Jerry Valtanen

Predicting the Annual Salary Costs of Finnish Companies with Machine Learning

Master's Thesis in Governance of Digitalization
Supervisor: Xiaolu Wang
Faculty of Social Sciences, Business and Economics
Åbo Akademi University

Helsinki 2021

ÅBO AKADEMI UNIVERSITY – Faculty of Social Sciences, Business and Economics
Abstract for Master's thesis

Subject: Governance of Digitalization	
Writer: Jerry Valtanen	
Title: Predicting the Annual Salary Costs of Finnish Companies with Machine Learning	
Supervisor: Xiaolu Wang	Supervisor:

Abstract:

The revenues of the Finnish Mutual Pension Insurance Companies are determined by the annual salary costs of their customer companies, as the average premium of a pension insurance is projected to be 24.4% in 2021. Therefore, knowing the annual salary costs of potential customers is crucial for determining the potential revenue that the mutual pension insurance company will receive. Suomen Asiakastieto is a public database about Finnish companies and their financial information, but some companies do not share their annual salary costs with Suomen Asiakastieto. This creates issues determining the potential revenue and assigning the customer to the appropriate salespeople. The purpose of this thesis was to find the suitable machine learning models to predict the salary costs of those companies, that have not shared their financial information with Suomen Asiakastieto. Linear Regression was used as the baseline of this thesis, as it is a simple model to implement, and it is computationally inexpensive. Neural Networks, Support Vector Regression, Decision Tree and Random Forest models were compared to the Linear Regression model and their performance was estimated using cross-validation. The results suggest that machine learning models can be used in solving the problem quite accurately, but more quality and accurate data is needed for more accurate results.

Keywords: Machine Learning, Linear Regression, Data Science, Supervised Learning, Random Forest, Support Vector Regression, Neural Networks, Decision Tree, Predictive Analytics

Date: 17.6.2021	Number of pages: 73

Table of Contents

1 Introduction 3
1.1 Problem Statement 3
1.2 Background 4
1.3 Research Questions 5
1.4 The Structure of The Thesis 6
2 Machine Learning 7
2.1 Unsupervised Learning 9
2.2 Supervised Learning 10
2.2.1 Regression methods 10
2.2.2 Neural Networks 18
2.2.3 Support Vector Machines 21
2.2.4 Regression trees 23
2.2.5 Random Forest 25
2.3 Handling missing data 27
2.4 Sampling and pre-processing the data 28
2.5 Model Selection & Parameter Tuning 31
2.6 Evaluating the performance and accuracy of the model 31
2.7 Related work 32
3 Empirical Study 34
3.1 Dataset and preprocessing 34
3.2 Linear Regression model and the Machine Learning algorithms in Python 46
3.3 K-Fold Cross-Validation with Python 47
3.4 Machine Learning Model Parameter Tuning 48
3.5 Analyzing the results 51
3.5.1 Random Forest 51
3.5.2 Linear Regression 56
3.5.3 Neural Network 60
3.5.4 Decision Tree 62
3.5.5 Support Vector Regression 64
4 Conclusion 69
4.1 Answers to Research Questions 69
4.2 Future research 70
References 72

1 Introduction

1.1 Problem Statement

The aim of this thesis is to find the most accurate machine learning model to predict the annual salary costs of Finnish companies, and the motivation for this thesis comes from the authors occupation. The authors main assignment is to find potential customers for a Finnish mutual pension insurance company, and the mutual pension insurance company's revenue is based upon the annual salary costs of its customers. In 2021, the projected premium of a pension insurance will be 24.4% from the annual salary costs, as well as some other fixed costs that will not be mentioned in this thesis. Therefore, knowing the annual salary costs of Finnish companies is a vital starting point in determining the potential customers for the mutual pension insurance company.

The mutual pension insurance company collects data from Suomen Asiakastieto, which is a database about Finnish companies. The database contains annual financial information and characteristics of all Finnish companies. Some of these companies do not share their salary data with Suomen Asiakastieto, which creates issues when determining the potential revenue coming from that company. Furthermore, this creates problems assigning these potential customers to the appropriate salespeople. Determining the potentiality of a company is a much more complex task than just determining the potential revenue received from the customer, but the salary costs are the starting point of that process. The data were retrieved from a Suomen Asiakastieto SQL database and will be cleaned and preprocessed for modelling. Data will then be fed for the machine learning models to teach the algorithms to find patterns in the data, and to make predictions about the unknown salary costs.

The starting point for the models in this study will be linear regression, whose performance will be compared to the other selected machine learning models. The other machine learning models used in this paper are Support Vector Machines, Artificial Neural Network, Decision Trees and Random Forests. This prediction problem is a regression problem, as it deals with predicting continuous numerical values, and not a classification problem, where datasets are classified into predefined categories. The models will be created with Python, as the author is most familiar with that programming language.

This thesis builds its foundation upon previous research conducted in the field of different machine learning models and regression analysis. The results and the model building chapters will use some coding related sources as well, as these parts focus on implementing and optimizing selected models.

1.2 Background

One field within the broader scope of artificial intelligence is machine learning, which has good capabilities of predicting the probabilities of future outcomes. The learning is done by learning from past events. Regression methods and machine learning have been previously used for the kind of problems that this thesis is trying to solve, which is why those methods were chosen for this study as well. Das et al., (2020) used regression techniques, such as linear and polynomial regression, to predict the salaries of individual employees after a certain period of time. They used historical data of salary increases to predict the future increases in one's salary level. Koskinen et al. (2005) were trying to predict the future salaries for individual employees, which were to be used for planning the future mutual pension insurance costs. They also used linear regression models for their predictions, but the model had some model parameters as extensions, which can be called as mixed models (Koskinen et al. 2005). Previous research in this field revolves around individual salary levels, and none of those studies consider the total annual salary costs of an entire company. Therefore, this thesis is somewhat unique and one of the first of its kind.

Machine learning is a powerful tool that utilizes mathematics and computer science to create mathematical models, which are capable learning from data and making predictions. Machine learning has been gaining attention in the past few years as the related technologies have become more accessible and more efficient. Hence, it has enabled companies to utilize tools such as image recognition, artificial neural networks and other prediction models. The paper written by Alom et al. (2019) presents real-life cases solved and improved by machine learning algorithms, which were mainly deep learning models such as speech recognition, machine translation and medical information processing.

1.3 Research Questions

This thesis tries to answer the question as to which is the most accurate and feasible statistical model or machine learning algorithm to predict annual salary costs of Finnish companies using public financial data of these companies? The thesis also endeavors to determine which are the most significant features to predict these annual salary costs of Finnish companies. Therefore, the research questions can be defined as follows:

- RQ1: What are the most important features that help predicting the annual salary costs of Finnish companies?
    - Predicting something using machine learning needs features, that affect the target variable in question. The goal is to find these features using data analytics and the data from Suomen Asiakastieto, so that the target variable can be accurately predicted
- RQ2: What is the most accurate machine learning algorithm to predict the target variable?
    - Many machine learning algorithms are suitable for the task, such as linear regression, support vector regression and neural network regression, and the goal is to determine the most accurate algorithm that the author can later use in his day job as a production model.
- RQ3: Are the more sophisticated and computationally expensive machine learning models better than a Linear Regression model?
    - Determining whether the more complex and sophisticated machine learning models are more accurate than a Linear Regression model is interesting, as Linear Regression is computationally more efficient than many machine learning models. Furthermore, the Linear Regression model is easier to interpret. Therefore, the complex models need to be considerably better performing in order for them to be considered as production models

The author will try to find the answers to these questions by conducting a thorough literature review on machine learning, and by using the data from Suomen Asiakastieto to build and

compare different machine learning models, which could also provide valuable business insights.

1.4 The Structure of The Thesis

Chapter 2 will introduce the literature about machine learning and artificial intelligence. It will begin by defining the different methods of machine learning and the general concepts of the field. Chapter 2 will also delve into the different types of machine learning, such as supervised and unsupervised learning, which are crucial concepts for this thesis. More detailed information about the different statistical and machine learning methods, such as linear regression, support vector machines and artificial neural networks, will also be given in chapter 2. Hence, the readers will be familiarized with the models that are going to be used. The second chapter will also define the different methods of evaluating the prediction models, such as the root squared error metrics (RMSE) and the R^2-score, also known as the coefficient of determination. Finally, the chapter will introduce the methods most commonly used in data exploring and preprocessing, the different re-sampling tools and how to do hyper parameter tuning and model selection.

The third chapter will begin by introducing the dataset that was used in this research, and how the dataset was cleaned and pre-processed. Also, a thorough exploratory data analysis will be presented, so that the reader will understand the problem clearly. Chapter 3 will also combine all the information about the machine learning models and statistical methods that were introduced in the first two chapters. Furthermore, the chapter will present how those machine learning models were optimized for achieving better results. Finally, the chapter will look at the results that were acquired by the chosen optimized models and analyse which of the models were the most relevant for the given problem.

Chapter 4 will bind the whole thesis together and tries to answer to the research questions that were presented in chapter 1. Finally, the fourth chapter will also provide possible suggestions for future studies made in a similar field as this research.

2 Machine Learning

The purpose of this chapter is to provide the reader knowledge about machine learning by thoroughly reviewing the current literature about statistical and machine learning models. This information will also help the author in the formulation of the machine learning models, and it might also help him avoiding some issues along the way. Chapter 2 will also explain some important categories of machine learning, such as supervised and unsupervised learning, how different machine learning models work and how regression methods can be evaluated.

The amount of data that humans create daily is increasing exponentially, and the maintenance and handling of this data is becoming more important. The rise of big data has created a need to process vast amounts of data as efficiently as possible, which has made the need for tools for such operations important and wanted. In order for someone to solve a problem with a computer, one needs to compile an algorithm to do it and teach the machine how to do it. (Alpaydin, E., 2014)

Artificial Intelligence simulates the human intelligence, which is done via computers. Machine learning, or automated learning, is a subcategory of Artificial Intelligence, where the model tries to make a machine to learn from past experiences. In other words, this means turning experience into expertise and knowledge. (Shalev-Shwartz, S. & Ben-David, S., 2014) Think of a classic example of email spam messages, where the spam filter tries to recognize spam messages using the pattern it learned from past email messages. In this case, the machine knows the content of the messages and whether they were spam messages or not. The machine then tries to find the same patterns from new messages and see how likely those new messages are spam messages. The machine continues learning as the amount of data increases and adjusts itself iteratively. (Alpaydin, E., 2014)

The resurgence of different technologies, such as remote and real-time data collecting, has made it possible to collect vast amounts of data about almost anything. This has enabled the use of machine learning in scenarios that seem completely random and would be thought difficult to predict. (Alpaydin, E., 2014) Consider consumer behavior, where people do not go to a grocery store and buy products at random, but they do buy certain items at the same

time. These are the kinds of patterns that a machine learning model is trying to find, and which companies then use to extract valuable information.

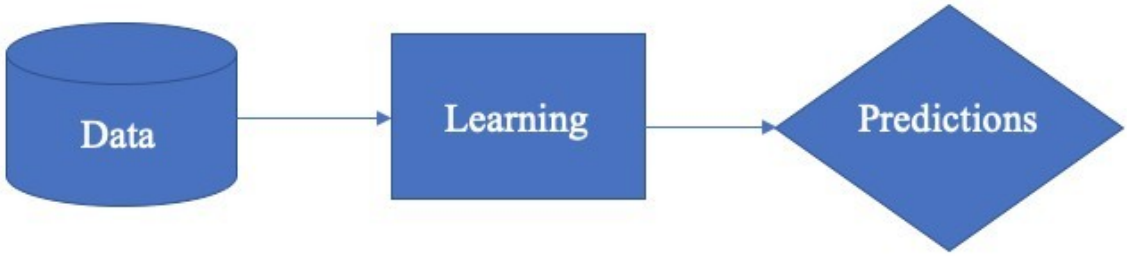

Figure 1 - Machine Learning Process

One of the simplest ways for a machine to learn is the classic "trial and error"-method, where the machine tries to do an action at random and stores the outcome to its memory. Eventually the machine repeats this process until the desired outcome is found. This type of learning is called route learning, and it was one of the first methods taught to a machine that was supposed to learn to play chess.

Figure 1 shows the process in a simplified form: data are a different set of features for the machine, which the machine uses for learning and finding the patterns in the data. From the learning process the machine will "extract" the patterns and will use it for generalization, i.e., making prediction for future values and generalizing those predictions to all unseen observations. This function can be simply be written down as a function f: X à Y, where the data X are used to create the predictions Y. This is a simplified function as the function itself depends on the problem at hand, and the algorithm used. The X and Y variables are "application-specific", meaning that they represent the data objects and the related predictions respectively. (Bekkerman R. et al, 2012)

The selection of the previously mentioned models and also the features can be a heuristic process, as there is no need, nor would it be feasible to use every possible model available for making predictions. (Bekkerman R. et al, 2012) Heuristic process means that one chooses the suitable models and features for the problem from their previous experiences and knowledge. (Shalev-Shwartz, S. & Ben-David, S., 2014) Deisenroth et al. (2020) suggest that the feature selection also requires domain knowledge and feature engineering, so that the features are

useful. Such feature engineering might include converting categorical variables into numerical dummy variables, turning postal numbers into longitude and latitude values, scaling values and many more different types of transformations.

2.1 Unsupervised Learning

Humans are capable of learning in multiple ways, which also applies for machines. The first learning type is called unsupervised learning, where no target variable is given in the learning data. (Bekkerman R. et al, 2012, Alpaydin, E., 2014) Awad & Khanne (2015, p. 4) describe unsupervised learning as a technique that "groups instances without a prespecified dependent attribute" and Hastie et al. (2017) continue by stating that it is "learning without a teacher". Thus, the unsupervised learning methods do not have any specific error metrics as there are no right or wrong answers that can be measured. Unsupervised learning algorithms try to find hidden patterns, regularities and structures in the datasets, but the final target variables are unlabeled. (Awad, M. & Khanne, R. 2015) One method of unsupervised learning is finding similar groups in the input data, which is defined as a clustering model. For example, customer segmentation is a form of clustering, where the clusters are segments in the customer base and the outliers may imply the existence of a niche market. (Alpaydin, E., 2014)

Balaji & Srivatsa (2012) used unsupervised learning methods in their paper "Customer Segmentation for Decision Support using Clustering and Association Rule based approaches". Their objective was to find the right segmentation methods to help decision makers to provide the right products to the right customers in the health insurance field. They utilized the Waikato Environment for Knowledge analysis workbench, which is a free software for data mining and machine learning.

Chan (2005) used a Neural Network Model to divide the customers of an Online Auction platform into segments. Chan had to deal with the lack of quality data, as the online auction platform only shared the customers id, so they needed to conduct some datamining to gather the relevant features. They utilized features such as the last bid time, overall auction length, buyer and seller ratings etc. These features were used to segment the sellers and buyers into homogeneous groups of impulsive, analytical and patient dealers/customers.

2.2 Supervised Learning

The second method for a machine to learn is called supervised learning, where the machine is given the learning data and the "right answers", or target variables. From the data the machine tries to find the patterns that will lead to sensible generalized predictions for any unseen data. This means that the supervised learning algorithms take advantage of the input data in order for them to be able to build the aforementioned prediction function f. The training data can be described as labeled examples $(x, y) \in X \times Y$, where x is the data and y are the truth predictions for x. "The ultimate goal of supervised learning is to identify a function f, that produces accurate predictions on test data" (Bekkerman R. et al., 2012, p. 2)

There are two types of data that will be utilized in the model building and measuring processes: training data and testing data. The difference between the training data and the test data is that training data is used to build up a model, to fit the data and to find the patterns in the data. On the other hand, the testing data is used to validate the model's accuracy and performance. The ultimate goal is to fit the model so that the features relate to the future observations as accurately as possible (prediction), or that a better understanding is obtained about the relationship of these two (inference). (James et al., 2017)

Two of the most common methods of supervised learning are classification methods and regression methods. The difference between these two methods is that regression is trying to predict infinite, continuous numerical outputs, whereas classification methods try to classify the predictions to previously determined output categories. The most often used classification values are binary, i.e., 0 and 1. The problem of this thesis is a regression problem, so there is no need to explain classification further.

2.2.1 Regression methods

As mentioned at the end of the previous chapter, one form of supervised learning are regression models. One of the most popular regression methods is linear regression. Regression analysis is a form of statistical learning, where the modeler tries to investigate and model the relationship between two or more variables. (Montgomery et al., 2001, James et

al., 2017) Linear regression will be the baseline for this thesis and for solving the problem, as the thesis is trying to predict a continuous value. The objective of linear regression is to find a function that models the training data but is also able to make generalized predictions to future and unknown observations. (Deisenroth et al., 2020)

Simple linear regression is a very straight forward method of predicting a continuous numerical variable. It takes a single predictor feature X to predict the quantitative target variable Y. The most important assumption is that there is an approximately linear relationship between those two variables. This linear relationship can be written as:

$$Y \approx \beta_0 + \beta_1 X + \varepsilon$$

This equation can be read as "regressing Y onto X" (James et al., 2017, p.63), which means predicting the value Y on the unit increase in X. It also means that one is estimating the Y, not saying that the value is precise. (James et al. 2017) The parameters β_0 and β_1 are unknown and they are called regression coefficients. For example, the β_0 can be interpreted as the intercept term of the model, where it tells the expected value of the predicted value Y when X is zero. The β_1 coefficient tells the average increase in the predicted value Y when X increases by one unit. (Montogmery et al., 2001, James et al., 2017) The epsilon sign ε describes random noise, or error. The model estimates this noise and tries to explain all the possible errors that are missed by the model. This is done because the true relationship of the variables most likely is not completely linear. (James et al., 2017) With the error term one has the assumption that the mean of the error term is zero and it has an unknown variance. One must also assume that the error terms of the different coefficients are not correlated, i.e., have no linear relationship. (Montgomery et al., 2001)

For example, consider the housing prices, where it can be said that the price (Y) rises on the increase of the size of the house (X):

$$\text{Price of the house} \approx \beta_0 + \beta_1 \times \text{Size of the House} + \varepsilon,$$

but for the prediction one would give the model estimated values:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

and in this case $\hat{y}$ means the prediction for Y. In the simplest form this linear regression model can be read as: the predicted amount of the price of the house ($\hat{y}$ increases per the unit increase in the size of the house.

2.2.1.1 Estimating the coefficients

As mentioned in the previous chapter, the regression coefficient parameters β_0 and β_1 are unknown and they will be estimated based on the data that are presented to the model. They are the estimated as the linear combinations of the collected Y values. The method that is used in estimating the regression coefficients is the least squares method. This means that one tries to minimize the sum of squared differences between the observed Y and the regression line. The difference between the observed value Y and the estimated value $\hat{y}$ is called a residual, which is used in the evaluation of the model adequacy. (Montgomery et al., 2001, James et al., 2017) The difference between the observed and estimated value can be seen in figure 3, where the estimated values are presented as a regression line and the observed values and residuals are shown compared to the regression line.

The line that the regression model generates is called the population regression line, which shows the most accurate approximation for the relationship of the independent variable X and the dependent variable Y. This regression line can be seen in the figure 2, where there are several possible regression lines, but the best fit is represented as a black line.

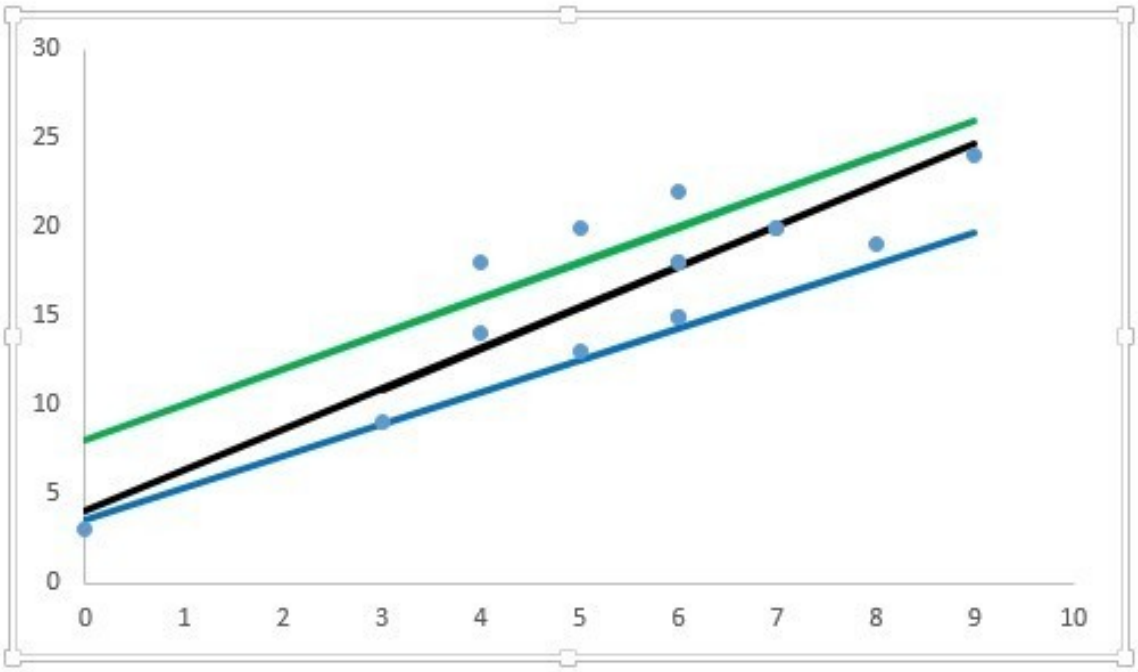

Figure 2 - Many possible regression lines, with black line being the best fit. Retrieved from: https://miro.medium.com/max/1024/1*rQoiu4Y51IUMDsOE8aGCyw.jpeg

After estimating the regression coefficients, one needs to evaluate whether or not there is a relationship between the independent variable X and the dependent variable Y. This is done by computing p-values for the regression coefficients, and if these p-values are below the set threshold, normally 0.05 or 0.01, one declines the hypotheses that there is no relationship between the variables and declares that there is a relationship between the variables. (James et al., 2017) A coefficient table can be seen in table 1.

	Coefficient	p-value
Intercept	$\beta 0$	< 0.01
Size of the house	$\beta 1$	< 0.01

Table 1 - Coefficient table

2.2.1.2 Model Accuracy

Evaluating the model and how well it describes the data and the relationship between the variables X and Y is extremely important. The two most used accuracy metrics are residual standard error, or RSE, and the R^2-score, also called the coefficient of determination. (Montgomery et al., 2001)

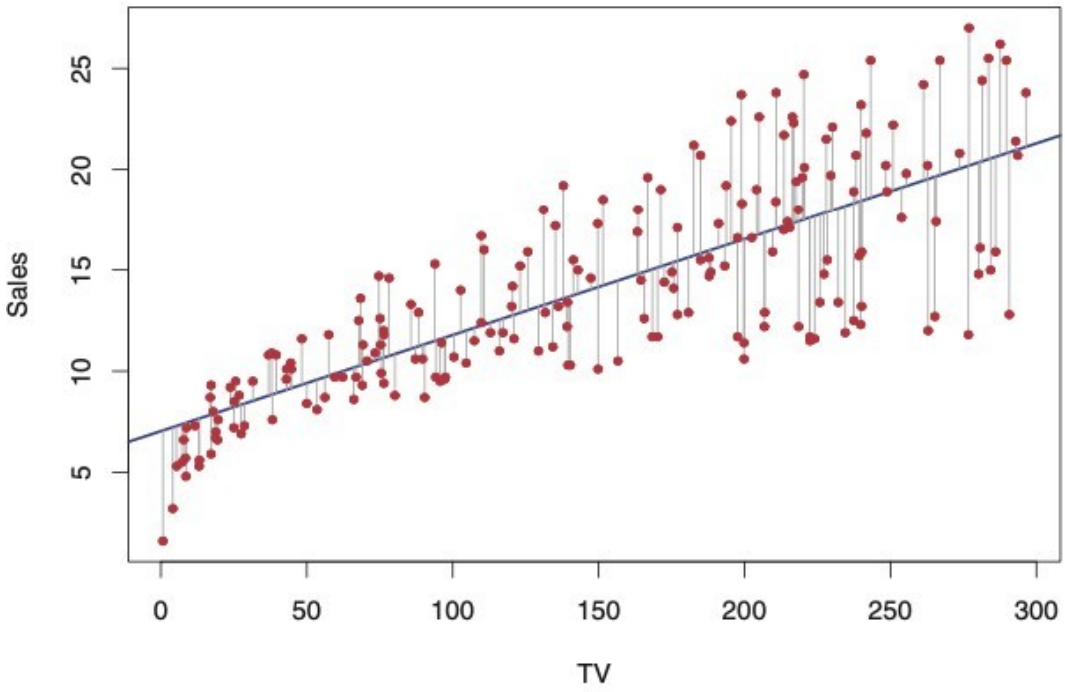

Figure 3 - Linear Regression line and residual lines (James et al., 2017)

The residual standard error is considered as the estimation of the standard deviation of the random noise term ϵ. Simply said it tries to estimate the average amount that the response variable Y is off from the actual regression line. (James et al., 2017) This means that if the predicted value $\hat{y}$ is far off from the observed value y, the residual standard error will increase and vice versa. The RSE is determined with the following equation:

$$RSE = \sqrt{1/(n-2)} * \text{nnnnnnnn } Sum \quad of \quad Squares$$

Where residual sum of squares is the squared differences between the predicted value $\hat{y}$ and the actual value y. The expression can be written as

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y})^2$$

As RSE is measured in the same units as the response variable Y, interpreting the results is sometimes difficult as there is no clear threshold for a good model. The R^2 statistic, also known as the coefficient of determination, tries to avoid this issue and it tries to explain how much of the variability of the relationship is explained by the model. The value of R^2 always lies in between 0 and 1 and the closer it is to 1, the more the model explains the variability in

the data. This can also be read as that there is 0-100% less variation around the prediction line than the mean. (Montgomery et al., 2001, James et al., 2017) The R^2 statistic can be calculated with the following formula:

$$R^2 = 1 - \frac{\&"}{("}$$

Where TSS is the total sum of squares and it is calculated as the squared difference of the observed value yi and the mean value of y, or $y@$This formula is written as

$$TSS = \ ?(yi - y@)^2$$

The modeler needs to remember that the R^2 statistic is a measure of correlation, not accuracy. Thus, if the value of the R^2 statistic is for example 0.85, it implies that the model describes 85% of the variation in the outcome. (Alpaydin, E., 2014)

2.2.1.3 Multiple Linear Regression

Simple linear regression is an efficient tool when the value that one is trying to predict is dependent on a single variable, but in the case of this study the author is interested in the annual salary costs of Finnish companies. Hence, one could safely argue that the annual salary costs are affected by more than one feature, such as the number of employees, the industry and the city in which the company operates in. In this case the author will use multiple linear regression, that takes into account multiple features that affect the target variable Y. One could run individual simple linear regression models for all these independent variables, but this would not be the most accurate way of predicting the target variable, as these models do not take into consideration the relationship between the three features. This could lead to untrustworthy predictions and therefore it should be avoided. (James et al. 2017)

As mentioned previously, multiple linear regression model takes into account more than one independent variable X that explains our dependent variable Y. This means that the equation for multiple linear regression is:

$$y = \ \beta 0 + \beta 1X\ 1 + \quad n2n2 + \cdots + \quad \beta iXi + \quad \varepsilon$$

which can be interpreted as the average response of target variable y per unit change in feature Xi, when all other independent variables are held constant. (Montgomery et al., 2001, James et al., 2017) Once again consider the housing price example, where the model to predict a price of the house could look something like this:

$$Price\,of\ the\ house \approx \beta_0 + \beta_1 \times size\ of\ the\ house + \beta_2 \times the\ condition\ of\ the\ house + random\ error.$$

Also, the variables can be interpreted as following: β0, or the intercept, is the price of the house when the size and the condition of the house are zero. The β1 can be interpreted as the unit increase in the price of the house when the size of the rises by on unit and the condition of the house is constant. Finally, the β2 can be read as the increase in the price of the house when the condition of the house increases by unit and the size of the house stays constant.

The estimation of the regression coefficients is the same least squared approach as is used in the simple linear regression model, where the coefficients are chosen by minimizing the previously mentioned sum of squared residuals, or RSS.

The important question to ask in the multiple linear regression setting is whether there is a relationship between the features and the target variable, that the model is trying to predict. This can be calculated with a simple hypotheses test, where one tries to find whether or not the independent variables βi are 0. (James et al., 2017) If this hypotheses test proves that there is a relationship between the independent variables and the dependent variable, one needs to choose a method for variable selection, i.e., which variables describe the dependent variable the most and which are not that descriptive. There are three widely used and generally approved approaches for the variable selection:

1. Forward selection: This selection method starts with an empty model, where only the intercept coefficient β0 is introduced and it assumes that there are no predictors, so the predictor values are non-existent. Then p simple linear models are fitted and the variable that has the lowest residual sum of squares is added to the empty model. This process is iterated until a pre-determined stopping rule is met. (Montgomery et al., 2001, James et al., 2017)

2. Backward selection/elimination: The method is the opposite of forward selection, where descriptive variables were added one by one to an empty model. In backward selection all the considered independent variables are chosen for the model and removed if they do not meet the determined threshold, for example t-statistic or p-value. This is method is run until all the variables meet the selected threshold. (Montgomery et al., 2001, James et al., 2017)
3. Mixed selection: This method starts also from the empty model that only contains the intercept coefficient β0 and adds one independent variable at a time to the model. The difference between the forward selection method is that the mixed selection model can delete any of the variables if the selected threshold value (p-value for example) rises too high in any point of the iteration. (Montgomery et al., 2001)

Also, another factor to consider is the attribute of the independent variables, which means whether they are quantitative or qualitative. Qualitative variables mean variables that are not numerical, such as gender (male or female). This would be visible if one would predict the annual salaries of individuals, as gender seems to have an impact on the salaries of individuals. Qualitative variables need to be computed as dummy variable, so that the variable Xi takes a value of 1 or 0, depending on whether the observed sample is a female or a male. If there are more than 2 levels in the qualitative variable, one should create dummy variables for the levels, and these variables will take the value 1 or 0 whether or not the observed sample meets that level or not. (James et al.,2017)

After the variables have been chosen, the model is fitted, and the fit will be calculated. This is done the same way as in simple linear regression, i.e., the RSE or R^2 statistics. The issue with the R^2 statistic is that the value of it increases with every variable added to the model and will result in overfitting if too many variables are used. (James et al., 2017) A proper method is to use the RSE statistic on the side as well, so that a middle ground is found.

2.2.1.4 Potential problems with Linear Regression

Many possible issues may occur in linear regression modelling, such as:

1. Non-linearity between the feature(s) and the target variable
   - One must always assume that there is a linear relationship between the features and target variable. If there is not, all of the predictions are suspect for error.

2. Correlation of the error terms
    - If there is a correlation between the error terms, then the model underestimates the true standard error, and the confidence interval will be too narrow.
3. Non-constant variance of the error terms
    - If the error or residuals form a funnel shape, the model might have heteroscedasticity and the residuals tend to increase as the fitted value increases
4. Outliers
    - As is often the case in statistics, outliers will skew the model and provide inaccurate predictions. This is also the case in linear regression, as outliers will move the least square line toward the outlier and will affect the fit of the model.
5. Collinearity
    - Collinearity means a situation where two or more features are related to each other, i.e., they have a linear relationship. This means that it is hard to evaluate how a feature affects the target variable.

The handling of these problems is a fine art, and one needs to take into consideration these issues when building a linear regression model. (James et al., 2017)

2.2.2 Neural Networks

Artificial Neural Networks, or ANNs, are a method of machine learning where a computational model is created to mimic the human brain and its construct. Agatonovic-Kustrin and Beresford (1999, p.1) explain the ANNs in their research "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research"; "Artificial neural networks (ANNs) are biologically inspired computer programs designed to simulate the way in which the human brain processes information. ANNs gather their knowledge by detecting the patterns and relationships in data and learn (or are trained) through experience, not from programming."

Some important concepts in neural networks are the layers (seen in figure 4 below), where the input layer includes the input values, or the features, which are then multiplied by the connection weights and fed to the neurons in the hidden layer through those connection coefficients. (Agatonovic-Kustrin S. & Beresford R., 1999) These weights are unknown parameters, and the model tries to find the values that create a model that describes and fits the data well. A hidden layer is not as mythical as its name might suggest, as it's just a layer

that isn't an input or an output layer. The hidden layer consists of the linear combinations of the input values. The output layer is the sum of all the weighted inputs, that first have been passed through the hidden layer neurons and their activation functions. The number of hidden layers determines whether the neural network is considered as deep-learning or not. More than one hidden layer constitutes as deep learning. (Hastie, T., Tibshirani, R., & Friedman, J. 2017).

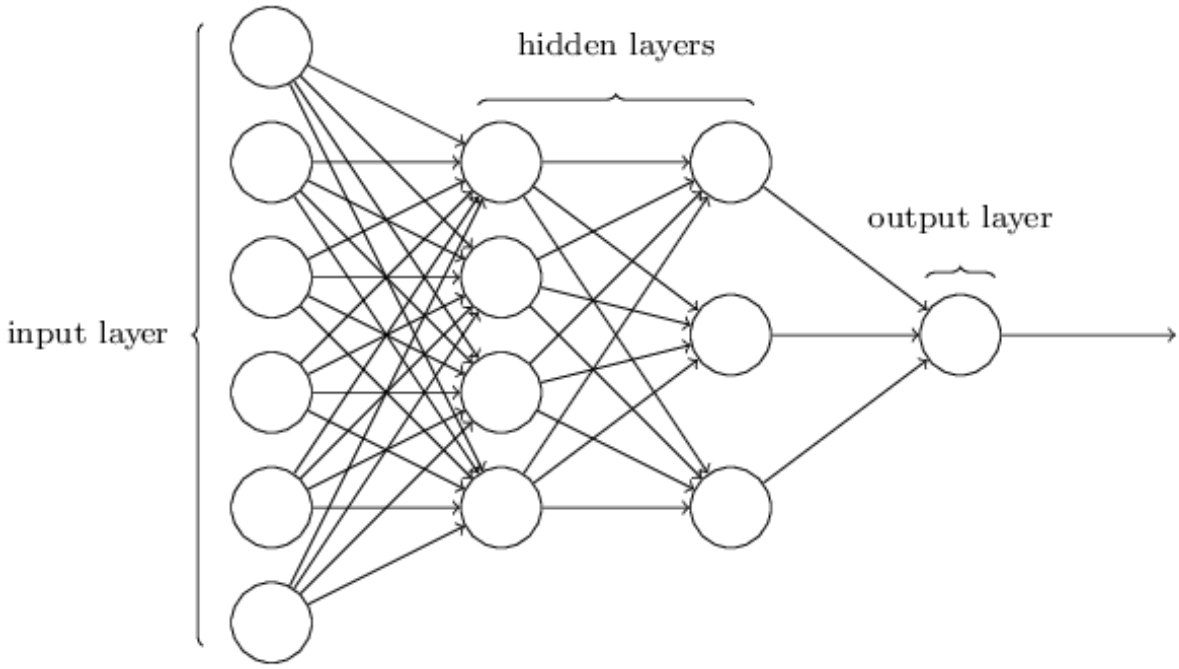

Figure 4 - Artificial Neural Network

The neural network models try to minimize the sum of squared residuals, which are the sum of squares difference between the predicted value and the observed value, but it is important to remember that neural networks are nonlinear statistical models. "Neural networks can be seen as a non-linear generalization of a linear model, both for regression and classification." (Hastie, T., Tibshirani, R., & Friedman, J. 2017, p. 394).

The way artificial neural network models adjust the weights to find the best fit is called back-propagation rule. (Agatonovic-Kustrin S. & Beresford R., 1999) The model tries to map the given features iteratively to adjust the weights after every layer of the model. The model gives information about the features forward for the next neurons, and so that the optimization of the weights happens between the neurons, in the connections or synapses.

The backward propagation is done during the learning phase, where neural network knows the input and output data. The model then tries to find the weights that minimizes difference between the predicted and target values. (Agatonovic-Kustrin S. & Beresford R., 1999; Kuhn, M., & Johnson, K., 2013; Hastie, T., Tibshirani, R., & Friedman, J. 2017) The model knows whether it was wrong and adjusts the weights accordingly. This is done by sending the information back to the synapses and neurons, which calculate the weights again so that the model finds a local minimum that minimizes the squared difference between the observed value and the predicted value. This process is called the gradient descent, and it is done via numerous training cycles. Although, model should not be allowed to run too many times and it should not have too many weights, as neural networks have a tendency to overfit the model and therefore make the generalization of the model to disappear. Overfitting can also happen if the model has too many hidden layers. (Hastie, T., Tibshirani, R., & Friedman, J. 2017).

2.2.2.1 Issues with Neural Networks

ANN's have some well-documented issues, such as the models are often overly parametrized, which have created the need for proper guidelines:

1. Starting values for weights need to be near zero, so that the model starts as a nearly linear model and becomes less and less linear as the weights increase. Using exactly zero weights does not move the model and too high weight values lead to bad results in the predictions. (Hastie, T., Tibshirani, R., & Friedman, J. 2017)
2. Overfitting is an often-occurring problem with ANN's, as the model tend to have too many weights assigned to them and therefore the model "will overfit the data at the global minimum". (Hastie, T., Tibshirani, R., & Friedman, J. 2017, p. 398) This is because of the large number of regression coefficients in the model. (Kuhn, M., & Johnson, K., 2013) This can be avoided by assigning the model with a stopping rule, where the model doesn't exceed the stopping rule when teaching itself. This rule can be for example a rule where the model stops when a threshold value for a regression coefficient, so that the impact of that regression coefficient to the model needs to significant enough. A good way to control overfitting is also to control the hyperparameter learning rate. Learning rate affects how fast the Neural Networks backpropagation converges, i.e., large learning rate may cause the oscillation of the algorithm and a small learning rate may cause a slow convergence. (Awad & Khanna, 2015)

3. Scaling the input values is a good practice as this leads to equal regularization process. Scaling could be done so that the inputs have a mean zero and standard deviation one. (Hastie, T., Tibshirani, R., & Friedman, J. 2017)
4. The number of hidden units and layers need to be well thought. If the number of hidden units is too low, the model might not be able to find the non-linearities in the data. If there are too many hidden layers the model can move the extra weights too close to zero, which make them pretty much useless anyway. (Hastie, T., Tibshirani, R., & Friedman, J. 2017)

2.2.3 Support Vector Machines

Support Vector Machines, or SVM's, are mostly used for classification problems, where nonlinear decision boundaries are utilized. (Hastie, T., Tibshirani, R., & Friedman, J. 2017) SVM's can also be used in regression problems, when they are called Support Vector Regressions, or SVR's. The SVR's, as well as linear regression models, try to find parameters that minimize the sum of squared errors (shown in figure 5). The difference between linear regression models and SVR's is that in SVR models the samples that have small residuals do not contribute so much on the definition of the prediction line. (Kuhn, M., & Johnson, K., 2013)

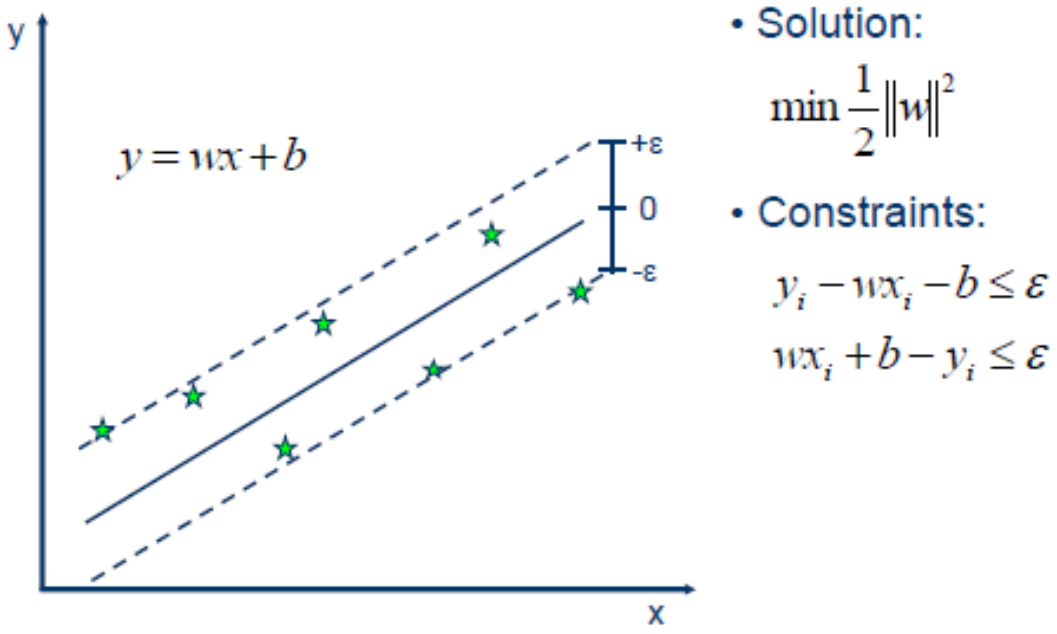

Figure 5 - Support Vector Regression graph. Retrieved from https://www.saedsayad.com/images/SVR_1.png

The SVR models utilize a loss function, which penalizes the high and low misestimates equally. This makes it a symmetrical loss function. (Kuhn, M., & Johnson, K., 2013, Awad, M., & Khanna R., 2015) In linear regression one bad sample residual can affect the model parameters immensely, whereas in SVR this issue is tackled as the cost function squares only the residuals that fall below a given threshold value. Furthermore, SVR uses the absolute residual value when the residual is above the threshold value. (Kuhn, M., & Johnson, K., 2013, Hastie, T., Tibshirani, R., & Friedman, J. 2017) The difference between the optimal fits that the Linear Regression and Support Vector Regression find can be seen in figure 6. This also means that if the given threshold value is large enough, only the outlier samples affect the model parameters. This might seem a bit counter intuitive compared to linear regression models, but this method has been shown to be quite successful when building an SVR model. (Kuhn, M., & Johnson, K., 2013) This is because the training and test data set are assumed to follow independent and identical distributions, which means they are following the same probability function. (Kuhn, M., & Johnson, K., 2013, Awad, M., & Khanna R., 2015)

In SVR models there are as many describing parameters as there are data points, which could be thought as over parameterized. In fact, the cost function doesn't penalize the smaller residuals which gives the model good generalization properties overall. With Support Vector Machines one needs to decide which kernel to use, which also depends on the problem. A kernel is a mathematical function, that takes the input data and transforms it into the form that is required. (Kuhn, M., & Johnson, K., 2013) There are several kernels to use in Support Vector Machines, such as radial basis function (RBF), linear kernel and polynomial kernel. There is also the regularization parameter C, which tells the support vector machine how much we want to penalize for miss classifying or predicting our outcome. Therefore, the higher the C, the lower the rate for wrong predictions. Besides the parameter C there is the parameter Gamma, which determines how far does the influence of a single data point reach in the given problem. If the gamma value is high, only the closest data points will be considered in the hyperplane and vice versa.

The SVR model also needs the data points used in the training dataset to calculate new predictions. This means that the larger the training dataset is, the less compact the SVR algorithm is compared to other machine learning algorithms. There's no need to panic though, as some of the parameters for different datapoints are almost zero, so those parameters will not be utilized in the prediction calculation as only the parameters that are far

beyond zero are needed. These datapoint parameters define the regression line of the SVR model, thus they are the support vectors of the predicted regression line. (Kuhn, M., & Johnson, K., 2013)

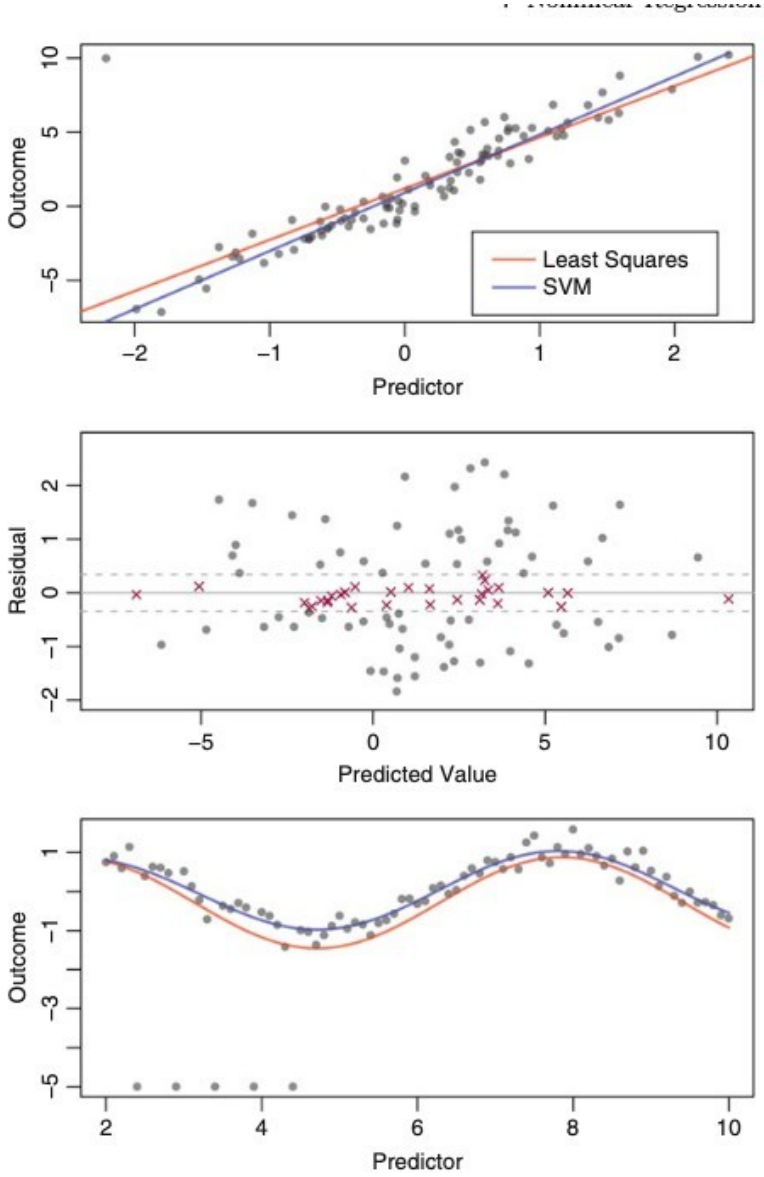

Figure 6 - ted values mapped or ·s (shown as grey colorec l lines are ± ε

2.2.4 Regr

Regression t lecision tree category. Decision trees are considered as nonparametric methods, where one doesn't assume any parametric form for the classes and the form of the tree is not fixed prior the model building. Thus, they can be utilized in classification as well as in regression problems. The decision trees are models where the local region is found by splitting the data in small number of steps. (Alpaydin, E., 2014) The steps can be seen in figure 7. The splitting process is done until the a given criterion is minimized, which could make the tree really deep. This can be avoided by giving the deepness a threshold value, where the splitting process ends when the threshold value is reached. (Kuhn, M., & Johnson, K., 2013)

```
if Predictor A >= 1.7 then
|   if Predictor B >= 202.1 then Outcome = 1.3
|   else Outcome = 5.6
else Outcome = 2.5
```

Figure 7 - Simple regression tree with if statements. Figure from Kuhn, M., & Johnson, K. (2013)

A decision tree constitutes from decision nodes and terminal leaves, where each decision node is a test function that defines an outcome that later label the branches. (Figure 8 and Alpaydin, E., 2014) This means that for every input the test function is applied, and the outcome determines which branch it goes to. The process begins from the bottom, or the root, and it is run until the outcome runs into a leaf node, where the value in the leaf is the output. (Kuhn, M., & Johnson, K., 2013)

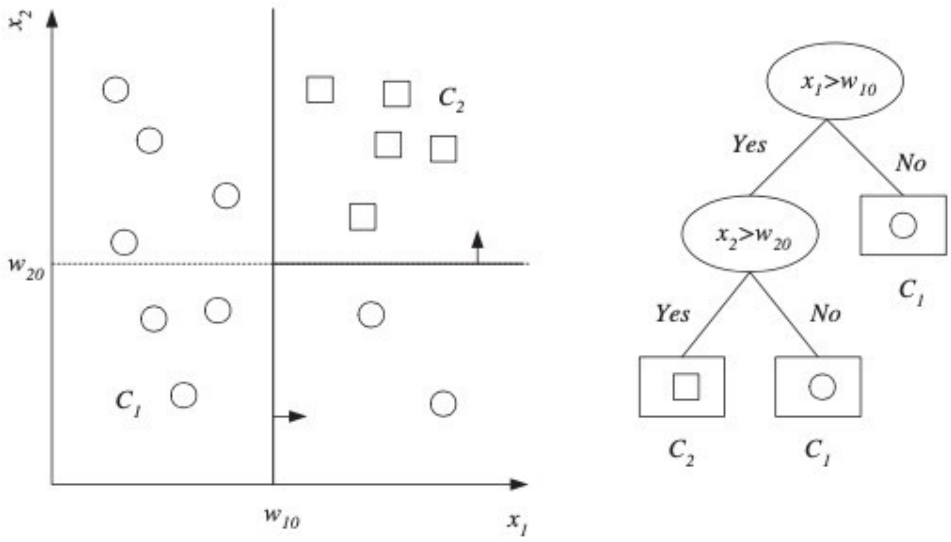

Figure 8 - Dataset and corresponding decision tree. Figure from Alpaydin, E. (2014

Deciding the root for the tree is done by calculating a threshold value for each predictor and then calculating the SSR for each predictor, finally choosing the one with the lowest SSR. From there the tree is grown by doing this for each predictor and finally the lowest SSR is chosen. A limitation for this method exists though, as each terminal node uses the average of the training outcomes for the prediction value, which means that the model doesn't predict well the values that are really low or high (Kuhn, M., & Johnson, K., 2013)

As mentioned before, the deepness of the regression tree can be controlled by giving it a threshold value that the model doesn't exceed. This should be done because the regression and classification trees tend to overfit the training data. The threshold value can be assigned by pruning. Pruning means the removal of badly performing branches within the decision tree. (Kuhn, M., & Johnson, K., 2013, Alpaydin, E., 2014)

One way to prune is to decide a value for splitting a node, for example 20 observations, no matter how big the error is. This is because the existence of too few observations creates variance and makes the generalization process inaccurate. (Alpaydin, E., 2014) Another method is called cost-complexity pruning, which means that a full-grown tree will be pruned with a function that determines which branches will be removed. (Hastie, T., Tibshirani, R., & Friedman, J. 2017) The function can be simplified as:

$$Tree\ \ Score = \ \ Sum\ \ of\ \ Squared\ \ Residuals + \ n* \ n$$

where the idea is to calculate the subtree that minimizes the tree score with tuning parameter α. The tuning parameter describes the tradeoff that the model has between the depth of the tree and how well the tree fits the data. (Hastie, T., Tibshirani, R., & Friedman, J. 2017) The tuning parameter is then increased until pruning leaves gives a lower tree score than the current tree, and it is continued until a single-node tree is left. Then cross-validation is done to calculate the mean value for the tuning parameter, which minimizes the sum of squared residuals.

There are two different times to do pruning: pre-pruning and post-pruning. In the first option one doesn't let the tree grow to its full size. Post-pruning is the opposite and more accurate method, where the tree is grown to its full size, and the pure and overfitting subtrees are removed (i.e., where there is no training error)

2.2.5 Random Forest

Random forest is a technique that tries to avoid the issue of variance in decision trees and where bagging, also known bootstrap aggregating, methods are utilized. Bagging is a method

where a bootstrapped dataset is created from the original dataset, and an unpruned tree is trained from this dataset. The trees are also built by choosing random predictor variables from the data set and the trees are trained using an individual bootstrapped dataset for every decision tree. This step is then iterated as many times as is seen fit and a forest of those trees is created. The forest constitutes from a number of de-correlated decision trees and the average of the prediction outputs is used in the prediction process. (Kuhn, M., & Johnson, K., 2013, Hastie, T., Tibshirani, R., & Friedman, J. 2017)

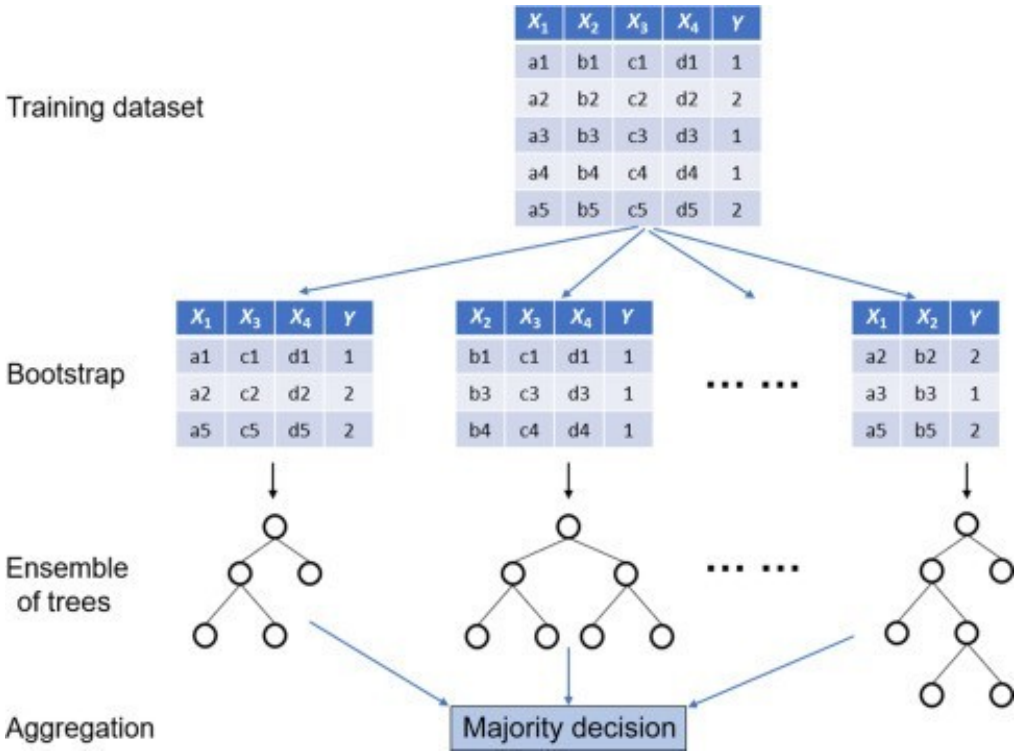

Figure 9 - Random Forest process well explained. Starting from the dataset and bootstrapping datasets for every individual decision tree. Figure from: https://ars.els-cdn.com/content/image/3-s2.0-B9780128177365000090-f09-17-9780128177365.jpg?_

The method of a Random Forest can be seen in figure 9. One important parameter that needs to be considered in the random forest model is the amount of the randomly selected features in each of the decision trees. The second parameter is the size of individual node and when to stop dividing the nodes. Hastie, T., Tibshirani, R., & Friedman, J. (2017) state that the number of randomly selected variables could be the number of features divided by three ($p/3$), but the minimum node size should be 5. Exceptions exist in random forests as well, but these a good starting point for the model building and good general initial guidelines to follow. The third important parameter is the size of the forest, i.e., how many trees should be

included in the forest model. The random forest algorithm doesn't have the same tendency of over-fitting as do the single decision tree models, so growing the number of individual trees in the model do not affect the accuracy of the model. Kuhn, M., & Johnson, K., (2013) suggest starting from 1000 initial trees, and fixing the model iteratively from that point on.

2.3 Handling missing data

In a perfect world all data would contain as much information as possible, and in a way that the data would be ready to be used straight away. As the world is not perfect, one needs to pre-process data before it is usable for problem solving. Kuhn, M., & Johnson, K., (2013) state that one of the most common issues with data are missing datapoints within the dataset. This means that a datapoint is completely missing, or null. Missing data could also mean that it's defined as zero, but there is no chance that this datapoint could ever take any other value than 0. If this is the case, it means that the data is structurally missing.

One important factor to consider first is the reason why there are missing values within the dataset and whether or not those missing datapoints affect the efficiency and accuracy of the analysis. Both Hastie et al. (2017) and Kuhn & Johnson (2013) give an example of drug testing and the side effects of that drug; if a drug has major side effects and taking measurements were impossible at the time, this unfortunate event would lead to missing data. These missing data points would also affect the outcome of the analysis and the model; therefore, they would include informative missingness. Another example of informative missingness would be customer ratings; people tend to rate products and services when they either hate them or love them, but not when their opinions are neutral. (Kuhn, M., & Johnson, K., 2013)

Data can also be censored, which should not be mixed with missing data. Censored data means the case where one doesn't have an exact observation, but they know that the data can't be below or above some threshold value. Consider a case in sales: when a salesperson presents an offer to the customer, and the customer has not approved or disapproved the offer. The salesperson does not have an ending date for the offer, but they know that the offer has been valid from the date that the offer was presented until today at least. (Kuhn, M., & Johnson, K., 2013) Censored data should be either thought as missing data in the

preprocessing phase, or to use the threshold value that one knows the censored data value is at the minimum.

Handling missing data is an important process and the whole preprocessing phase is more of an art than a science. One of the most common ways to handle missing data is removing the feature that has missing values, though it should be done only if the percentage of those missing values is too big for trustworthy predictions. Thus, the removal can be done if the percentage of those missing values within the predictor is not too big, and one has determined that the missingness of those datapoints is not informative. (Kuhn, M., & Johnson, K., 2013)

If one doesn't want to remove the missing values from the dataset, they can also use a technique called imputation. Imputation is a technique where they would use the other predictors in the dataset to estimate the missing values within the dataset. For example, one could calculate the mean value of group A for the feature X and fill the missing values that belong to the group A with that mean value. Imputing values will create more uncertainty within the model, and also tends to increase the computational time that is required when building the model. Still, the estimates of the model performance will be more honest when doing so. (Kuhn, M., & Johnson, K., 2013) Hastie et al. (2017) suggest that finding the relationship between different features should be done via exploratory data analysis. Hence, if there is a strong correlation between the feature that has missing values and another feature that doesn't have missing values than the model can be thought as giving good predictions without the feature with the missing values.

One other way to compute the missing values is the K-nearest neighbor method, where the nearest samples of missing data point are used, and their average is used to fill the missing value. This method has the advantage of being quite sure that the missing value is indeed within the range of those neighbor values, but the disadvantage is the computational time and the entire dataset needed to compute these values. (Liu et al., 2016)

2.4 Sampling and pre-processing the data

Over-fitting is an issue that has been mentioned several times in this paper, and also some techniques have been introduced to avoid over-fitting and the inability to generalize the results of the prediction model. In short, over-fitting is a situation where besides the patterns

of the data, the model has also picked up and utilized the random noise that each of the datapoint and sample have. This creates issues further down the line when the model is presented some unseen data that the model needs to predict, as the model is a personal model for the training data and not good for generalization. (Alpaydin, E., 20142014) Overfitting can be seen in figure 10, where the model 1 has picked up every class even though they are not even close to the main decision boundary.

In this chapter one should assume that the dataset has been cleaned, and that the data does represent the entire sample population. Selecting the machine learning algorithm that will be used is a decisive process and it depends on the problem at hand as well as the dataset that is available. Alpaydin, E. (2014) introduces the theorem of "no free lunch", which states that there is no best algorithm that solves all problems, but there are many algorithms that work well for certain situations and poorly on others.

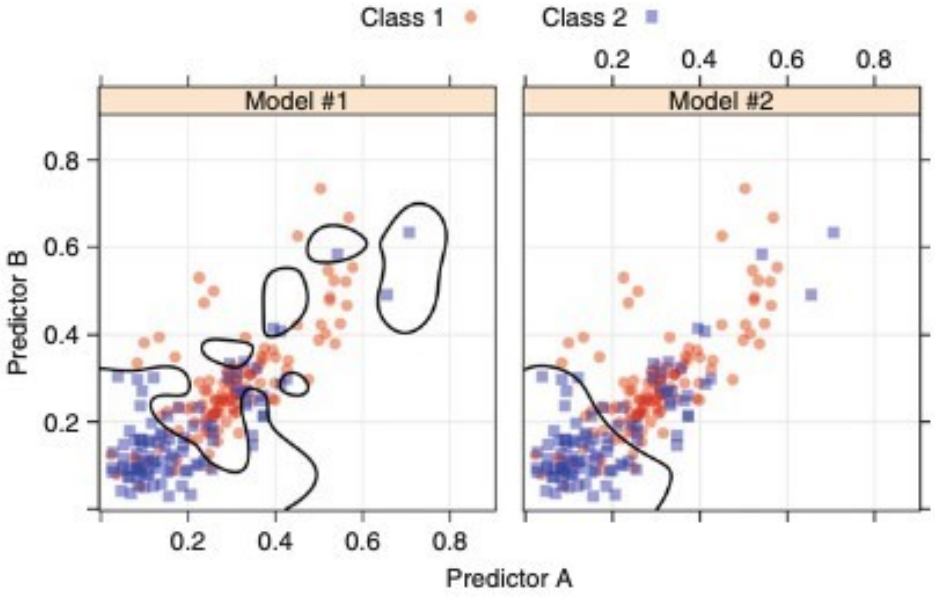

Figure 10 - Over fitting visualization.. Figure from Kuhn, M., & Johnson, K. (2013). Applied predictive modeling (Vol. 26)

Kuhn, M., & Johnson, K. (2013) state the idea that the modeler needs to find the right parameters that produce the most accurate and realistic predictions for the problem at hand. A globally accepted method of model tuning is data splitting, which means the splitting of the data set into a randomly created training dataset, which is used to find the patterns in the data

and trains the model create a model that makes accurate predictions, and into a randomly created test dataset that is used to evaluate the model and its accuracy. This split is usually around 70%-30% and 80%-20%. Alpaydin, E. (2014) also suggests using the training set for cross validation, which means creating several training and test sets for tuning and testing the model. When selecting randomly the testing and training datasets, the modeler can select from several re-sampling techniques to reduce the noise and bias within the dataset. Such re-sampling techniques are k-fold cross validation and bootstrapping. (Kuhn, M., & Johnson, K., 2013, Alpaydin, E., 2014, Hastie et al., 2017)

K-fold cross validation tries to avoid the sampling dilemma by introducing a procedure where the dataset X randomly divided into the same sized sub-datasets. The number of those datasets is where the number of this technique comes from, as the number of datasets created is K. Then the model is trained with the number of datasets minus one, or K – 1, and then tested with the remaining 1 dataset. This process is then repeated until the model is trained and tested with every created dataset being the testing set k-1. (Kuhn, M., & Johnson, K., 2013, Alpaydin, E., 2014) This iterative process can be seen in figure 11, where the fold number (or k) is 5 and the model is trained and tested in with each fold.

Figure 11 - K-fold cross validation on a dataset, where k = 5. Source: https://static.packt-cdn.com/products/9781789617740/graphics/b04c27c5-7e3f-428a-9aa6-bb3ebcd3584c.png

Bootstrapping is a re-sampling technique where multiple samples from a single sample are created. In bootstrapping, random observations are picked for the new samples, but the same observation can be picked for the same new dataset more than once, as the picked observation will not be excluded from the dataset. The model is then trained using the bootstrapped

datasets, and Alpaydin (2014) suggests using the original dataset as a validation set for the model. This method is also emphasized by Kuhn, M. & Johnson, K. (2013). The model can be tested with several bootstrapped datasets and the results of those models will be summarized to find the best results. (Kuhn, M., & Johnson, K., 2013)

2.5 Model Selection & Parameter Tuning

Once the data has been preprocessed and cleaned, it is ready for the next phase which is the model selection and parameter tuning phase. Alpaydin (2014) states that model selection is choosing the right bias and that the aim of machine learning is to predict previously unseen observations and therefore being able to generalize the predictions for new instances. Thus, in the model selection process it is important to remember which kind of output is predicted (i.e., classification or continuous numerical variable).

Almost every model has hyper parameters that should be optimized, which is an iterative process where the algorithm tries to find the best possible combinations of parameters in each iteration. Those combinations can be tested either by doing a grid search, where every possible combination is tested. This is an exhaustive and computationally expensive process, but it can be avoided by doing a randomized search. Randomized search tests a given number of hyper parameter combinations, where the number of combinations can be between 1-N. Finally, the best possible combination is reached for the specific problem. One important thing to note is that no perfect model exists, as every problem is different and therefore the best model for a specific problem needs to be found every time.

2.6 Evaluating the performance and accuracy of the model

The evaluation of the model differs whether the model is classifying the outcome or predicting an infinite numeric outcome. As the author is trying to predict an infinite numeric target variable, i.e., salary costs, he will focus on the evaluation of regression models. In the linear regression chapter both RSE and the R^2-metric were presented.

Root mean square error is a method that is normally used in regression model measurements and it is a method where the errors/residuals are squared and summed, their mean is calculated and finally the square root is taken. One of the advantages of RMSE is that it is the

average of a single observation error, and it has the same unit measurement as the data, thus making the interpretability easier. (Alpaydin, E., 2014, Hastie et al., 2017)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Aplaydin (2014, p. 95) summarizes the RMSE metric as follows: “The value is usually interpreted as either how far (on average) the residuals are from zero or as the average distance between the observed values and the model predictions.”

2.7 Related work

There have been several different studies and research about predicting salaries that companies pay, but these studies mainly revolve around the individual employee. Such research was done by Li et al. (Year not found), where they were trying to predict the salary of a job posting based on the advertisement alone using machine learning and data scraping methods. They also used linear regression as their baseline method and compared the other models to it. The other models used were support vector regressor, logistic regressor, nearest centroid, random forest and K-Neighbors Regressor. What they discovered was that random forest regressor was the most accurate model for their problem.

Bansal et al. (2021) compared how the simple linear regression performed compared to multiple linear regression in predicting both the house prices and salaries of individual employees. What they found out was that in both cases simple linear regression was not as efficient as multiple linear regression as both problems are more complex than a simple linear regression model is able to describe.

Das et al. (2020) also tried to predict the salaries of individual employees with different regression techniques, but they were interested in the development of the salary level of a single employee during his/her career after a certain period of time. Their chosen models were linear and polynomial regressions, and they suggested the use of k-nearest regression in future

Jackman and Reid (Year unknown) used different regression methods to predict the individual job salaries from text descriptions, where they web scraped the job advertisements as their input data. They used mean absolute error as their metric and in their problem the random forest regression outperformed all the other models. The other models they tried were maximum likelihood, lasso regression, neural network and dropout neural network.

Koskinen et al. (2005) were trying to predict the future salaries for individual employees, given pas salaries, to be used in planning purposes; to predict the future pensions to come and the amount of individual risk. This research is interesting as well as it was done for pension insurance companies and provides vital information for our field of work. In this study Koskinen et al. used a basic linear regression model with some model parameter extensions, which can be called mixed models. What they discovered was that the prediction error was much smaller in the middle wage categories than in the high- and low-end categories.

Martín et al. (2018) tried also tried to predict the salaries of individual positions in the IT Job Market with high-dimensional samples, where they had samples with 2000 features. They utilized basic linear regression models to predict the continuous value of the salary as a baseline and compared models such as support vector machines and random forests to that baseline, as well as K-Means clustering to predict the salary range of individual job postings.

Wang et al. (2019) used convolutional neural networks to predict the salaries of individual job advertisements in Japan, where they used web scraping methods to gather the data and natural language processing to create features from those scraped job advertisements. Their issues were mainly with the natural language processing related and they tried to train a single model to be as efficient as possible, rather than finding the best possible model.

The previous studies conducted in a similar field as the topic of this thesis shows that there are several potential models that can be utilized in this kind of a problem and that the models chosen for this thesis were some of the better performing models in the previous studies. The issue with the previous studies was that they were focusing on the individual salaries of an employee, and not the annual salary costs of an entire company. Thus, it is fair to say that this thesis topic is unique as there does not seem to be exactly the same research to be found in the field, but the previously mentioned studies can be used as reference points and the findings in those studies provide insights for this problem as well.

3 Empirical Study

The purpose of this study was to determine the most accurate machine learning models to predict the annual salary costs of Finnish companies. The aim was also to compare the more sophisticated and complex Machine Learning models to a Linear Regression model. After preprocessing the data and keeping the relevant features, the author was left with the following steps:

1. Compile a Linear Regression model and the different Machine Learning models
2. Evaluate and compare the model performances

The data were obtained from Suomen Asiakastieto public database, which consists of data about Finnish companies. The data contained financial and categorical data such as revenue, location, staff size and company form.

3.1 Dataset and preprocessing

The initial step for the study was to discover the most descriptive features for the different machine learning models. As mentioned before, the dataset consists of numerical and categorical data about Finnish companies that are registered as employers in the Finnish Employer Registry. In total, the dataset had 13 columns (table 2) and 115 535 rows, where each row corresponds a single company. First 9 columns were the general company information columns (such as business Id, company name, company form, city) and the rest were financial information related (revenue, annual salary costs and the age of the company). The AgeAsYears-column was calculated when the data were retrieved from the SQL-database, as there was only the founding date of the company. This date value would have been too specific for a machine learning model to interpret. Out of the 115 535 companies, 41 592 were missing the annual salary costs information and those were the companies that motivated the topic of this thesis.

Column	Type
Business_ID	object
Company	object
Company_form	object
Industry_name	object
Revenue	float64
Wages	float64
Staff_size	object
Revenue_class	object
City	object
County	object
Trade_registry_date	object
Founding_date	datetime64
AgeAsYears	float64

Table 2 - An overview of the datasets columns and their data types

At first there was an intention to use the numerical revenue values to predict the annual salary costs, as the revenue and annual salary costs had a correlation of 0.62 (or after log10-transformation 0.81, see figure 12 below). Unfortunately, out of the 41 592 companies that were missing the annual salary cost data, 39 989 were also missing the revenue data. Hence, using the numerical revenue as a feature would not be feasible in a real-life situation and therefore the numerical revenue column was dropped from the final model. The AgeAsYears-column was also removed from the final models, as the correlation was not strong enough (0.32). The removal of the numerical revenue features meant that the models would not have any numerical features as predictors. Hence, meaning that all of the features would be categorical.

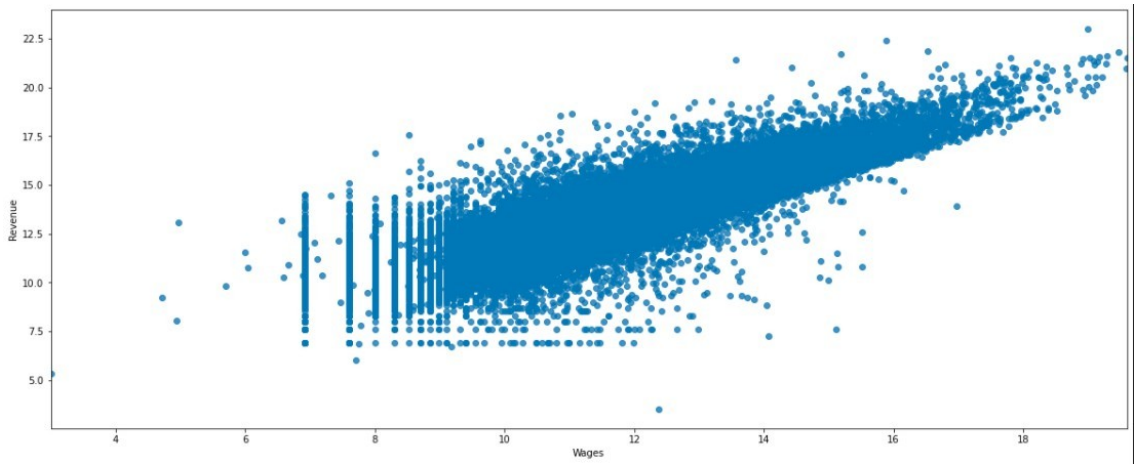

Figure 12 - Log10(Wages) and Log10(Revenue) relationship

The dataset contained companies from the absolute smallest self-employed people all the way to the multinational corporations. Therefore, the dataset was vastly skewed to the right, meaning that there were more smaller companies in the dataset. The skewness has a tendency to affect the model performances, so the skewness needed to be addressed before building and comparing the models. The distribution of the different sized companies can be seen in figure 13 and 14, where the frequencies of the categorical features staff size and revenue class are visualized.

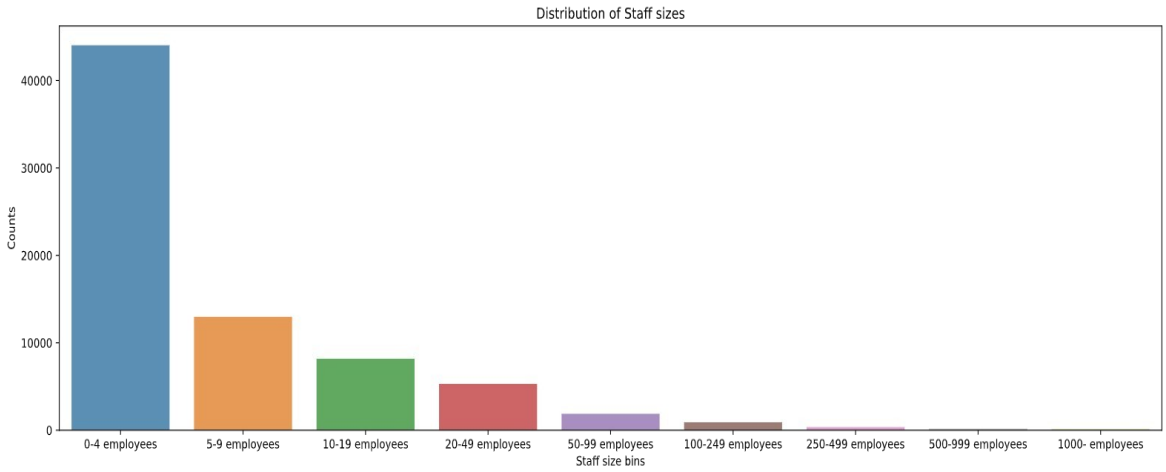

Figure 13 - Distribution of companies by the staff size category

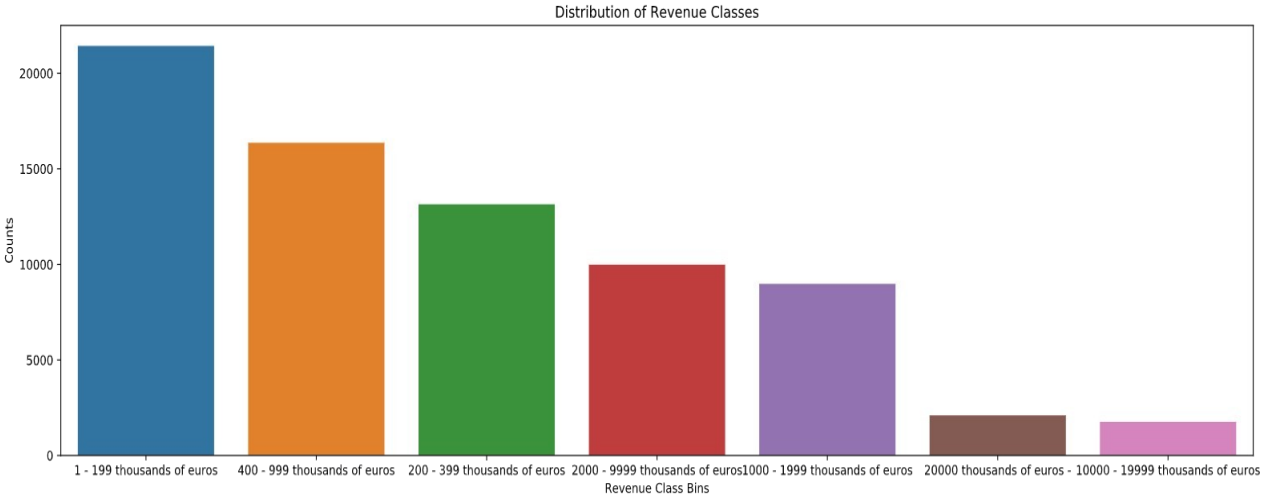

Figure 14 - Distribution of companies by the revenue class

Furthermore, the dataset needed some basic data cleaning, such as transforming some of the annual salary cost values from negative to positive. Finally, some outliers existed in the annual salary cost column, and these outliers were removed. The removing was done by finding the outlier values from each staff size-category, which were either below the 5% or above the 95% quantile. For example, the staff size category "0-4 employees" had a 5% quantile of 8000€ and a 95% quantile of ~192 000€, so every annual salary cost value smaller than 8000€ and larger than 192 000€ in that staff size category were removed from the dataset.

After all the preprocessing and data transformations were finished, some more exploratory data analysis needed to be done so that the author would understand the data and the relationships between the features and the target variable better. As the author was dealing with only categorical variables as possible features, the features needed to be plotted against the target variable. The staff size feature was studied first, and in figure 15 we can see that staff size has an effect on the annual salary cost, as the average annual salary costs increase as the staff size category increases.

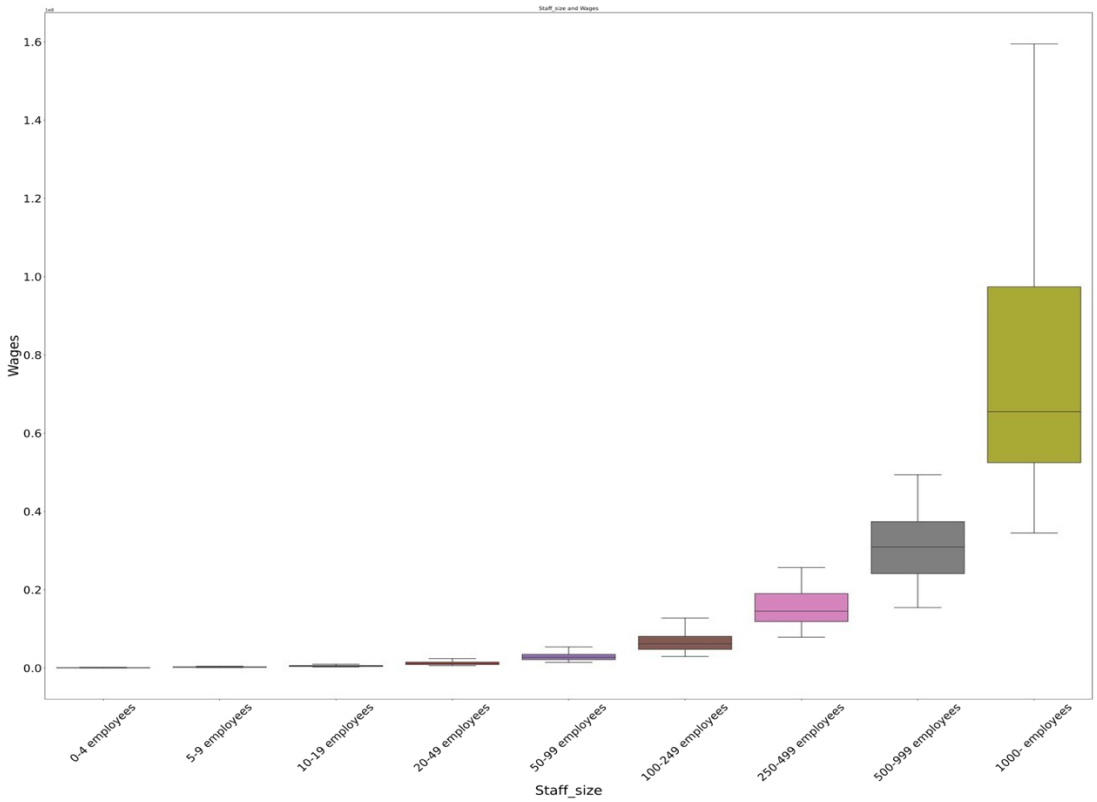

Figure 15 - Staff size and wages

As previously mentioned, the correlation between the revenue and annual salary costs was 0.63. Thus, author wanted to confirm that the fairly strong correlation can be noticed in the categorical bins of the revenue class as well. Figure 16 shows that the relationship is also visible in the categorical bins of the revenue class, as the average annual salary costs increase along the revenue class increasing.

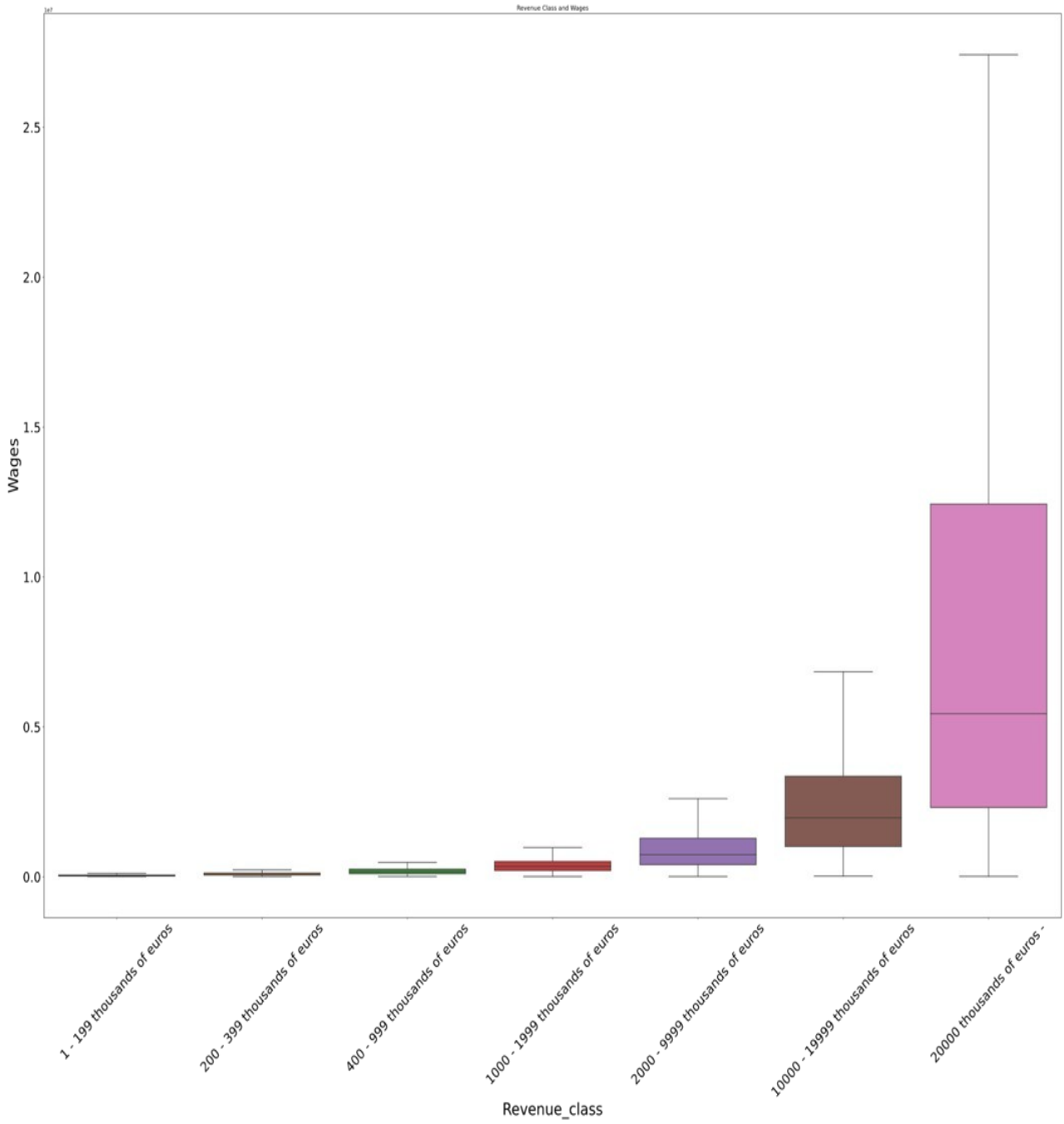

Figure 16 - Revenue class and wages

Figure 17 explores the relationship of industry and annual salary costs in different staff sizes. In this figure it is visible that some industries have larger annual salary costs than others within the same staff size category. The figure also proposes that the industry feature could be useful in the model building phase.

The same can be seen in figure 18, where the distribution of annual salary costs are visualized in the context of county and staff size, i.e., some counties have higher average wages in the same staff size class as other counties. Lastly, the relationship between company form and wages was investigated. Figure 19 shows there might be a relationship between these two variables as well. In the end, the final features selected for the models were company form, industry, county, staff size and revenue class. These categorical features were transformed into dummy variables, which means assigning a 0 or a 1 depending on if the observation belongs to a given category. The final feature set can be seen in table 3, where the features are represented in their dummy form. Furthermore, when creating the dummy variables, the first categories in every feature were dropped to avoid the so called "dummy variable trap", which is the multicollinearity of two or more dummy features.

#	Feature	Type
1.	Company_form_AYH	int
2.	Company_form_KOY	int
3.	Company_form_KY	int
4.	Company_form_OK	int
5.	Company_form_OY	int
6.	Company_form_OYJ	int
7.	Company_form_SL	int
8.	Company_form_SÄÄ	int
9.	Company_form_VLL	int
10.	Company_form_YEH	int
11.	Industry_name_Hallinto- ja tukipalvelutoiminta	int
12.	Industry_name_Informaatio ja viestintä	int
13.	Industry_name_Julkinen hallinto ja maanpuolustus	int
14.	Industry_name_Kaivostoiminta ja louhinta	int
15.	Industry_name_Kansainvälisten organisaatioiden ja toimielinten toiminta	int
16.	Industry_name_Kiinteistöalan toiminta	int
17.	Industry_name_Koulutus	int
18.	Industry_name_Kuljetus ja varastointi	int
19.	Industry_name_Maatalous, metsätalous ja kalatalous	int
20.	Industry_name_Majoitus- ja ravitsemistoiminta	int
21.	Industry_name_Muu palvelutoiminta	int
22.	Industry_name_Rahoitus- ja vakuutustoiminta	int
23.	Industry_name_Rakentaminen	int
24.	Industry_name_Sähkö-, kaasu- ja lämpöhuolto, jäähdytysliiketoiminta	int
25.	Industry_name_Taiteet, viihde ja virkistys	int
26.	Industry_name_Teollisuus	int
27.	Industry_name_Terveys- ja sosiaalipalvelut	int
28.	Industry_name_Toimiala tuntematon	int

29.	Industry_name_Tukku- ja vähittäiskauppa	int
30.	Industry_name_Vesihuolto, viemäri- ja jätevesihuolto, jätehuolto ja muu ympäristön puhtaanapito	int
31.	Staff_size_5-9 employees	int
32.	Staff_size_10-19 employees	int
33.	Staff_size_20-49 employees	int
34.	Staff_size_50-99 employees	int
35.	Staff_size_100-249 employees	int
36.	Staff_size_250-499 employees	int
37.	Staff_size_500-999 employees	int
38.	Staff_size_1000- employees	int
39.	County_Etelä-Karjalan maakunta	int
40.	County_Etelä-Pohjanmaan maakunta	int
41.	County_Etelä-Savon maakunta	int
42.	County_Kainuun maakunta	int
43.	County_Kanta-Hämeen maakunta	int
44.	County_Keski-Pohjanmaan maakunta	int
45.	County_Keski-Suomen maakunta	int
46.	County_Kymenlaakson maakunta	int
47.	County_Lapin maakunta	int
48.	County_Pirkanmaan maakunta	int
49.	County_Pohjanmaan maakunta	int
50.	County_Pohjois-Karjalan maakunta	int
51.	County_Pohjois-Pohjanmaan maakunta	int
52.	County_Pohjois-Savon maakunta	int
53.	County_Päijät-Hämeen maakunta	int
54.	County_Satakunnan maakunta	int
55.	County_Uudenmaan maakunta	int
56.	County_Varsinais-Suomen maakunta	int
57.	Revenue_class_200 - 399 thousand euros	int
58.	Revenue_class_400 - 999 thousand euros	int
59.	Revenue_class_1000 - 1999 thousand euros	int
60.	Revenue_class_2000 - 9999 thousand euros	int
61.	Revenue_class_10000 - 19999 thousand euros	int
62.	Revenue_class_20000 – thousand euros	int

Table 3 - The Features in their Dummy-form

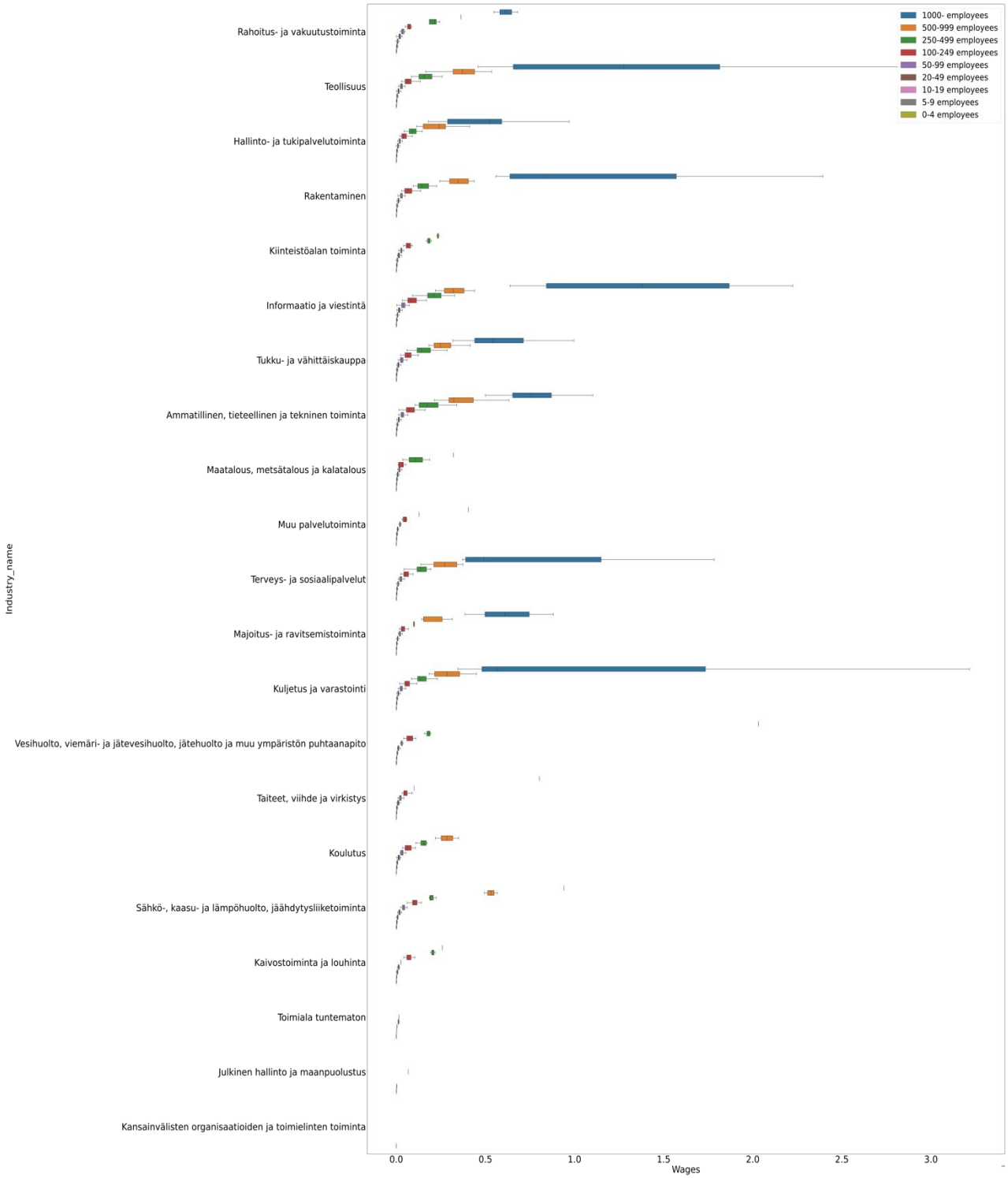

Figure 17 – The relationship between Industry, Staff Size and Annual Salary Costs

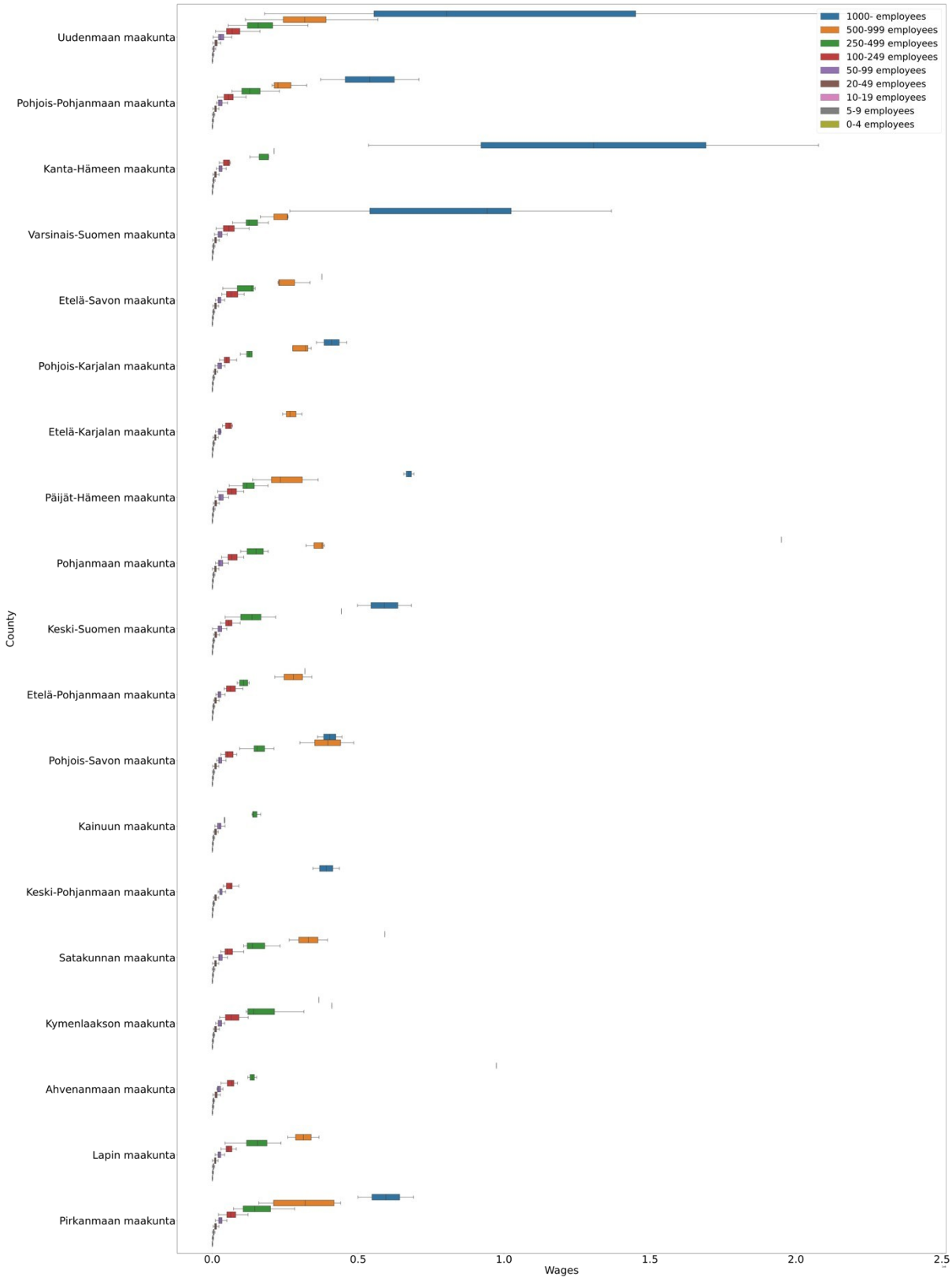

Figure 18 – The relationship between County, Staff Size and Annual Salary Costs

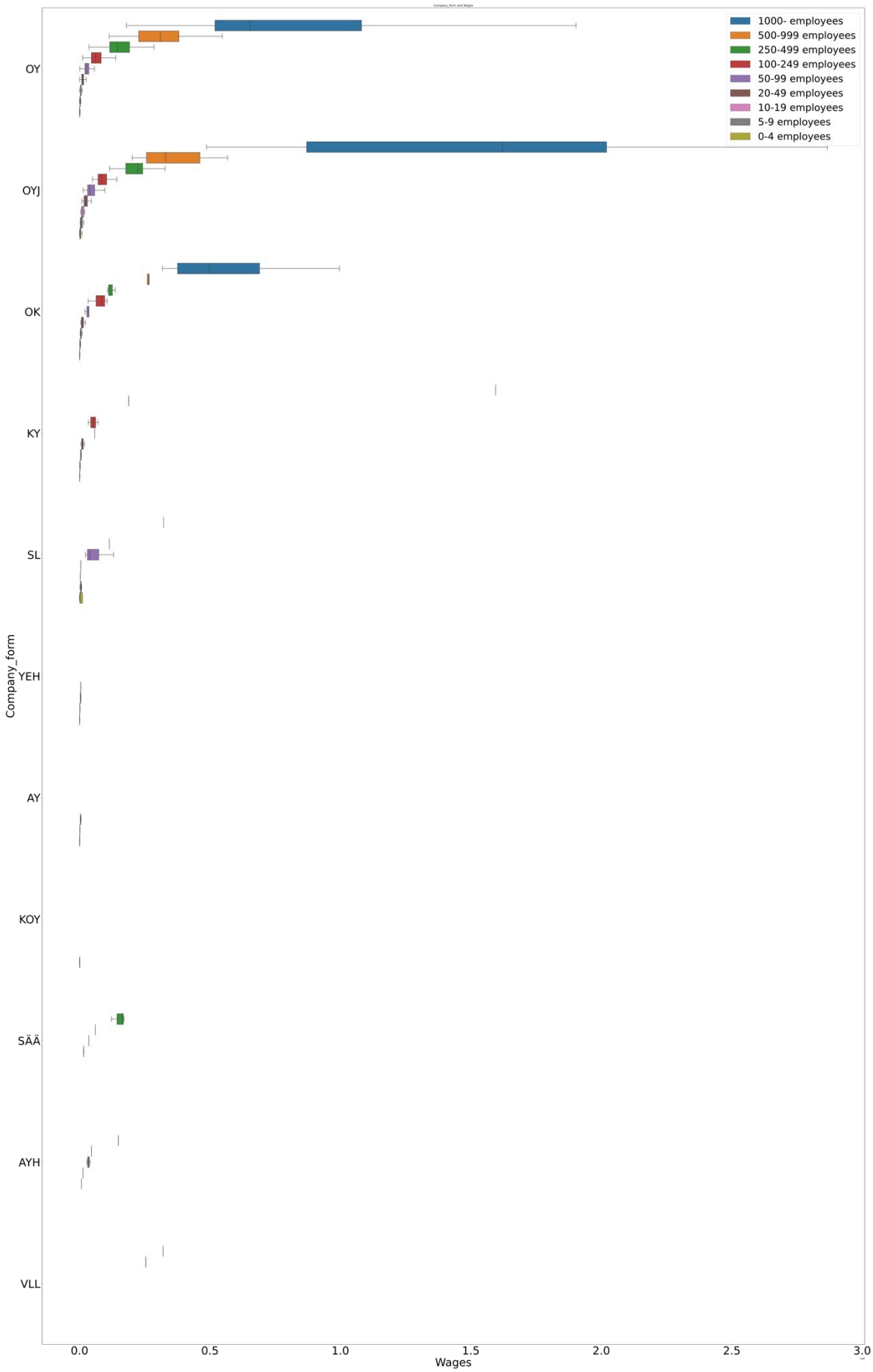

Figure 19 – The relationship between Company Form, Staff Size and Annual Salary Costs

Finally, the annual salary costs column was transformed into its logarithm base 10 form. The transformation was necessary so that the support vector machines, and neural networks wouldn't produce poor results, as especially support vector machines might be negatively affected by values that are in different scales. The effect of the transformation can be seen in figures 20 and 21, where the distribution of the annual salary costs transforms from highly right skewed to a decently normal distribution.

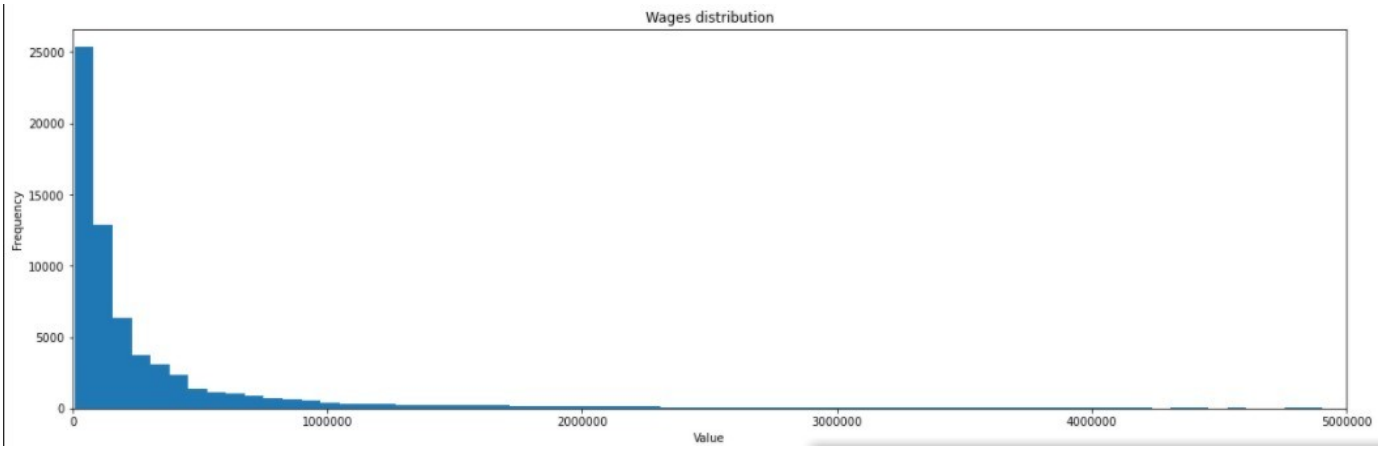

Figure 20 – Annual salary cost distribution before log-transformation

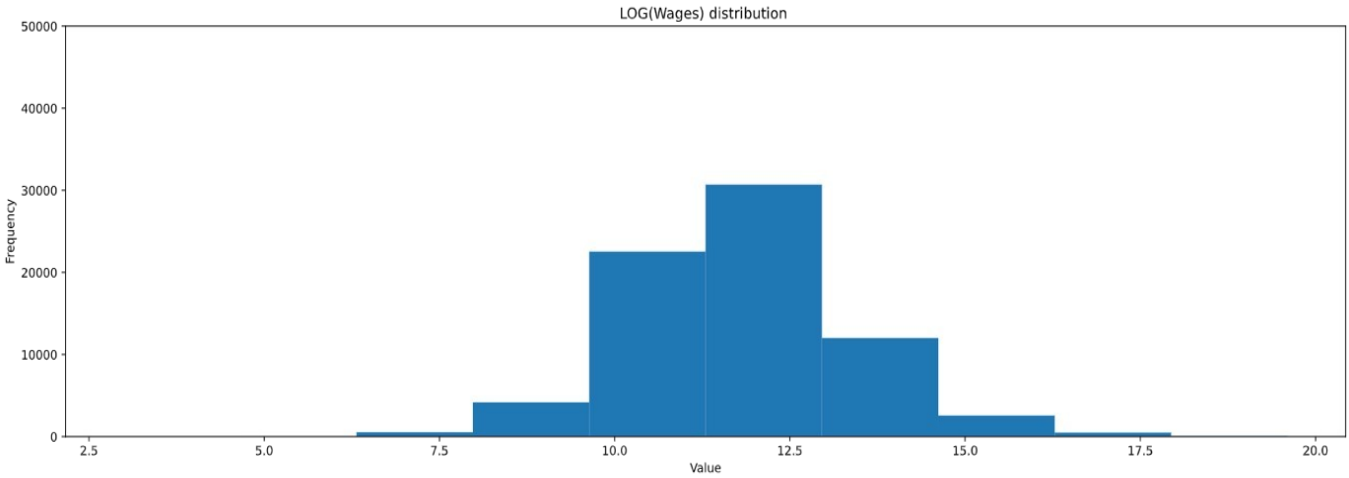

Figure 21 - Annual salary cost distribution after log-transformation

When calculating the final R^2-score and the root mean square error for each model, the predicted values and the actual values were transformed back to their normal form, so the interpretability of those metrics was easier and made sense in the given problem.

3.2 Linear Regression model and the Machine Learning algorithms in Python

For this study the programming language Python was chosen, as it is a language the author is the most familiar with. Python also has a vast number of libraries, that are easy to understand and efficient. The linear regression model was fit using the LinearRegression.fit() function from the scikit-learn – package:

$$estimator = LinearRegression.fit(x, y)$$

This formula tells the program the features and the target variable that it needs to include in the model, as the formula is $y = \beta 0 + \beta 1X1 + n2n2 + \cdots + \beta iXi + \varepsilon$ where y is the target variable and X's are the features. The scikit-learn package was also used for the Decision Tree, the Random Forest and Support Vector Regression algorithms and those algorithms take the same input as the Linear Regression function.

$$estimator = DecisionTreeRegressor.fit(x, y)$$

$$estimator = RandomForestRegressor.fit(x, y)$$

$$estimator = SupportVectorRegressor.fit(x, y)$$

For the Artificial Neural Network, the Keras Tensorflow package was used. With Keras Tensorflow, the user is responsible for defining every layer of the neural network, the number of neurons in those layers and the activation functions used in each of these layers. The sample code below shows an example of how a Keras Tensorflow ANN was compiled with Python.

```
model = Sequential()
model.add(Dense(input_dim, input_shape=(input_dim, ), activation='relu'))
model.add(Dense(32, input_shape=(input_dim, ), activation='relu'))


# Output layer
model.add(Dense(1, activation='relu'))

#  Compile model
optimizer=Adam(lr=0.001)

model.compile(loss='mean_squared_error',
          optimizer=optimizer,
          metrics=['mean_squared_error'])

model.fit(x_train,y_train,validation_data=(x_test,y_test),
    verbose=0, epochs=50, batch_size=64)


pred = model.predict(x_test)
pred = np.array(pred)
y_test = np.array(y_test)

#Transform back to normal values
pred = 10**pred
y_test = 10**y_test

#Print metrics
print(f" (R2): ",metrics.r2_score(y_test, pred))
print("RMSE: %.2f" % math.sqrt(metrics.mean_squared_error(y_test, pred)))
```

3.3 K-Fold Cross-Validation with Python

As mentioned in earlier chapters (e.g., 2.1.1.3 & 2.1.2.1), over- and underfitting are serious issues that one is trying to avoid in machine learning models, as those create issues in the generalization of the model. In order for a model to avoid under- or overfitting, some re-sampling methods might need to be used. In this thesis, the K-Fold Cross-Validation with 5 folds was used as a re-sampling method, as the size of the dataset is so large that using more folds would be computationally too expensive and inefficient. K-fold cross-validation means dividing the dataset into K-minus-1 folds, These K-minus-1 folds will be used for training and the last fold for testing. The process will then be repeated so that each fold is used for testing at least once. For each fold the R^2 statistic was calculated and the mean of those R^2 scores was taken, so that the overall score of the model can be assessed. Below is a sample

code of how the K-Fold Cross-Validation method was utilized for the Linear Regression model.

```
kf = KFold(n_splits=5, shuffle = True, random_state=42)

new_y = []
new_pred = []

fold = 0
for train_index, test_index in kf.split(x,y):
        print("TRAIN:", train_index, "TEST:", test_index)
        x_train, x_test = x.iloc[train_index], x.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
        fold+=1
        print(f"Fold #{fold}")

        regression = LinearRegression()

        regression.fit(x_train, y_train)

        pred = regression.predict(x_test)
        pred = np.array(pred)
        y_test = np.array(y_test)

        pred = 10**pred
        y_test = 10**y_test

        new_y.append(y_test)
        new_pred.append(pred)

        # Measure this fold's R2
        print(f"Fold score (R2): ",metrics.r2_score(y_test, pred))

new_y = np.concatenate(new_y)
new_pred = np.concatenate(new_pred)

print(f"Final, out of sample score (R2): ",metrics.r2_score(new_y, new_pred))
print("RMSE for Linear Regression: %.2f" % math.sqrt(metrics.mean_squared_error(new_y, new_pred)))
```

3.4 Machine Learning Model Parameter Tuning

After the data was preprocessed and cleaned, the best possible accuracy and performance for the different chosen machine learning models was tried to achieve. In order for achieving the best possible accuracies, the hyper parameters needed to be tuned for the different models. Finding the best possible hyper parameter combinations can be done manually by building individual models with different parameters, or by writing a piece of code that does it for you. In this study, the RandomizedSearchCV-package in Python was used, which is simpler than manually finding the best parameters. The RandomizedSearchCV-package takes different

hyper parameters as input and assigns randomly the chosen number of given parameter combinations. Finally, all of those combinations are given to the model, so that the best possible combination for each model can be found and used for predicting the target variable.

For each hyper parameter combination of a given model, a K-Fold Cross-Validation was used to calculate the mean of R^2-score and the root mean squared error of every fold. Finally, the best possible hyper parameter combinations for each of the models were identified. In Table 4 below, the best hyper parameter combinations for all of the models can be seen, along with the R^2-scores and root mean square errors of those combinations.

All of the models have a different set of hyper parameters, some of which have already been presented in earlier chapters (such as NN's neurons, hidden layers and learning rate, as well as SVR's gamma, c and kernels). Starting with neural network and its hyper parameters: epochs define how many times the entire dataset is fed forward and backward through the neural network model. The entire dataset can't be passed through the neural network model at once, so the dataset needs to be divided into batches. (Brownlee, 2016)
The batch size hyper parameter defines how many training samples are in a single batch. Optimizer parameter defines the algorithm that is used to change the attributes of the neural network, and they are solving the optimization problem by minimizing the function that is coded in the optimizer. (Brownlee, 2016)

The hyper parameters for a decision tree regressor follow the tree structure of the model, where the max_depth-parameter determines the maximum depth of the tree. From scikit-learn library's documentation we see that "if None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples." (scikit-learn.org) Min_samples_split-parameter tells the model the minimum number of samples required to split an internal node of the tree, whereas min_samples_leaf tells the minimum number of samples required for it to be at a leaf node. Criterion determines the criteria with which an individual node is split. Finally, the max_leaf_nodes determine the maximum number of leaf nodes in the model.

Random forest regressor has similar hyper parameters as the decision tree regressor, as it built from several decision trees. The n_estimators parameter tells the number of decision

trees in the random forest regressor model. The rest of the hyper parameters are the same as they are in the decision tree models, as they control the single decision trees in the model.

MODEL	OPTIMAL PARAMETERS	R^2 - SCORE	ROOT MEAN SQUARE ERROR
Linear regression		0.793	1,915,840€
Neural network	{Input dimension: 63, Hidden layer 1: 32 neurons, Learning rate: 0.001, Epochs: 50, Batch size: 64, Optimizer: Adam}	0.791	1,942,252€
Random forest	{'n_estimators': 100, 'min_samples_split': 8, 'min_samples_leaf': 1, 'max_leaf_nodes': None, ' max_depth': None, 'criterion': 'mse'}	0.795	1,835,503€
Decision tree	{'criterion': 'mse', 'max_depth': 8, 'max_leaf_nodes': 100, 'min_samples_leaf': 40, 'min_samples_split': 10}	0.779	2,006,212€
Support Vector Regression	{'kernel': 'linear', 'gamma': 0.01, 'C': 1}	0.787	1,969,923€

Table 4 - The optimal hyper parameters for each model

The highest scores were achieved with Random Forest Regressor, with a R^2-score of 0.795 and a root mean square error of 1,835,503€. The Random Forest model barely beat the linear regression model, that had a R^2-score of 0.793 and a root mean square error of 1,915,840€. The Neural Network with the tuned hyper parameters had a R^2- score of 0.791 and a root mean square error of 1,942,252€. The fourth and fifth highest R^2- scores were Support Vector Regression and Decision Tree Regression, with the R^2-scores of 0.787 and 0.779 and a root mean square errors of 1,969,923€ and 2,006,212€, respectively. The score for an untuned Random Forest was 0.73, with a root mean square error 2,011,232€, and an untuned Neural Network model had the R^2-score of 0.05. Thus, this shows that hyperparameter tuning is important in order for finding the most suitable model for a given problem and trying to tackle the bias-variance tradeoff.

3.5 Analyzing the results

3.5.1 Random Forest

After an exhaustive period of tuning the hyper parameters and finding the most accurate model, the next important step was to delve deeper into to the results and see what information can be withdrawn from the model's performance and its features. Therefore, an intriguing part was to see what features were the most important in the Random Forest model as predictors. From figure 22 it can be noted that by far the most important features were the revenue class categories, followed by the staff size categories. The importance of revenue was previously noticed in the exploratory data analysis, when studying the correlations between the features and the target variable. Surprisingly, the different industry and county features were not that important as predictors.

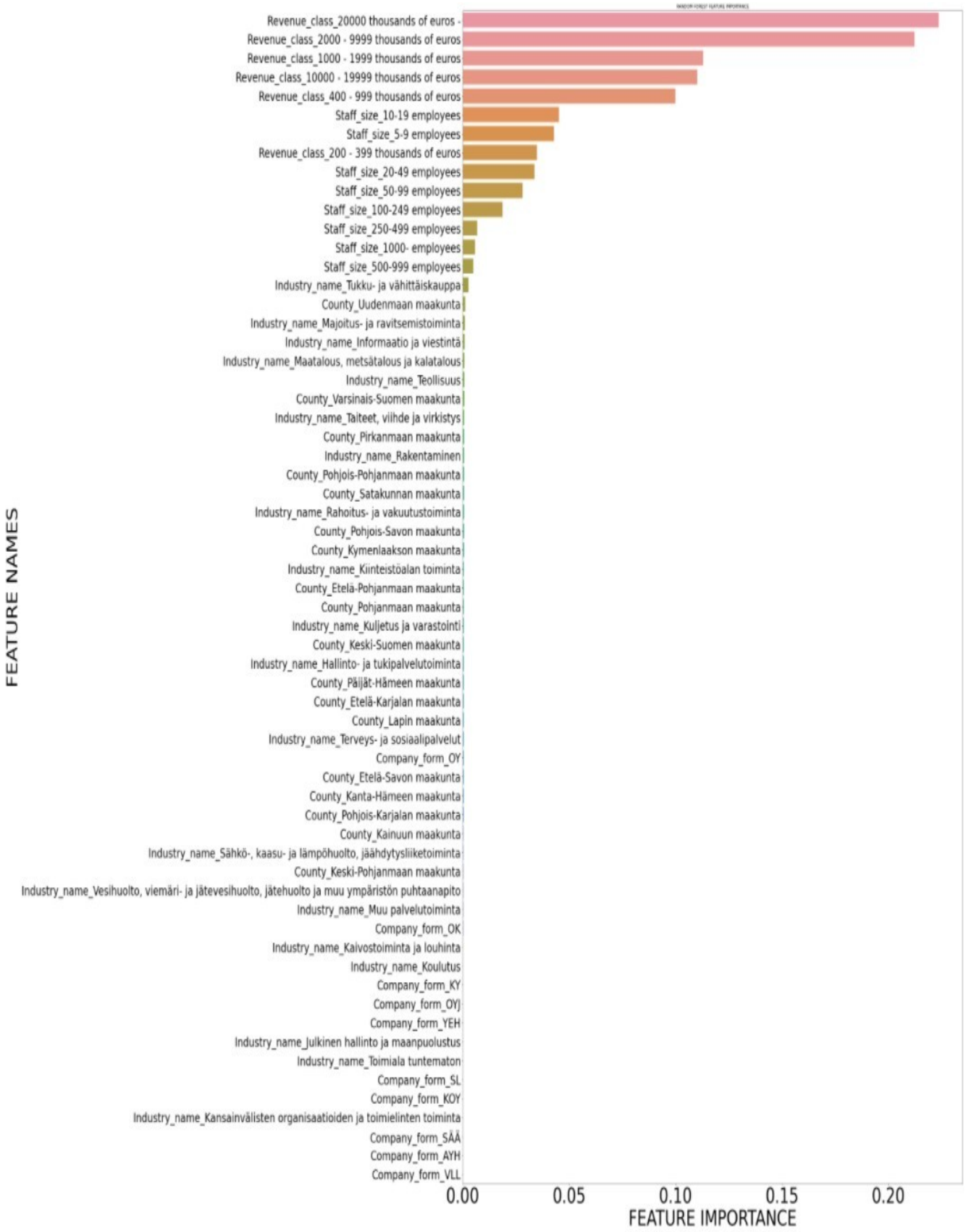

Figure 22 - Feature importance with all the selected features

Figure 23 plots the predicted values of the Random Forest model against the actual values. This plot seems to follow a decently linear pattern. Looking at graph, it is noticeable that the test set had more smaller companies and that the predictions in those companies tend to be further from the actual value. The opposite is true for the bigger companies, which are closer to the optimal line.

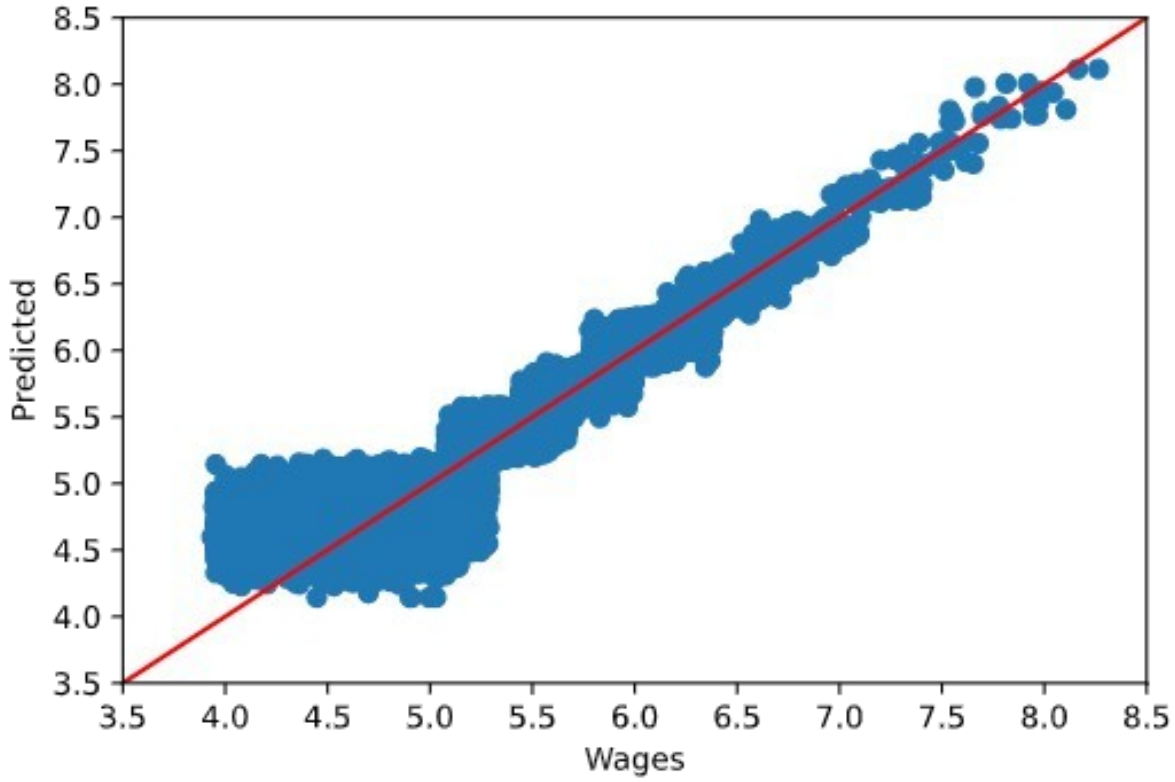

Figure 23 - Random Forest models log10(Predicted values) vs log10(actual values)

Table 5 shows how well the Random Forest regressor performed in the different revenue class categories, so that it can be determined if some categories are better predicted and explained by the Random Forest model. The reason for this calculation was that the author and his employer had an interest to know if there were certain revenue or staff size classes that the models were struggling with, or performing exceptionally well in. Below is a sample code that shows how those metrics were calculated per each staff size group. The revenue class calculations were achieved by changing the value in the “group by”-function to ‘revenue class’.

```
y_hats_df = pd.DataFrame( data = pred,
                                    index = x_test.index.copy()
                                   )

df_out = pd.merge(df,
                          y_hats_df,
                          how = 'left',
                          left_index = True,
                          right_index = True)

df_out.dropna( inplace=True )

df_out['Prediction_error'] = df_out['Wages'] - df_out[0]

def r2_rmse( g ):
   r2 = metrics.r2_score( g['Wages'], g[0] )
   rmse = np.sqrt( metrics.mean_squared_error( g['Wages'], g[0] ) )
   return pd.Series( dict(  r2 = r2, rmse = rmse ) )

df_out.groupby( 'Staff_size' ).apply( r2_rmse ).reset_index()
```

From table 5 it can be noticed that the Random Forest model describes the relationships between the different revenue class categories and the target variable well, at least according to the R^2-score. The model has some difficulties in the second smallest class, and it is interesting that the RMSE is lower in this class compared to the smaller class, even though the R^2-score is lower. From table 5 we can see that the RMSE increases as the revenue class size increases, but also the annual salary costs increase in this category as well. If we look at the figure 23, we see that the larger companies' predictions are closer to the optimal line, even though the RMSE increases. This is due to the logarithmic scale.

Revenue_class	R^2	RMSE
1 - 199 thousand euros	0.880	47,803
200 - 399 thousand euros	0.492	43,532
400 - 999 thousand euros	0.741	65,942
1 000 – 1 999 thousand euros	0.799	117,343
2 000 – 9 999 thousand euros	0.836	353,468
10 000 – 19 999 thousand euros	0.843	857,129
20 000 thousand euros -	0.868	7,684,170

Table 5 – Random Forest and Revenue class

Table 6 shows the R^2-score and the root mean squared error for the different staff size categories. The RMSE rises as the staff size rises, which seems normal as the wages also increase when the staff size increases. Thus, this could mean that the relative error might not be as big as the absolute error. Looking at the table 6, it can be seen that the random forest model explains decently the variance in all of the staff sizes, except the 250-499 and the 500-999 employees' class, where it explains the variance close to how a mean line would.

Staff_size	R^2	RMSE
0-4 employees	0.255	38,947
5-9 employees	0.214	71,622
10-19 employees	0.271	143,167
20-49 employees	0.179	390,141
50-99 employees	0.228	772,148
100-249 employees	0.355	1,950,947
250-499 employees	0.016	4,636,712
500-999 employees	0.089	8,068,708
1000- employees	0.432	28,346,190

Table 6 - Random Forest and Staff Size class

3.5.2 Linear Regression

After studying the most accurate and the most descriptive model, the performances of the other models were then explored. This meant exploring how the other models performed in predicting the annual salary costs overall, and how much of the variance did they explain in the different staff size and revenue class categories. Figure 24 shows that the linear regression predictions follow a somewhat linear line, when plotted against the actual values. The model seems to have the same issues in the smaller companies as did the Random Forest model, i.e., some predictions fall far away from the optimal line. Some outliers also exist in the medium sized companies, where the model underpredicts the salary costs. Overall, the predictions fall close to the optimal line of the actual observations.

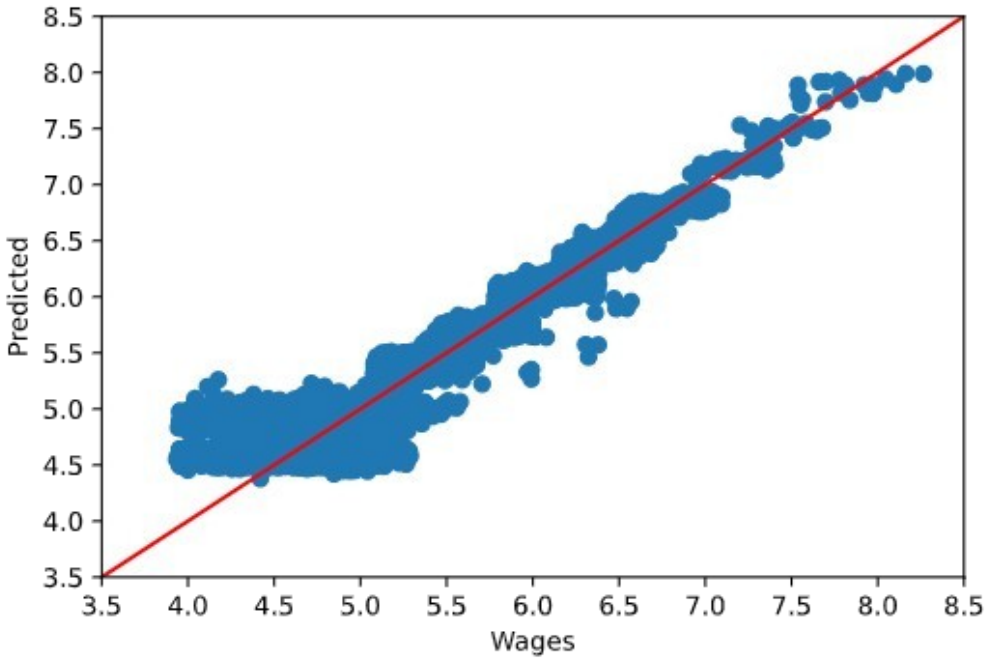

Figure 24 - log10(Predicted values) vs log10(actual values), linear regression

Table 7 shows that the linear regression model performs well in the larger revenue classes but has issues in the smaller categories. This supports the findings that were emphasized in figure 24. Table 8 shows that linear regression performs evenly in the different staff size classes, except the second biggest one, where the model explains the variance almost as well as a mean line would. Figures 25 and 26 are visual representations of the lowest and highest fits of the revenue classes.

Revenue_class	R^2	RMSE
1 - 199 thousand euros	0.382	108,643
200 - 399 thousand euros	0.488	43,731
400 - 999 thousand euros	0.731	67,139
1 000 – 1 999 thousand euros	0.775	124,084
2000 - 9999 thousand euros	0.830	359,298
10 000 – 19 999 thousand euros	0.811	939,293
20 000 thousand euros -	0.841	8,440,330

Table 7 - Linear Regression and Revenue Class

Staff_size	R^2	RMSE
0-4 employees	0.255	38,929
5-9 employees	0.180	73,151
10-19 employees	0.216	148,527
20-49 employees	0.167	392,872.
50-99 employees	0.159	805,831
100-249 employees	0.265	2,083,550
250-499 employees	0.271	3,990,915
500-999 employees	0.070	8,154,306
1000- employees	0.282	3,186,4443

Table 8 - Linear Regression and Staff Size class

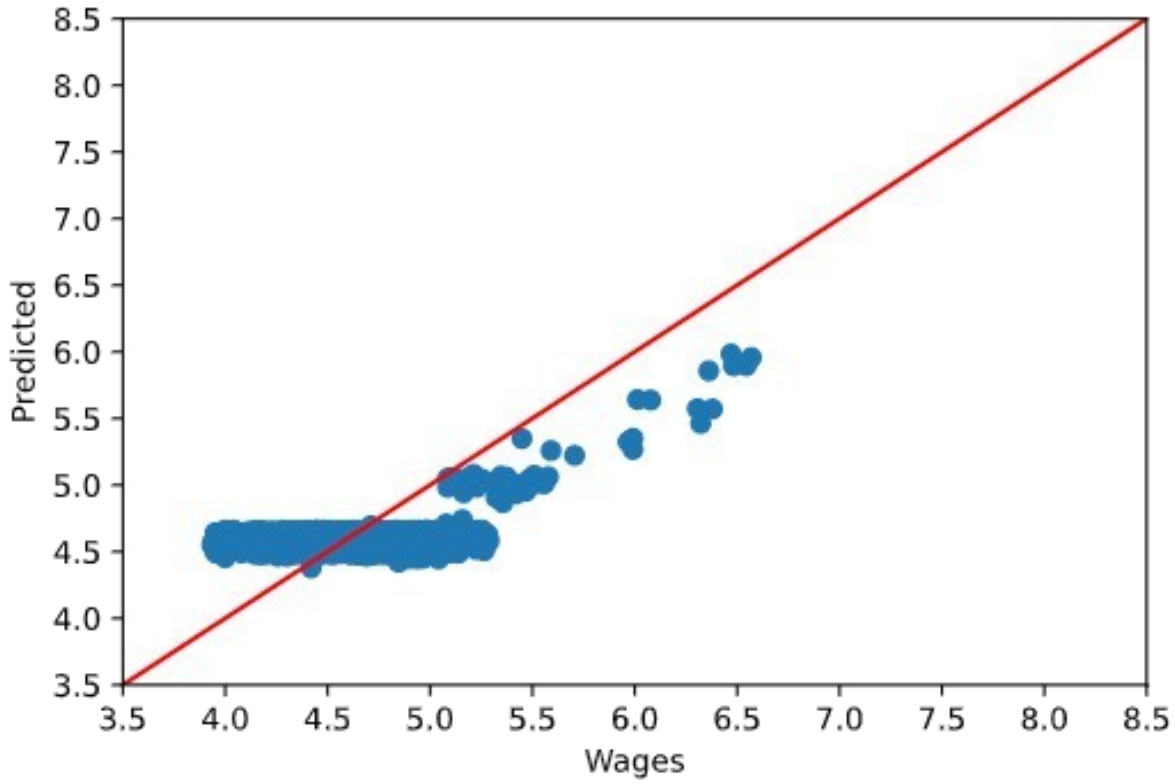

Figure 25 - The fit for companies in the class “1-199 thousand euros”

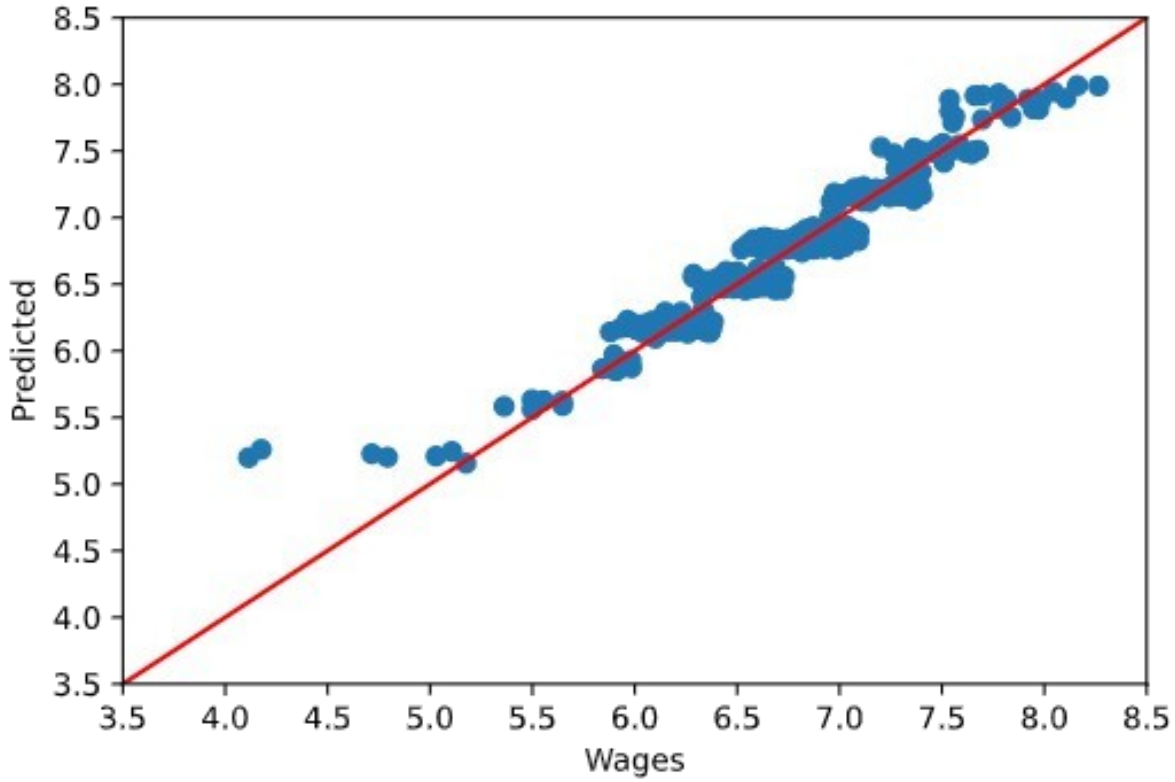

Figure 26 - The fit for companies in the revenue class 2000 - thousand euros

Looking at the residual plot in figure 27, we can see that the log(residual) distribution of the linear regression model is normal, which is good as Montgomery et al. (2001, p.17) state that "residuals play a key role in evaluating the model adequacy." As previously mentioned in chapter 2.1.1.1, residuals are the difference between the actual value and the predicted value (residual = actual value – predicted value), and it can be considered "as the deviation between data and the fit and the variability in the response variable not explained by the regression model". (Montgomery et al. 2001, p. 130) Figure 28 shows that the residuals follow a somewhat random pattern, where there is a slight variance decrease when the annual salary cost prediction increases.

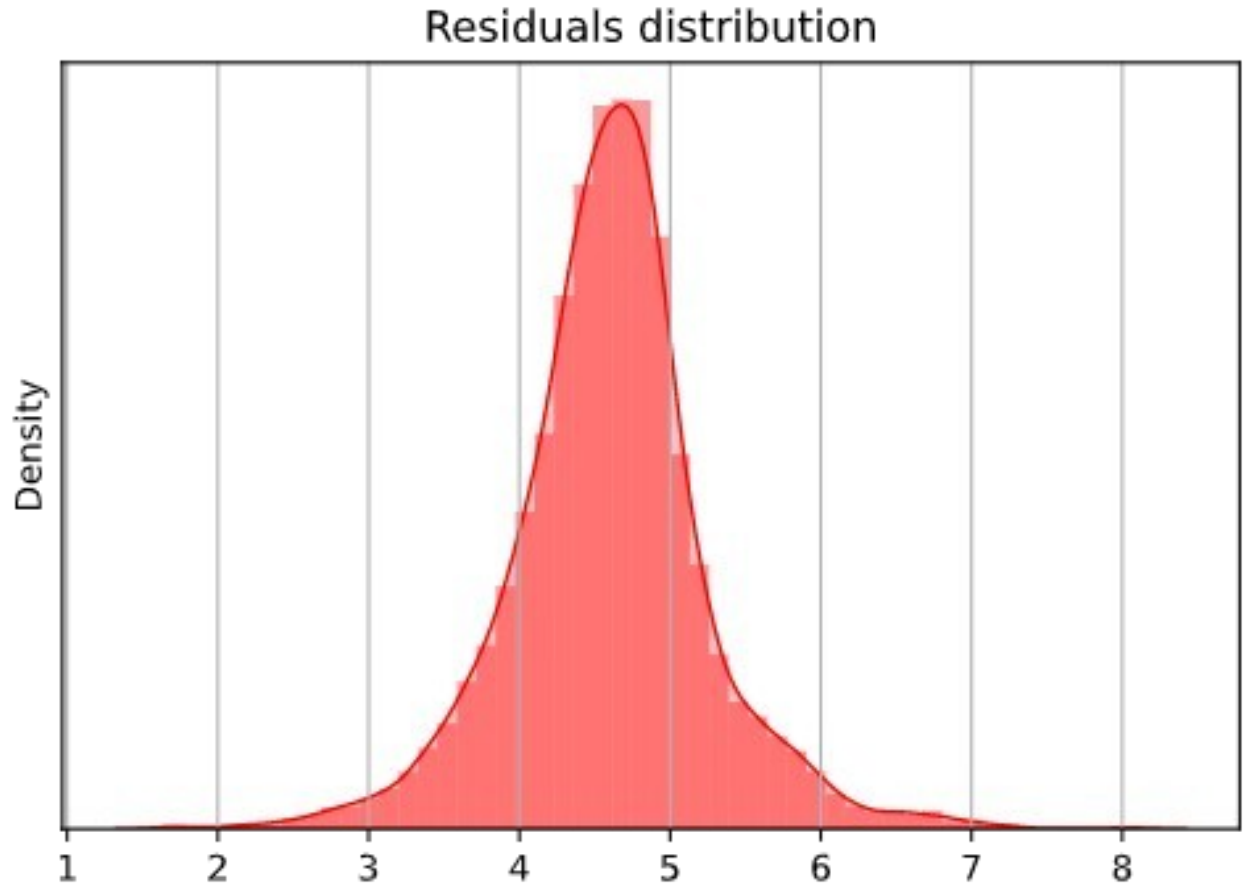

Figure 27 - log10(Residua)l distribution of the Linear Regression model

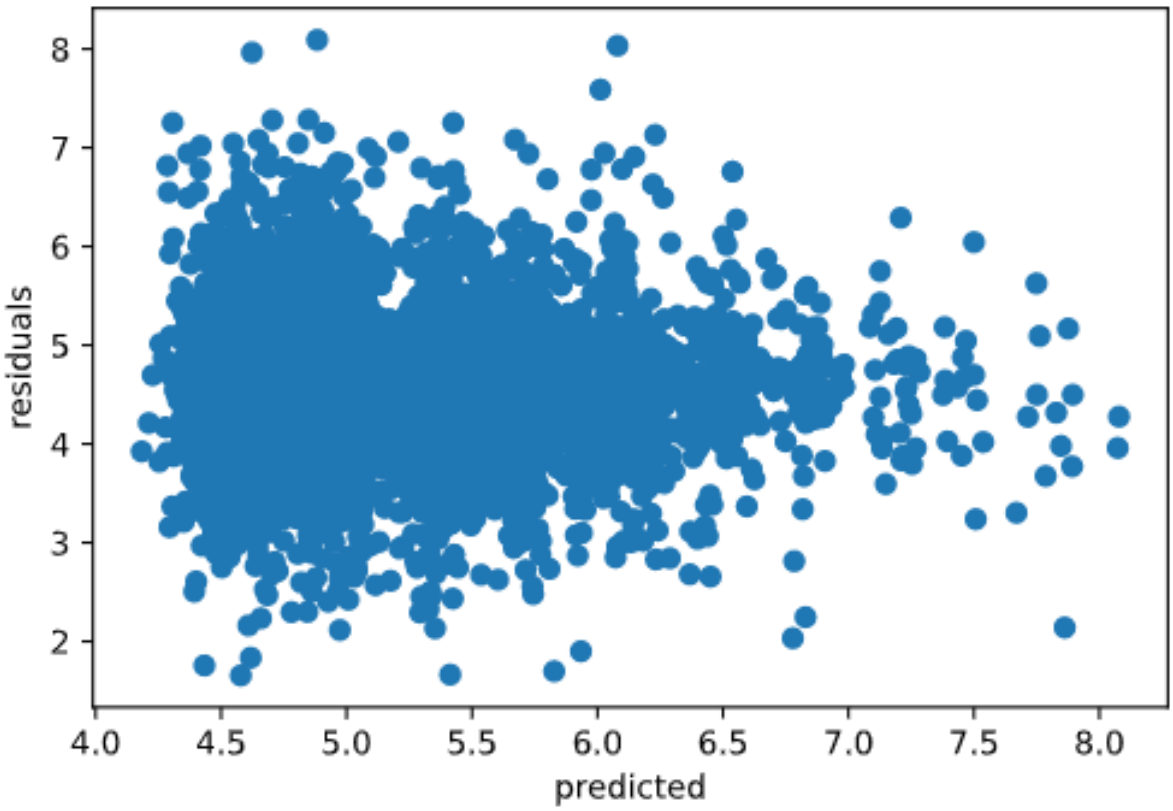

Figure 28 - Residual distribution in the overall model with log10(predictions) and log10(residuals)

3.5.3 Neural Network

Figure 29 shows how the Neural Network with 1 hidden layer performed overall. The model performs fairly well, with the same issues in the smaller companies as the previous models, but the linearity is clearly visible in the graph and the trend is similar to the previous models. Some outliers exist in the smaller companies especially, and in the larger companies some under predicting can be seen.

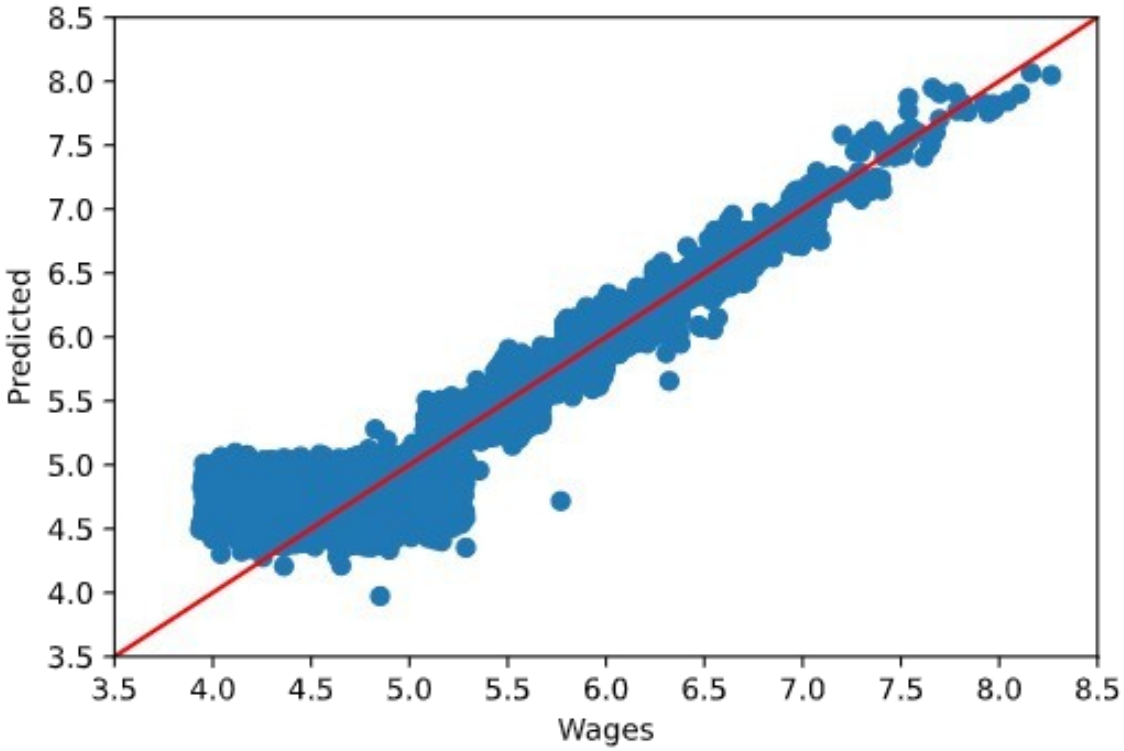

Figure 29 - Neural Network predicted values plotted against the actual values

Table 9 shows that the Neural Network performs well explaining the variance across the revenue classes, and especially well in the biggest revenue class. The model also has the smallest RMSE in the biggest class, which supports the good R^2-score. Table 10 shows that the neural network works decently in the smaller staff sizes, exceptionally well in the largest staff size and poorly in the second largest.

Revenue_class	R^2	RMSE
1 - 199 thousand euros	0.783	64,323
200 - 399 thousand euros	0.505	42,985
400 - 999 thousand euros	0.736	66,483
1 000 – 1 999 thousand euros	0.777	123,714
2 000 – 9 999 thousand euros	0.822	367,459.
10 000 – 19 999 thousand euros	0.808	946,716
20 000 thousand euros -	0.900	6,666,216

Table 9 - Neural Network and Revenue Class

Staff_size	R^2	RMSE
0-4 employees	0.283	38,208
5-9 employees	0.205	72,001
10-19 employees	0.230	147,146.
20-49 employees	0.125	402,832
50-99 employees	0.227	772,677.
100-249 employees	0.223	2,141,057
250-499 employees	0.106	4,420,089
500-999 employees	-0.177	9,174,543
1000- employees	0.613	23,391,049

Table 10 - Neural Network and Staff Size

3.5.4 Decision Tree

Figure 30 shows that there are visually noticeable clusters in the predictions of the Decision Tree model, as the Decision Tree model tends to find a mean value of some groups and assigns the predictions according to those means. This limitation was also mentioned in chapter 2, where the Decision Trees tendency of mean predictions was mentioned. There it was also mentioned that the decision tree algorithms tend to struggle with the smallest and largest values and their predictions. The number of outliers is also vastly greater in Decision Tree models predictions than in any other model, especially in the smaller companies.

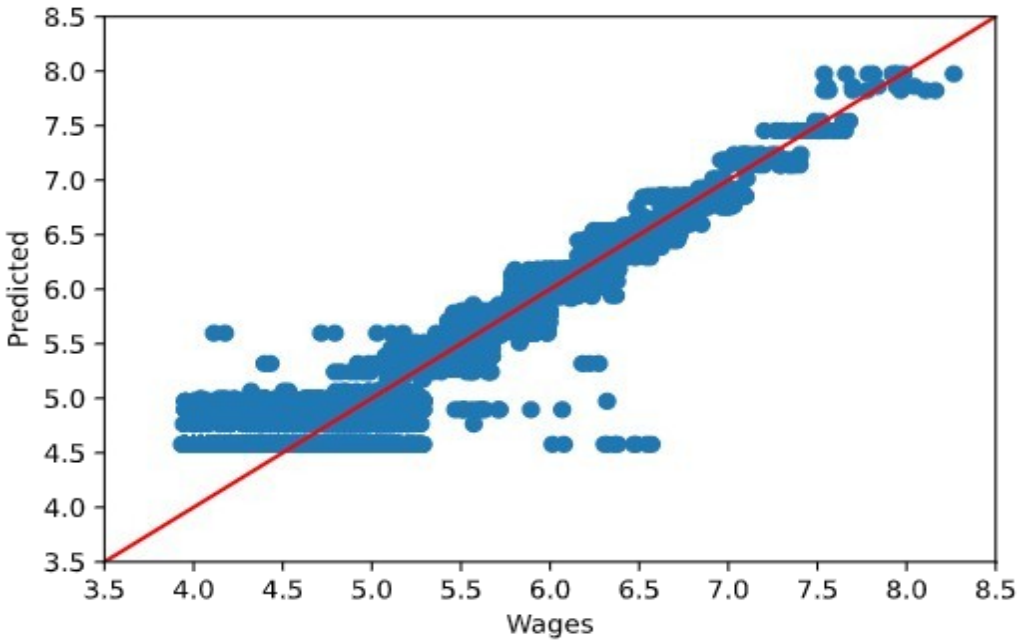

Figure 30 - Decision Tree predictions plotted against the actual values

Table 11 proves that the decision tree model has some serious issues in the smaller revenue class, where it explains the variance almost as poorly as a mean line would. This can be seen in figure 30, where the predictions seem to be some mean values of a group. The RMSE of that class is also the biggest across all the models. As the revenue class increases, the Decision Tree models variance explanation quality increases as well, with the RMSE following the same trend with the other models.

Revenue_class	R^2	RMSE
1 - 199 thousand euros	0.031	136,058
200 - 399 thousand euros	0.219	54,006
400 - 999 thousand euros	0.723	68,181
1 000 – 1 999 thousand euros	0.694	145,003
2 000 – 9 999 thousand euros	0.831	357,877
10 000 – 19 999 thousand euros	0.827	900,099
20 000 thousand euros -	0.784	9,829,409

Table 11 - Decision Tree and Revenue Class

Table 12 shows that the Decision Tree models quality varies vastly in the staff size class, as the model performs as well as the other models, but in some it explains the variance almost as poorly as a mean line would. The RMSE are also a bit higher than they were in the other models.

Staff_size	R^2	RMSE
0-4 employees	0.205	40,216
5-9 employees	0.165	73,826
10-19 employees	0.167	153,098
20-49 employees	0.156	395,530
50-99 employees	0.009	874,900
100-249 employees	0.218	2,148,315
250-499 employees	0.073	4,501,353
500-999 employees	0.222	7,457,731
1000- employees	-0.006	37,752,478

Table 12 - Decision Tree and Staff size Class

3.5.5 Support Vector Regression

Support Vector Regression has a similar trend as the other models, which can be seen in figure 31. Similar issues are prevalent in the small company predictions, where the predictions fall on both sides of the optimal line. Furthermore, the SVR has more outlier predictions in the medium sized companies than the other models, and the SVR model seems to have tendency to under predict the medium and large companies.

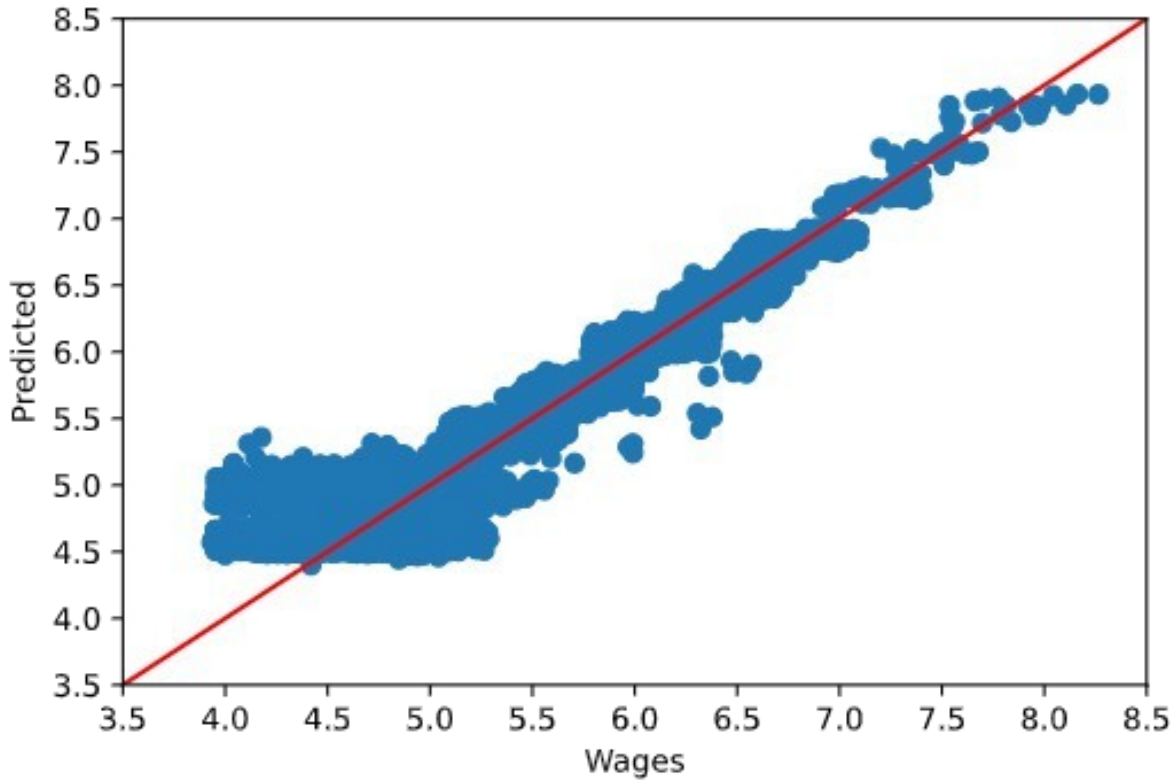

Figure 31 - Support Vector Regressor overall predictions plotted against actual observed

The Support Vector Regression explains the variance in the larger revenue classes as well as the other models, and just like the other models it has difficulties in the smaller classes. The RMSE in the smallest revenue class is the second highest across all the models, which is worth noting. Compared to the revenue classes, the Support Vector Regressor explains the variance best in the smallest staff size category, and the worst in the biggest two categories. In the smallest staff size SVR has the lowest RMSE, but it isn't able to explain the variance so well.

Revenue_class	R^2	RMSE
1 - 199 thousand euros	0.342	112,129
200 - 399 thousand euros	0.507	42,877
400 - 999 thousand euros	0.750	64,730
1 000 – 1 999 thousand euros	0.787	120,773
2 000 – 9 999 thousand euros	0.834	354,956
10 000 – 19 999 thousand euros	0.820	917,564
20 000 thousand euros -	0.818	9,015,439

Table 13 - Support Vector Regression and Revenue Class

Staff_size	R^2	RMSE
0-4 employees	0.283	38,199
5-9 employees	0.188	72,773
10-19 employees	0.238	146,463
20-49 employees	0.174	391,434
50-99 employees	0.177	797,428
100-249 employees	0.272	2,072,681
250-499 employees	0.253	4,039,207
500-999 employees	0.050	8,240,612
1000- employees	0.169	34,291,263

Table 14 - Support Vector Regression and Staff Size class

Overall, all the models found a somewhat linear pattern in their predictions when looking at the predicted values plotted against the actual values. Some models, such as SVR and Decision Tree, had more outlier predictions than others, which could be caused by the somewhat poor quality of the data. With more numeric features, the prediction quality could be improved. Especially the Decision Tree model could most likely get rid of the clusters of mean predictions, as it would have more values to split a node.

The quality of the data could be improved by enriching the data with the actual revenue values of the companies, in the case of the companies that are missing the revenue value in

the Suomen Asiakastieto dataset. By enriching this data with the actual numeric revenue values, the prediction quality would improve, as is shown in figure 32. This figure shows how Linear Regression models' predictions improve with the use of the numerical revenue feature. The predicted values are closer to each other and the optimal prediction line. This is also shown by the accuracy metrics, as the mean R^2 score of the Linear Regression model is 0.93, with a RMSE of 872,652€. This means that the prediction accuracy improves by 1 million euros in the case of the Linear Regression model. Also, finding and removing the outliers in the revenue feature could be more practical with the numerical revenue feature. Looking back, the author could have studied the frequency of the different revenue classes in the different staff size categories, so that the outliers could have been removed. Furthermore, the quality of the staff size data could be enriched by obtaining the actual number of staff that the companies have. This could also improve the performance of the different models, as the relationship of the staff size feature and the target variable would be more defined and easier to calculate.

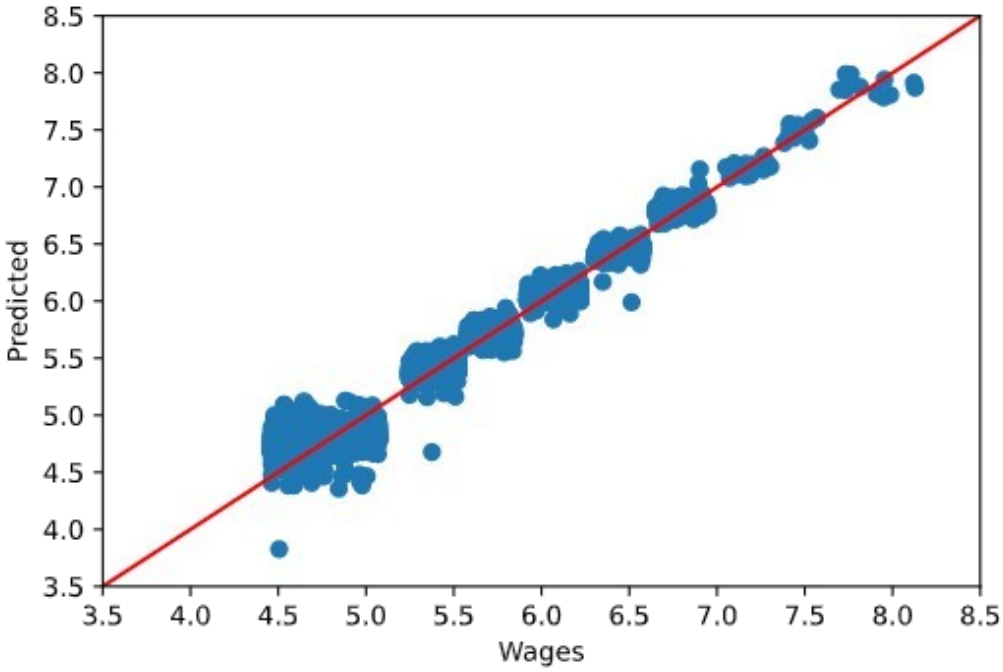

Figure 32 – Linear Regression predictions with the numerical revenue feature as a predictor

Some of the models, such as Random Forest and Neural Network, perform quite well across the revenue class categories and some perform quite well in some of the categories, but fairly poorly in some. Looking at all the models and their performance metrics, it is evident that all the models were able to find a pattern in the biggest revenue class, as all models had a R^2

score of 0.78 or more. The most difficult class was the smallest one, as most of the $R^{\%}-$ scores were under 0.5, with 0.03 being the worst. Hence, it could be said that all the models had difficulties explaining the variance in the smaller staff size categories.

4 Conclusion

The objective of this thesis was to understand how simple machine learning models, such as Linear Regression, and more sophisticated machine learning models can be used in predicting the annual salary costs of Finnish companies using public information about those companies. The motivation for the topic came from the authors employer, a mutual pension insurance company, as the mutual pension insurance company's revenue is based on the annual salary costs of its customers. Knowing the annual salary costs is the basis of determining the potential revenue that the mutual pension insurance company would receive from a new customer.

The topic was approached by reviewing the previous studies made in the field of salary prediction. Existing literature about machine learning was reviewed to form an understanding about the different machine learning algorithms and assessing the quality of different machine learning models. The chosen quality metrics in this study were the R^2-score and the Root Mean Square Error. The data used in the study were public data about Finnish companies and it was retrieved from Suomen Asiakastieto.

4.1 Answers to Research Questions

The research questions for the study were:

RQ1: What are the main features that determine the salary level of Finnish companies?

RQ2: What is the most accurate Machine Learning Algorithm to predict our target variable?

RQ3: Are the more sophisticated and computationally expensive machine learning models better than a Linear Regression model?

The answer to the first question, for this specific dataset was the revenue class. This was also supported by the fact that the R^2-scores across the models were better for those revenue class categories. The second most important feature were the different staff sizes, which had decent R^2-scores in every model. The other variables are somewhat questionable features, and it

might be feasible to remove them from the model. More accurate data could be beneficial for the models, as the categorical variables are too vast in scale, and therefore the prediction accuracies might suffer. By enriching the data with numerical variables, the models might make more accurate predictions.

The answer to the second research question is Random Forest Regressor, which had the average fold R^2-score of 0.795 and an average root mean squared error of 1,835,503€. This was followed by Linear Regression with a R^2score of 0.793. The computationally more expensive models, Neural Network and Support Vector Regressor, followed with a $R^2 - score$ of 0.791 and a root mean squared error of 1,942,252€ and 0.787 / 1,969,923€. The Decision Tree Regressor was the final model, and it had a $R^2 - score$ of 0.779 and RMSE of 2,006,212€. All the results were the mean scores of 5-folds in KFold Cross-Validation.

The answer to the third question is somewhat subjective, as the Linear Regression model performed really well compared to the more complex machine learning models. Though the Random Forest model had the lowest RMSE and the highest $R^2 - score$, Linear Regression is computationally less expensive and faster to compute as it does not need hyper parameter optimization. The $R^2 - score$ and RMSE of the Linear Regression model were near to the Random Forest model, which support the fact that Linear Regression should not be dismissed when considering a feasible model. Computationally the most expensive models, the Support Vector Regressor and the Neural Network, had worse performance according to the chosen accuracy metrics. Thus, by using Linear Regression the predictions are on par with the machine learning models but formulating, tuning and running the model takes less time and computational power.

4.2 Future research

The focus of this this study was to predict the annual salary costs of Finnish companies by utilizing public data about those companies. Previous research (such as Martín et al. (2017) & Bansal et al. (2021) has mainly focused on predicting the annual salary of individuals, which made this study one of the first in this specific field, but previous research was used as a framework as the problem was similar in nature.

The models had difficulties predicting the wages of the smaller companies, which might be due to the fact that the scale of the categorical variables is too vast. Consider the smallest revenue class category, where there are companies with a revenue from 1000€ to 199 000€, which makes the smallest observation 199 times smaller than the biggest observation. Thus, for future research it is suggested that the data would be enriched with the actual revenue value, if possible, or at least with more detailed categories. Furthermore, it would be advisable to enrich the data with the actual staff size, as the staff size class has similar scaling issues. There are service providers, such as Vainu.io, that offer enriched data about Finnish companies, but it comes at a significant cost. With this better data it would be advisable to study if the data has outlier values in the revenue feature, hence the revenue is such an important feature that these revenue outliers could skew the predictions immensely, especially in the smaller companies. Furthermore, outliers of revenue in the staff size feature should be studied for the same reasons.

Finally, previous research in the field of salary prediction, such as Das et al. (2020), suggested the use of K-nearest regression, but this study did not use K-nearest regression as a prediction model. Therefore, if future researchers are able to obtain the previously mentioned enriched data, the K-nearest regression could be an interesting method to utilize next to the models that were used in this study. The use of XGBoost could be interesting, as XGBoost has proved to be a scalable and reliable machine learning model. (Bentéjac et al., 2019) Bentéjac et al. (2019) observed that gradient boosting models, such as XGBoost, might be more accurate in some scenarios than for example Random Forest models. They continue by stating that gradient boosting methods have won many machine learning competitions and/or provided very respectable results in general.

References

Agatonovic-Kustrin S. & Beresford R., (2000) Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research,
Journal of Pharmaceutical and Biomedical Analysis, Volume 22, Issue 5, June 2000, Pages 717-727

Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, Hasan M, Van Essen BC, Awwal AAS, Asari VK. A State-of-the-Art Survey on Deep Learning Theory and Architectures. Electronics. 2019; 8(3):292. https://doi.org/10.3390/electronics8030292

Alpaydin E., (2014). Introduction to Machine Learning. Adaptive Computation and Machine Learning. The MIT Press

Awad M. & Khanna R. (2015) Efficient Learning Machines: Theories, Concepts, and Applications for engineers and System Designers, Apress Open

Balaji S. & Srivatsa S.K, (2012) Customer Segmentation for Decision Support using Clustering and Association Rule based approaches. International Journal of Computer Science & Engineering Technology (IJCSET) Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.6935&rep=rep1&type=pdf

Bansal U., Narang A., Sachdeva A., Kashyap I & Panda S.P., (2021) Empirical analysis of regression techniques by house price and salary prediction. IOP Conf. Ser.: Mater. Sci. Eng. 1022 012110

Bekkerman R., Bilenko M., & Langford J. (2012). Scaling up Machine Learning. Cambridge University Press

Benner, J. (2020). Cross-Validation and Hyperparameter Tuning: How to Optimise your Machine Learning Model. Retrieved from: https://towardsdatascience.com/cross-validation-and-hyperparameter-tuning-how-to-optimise-your-machine-learning-model-13f005af9d7d

Bentéjac, C, Csörgo A. & Martínez-Muñoz G. (2019) A Comparative Analysis of XGBoost . Retrieved from: https://www.researchgate.net/publication/337048557_A_Comparative_Analysis_of_XGBoost

Brownlee, J. (2016). Deep Learning with Python: Develop Deep Learning Models on Theano and TensorFlow using Keras (edition: v1.7).

Chan, C-C, H (2005) Online Auction Customer Segmentation Using a Neural Network Model. International Journal of Applied Science and Engineering 2005. 3, 2: 101-109

Das S., Barik R. & Mukherjee A. (2020) Salary Predixtion Using Regression Techniques. SSRN Electronic Journal · January 2020

Deisenroth M.P., Aldo Faisal A. & Soon Ong C. (2020) Mathematics for Machine Learning. Cambridge University Press

Hastie, T., Tibshirani, R., & Friedman, J. (2017). The elements of statistical learning (Vol. 2, No. 12). New York, NY, USA: Springer series in statistics.

Jackman S. & Reid G., (Year Unknown). Predicting Job Salaries from Text Descriptions. Retrieved from https://open.library.ubc.ca/

James G., Witten D., Hastie T. & Tibshirani R. (2017). Introduction to Statistical Learning: with applications in R. 8th edition. New York, NY, USA

Koskinen L., Nummi T., Salonen J. (2005) Modelling and Predicting Individual Salaries: A Study of Finland's Unique Dataset

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling (Vol. 26). New York: Springer.

Manna, S. (2020). K-Fold Cross Validation for Deep Learning Models using Keras. Retrieved from: https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538

Martín, I., Mariello, A., Battiti, R. & Hernández J., A. (2018). Salary Prediction in the IT Job Market with Few High-Dimensional Samples: A Spanish Case Study International Journal of Computational Intelligence Systems, Vol. 11 (2018) 1192-1209

Liu, Z. G., Pan, Q., Dezert, J., & Martin, A. (2016). Adaptive imputation of missing values for incomplete pattern classification. Pattern Recognition, 52, 85-95.

Montgomery, D. C., Peck, E. A., and Vining, G. G. (2001).
Introduction to Linear Regression Analysis 5th Edition. New York: Wiley.

Scikit-Learn, A Python library, www.scikit-learn.org

Shalev-Shwartz S., Ben-David S., (2014) Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press

Wang Z., Sugaya S. & Nguyen P.T. (2019) Salary Prediction using Bidirectional-GRU-CNN Model Retrieved from: https://anlp.jp/proceedings/annual_meeting/2019/pdf_dir/F3-1.pdf

Werloot, C. (2021) How to use K-fold Cross Validation with TensorFlow 2 and Keras? . Retrieved from: https://www.machinecurve.com/index.php/2020/02/18/how-to-use-k-fold-cross-validation-with-keras/