44 | 5112 | 6619 | ∼0.0 |
| K-means cluster decision tree | 0.44 | 0.44 | 5159 | 6644 | ∼0.0 |
| Benchmark 3: common sense groups | 0.42 | 0.42 | 5183 | 6748 | ∼0.0 |
| Benchmark 2: 3-predictor lin. reg. | 0.41 | 0.41 | 5320 | 6829 | ∼0.0 |
| Post-pruned decision tree | 0.35 | 0.39 | 5560 | 7151 | < 0.1 |
| ElasticNet (no PCA) | 0.33 | 0.33 | 5665 | 7259 | 0.1 |
| SVR (no PCA) | 0.24 | 0.52 | 5768 | 7738 | 0.2 |
| Pre-pruned decision tree | 0.30 | 0.34 | 5795 | 7395 | < 0.1 |
| Benchmark 1: estimates from table | -0.29 | 0.33 | 7672 | 10066 | ∼0.0 |
| SVR (PCA 0.8) | Fails to converge in childcare expenses | | | | |

**Table 6.9:** Prediction performance metrics when predicting total expenses using the distribution with upper tails dropped at mean + 2 std and lower tails floored at minimum references. Sorted by MAE, from best to worst.

| **Predictions for total expenses by:** | $R^2$ | EV | MAE | RMSE | Time (h) |
|---|---|---|---|---|---|
| CatBoostRegressor (no PCA) | 0.49 | 0.49 | 4759 | 6326 | 0.2 |
| XGBRegressor (no PCA) | 0.49 | 0.49 | 4798 | 6310 | $< 0.1$ |
| GaussianProcessRegressor (no PCA) | 0.46 | 0.46 | 4927 | 6518 | 0.1 |
| Bayesian ridge | 0.45 | 0.45 | 4934 | 6578 | $\sim$0.0 |
| CatBoostRegressor (PCA 0.8) | 0.47 | 0.47 | 4936 | 6470 | 1.2 |
| XGBRFRegressor (no PCA) | 0.48 | 0.48 | 4957 | 6417 | $< 0.1$ |
| Linear regression | 0.44 | 0.44 | 4963 | 6621 | $\sim$0.0 |
| GaussianProcessRegressor (PCA 0.8) | 0.45 | 0.45 | 4989 | 6584 | 0.1 |
| Benchmark 3: common sense groups | 0.44 | 0.44 | 5047 | 6635 | $\sim$0.0 |
| ElasticNet (no PCA) | 0.45 | 0.45 | 5062 | 6571 | $\ll 0.1$ |
| XGBRegressor (PCA 0.8) | 0.44 | 0.44 | 5074 | 6649 | $< 0.1$ |
| ElasticNet (PCA 0.8) | 0.44 | 0.44 | 5138 | 6643 | $\ll 0.1$ |
| XGBRFRegressor (PCA 0.8) | 0.44 | 0.44 | 5141 | 6633 | $< 0.1$ |
| SVR (no PCA) | 0.40 | 0.41 | 5179 | 6860 | $< 0.1$ |
| Benchmark 2: 3-predictor lin. reg. | 0.32 | 0.32 | 5517 | 7291 | $\sim$0.0 |
| K-means cluster decision tree | 0.19 | 0.33 | 6220 | 7990 | $\sim$0.0 |
| Pre-pruned decision tree | 0.09 | 0.27 | 6438 | 8446 | $\sim$0.0 |
| Post-pruned decision tree | 0.12 | 0.29 | 6547 | 8331 | $\sim$0.0 |
| SVR (PCA 0.8) | 0.02 | 0.02 | 7002 | 8793 | 0.1 |
| Benchmark 1: estimates from table | -0.30 | 0.07 | 7655 | 10128 | $\sim$0.0 |
| Benchmark 4: K-means clusters | -16.76 | -1.71 | 34394 | 37368 | $\sim$0.0 |

# 7. Conclusions

Let us conclude the thesis by reflecting on the validity of the work done, providing justification for some of the focal choices that we made, drawing conclusions from the results, presenting ideas for improvement and refinement, and finally discussing some new directions where expenditure estimation in loan origination could be taken.

## 7.1 Reflection and Justification

We did our utmost to follow best practices in pre-processing and preparing the data for modeling, building the modeling pipeline, and evaluating the models. Perhaps our biggest doubt in the process came in the pre-processing and preparation stage: what should be the correct way to handle outliers? This is a question that is difficult to answer since we do not have a dataset of "ground truth without outliers" to evaluate the model performance against after using different ways of outlier handling.

Nevertheless, at least handling the lower tails at the minimum references seems perfectly justified, because our trained model should not predict values lower than those references anyway. What remains an unanswered questions is whether the correct way to handle the lower tail outliers is by dropping or flooring. If we knew the reasons behind the low values for those observations better, we could make more informed, even case-specific, decisions between flooring and dropping. But since we do not have such information, we must choose between the two ways to use for all the outliers, and as such we are probably better off flooring, because this way we do not throw away the tremendous number of observations that fall below the minimum references.

As for the upper tails, it is perhaps more questionable whether we should handle the outliers at all, and if we do, where we should set the threshold of calling an observation an outlier. In any case, even with as low a threshold as two standard deviations above the mean, the number of upper tail outliers was much smaller than the number of lower tail outliers based on the minimum references, as we saw in Table 5.1. Therefore, dropping this number of observations at the upper tail was deemed justified.

In the modeling pipeline, we included several model families as options to be used for the black box models, choosing which we used our best judgment as to which ones

could provide the highest predictive power or the best compromise between predictive power and interpretability. We took into account the fact that the number of observations available for training in each prediction problem was at most only under 3,000, and therefore, e.g., neural networks, which typically require much more data to stand out, were deemed unsuitable for this problem. Moreover, problems with cross-sectional tabular data are usually considered to be where particularly ensemble methods shine, and that is why we expected the best predictive power from the gradient boosted trees, and perhaps the random forests, too. However, 3,000 observations may have been too little data also for them to really stand out from classical models like linear regression, therefore explaining why we did not see them dominating in predictive power. Another possibility is that the features extracted, selected and used for the modeling simply did not contain enough signal to make any model to be able to provide more predictive power than what we saw in the results. In a following section, we will suggest some ideas on how we could go about trying to improve the predictive power.

## 7.2 Conclusion to the Research Question

The research question presented in Introduction was: using the statistical microdataset of household consumption expenditures, how good solutions can we find for estimating or predicting household expenditures (used as a proxy for loan applicants' expenditures) when considered in terms of a) predictive power, and b) a combination of or a suitable compromise between predictive power and interpretability? The results summarized in Tables 6.2-6.9 make a strong case that there are many options for obtaining better predictions by predictive power than what the benchmarks were capable of. In some prediction problems, the black box models we used provided the best predictive power, although usually with only a little margin to the best interpretable models. Moreover, the black box models provided little to no interpretability. In contrast, the decision tree-based solutions and the linear regression-based solutions provided also interpretability of their predictions: the decision tree-based models in the form of a set of decision rules, and the linear regression-based models in the form of estimates of regression coefficients.

Out of all the models we used, the ordinary linear regression model with a full set of features might provide the best combination of both predictive power and interpretability, although its rather high number of estimates of coefficients may be somewhat overwhelming to interpret. With the careful data preparation that we conducted, the Bayesian ridge did not provide clear benefit over the ordinary linear regression. If simpler interpretation is desired, then the number of features and consequently the number of coefficient estimates in the linear regression model could be reduced, per-

haps all the way to match with those of the three-regressor coefficient estimates of benchmark 1 and benchmark 2. However, benchmark 3, the "common sense groups", in fact provided better predictive performance than benchmark 2 (which in turn beat benchmark 1), while its interpretation is at least as simple as that of benchmark 2.

A predictive power somewhere between the full-blown linear regression model and the benchmarks is provided by the decision tree-based solutions. Decision trees provide an easy interpretation in the form of a set of rules, but these rules are not as intuitive as, e.g., the rules that make up the "common sense groups" of benchmark 3.

All in all, this work suggests an assortment of options for models to use for predicting loan applicant's expenditures, out of which one must take great care in deciding the best solution for their purpose. For example, one must decide what the needed level and type of interpretability is, and whether a single one of the solutions should be used for all 12 prediction problems, or if a different solution, the best solution for each particular problem, could possibly be used for each of them.

## 7.3 Improvement and Refinement

To get better predictive power from the black box models, the best bet would probably be to put more time and effort in hyperparameter search. This would include trying to get a better understanding of the meaning of the different hyperparameters of each model and learning the best practices of searching for their optimal values. Studying machine learning practitioners' works available online, e.g., their entries in prediction competitions, might prove useful. Another approach would be to put more effort in crafting features, i.e., feature engineering. This would require more domain knowledge about, e.g., the household consumption survey and the data available in loan application processes.

If any of the black box models provided significantly better predictive power than all the other options, it would have made a tempting solution to propose for expenditure prediction despite lack of interpretability. In such a case, to remedy the interpretability issue, we could have tried fitting the black box model as a multi-target problem (if supported), which would have given us only one model for all 12 prediction problems, somewhat facilitating interpretation, although with the cost of reduced predictive power.

We could also explore more the potential of using different clustering algorithms for partitioning the training data and then assigning new observations to these clusters for determining predictions for them. *Supervised clustering* might be an interesting family of methods in this regard.

*Robust regression* might be another interesting concept to use with this dataset

with its numerous outlier-looking observations. It could provide improved predictive performance, while retaining the interpretability of linear regression and letting us omit outlier handling with its associated problems.

## 7.4 Discussion

There is potential, and evidence, that requiring pieces of information related to a topical domain known as *environment, social, and governance* (ESG) in the loan application process could prove very beneficial – not only for the lender, but also for the borrower, and even society at large. For example, green thinking and environmental sustainability considerations fall under the ESG umbrella. In fact, considering the effect of energy consumption related ESG aspects on loan applicant's expenditures was initially one central idea in this research, because conventional mortgage loan underwriting criteria are based on metrics that do not capture, e.g., the differences in energy and transportation expenses between different properties and households [15]. Thus, it is a topical question in credit score modeling whether incorporating new variables that represent energy-efficiency aspects, such as *energy efficiency performance* (EPC) rating of the property or the motive power of the applicant's vehicles, into the mortgage credit scoring model would help to differentiate the credit risk of applicants better [39]. However, it turned out that our dataset of household expenditures did not contain information needed for taking these two aspects into account, so they fell out of the scope of this study. If suitable data is available in the future, this would make an interesting and timely research direction with a potentially big impact.

Another direction to consider extending this research to is the domain of *longitudinal analysis*. This study was only concerned with cross-sectional data and correlations of the household consumption survey of 2016. However, Statistics Finland has conducted these surveys approximately every 5-7 years, so there is more data available than just the latest "cross-section". Moreover, the data from the latest survey from the year 2022 was published during the course of this work. If enough households have been participating in several of these surveys over the years, perhaps some additional insight could be drawn from treating the data as a longitudinal dataset with auto-correlations and cross-correlations on top of the cross-sectional correlations.

One aspect to be taken into account if any of the solution propositions of this work are considered to be deployed is their *fairness*. Some variables characterizing the loan applicant, such as gender, are considered *protected attributes* and should not crucially affect the loan decision. Variables such as gender are straightforward to account for, but there are also more complicated, hidden associations between variables that may lead to unfair decisions being made with machine learning models [30].

# Bibliography


[1] C. Albon. *Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning*. O'Reilly Media, Incorporated, 2018.

[2] R. B. Avery, R. W. Bostic, P. S. Calem, and G. B. Canner. Credit risk, credit scoring, and the performance of home mortgages. *Federal Reserve Bulletin*, 82:621–648, 1996.

[3] J. Banasik, J. Crook, and L. Thomas. Recalibrating scorecards. *Journal of the Operational Research Society*, 52:981–988, 2001.

[4] M. Barès and É. Bossé. *Relational Calculus for Actionable Knowledge*. Information Fusion and Data Science. Springer International Publishing, 2022.

[5] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[6] M. S. Ben-Shachar. Stat's What It's All About - What ARE the Assumptions of Linear Regression [online publication]. https://blog.msbstats.info/posts/2018-08-30-linear-regression-assumptions, accessed on 15th February 2024.

[7] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artif. Intell. Rev.*, 54(3):1937–1967, mar 2021.

[8] M. Billio, M. Costola, L. Pelizzon, and M. Riedel. Buildings' energy efficiency and the probability of mortgage default: The dutch case. *The Journal of Real Estate Finance and Economics*, 65(3):419–450, 2022.

[9] L. Breiman. Statistical modeling: the two cultures. *Statist. Sci.*, 16(3):199–231, 2001. With comments and a rejoinder by the author.

[10] J. Brownlee. *Machine Learning Algorithms From Scratch with Python*. Machine Learning Mastery, 2016.

[11] J. Brownlee. *Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models and Work Projects End-to-end*. Machine Learning Mastery, 2016.

[12] J. Brownlee. *XGBoost With Python: Gradient Boosted Trees with XGBoost and scikit-learn*. Machine Learning Mastery, 2016.

[13] J. Brownlee. *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery, 2020.

[14] J. Brownlee. *Ensemble Learning Algorithms With Python: Make Better Predictions with Bagging, Boosting, and Stacking*. Machine Learning Mastery, 2021.

[15] L. Burt, D. B. Goldstein, and S. E. Leeds. A path towards incorporating energy and transportation costs into mortgage underwriting: Shifting to fact-based analysis. 2010.

[16] G. C. Calafiore and L. El Ghaoui. *Optimization Models*. Cambridge University Press, 2014.

[17] J. Devore. *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2015.

[18] P. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, Oct 2012.

[19] B. Efron. Prediction, estimation, and attribution. *Journal of the American Statistical Association*, 115(530):636–655, 2020.

[20] European Banking Authority. Interactive single rulebook - Capital Requirements Regulation (CRR) [online publication]. https://www.eba.europa.eu/regulation-and-policy/single-rulebook/interactive-single-rulebook/16022, accessed on 13th February 2024.

[21] S. M. Finlay. Predictive models of expenditure and over-indebtedness for assessing the affordability of new consumer credit applications. *Journal of the Operational Research Society*, 57(6):655–669, 2006.

[22] P. Florek and A. Zagdański. Benchmarking state-of-the-art gradient boosting algorithms for classification [preprint]. arXiv:2305.17094, 2023.

[23] A. Gelman, J. Hill, and A. Vehtari. *Regression and Other Stories*. Analytical Methods for Social Research. Cambridge University Press, 2020.

[24] A. Gelman and C. P. Robert. Not only defended but also applied: The perceived absurdity of bayesian inference. *The American Statistician*, 67(1):1–5, 2013.

[25] L.-G. Giraudet, A. Petronevich, and L. Faucheux. Differentiated green loans. *Energy Policy*, 149:111861, 2021.

[26] Google. Regularization for Simplicity - $L_2$ Regularization - Machine Learning - Google for Developers [online publication]. https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization, accessed on 14th February 2024.

[27] Google. Regularization for Sparsity - $L_1$ Regularization - Machine Learning - Google for Developers [online publication]. https://developers.google.com/machine-learning/crash-course/regularization-for-sparsity/l1-regularization, accessed on 14th February 2024.

[28] M. Gruber. *Improving Efficiency by Shrinkage: The James-Stein and Ridge Regression Estimators*. 11 2017.

[29] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[30] T. Hellström, V. Dignum, and S. Bensch. Bias in machine learning – what is it good (and bad) for? *CoRR*, abs/2004.00686, 2020.

[31] C. Hernández and M. Sandoval. Building models based on artificial neural networks to predict entrepreneurial intentions among undergraduate students. In *Proceedings of the 4th European International Conference on Industrial Engineering andr Operations Management*. IEOM Society International, 2021.

[32] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3), 2021.

[33] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer US, 2021.

[34] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Trans. Knowl. Discov. Data*, 6(4), dec 2012.

[35] R. K. Kaufmann, N. Gonzalez, T. A. Nickerson, and T. S. Nesbit. Do household energy expenditures affect mortgage delinquency rates? *Energy Economics*, 33(2):188–194, 2011.

[36] Kielikone Oy. MOT SFS Standards: riski [online publication]. http://www.sanakirja.fi/finnish-english/riski, accessed on 13th February 2024.

[37] Lehtinen, Anna-Riitta. Mitä eläminen maksaa? : kohtuullisen minimin viitebudjettien hintapäivitys vuodelle 2021 [online publication]. http://hdl.handle.net/10138/333191, accessed on 29th February 2024.

[38] M. A. Lones. How to avoid machine learning pitfalls: a guide for academic researchers [preprint]. arXiv:2108.02497v4, 2024.

[39] Loriana Pelizzon. Creating an Energy Efficient Mortgage for Europe: Review of the impact of energy efficiency on probability of default [online publication]. https://www.igbc.ie/wp-content/uploads/2017/10/EeMAP-Technical-Report-on-the-Impact-of-Energy-Efficiency-on-Probability-of-default.pdf, accessed on 13th February 2024.

[40] T. Mitchell. *Machine Learning.* McGraw Hill series in computer science. McGraw Hill, 2017.

[41] C. Molnar. *Modeling Mindsets: The Many Cultures of Learning from Data.* Christoph Molnar, 2022.

[42] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 841–848, Cambridge, MA, USA, 2001. MIT Press.

[43] Nosedal, Al. Inference for a Population Proportion [lecture notes]. https://mcs.utm.utoronto.ca/~nosedal/sta313/sta313-proportions.pdf, accessed on 14th February 2024. University of Toronto.

[44] Official Statistics of Finland (OSF). Households' consumption [online publication]. https://stat.fi/en/statistics/ktutk, accessed on 29th February 2024.

[45] J. W. Osborne. *Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data.* SAGE Publications, Inc., 2nd edition, 2013.

[46] P. Palmroos. *Essays on Modeling and Analysis of Mortgage Loan Pools and the Delphi Method in Forecasting of Financial Variables.* Aalto University, 2016.

[47] S. Raschka. Model evaluation, model selection, and algorithm selection in machine learning [preprint]. arXiv:1811.12808v3, 2020.

[48] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.

[49] S. Y. Rauterkus, G. I. Thrall, and E. Hangen. Location efficiency and mortgage default. *The Journal of Sustainable Real Estate*, 2(1):117–142, 2010.

[50] M. Rönkkö, E. Aalto, H. Tenhunen, and M. I. Aguirre-Urreta. Eight simple guidelines for improved understanding of transformations and nonlinear effects. *Organizational Research Methods*, 25(1):48–87, 2022.

[51] scikit-learn. 1.1. Linear Models - scikit-learn 1.4.1 documentation [online publication]. https://scikit-learn.org/stable/modules/linear_model.html, accessed on 16th February 2024.

[52] scikit-learn. 1.4. Support Vector Machines - scikit-learn 1.4.1 documentation [online publication]. https://scikit-learn.org/stable/modules/svm.html, accessed on 16th February 2024.

[53] scikit-learn. 1.7. Gaussian Processes - scikit-learn 1.4.1 documentation [online publication]. https://scikit-learn.org/stable/modules/gaussian_process.html, accessed on 14th February 2024.

[54] scikit-learn. 3.3. Metrics and scoring - quantifying the quality of predictions - scikit-learn 1.4.1 documentation [online publication]. https://scikit-learn.org/stable/modules/model_evaluation.html, accessed on 14th February 2024.

[55] scikit-learn. sklearn.linear_model.LinearRegression - scikit-learn 1.4.1 documentation [online publication]. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html, accessed on 14th February 2024.

[56] G. Shmueli. To Explain or to Predict? *Statistical Science*, 25(3):289 – 310, 2010.

[57] Stephen Richardson. Creating an Energy Efficient Mortgage for Europe: Towards a new market standard [online publication]. https://www.igbc.ie/wp-content/uploads/2018/09/Creating-an-Energy-Efficient-Mortgage-for-Europe-Towards-a-New-Market-Standard-Web.pdf, accessed on 13th February 2024.

[58] L. Thomas, J. Crook, and D. Edelman. *Credit Scoring and Its Applications.* SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2017.

[59] xgboost developers. Categorical Data - xgboost 2.1.0-dev documentation [online publication]. https://xgboost.readthedocs.io/en/latest/tutorials/categorical.html, accessed on 14th February 2024.

[60] xgboost developers. Random Forests(TM) in XGBoost - xgboost 2.1.0-dev documentation [online publication]. https://xgboost.readthedocs.io/en/latest/tutorials/rf.html, accessed on 14th February 2024.

[61] xgboost developers. XGBoost Parameters - xgboost 2.1.0-dev documentation [online publication]. https://xgboost.readthedocs.io/en/latest/parameter.html, accessed on 14th February 2024.

[62] Yandex LLC. Categorical features - CatBoost documentation [online publication]. https://catboost.ai/en/docs/features/categorical-features, accessed on 14th February 2024.

# Appendix A. Descriptions of Variables

**Table A.1:** Preliminary set of features to proceed to data pre-processing and preparation stage with. The information is from the variable description sheet of the microdataset.

| | Name | Description | Values |
|---|---|---|---|
| 1 | nuts3 | Region/county | 1 = Uusimaa<br>2 = Varsinais-suomi<br>4 = Satakunta<br>5 = Kanta-Häme<br>6 = Pirkanmaa<br>7 = Päijät-Häme<br>8 = Kymenlaakso<br>9 = Etelä-Karjala<br>10 = Etelä-Savo<br>11 = Pohjois-Savo<br>12 = Pohjois-Karjala<br>13 = Keski-Suomi<br>14 = Etelä-Pohjanmaa<br>15 = Pohjanmaa<br>16 = Keski-Pohjanmaa<br>17 = Pohjois-Pohjanmaa<br>18 = Kainuu<br>19 = Lappi<br>21 = Ahvenanmaa |
| 2 | paak | Capital region<br>(i.e. the municipalities of Helsinki, Espoo, Kauniainen and Vantaa) | 1 = yes<br>2 = no |
| 3 | maka | Rural-urban classification | Bl = N/A<br>K1 = Inner urban area<br>K2 = Outer urban area<br>K3 = Urban fringe<br>M4 = Rural local center<br>M5 = Countryside close to urban area<br>M6 = Countryside proper<br>M7 = Sparsely populated countryside |

| | | | |
|---|---|---|---|
| 4 | aikuisia | Number of adults. A person was regarded an adult if they turned 18 in the same calendar year as when the survey was conducted (2016). A person younger than 18 was regarded an adult if they were the reference person or the spouse of the reference person in the household. | >= 1 |
| 5 | ellu | Number of over 64-year-olds | >= 0 |
| 6 | yli75v | Number of over 75 -year-olds | >= 0 |
| 7 | skidi | Number of under 3-year-olds | >= 0 |
| 8 | muku | Number of under 7-year-olds | >= 0 |
| 9 | peru | Number of 7-12 year-olds | >= 0 |
| 10 | murk | Number of 13-16 year-olds | >= 0 |
| 11 | isot | Number of non-working 17-24 year-olds who were not married, in unmarried cohabitation with a partner, divorced or widowed. | >= 0 |
| 12 | kaytetmk | Net income of the household (earned income, entrepreneur's income, capital income, income transfers, e.g., social benefits, received) | Euros per year |
| 13 | talotyyp | Dwelling type | 1 = one-family house (detached houses and farmhouses)<br>2 = two-family house (semi-detached houses/duplex houses and farmhouses)<br>3 = Row houses, terraced houses and linked houses<br>4 = apartment buildings<br>5 = other |

| | | | |
|---|---|---|---|
| 14 | ravuosi | Year of construction | 1919− |
| 15 | paalammi | Main source of heating | 1 = district heating<br>2 = direct electric heating<br>3 = electric storage heating<br>4 = oil district heating<br>(building or property specific)<br>5 = other type of district heating<br>(building or property specific)<br>6 = stove, fireplace or other<br>heating type using solid fuels<br>7 = geothermal heat<br>8 = air source heat pump<br>9 = other |
| 16 | aspintaa | Dwelling area | Square meters |
| 17 | huonluku | Number of rooms in the dwelling | > 0 |
| 18 | toimivia | Number of economically active household members (practicing a profession: either an employee or an entrepreneur) | >= 0 |
| 19 | vamto | Status in employment of the reference person (the responder) of the household | 0 = not practicing a profession<br>1 = practicing a profession |
| 20 | pamto | Status in employment of the spouse of the reference person | 0 = not practicing a profession<br>1 = practicing a profession<br>bl = no spouse |
| 21 | pika | Age of the reference person at the end of the calendar year of the survey | Years |
| 22 | psup | Gender of the reference person | 1 = man<br>2 = woman |

| | | | |
|---|---|---|---|
| 23 | soss90 | Socioeconomic status of the reference person | 10 = agricultural entrepreneur<br>20 = entrepreneur<br>30 = upper white-collar worker<br>40 = lower white-collar worker<br>50 = blue-collar worker<br>60 = student or school child<br>70 = retiree or pensioner<br>80 = long-term unemployed<br>90 = other non-working (e.g. homemaker or stay-at-home parent)<br>99 = socioeconomic status unknown |
| 24 | vaautom | Number of passenger cars or vans (no company-owned cars) | >= 0 |
| 25 | vamautom | Number of recreational vehicles | >= 0 |
| 26 | vatsautm | Number of company-owned cars | >= 0 |
| 27 | vaasvaum | Number of trailers and sleeping caravans | >= 0 |
| 28 | vavenem | Number of motor or sailing boat | >= 0 |
| 29 | muuas | Possession of a second home or a buy-to-let home | 0 = no<br>1 = yes |
| 30 | muuasulk | Possession of a second home or a buy-to-let home abroad | 0 = no<br>1 = yes |
| 31 | vapaikas | Possession of a recreational dwelling | 0 = no<br>1 = yes |
| 32 | vapaulk | Possession of a recreational dwelling abroad | 0 = no<br>1 = yes |