Statistical Machine Learning from Classification Perspective:

Prediction of Household Ties for Economical Decision Making

Kristoffer Brodin

Supervisor at Handelsbanken: Jovan Zamac

Supervisor at KTH: Tatjana Pavlenko

Examiner: Tatjana Pavlenko

October 2017

Abstract

In modern society, many companies have large data records over their individual customers, containing information about attributes, such as name, gender, marital status, address, etc. These attributes can be used to link costumers together, depending on whether they share some sort of relationship with each other or not. In this thesis the goal is to investigate and compare methods to predict relationships between individuals in the terms of what we define as a household relationship, i.e. we wish to identify which individuals are sharing living expenses with one another. The objective is to explore the ability of three supervised statistical machine learning methods, namely, logistic regression (LR), artificial neural networks (ANN) and the support vector machine (SVM), to predict these household relationships and evaluate their predictive performance for different settings on their corresponding tuning parameters. Data over a limited population of individuals, containing information about household affiliation and attributes, were available for this task. In order to apply these methods, the problem had to be formulated on a form enabling supervised learning, i.e. a target Y and input predictors $\mathbf{X} = (X_1, ..., X_p)$, based on the set of p attributes associated with each individual, had to be derived. We have presented a technique which forms pairs of individuals under the hypothesis H_0, that they share a household relationship, and then a test of significance is constructed. This technique transforms the problem into a standard binary classification problem. A sample of observations could be generated by randomly pair individuals and using the available data over each individual to code the corresponding outcome on Y and $\mathbf{X}$ for each random pair. For evaluation and tuning of the three supervised learning methods, the sample was split into a training set, a validation set and a test set.

We have seen that the prediction error, in term of misclassification rate, is very small for all three methods since the two classes, H_0 is true, and H_0 is false, are far away from each other and well separable. The data have shown pronounced linear separability, generally resulting in minor differences in misclassification rate as the tuning parameters are modified. However, some variations in the prediction results due to tuning have been observed, and if also considering computational time and requirements on computational power, optimal settings on the tuning parameters could be determined for each method. Comparing LR, ANN and SVM, using optimal tuning settings, the results from testing have shown that there is no significant difference between the three methods performances and they all predict well. Nevertheless, due to difference in complexity between the methods, we have concluded that SVM is the least suitable method to use, whereas LR most suitable. However, the ANN handles complex and non-linear data better than LR, therefore, for future application of the model, where data might not have such a pronounced linear separability, we find it suitable to consider ANN as well.

This thesis has been written at Svenska Handelsbanken, one of the large major banks in Sweden, with offices all around the world. Their headquarters are situated in Kungsträdgården, Stockholm. Computations has been performed using *SAS* software and data have been processed in *SQL* relational database management system.

Statistisk maskin inlärning från klassificeringsperspektiv:
prediktion av hushållsrelationer för ekonomiskt beslutsfattande

Sammanfattning

I det moderna samhället har många företag stora datasamlingar över sina enskilda kunder, innehållande information om attribut, så som namn, kön, civilstatus, adress etc. Dessa attribut kan användas för att länka samman kunderna beroende på om de delar någon form av relation till varandra eller ej. I denna avhandling är målet att undersöka och jämföra metoder för att prediktera relationer mellan individer i termer av vad vi definierar som en hushållsrelation, d.v.s. vi vill identifiera vilka individer som delar levnadskostnader med varandra. Målsättningen är att undersöka möjligheten för tre övervakade statistiska maskininlärningsmetoder, nämligen, logistisk regression (LR), artificiella neurala nätverk (ANN) och stödvektormaskinen (SVM), för att prediktera dessa hushållsrelationer och utvärdera deras prediktiva prestanda för olika inställningar på deras motsvarande inställningsparametrar. Data över en begränsad mängd individer, innehållande information om hushållsrelation och attribut, var tillgänglig för denna uppgift. För att tillämpa dessa metoder måste problemet formuleras på en form som möjliggör övervakat lärande, d.v.s. en målvariabel Y och prediktorer $\mathbf{X} = (X_1, ..., X_p)$, baserat på uppsättningen av p attribut associerade med varje individ, måste härledas. Vi har presenterat en teknik som utgörs av att skapa par av individer under hypotesen H_0, att de delar ett hushållsförhållande, och sedan konstrueras ett signifikanstest. Denna teknik omvandlar problemet till ett standard binärt klassificeringsproblem. Ett stickprov av observationer, för att träna metoderna, kunde genereras av att slumpmässigt para individer och använda informationen från datasamlingarna för att koda motsvarande utfall på Y och $\mathbf{X}$ för varje slumpmässigt par. För utvärdering och avstämning av de tre övervakade inlärningsmetoderna delades provet in i ett träningsset, ett valideringsset och ett testset.

Vi har sett att prediktionsfelet, i form av felklassificeringsfrekvens, är mycket litet för alla metoder och de två klasserna, H_0 är sann, och H_0 är falsk, ligger långt ifrån varandra och väl separabla. Data har visat sig ha en uttalad linjär separabilitet, vilket generellt resulterar i mycket små skillnader i felklassificeringsfrekvens då inställningsparametrarna modifieras. Dock har vissa variationer i prediktiv presentanda p.g.a. inställningskonfiguration ändå observerats, och om hänsyn även tages till beräkningstid och beräkningsk-raft, har optimala inställningsparametrar ändå kunnat fastställas för respektive metod. Jämförs därefter LR, ANN och SVM, med optimala parameterinställningar, visar resultaten från testningen att det inte finns någon signifikant skillnad

mellan metodernas prestanda och de predikterar alla väl. På grund av skillnad i komplexitet mellan metoderna, har det dock konstaterats att SVM är den minst lämpliga metoden att använda medan LR är lämpligast. ANN hanterar dock komplex och icke-linjära data bättre än LR, därför, för framtida tillämpning av modellen, där data kanske inte uppvisar lika linjär separabilitet, tycker vi att det är lämpligt att även överväga ANN.

Denna uppsats har skrivits på Svenska Handelsbanken, en av storbankerna i Sverige, med kontor över hela världen. Huvudkontoret är beläget i Kungsträdgården, Stockholm. Beräkningar har utförts i programvaran *SAS* och datahantering i databashanteraren *SQL*.

Acknowledgements

First I want to thank Jovan Zamac, my supervisor at Svenska Handelsbanken, for the idea behind this thesis, his advice and guidance. I would also like to thank Tatjana Pavlenko, my supervisor at KTH, for her input and help to finish the thesis.

Contents

Notations

Capital letters such as X and Y denote generic aspects of variables, whereas the corresponding small letters x and y denote observations/outcomes of these variables. X is used for predictor variables and Y for the corresponding target variable. Bold letters indicate a vector e.g. $\mathbf{X}$. For instance, we could have the p-vector of predictor variables $\mathbf{X} = (X_1, ..., X_p)$ as the input in a prediction model. The corresponding sample of n observations on the predictors is then denoted by $\mathbf{x}_1, ..., \mathbf{x}_n$, where for observation i, $\quad i = 1, ..., n$, $\mathbf{x}_i = (x_{1i}, ..., x_{pi})$.

1 Introduction

In modern society, information about people's relationships, preferences, interests, etc. is a valuable resource for many companies. Data containing information about current and future customers is an important key for profitability, service and development. Banks, social media companies, insurance companies, investors, etc. are all examples of companies who have much to gain by collecting such data. It can for example be used for financial and economic decision making, optimizing construction layouts and for the manufacturing and processing of goods and services. In this thesis we look at a particular aspect of this area, namely, the ability to predict relationships between individuals for economical decision making.

Many companies have large data records over their individual customers, containing information about attributes, such as name, gender, marital status, address, etc. However, these attributes only provide information about the customers independently, where knowledge about their relationships and connection to each other is often limited. For instance, marital status can tell us that an individual is married to someone, but not to whom. The possibility of finding links between costumers and making a connection between them may be very valuable since customers often affect each other in their decisions. For instance, consider an individual looking at making an investment and suppose he or she lives in a relationship with someone else e.g. marriage. Then this partner will likely be a part of the decisions concerning this investment since it affects both partners. However, the company providing the investment opportunity will only see the individual making the investment but not his or her partner. In the company perspective, if both individuals are costumers, they should rather be seen as one unit making a joint investment and not as separate individuals.

In this thesis the goal is to investigate and compare methods to predict relationships between people in the terms of what we define as a household relationship, i.e. we wish to identify individuals' household affiliation by linking all individuals who share a household relationship together.

Definition 1 (Household Relationship) *A household relationship is defined to be a group of one or several people who share living expenses.*

People sharing living expenses also make joint economic decisions. In this thesis we verify a household relationship by using a questionnaire given to customers at Svenska Handelsbanken. In the questionnaire the customers are asked to provide information about which other individuals he or she share a household relationship with according to definition 1. A household relationship is thus self-defined by the individuals and we do not to verify that they actually share living expenses in any other way. A customer does not have to share a household relationship with any other individual and thus the definition of a household relationship includes both one single individual and many individuals. Definition 1 is what is referred to when the terms "household relationship", "household affiliation" or "household link" are

used henceforth in this thesis.

Every individual possesses a number of attributes such as gender, marital status, address etc. From these attributes our goal is to predict household links between individuals. The questionnaire and data records over customers at Svenska Handelsbanken, provides the necessary data for this task.

1.1 Problem Statement

The objective of this thesis is to explore different models within statistical machine learning for prediction of people's household affiliation. For a population of 100000 individuals, their household relationship, see definition 1, should be identified by linking the individuals together according to their household affiliation. A selected few models within supervised learning, namely, logistic regression (LR), artificial neural networks (ANN) and the support vector machine (SVM), are tested and compared for this purpose.

The objective can be divided into two main parts. The first part is to formulate the problem on a mathematical form making supervised learning models applicable, i.e. derive a target variable Y, defining a household link, and derive a vector of p input predictor variables $\mathbf{X} = (X_1, ..., X_p)$ based on the set of p attributes registered on each individual. Thereafter, from the derivation of target and predictors, using the data over the individuals, a sample of n pairs of observations, $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_n, y_n)$, must be constructed. The second part, using this sample, is to estimate the unknown function $f(\mathbf{X})$, relating the attribute predictors $\mathbf{X}$ and the target household relationship Y, i.e. we have $Y = f(\mathbf{X}) + \epsilon$, where ϵ is a random error, independent of $\mathbf{X}$ and with mean zero. The main objective is to make a comparison of the three supervised learning models ability to estimate f, minimize ϵ and evaluate their predictive performance for different settings on corresponding tuning parameters. We use the constructed sample to first train and tune the models, thereafter to evaluate and compare how well they predict household relationships.

1.2 Motivation of Solution Approach

Initially the task of trying to find links between individuals in a large population, tying them together according to their household affiliation, might seem very difficult. Supervised learning models are based on having a target variable Y, predictors $\mathbf{X} = (X_1, ..., X_p)$ and aims to find the function f relating Y with $\mathbf{X}$. However, it is far from obvious how to transform the problem of identifying household links into the form of target and predictors and how to generate a sample to train the models. Therefore, one could consider the unsupervised clustering approach instead. With an unsupervised approach there exists no target variable. Instead the attributes would be analyzed directly to sort out individuals' household affiliation. The idea is to cluster customers into groups based on their attributes and where each group forms a unique household relationship. However, clustering data correctly into such

small groups is almost impossible. Clustering methods works well to sort data in to larger pre-determined categories but are not suitable for prediction problems of this kind.

Conclusively only supervised models are considered in this thesis and a target Y must be defined. This is done in the classification setting where the goal is to classify individuals into their household affiliation. However, it is not possible to derive a target variable which has a classification category for every possible unique household relationship or every individual in the data set. Generally, the number of household relationships is unknown, and this knowledge cannot be required. Instead a target must be derived in simpler way, with a small limited number of categories and where no information of number of household relationships is required. Our solution idea is to form pairs of individuals under the hypothesis that they share a household relationship and then test if this hypothesis holds true or not. Each individual is tested against a limited number of possible candidates which are believed to be a household partner. Thus, for individuals that are suspected to share a household relationship, we perform a test of the hypothesis on the form

$$H_0 : \text{The individuals share a household relationship.} \tag{1}$$

and the target variable could then be coded binary as

$$Y = \begin{cases} 1, & \text{if } H_0 \text{ is true,} \\ 0, & \text{if } H_0 \text{ is false.} \end{cases} \tag{2}$$

The problem has thus been reduced to a binary classification problem. In the data available there exist only two kinds of household relationships, single households and households consisting of two individuals, and therefore only these two cases are considered in this study. The idea could however be expanded to include households of more than two individuals as well. A sample for training may be generated by randomly pairing individuals two and two into fictive household relationships. Most of these pairs will of course share no household relationship with each other, but some will be paired into their household affiliation by random. The questionnaire, where household affiliation is provided by the individuals, serves as an identification key to identify for which random pairs H_0 is true and for which H_0 is false, i.e. how the target y should be coded for each random pair. The sample would thus not consist of single individuals, but of pairs of individuals. The attributes of each individual in a pair forms the predictor variables $\mathbf{x}$ for that pair. The sample can be used to train the prediction methods to distinguish between pairs of true household relationships and pairs of false household relationships, i.e. outcome on Y based on outcome on $\mathbf{X}$. The hypothesis idea, the definition of a target variable Y and predictor variables $\mathbf{X}$ and the generation of a sample of observations are more thoroughly described in section 2.

The three models within supervised learning that have been chosen for the comparison study, logistic regression (LR), artificial neural networks (ANN) and the support

vector machine (SVM), are some of the more common and applicable methods for both regression and classification. LR is perhaps the most widely used classification model and could therefore work as a baseline to compare the other two models towards. ANNs and the SVM are more refined models which can handle complex and non-linear problems. ANN's are not strictly restricted to only supervised learning, however, unsupervised learning is an exemption that will not be included in this thesis. For the LR-model there are no tuning parameters to set, whereas SVM's and in particular ANN's have several tuning parameters to set, and we will evaluate their predictive performance for different settings. Since the household prediction problem is a relatively new and unusual kind of problem, where the exact complexity of the solution is unknown, we expect the SVM and particularly ANN's to have the prospect of finding a solution to predicting these relationships where other more conventional statistical methods perhaps cannot.

1.3 Limitations and Prerequisites

- The data used in this thesis have been collected from customers at Svenska Handelsbanken, through a questionnaire and from other records over the customers. However, it could equally be collected from authorities or similar sources. It is important to note that choice of solution approach is based on the form of available data.

- A household relationship is defined by individuals through a questionnaire, hence, a household relationship is established at a fixed time point. However, relationships change over time and thus it is possible that an individual's household affiliation may have changed since the questionnaire was filled in. We do not consider this time aspect. Also, an individual is only allowed one unique household affiliation. An individual may not provide information of two separate household relationships with separate people.

- All data processing to form the sample of observations is done in SQL and the fitting and tuning of the prediction methods is made in SAS. However, some limitations in the fitting and tuning possibilities are induced by SAS e.g. limitations in the choice of kernel for the SVM.

1.4 Thesis Structure

The remaining part of this thesis is organized as follows. In section 2 we give the background of the dataset available, define the target variable y and the input predictors $\mathbf{x}$ properly. Also in this section a thorough description of how data is processed to from the sample of observations. In the final part of the section we present and discuss how to fit and tune the models and how to evaluate their performance. In section 3, mathematical background, the mathematical theory behind logistic regression, artificial neural networks and the support vector machine is presented. Each of the three methods have their separate section, first the theory and mathematical derivation of the method is presented, thereafter we describe how the

method is fitted and finally training and tuning aspects for respective method is discussed. In practice, the training and tuning process have a significant impact on how well the model predicts and therefore this part must be given proper consideration. In section 4, the results for the three statistical methods are presented. We give both results from the validation and the testing step and a comparison of the three prediction methods is made. In section 5 a discussion of solution approach, the obtained results and an attempt to explain our findings is made. The scope for future research is presented as well. Section 6 provides final conclusions.

2 Data preparation and Problem Setup

The basis for the comparison study of the three statistical method's ability to predict household links is the data over $m = 100000$ individuals, containing information about their attributes and their household affiliation. In section 1.2 we briefly introduced the solution idea on how to derive a target Y, predictors $\mathbf{X}$ and using these definitions and the data available to form a sample of observations. In this section we describe this idea more thoroughly.

The first step is to collect the information about the attributes and the household affiliation from the data records and the questionnaire into one data set. Every individual j is provided a unique personal ID in the form of a unique reference number $r_j, \quad j = 1, ..., m$. This number defines an individual in the data set. Information about household affiliation for each individual j is given by a numerical household identification ID $q_j, \quad j = 1, ..., m$ (not unique for every individual). This is our solution key, all individuals sharing a household relationship have equal household ID q_j. To check whether two individuals share a household relationship we simply check if their corresponding household identification ID match. The p attributes of respective individual we denote by $Z_1, Z_2, ..., Z_p$. Conclusively this information can be collected into a table with m rows, one row for each individual j, matching his/her reference number r_j, household identification key q_j and all the corresponding attributes $z_{1j}, z_{2j}, ..., z_{pj}$. This forms the basic data set to generate a sample of observations. The structure of this table can be seen in table 1.

r	q	Z_1	Z_2	$\cdots$	Z_p
r_1	q_1	z_{11}	z_{21}	$\cdots$	z_{p1}
r_1	q_2	z_{12}	z_{22}	$\cdots$	z_{p2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: *Table structure after the first step of collecting and ordering data. r_j, q_j, z_{1j}, ..., z_{pj}, $j = 1, ..., n$ are the reference numbers, household identification keys and all the attributes. The vertical dots symbolize all the elements following.*

As previously mentioned the household relationships are self-defined by the questionnaire. Therefore, there may be irregularities in the data and people may for some reason have provided incorrect information. In some cases, an individual may be associated with several separate household relationships i.e. having multiple household ID's. In this thesis only one unique household affiliation is allowed for each individual and thus individuals violating this criterion must be removed from the data set. If any information about household affiliation or attributes is missing without apparent reason, this individual will be removed from the data set as well. In conclusion, only individuals with one single household ID q_j and with all necessary information about his/her attributes, remain in the data set.

2.1 The Target Variable

We seek a solution to the household linkage problem through supervised learning methods in the classification setting. Thus, a target variable Y, mathematically defining a household relationship by a limited number of categories, must be derived. In section 1.2 the outline idea of how we do this was presented. Let us now do this properly. Since in general the number of households is unknown, the categories of the target Y cannot be defined requiring this knowledge. Letting each household be its own category is thus impossible. If all kinds of household constellations are considered, from single to including many individuals, the problem could become very complex. In our dataset there exits however only two possible kind of household constellations, namely single households and households consisting of two individuals, and therefore only these two cases need to be considered in this study, clearly simplifying the problem. In fact, if not considering children, which seldom have an individual relationship with the bank, a household relationship is seldom shared between more than two individuals. However, if one wish to predict households of more than 2 individuals, the approach described in this section could be extended to include more cases as well. This is discussed more further down and in section 5.

We start by first consider only one possible case, namely, households consisting of two individuals (no single) and the problem can be reformulated into the standard classification setting with a binary target Y. Suppose we pair two individuals j and ℓ, $j = 1, ...m, \quad \ell = 1, ...m, \quad j \neq \ell$, under the hypothesis

$$H_0 : \text{Individuals } j \text{ and } \ell \text{ share a household relationship.} \tag{3}$$

The goal is then to determine if this is true or false and a test of significance is constructed by coding the target Y as

$$Y = \begin{cases} 1, & \text{if } H_0 \text{ is true,} \\ 0, & \text{if } H_0 \text{ is false.} \end{cases} \tag{4}$$

For an individual j whose household affiliation should be investigated, he/she is simply test paired with other individuals $\ell, \quad \ell \neq j$, suspected be a possible household partner. Individuals form one or several pairs $\{j, \ell\}$ under H_0 and the problem has been simplified to the binary classification problem of predicting the class of each pair. If $y_{\{j,\ell\}} = 1$ is predicted, H_0 is considered to be true for that pair $\{j, \ell\}$. If $y_{\{j,\ell\}} = 0$ is predicted, H_0 is considered to be false for that pair $\{j, \ell\}$. In the end, if only considering households consisting of two individuals and with one unique household affiliation, for each individual, H_0 should be considered true for at most one pair. For instance, if an individual has been paired with ten other individuals, forming ten pairs under H_0, H_0 should be true for at most one of these ten pairs. Otherwise the individual has been predicted to share more than one unique household relationship. However, it is possible that this condition fails. In that case we may not sort out which pair has been predicted correctly and must investigate

these pairs further in another way. However, if using enough training data, considering multiple attributes as predictors and training the prediction methods well, this scenario will probably only occur in a few cases and most individuals will be classified correctly. In the case households consisting of more than two individuals are considered as well, one would of course allow for more than one matching pair, i.e. H_0 can be true for multiple pairs. Although, it might be hard to determine if multiple matches mean that several individuals share a household relationship or if some of them have been falsely matched. This is discussed further in section 5. For individuals who are not classified as $y = 1$ for any of his/her tested pairings, i.e. where H_0 is not considered true for any pair, when all reasonable possibilities have been tested, we conclude that this individual lives by himself/herself in a single household with high probability. Thus, with reasonable confidence, both single households and households consisting of two individuals can be identified from this pairing idea.

This idea of course assumes that there are a limited number of candidates each individual of interest can be tested towards. Fortunately, this is often the case. For instance, a good example on where we wish sort out household affiliation is to consider a large building with several apartments and a large group of people living there. Who is living in respective apartment is unknown and the problem is thus to predict which individuals are living together in one apartment. Then, clearly, there is a limited number of candidates to test. In the case when there are a very large number of possible pairings, this pairing approach of course becomes more complex, requiring longer computational time to test every possibility. However, as long as a limited number of individuals that should be paired is set, the pairing approach is always possible.

2.2 Sample Generation Procedure

From this pairing idea we may construct a sample to fit, tune and compare the prediction methods. The data set in table 1 is used to do this in two steps. First individuals are paired randomly into fictive households using random numbers from $U(0, 1)$. Individuals reference numbers form random pairs $\{r_j, r_\ell\}$, $j = 1, .., m$, $\ell = 1, ..., m$, with condition $j \neq \ell$. In those cases $q_j = q_\ell$ for a pair $\{j, \ell\}$, we code the corresponding observation on the target as $y_{\{j,\ell\}} = 1$ and H_0 is true for this random pair. For the vast majority of pairs we would however have $q_j \neq q_\ell$ with observation $y_{\{j,\ell\}} = 0$ on the target. This random pairing thus forms a sample of pairs for which H_0 is true for very few observations, i.e. with outcome $y = 1$. In order to have a better mix of observations on the target, the second step is to form a new smaller sample where individuals $j = 1, .., m$, $\ell = 1, ..., m$, $j \neq \ell$, are paired such that $q_j = q_\ell$. This smaller sample thus only consist of pairs where H_0 is true. These two samples can then be mixed in to one final large sample of observed pairs, with a proper distribution of both outcomes on the target Y. Conclusively, we pair individuals both randomly and based on their household affiliation to generate a sample with a sufficient mix of observations of pairs of individuals sharing a

ν	r	q	Z_1	Z_2	$\cdots$	Z_p
ν_1	r_1	q_1	z_{11}	z_{21}	$\cdots$	z_{p1}
ν_2	r_2	q_2	z_{12}	z_{22}	$\cdots$	z_{p2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2: *Table structure after adding a random number pairing key. Note that the table has been reorder by this key therefore the new r_1, q_1, z_{11}, etc. may not be the same as in previous tables. They just symbolize elements in the cells to clarify the structure of the tables created.*

r	q	Z_1	Z_2	$\cdots$	Z_p	r_{lag}	q_{lag}	$Z_{1,lag}$	$Z_{2,lag}$	$\cdots$	$Z_{p,lag}$
r_1	q_1	z_{11}	z_{21}	$\cdots$	z_{p1}	r_2	q_2	z_{12}	z_{22}	$\cdots$	z_{p2}
r_2	q_2	z_{12}	z_{22}	$\cdots$	$z_{p'2}$	r_3	q_3	z_{13}	z_{23}	$\cdots$	z_{p3}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 3: *Table structure after copying all columns and lagging all rows of the copy. The lagged copy is joined to the right of the table and the random number μ is removed from the table since it is of no use after the lagging procedure.*

household relationship and individuals not sharing a household relationship. Here is a more detailed description of how the paring process is done. This should not be seen as an exact way of how it must be done but serve as an example of how it can be done to generate the sample required.

2.2.1 Sample Generation Algorithm

1. For every unique reference number r_j in table 1, generate a random number ν_j from $U(0,1)$ and assign it to the corresponding r_j in a new column. This may be repeated several times to generate a sample of sufficient size. For the m unique r_j in table 1, we do three repetitions which will generate $3m$ random numbers as pairing keys. The structure of the resulting table can be seen in table 2.

2. Make a copy of the original table and lag the rows by one step, join this lagged table onto the original one. Thus, there will be two version of every column category, the original and the lagged. After the lagging procedure the random number column can be removed. The structure of the resulting table can be seen in table 3.

3. Merge all columns from the original table with the corresponding lagged double into arrays.

4. Check the household ID q_j column. If the two elements in an array are equal, $q_j = q_{j+1}$, the hypothesis H_0 is true for that pair. Thus, let $y_{\{j,j+1\}} = 1$ in

r	Y	q	Z_1	Z_2	$\cdots$	Z_p
$[r_1, r_2]$	$y_{\{1,2\}}$	$[q_1, q_2]$	$[z_{11}, z_{21}]$	$[z_{21}, z_{22}]$	$\cdots$	$[z_{p1}, z_{p2}]$
$[r_2, r_3]$	$y_{\{2,3\}}$	$[q_2, q_3]$	$[z_{12}, z_{13}]$	$[z_{22}, z_{23}]$	$\cdots$	$[z_{p2}, z_{p3}]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 4: *Table structure after the merging process and coding of the target variable Y.*

this case and otherwise let $y_{\{j,j+1\}} = 0$. The structure after the merging and coding of the target can be seen in table 4.

5. Generate a new table from table 1 where all reference numbers r_j and their corresponding attributes are paired using q_j as pairing key (group by q in SQL-setting) i.e. pairs $\{r_j, r_\ell\}$, $j = 1, .., m$, $\ell = 1, ..., m$, $j \neq \ell$ with condition $q_j = q_\ell$. Make sure there is only one unique q_j in the q-column and add a new column of $y_{\{j,\ell\}} = 1$ for all the households in this table. The structure of the resulting table is the same as table 4 apart from having only single elements in the q-column.

6. Join the table of random pairs with the table of q-generated pairs.

7. Finally reorder the resulting table by once again generate random numbers from $U(0, 1)$ and order the rows by these random numbers. This ensures a random order of observations. If one wish to extract a smaller subsample or divide data into training, validation and testing it is important that this extraction/division is random such that not only certain kinds of observations are included, e.g. $y = 0$. By reshuffling the sample by random this is ensured. The structure of the final table is as seen in table 4. Reference numbers r_j and household ID's q_j are of no interest in the sample for the prediction methods and can thus be removed from the table. Conclusively the final table has $n = 4m = 400000$ rows where each row corresponds to a pair $i = \{j, \ell\}$, $i = 1, ..., n$.

2.3 The Attributes

There are six kinds (categories) of attributes we consider for predicting household relationships, namely; address, last name, gender, age, marital status and account ownership at the bank. Apart from age, these kinds of attributes are all qualitative and must be coded as categorical. Note also that the attributes $z_{1j}, ..., z_p$ are associated to a single individual j, but the sample constructed in section 2.2 consists of pairs of individuals with pairs of attributes. Therefore, predictor variables should be coded w.r.t. a pair and not an individual himself/herself. For all six kinds of attributes the exact outcome is of minor interest for the household prediction problem. That an individual is living at a precise address or have a certain last name does not provide much information to predict his/her household relationship,

rather we are interested in whether the outcome is the same or not for individuals that have been paired together. For instance, having the same last-name, address or similar age will likely increase the probability of having a household relationship. Therefore, for two individuals that have been paired, we would like to define the attribute predictor variables $\mathbf{X} = (X_1, ..., X_p)$ based on whether the two individuals have an attribute in common, i.e. code them into binary variables telling us if an attribute is equal or not between two paired individuals. In what follows, we describe coding of predictor variables in detail for each of the six kinds of attributes.

Address

Address is a collective name for several attributes, where we look at city, postal code, street name, street number, and country (almost all individual are from the same country) as separately. Street name and street number are considered as both separate attributes and together as one attribute i.e. street name + street number. Apartment number, floor/stair or other extensions is also included in those cases this information exists. Suppose two individuals live on the same street with the same number but on different floors or with different apartment numbers, then the probability of them sharing a household relationship is not very large even though they live on the same street with the same number. Therefore, dividing address into several separate attributes provides more information than just letting it be one single attribute. Observe though that for most individuals, information about apartment number or floor/stair is not available and these attribute categories in many cases have missing values. Also, an individual may have multiple addresses registered, therefore we must consider three variants, the official address registered at the tax-authorities, a non-official address and the possibility for a costumer to provide a third address for e.g. a summer house. For most of the individuals in the data set these three kinds of addresses are the same though. Conclusively we have 10 kind of address attributes with 3 version of each, yielding a total of 30 address attributes that should be coded into predictor variables. By the principle idea described above, were our interest lies in whether city, street name etc.is the same or not for two paired individuals, we code the predictor variables for address as follows. For every pair $i = \{j, \ell\}$, $i = 1, ...n$ consisting of two individuals j, $j = 1, ..., m$ and ℓ, $\ell = 1, ..., m$, $j \neq \ell$, with an address attribute z_k^j and z_k^ℓ, $k = 1, ..., 30$ respectively, we code the corresponding predictor variable x_k, $k = 1, ..., 30$ for that attribute as

$$X_k = \begin{cases} 1, & \text{if} \quad Z_k^j = Z_k^\ell, \\ -1, & \text{if} \quad Z_k^j \neq Z_k^\ell. \end{cases} \tag{5}$$

To clarify; if for example the two individuals in a pair have the same official street name we get the outcome 1 on corresponding predictor variable for street name, whereas if they for example have different street number we get the outcome -1 on the predictor variable for street number. We do this coding for all 30 address attributes resulting in 30 predictor variables. If information about apartment number or floor/stair is missing, we just let the corresponding predictor variable be *null*.

Note that for the methods to be tested a $[-1, 1]$ coding is preferable compared to the standard binary coding $[0, 1]$, see the coming section (3.2.4).

Last Name

Last name also has more than one variant. We have the standard official last name but also the possibility for an individual to provide a self-chosen last name to the bank. Obviously in almost all cases there is no difference between these two variants. Thus, we have two attributes for last name and two corresponding predictor variables. They are coded exactly as the described by (11), i.e. for a pair of two individuals,

$$X_k = \begin{cases} 1, & \text{if they have the same last name,} \\ -1, & \text{otherwise,} \end{cases} \tag{6}$$

for $k = 31, 32$.

Account Ownership

Account ownership is an attribute telling us the account each individual has at the bank. We are interested in whether two individuals in a pair have any common account ownership since this likely increase the probability of sharing a household relationship as well. We thus code the corresponding predictor variable by the same manner as described by (11), i.e.

$$X_{33} = \begin{cases} 1, & \text{if they have a common account ownership,} \\ -1, & \text{otherwise.} \end{cases} \tag{7}$$

Gender

The attribute gender should be coded in an analogous way as previously described above. However, in this case, since opposite gender more likely imply a household relationship than having the same gender, following our convention that a positive number on the attribute variable imply positive impact on the probability of a household relationship, let

$$X_{34} = \begin{cases} 1, & \text{if they have opposite gender,} \\ -1, & \text{otherwise.} \end{cases} \tag{8}$$

Age

Age is a quantitative attribute and we are mostly interested in the age difference between the paired individuals (low age difference likely increases the probability of a household relationship). Therefore, for two paired individuals j and ℓ, $j \neq \ell$ the corresponding predictor variable is defined as

$$X_{35} = |\text{age of individual } j - \text{age of individual } \ell|. \tag{9}$$

We may also have an interest in different spans of age difference, therefore we also define the categorical variable

$$X_{36} = \begin{cases} 0, & \text{if} \quad |\text{age of individual } j - \text{age of individual } \ell| \leq 5, \\ 1, & \text{if} \quad 6 \geq |\text{age of individual } j - \text{age of individual } \ell| \leq 7, \\ 2, & \text{if} \quad 8 \geq |\text{age of individual } j - \text{age of individual } \ell| \leq 10, \\ 3, & \text{otherwise,} \end{cases} \tag{10}$$

Marital Status

For marital status we have only access to information on whether the individuals are married or not. Therefore, the predictor variable for marital status, for two paired individuals $j, \quad j = 1, ...m$ and $\ell, \quad \ell = 1, ...m$, $j \neq \ell$, is defined as

$$X_{37} = \begin{cases} 1, & \text{if both } j \text{ and } \ell \text{ are married,} \\ -1, & \text{otherwise.} \end{cases} \tag{11}$$

Following the data preprocessing steps of sections 2.1-2.3, we have final sample consisting of n observations $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_n, y_n)$ with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{1, 0\}$, with $p = 37$ and $n = 400000$. Each observation $(\mathbf{x}_i, y_i)$ corresponds to the paring of two individuals into a "fictive" household. The vector of predictors $\mathbf{x}$ tell us which attributes these two individuals have in common (or age difference) and the target y answer the question whether the hypothesis H_0 is true or not, i.e. if they share a real household relationship. The structure of the final sample can be seen in table 5.

Y	X_1	X_2	$\cdots$	X_p
y_1	x_{11}	x_{21}	$\cdots$	x_{p1}
y_2	x_{12}	x_{22}	$\cdots$	x_{p2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 5: *Table structure of the final sample.*

2.4 Training, Validation and Testing

Statistical machine learning methods are generally applicable to many different kind of problems and must thus be tuned to perform well for the household prediction problem at hand. Logistic regression has no tuning parameters to set, the support vector machine has some, whereas artificial neural networks are very flexible and the whole structure of the models determined by the settings on the tuning parameters. Also, one wants to evaluate the performance of the model, both to make a comparison study as is the goal of this thesis, but also to certify that the models are preforming well and the predicted results from them can be trusted.

There are several approaches in statistics and machine learning to achieve this. Many of them are formed with the assumption that the data available is limited, which is often the case, and evaluation should be done without sacrificing to much data for fitting, e.g. cross-validation. However, in our case the dataset is very large, and this problem does not exist. The simplest way for evaluation is then to the divide the sample of n observations into three separate sets: training, validation and testing. First the training set is used to fit the parameters of model, thereafter the validation set is used to validate and tune model. Using background knowledge and experience, start values on the tuning parameters can be set and then the validation set is used to evaluate the performance of the current setting, the model is re-tuned, and one may again evaluate whether performance has improved. In this manner the model can be tuned to maximize performance. Finally, the third data set, testing, is used to evaluate the real performance of the fitted and tuned model. It is the results from testing is what certifies that the model predicts sufficiently well. Note that these data sets are completely separated, no data used for training may be used for validation and no data for testing may have been used in training or validation. Then the whole idea of this process would be lost.

One of the most important aspects of dividing the data set in to three parts is to ensure the obtained solution is not overfitted. In the training process there is a risk that the fitted model will capture small irregularities in training data, not general to the whole data set and probably not present in any new data the models should be applied on later. If the model is adapted to these irregularities, prediction performance would decrease since consideration is taken to completely random aspects in the data that have no real general effect on the relationship between the predictors $\mathbf{X}$ and the target Y, e.g. outliers. By separating the data in these three parts, the fitted and tuned model's performance is evaluated on a completely new data set, i.e. the training set, and one can check that the model performs well on this new data set as well. Since this set has not been used to fit and tune the model, testing the model on this set gives a good indication on whether the model perform well when applied on new data or if it has been over-fitted and have been adapted too much to the data it was trained on. This issue is discussed further in section 3

There exists unfortunately no general rule on how to choose the number of observations in each of these data sets, since this depends on the so-called signal-to-noise ratio, i.e. the level of the desired output compared to the level of background disturbance in the data, and the number of available training observations. A common split is 50% for training, and 25% each for validation and testing ([1],p.222). This is also the chosen split in this thesis since we have a very large data set and have no limitations due to sample size.

2.4.1 Error Evaluation

Here is a more detailed description on how these three data sets are used to fit and tune the models. Denote the set of parameters to be directly fitted in respective

model by $\boldsymbol{\omega}$ and denote the set of tuning parameters (if existing for the model) by $\boldsymbol{\psi}$. The division into three data sets yields three corresponding kinds of classification/prediction errors to consider. The training error E_{tr} gives a measure on how much the fitted model deviates from the training data. In the training step the tuning parameters $\boldsymbol{\psi}$ are first fixed and the model fitted on the training set by choosing the set of parameters $\boldsymbol{\omega}$ which results in a small (optimal) training error E_{tr}. Observe that it is not necessarily optimal to find the global minimum of the training error since will likely consider outliers in the data and this will risk overfitting as just discussed. Therefore the optimal parameters $\boldsymbol{\omega}^*$ should be chosen such that E_{tr} becomes as small as possible but without overfitting the model.

The validation error E_{va} is obtained from the validation set. This error is used to tune the model, i.e. finding the optimal settings on $\boldsymbol{\psi}$. The tuning parameters $\boldsymbol{\psi}$ are fixed, $\boldsymbol{\omega}^*$ found from the training data and then the corresponding validation error E_{va} is evaluated on the validation set. $\boldsymbol{\psi}$ is modified and the procedure repeated once again which will yield a new E_{va} for new parameters $\boldsymbol{\omega}^*$ and $\boldsymbol{\psi}$. Then one can evaluate if E_{va} has decreased and thereby improved the predictive performance of the model. This should be done until no significant decrease on E_{va} can be seen, if one wish to find the optimal tuning parameters $\boldsymbol{\psi}^*$. The validation error also gives an indication on whether model has been overfitted. If E_{va} is significantly larger than E_{tr} the model is likely overfitted and the parameters $\boldsymbol{\omega}$ should be refitted (perhaps modifying the fitting procedure).

Finally, the test error E_{te} is obtained from the test set and the real goal is to train and tune the model such that E_{te} is minimized. This error gives a measure on the true performance of the model and a small E_{te} serves as a confirmation that for chosen parameters $\boldsymbol{\omega}$ and $\boldsymbol{\psi}$ the model predicts well. In an overfitted solution, the test error might be much larger than the training error, but with a small test error there is minor risk of an overfitted solution. The test error will be the main measure to compare the predictive performance between the three supervised learning models and the validation error will evaluate respective model's performance for different settings on the tuning parameters. In fact, E_{va} also serves as a good estimation of E_{te}.

Both E_{va} and E_{te} will be evaluated by the misclassification rate R_m, defined as

$$R_m = \frac{1}{n}\sum_{i=1}^{n} I_{y_i \neq \hat{y}_i}. \tag{12}$$

where I denotes the indicator function, ($\hat{y_i} = \hat{f}(\mathbf{x}_i)$ is the predicted classification of the target and y_i is the true observation of the target). It is rate measure of how many individuals that have been classified correctly. This rate error is however not as suitable for training the models since it does not give a measure on how well a data point have been classified. When fitting the models, it is a significant difference if a data point just falls within the decision boundary for the classification

rule or well inside with good margin. Therefore, other error measures are used for evaluating the training error E_{tr}. Choice of error measure for training depend on the supervised learning model and is therefore defined for each of them separately in their respective section, in section 3.

3 Mathematical Background

3.1 Logistic Regression

This section is based on the layout given in ([2],p.130-137) and ([1],p.119-120) about the theory and fitting of the logistic regression model.

For classification problems with a categorical target, logistic regression is perhaps one of the most common approaches. It can be seen as an expansion of normal linear regression into the classification setting. When the target Y is not quantitative, but qualitative, we may not directly regress the output from the input predictors $\mathbf{X}$. Instead in logistic regression we construct a regression model for the probability $P(Y = y|\mathbf{X} = \mathbf{x})$, i.e. the probability of the outcome (category) y of the target variable Y, given the predictors $\mathbf{X} = \mathbf{x}$. We must thus choose a function which maps the outcome from the regression of the input predictors $\mathbf{X}$ strictly into the interval $[0, 1]$. There are potentially a couple of possible choices but in logistic regression we use the logistic function. For a binary target $Y \in \{0, 1\}$ and p multiple predictors $\mathbf{X} = (X_1, ..., X_p)$ the logistic model for $Y = 1$ can be written

$$log\left(\frac{P(Y = 1|\mathbf{X} = \mathbf{x})}{1 - P(Y = 1|\mathbf{X} = \mathbf{x})}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p \tag{13}$$

where the left-hand side is the so-called log-odds for $Y = 1$. Equivalently we have

$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}}{1 + e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}} \tag{14}$$

where the right-hand side forms the logistic function. Thus, we also have for $Y = 0$

$$P(Y = 0|\mathbf{X} = \mathbf{x}) = 1 - P(Y = 1|\mathbf{X} = \mathbf{x}) = \frac{1}{1 + e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}}. \tag{15}$$

$\beta_0, \beta_1, ..., \beta_p$ are parameters to be estimated. We regress on the log-odds which inverse is the logistic function, mapping the regression in the interval $[0, 1]$, yielding a prediction for the sought probability. Define $p(\mathbf{x}) := P(Y = 1|\mathbf{X} = \mathbf{x})$ and introduce a threshold of 0.5 and the classifier $G(\mathbf{x})$ becomes

$$G(\mathbf{x}) = \begin{cases} 1, & p(x) \geq 0.5, \\ 0, & p(x) < 0.5. \end{cases} \tag{16}$$

If $p(x) \geq 0.5$, paired individuals are classified as: H_0 is true, and if $p(x) < 0.5$ they are classified as: H_0 is false. If the target variable has more than two categories, this model is not valid and has to be modified, however this is of no interest in this thesis. When the target Y has only two categories and can be coded as binary, standard linear regression is a possible approach as well. It can be shown that $\mathbf{X}\beta$ obtained from the linear regression would be an estimate of the probability $P(Y = 1|\mathbf{X} = \mathbf{x})$. However in normal regression the predicted response on the target can be hard to interpret as a probability measure since the estimate might fall outside the $[0, 1]$ interval ([2],p.130). Thus, logistic regression is the method of choice also in this case.

3.1.1 Fitting the Logistic Regression Model

The parameters $\beta_0, \boldsymbol{\beta} = (\beta_1, ..., \beta_p)$ can be estimated from the constructed training set using Maximum likelihood. Given the set of n pairs of training observations $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_n, y_n)$, we have

$$p(\mathbf{x}_i; \beta_0, \boldsymbol{\beta}) := P(Y = 1 | \mathbf{X} = \mathbf{x}_i; \beta_0, \boldsymbol{\beta}) = \frac{e^{\beta_0 + \boldsymbol{\beta}^T x_i}}{1 + e^{\beta_0 + \boldsymbol{\beta} x_i}}, \quad i = 1, ..., n. \tag{17}$$

For a binary output the Bernoulli distribution is appropriate to model the probability. The likelihood function can thus be written

$$L(\boldsymbol{\beta}) = \prod_{j: y_j = 1} p(\mathbf{x}_j; \beta_0, \boldsymbol{\beta}) \prod_{j^*: y_{j^*} = -1} (1 - p(\mathbf{x}_{j^*}; \beta_0, \boldsymbol{\beta})), \tag{18}$$

where j are all training observations with outcome $y_j = 1$ and j^* are all training observations with outcome $y_{j^*} = 1$, in total n observations. With the coding $Y \in \{0, 1\}$ and using (17) we may derive a simplified version of the log-likelihood as

$$\begin{aligned}
\ell(\beta_0, \boldsymbol{\beta}) :=& log(L(\beta_0, \boldsymbol{\beta})) \\
=& \sum_{j: y_j = 1} \log(p(\mathbf{x}_j; \beta_0, \boldsymbol{\beta})) + \sum_{j^*: y_{j^*} = -1} \log(1 - p(\mathbf{x}_{j^*}; \beta_0, \boldsymbol{\beta})) \\
=& \sum_{i=1}^{n} \frac{1 + y_i}{2} \log(p(\mathbf{x}_i; \beta_0, \boldsymbol{\beta})) + \frac{1 - y_i}{2} \log(1 - p(\mathbf{x}_i; \beta_0, \boldsymbol{\beta})) \\
=& \sum_{i=1}^{n} \frac{1 + y_i}{2} (\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i - \log(1 + \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i)) \\
&+ \frac{1 - y_i}{2} (- \log(1 + e^{\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i})) \\
=& \sum_{i=1}^{n} \frac{1 + y_i}{2} (\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) - \log(1 + e^{\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i})
\end{aligned} \tag{19}$$

The estimates $\hat{\beta}_0$, $\hat{\beta}_1$, ..., $\hat{\beta}_p$ are chosen such that the log-likelihood function $\ell(\beta_0, \boldsymbol{\beta})$ is maximized over all training observations (equivalent to maximizing $L(\beta_0, \boldsymbol{\beta})$). We do so by taking the derivative of $\ell(\beta_0, \boldsymbol{\beta})$ w.r.t. $\beta_0, \boldsymbol{\beta}$ an setting it to zero. It yields a set of $p + 1$ non-linear equations on the form

$$\begin{aligned}
\frac{\partial \ell(\beta_0, \boldsymbol{\beta})}{\partial \beta_0} &= \sum_{i=1}^{n} \frac{1 + y_i}{2} - p(\mathbf{x}_i; \beta_0, \boldsymbol{\beta}), \\
\frac{\partial \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{n} \mathbf{x}_i \left(\frac{1 + y_i}{2} - p(\mathbf{x}_i; \beta_0, \boldsymbol{\beta}) \right),
\end{aligned} \tag{20}$$

which we solve to find the parameters $\beta_0, \boldsymbol{\beta}$. The equation system is usually solved numerically by the Newton-Raphson algorithm (other approaches are also possible),

requiring us to take the second derivatives as well to find the Hessian. This is the algorithm used by *SAS*. For an introduction to the Newton-Raphson algorithm, see e.g. [6].

3.2 Artificial Neural Networks

This section is based on the layout given in ([1],p.392-401) about the background theory, fitting and training of artificial neural networks. Observe that this section first aims to provide a general description of the three supervised learning methods, thus the notation $\mathbf{X} = (X_1, ..., X_p)$, refer to a general vector of predictor variables, Y or Y_k, $k = 1, ..., K$ refer to a general target variable or the target categories for K-class classification. The notation Z_m, $m = 1, ...M$ are used to denote hidden layers in a ANN and have no relation with the previous usage of $Z_1, ..., Z_p$ to denote the individuals attributes before coding them in to predictor variables.

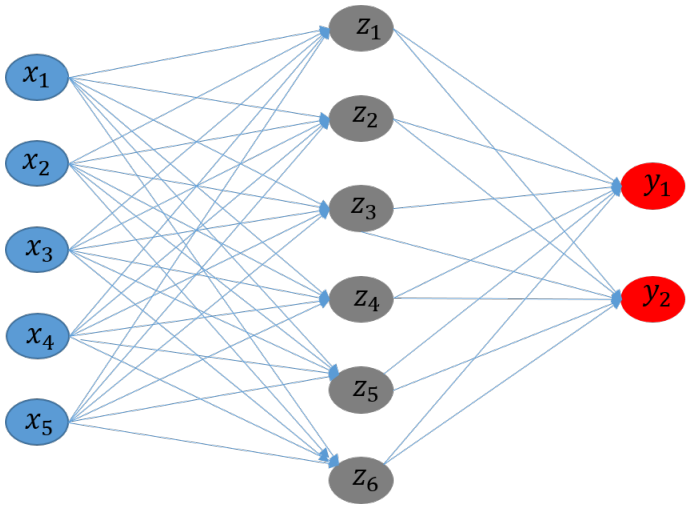

Figure 1: *Illustration of a neural network with five inputs $X_1, \ldots, X_5$, one hidden layer with six units $Z_1, \ldots, Z_6$ and an output layer with two output units Y_1, Y_2. Note that sometimes an extra bias unit of 1, symbolizing α_0, β_0, is included in the input and hidden layers in the network illustration, we have chosen not to do so here.*

An artificial neural network (ANN) is a regression or classification model in several stages. It is often represented by a network diagram ([1],p.392) as in figure 1 and is originally inspired from attempting to mimic brain activity ([3], p.316). In this thesis we are only interested in classification and therefore focus is on this application, however the network described below is general and applicable for regression as well. A network consists of an input layer of p nodes for the input variables $\mathbf{X} = (X_1, ..., X_p)$, an output layer with K nodes for the target measurements Y_k, $k = 1, ..., K$ and one or several middle layer(s), called the hidden layer(s) since the values on its hidden nodes are never observed directly ([1],p.393). For a network with one hidden layer we denote its nodes by Z_m, $m = 1, ...M$. For regression we have usually have $K = 1$ and only one output node. For K-class classification, there are K nodes in the right output layer, with the k'th unit modeling

the probability of class k. Each of the targets measurements Y_k are being coded as a binary variable, 1 for belonging to class k, 0 if not ([1],p.392). The target variable is thus modeled as complete layer where each node represents each class or equally each node represent each possible outcome the target variable.

The number of hidden layers in the network is a somewhat subjective choice that is set before the fitting procedure. On one hand more layers may yield better predictions but on the other hand also yields more complexity, less interpret-ability, the risk of over-fitting increases and more computations are required. Therefore, it seems unnecessary to set multiple hidden layers at the start rather we may increase the number hidden layers in the validation process to see if predictions can be improved this way. Also, the number of units in each hidden layer is a subjective choice that must be determined and evaluated by a similar procedure.

In a single hidden layer neural network the hidden nodes Z_m are modeled as linear combinations of the input variables $\mathbf{X} = (X_1, ..., X_p)$, and then the target Y_k is modeled as a function of linear combinations of the Z_m as

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \boldsymbol{\alpha}_m^T \mathbf{X}), \quad m = 1, ..., M, \\ T_k &= \beta_{0k} + \boldsymbol{\beta}_k^T \mathbf{Z}, \quad k = 1, ..., K, \\ f_k(\mathbf{X}) &= g_k(\mathbf{T}), \quad k = 1, ..., K, \end{aligned} \tag{21}$$

where the value on the k'th output node can be expressed as

$$Y_k = f_k(\mathbf{X}) + \epsilon_k. \tag{22}$$

We have $\mathbf{Z} = (Z_1, Z_2, ..., Z_M)$ for a choice of M hidden units and $\mathbf{T} = (T_1, ..., T_K)$ for K classes. α_0, $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_p)$ and β_0, $\boldsymbol{\beta} = (\beta_1, ..., \beta_M)$ are unknown parameters to be estimated. $\sigma(v)$ is the activation function which makes the non-linear transformation of the inputs $\mathbf{X}$ into the hidden units Z. There are a number of choices on σ but the most common one is the *sigmoid/logistic* $\sigma(v) = 1/(1 + e^{-v})$ ([1],p.392), which makes the first transformation a logistic regression. Different choices of activation functions can be seen in table 6.

Activation Function	$\sigma(u)$	Range of Values
Identity	u	$\mathbb{R}$
Sigmoid/Logistic	$(1 + e^{-u})^{-1}$	$(0, 1)$
Hyperbolic tangent	$\frac{e^u - e^{-u}}{e^u + e^{-u}}$	$(-1, 1)$
Exponential	e^u	$(0, \infty)$

Table 6: *Table of common activation functions in artificial neural networks.*

$g(\mathbf{T})$ is the output function which makes a final transformation of the outputs T from the hidden units Z into a suitable form for the output of interest Y. For K-class classification the most common transformation is by the $softmax$ function

$$g_k(\mathbf{T}) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}, \quad k = 1, ..., K \tag{23}$$

where K is the number of output units Y_k. It has the property $\sum_{k=1}^{K} g_k(\mathbf{T}) = 1$ and the output from $g_k(\mathbf{T})$ can thus be interpreted as a probability measure of class k. However for multiple regression with only one single output node we instead use the identity function $g(T) = T$ ([1],p.393).

$f_k(\mathbf{X})$ is a function which takes the vector of inputs $\mathbf{X}$ as argument and then gives the corresponding numerical output value from the output function $g_k(\mathbf{T})$ on each output node k. ϵ_k is the corresponding error on node k. In the case of K-class classification with the soft-max output function we would have $f_k(\mathbf{X}) \in [0, 1]$ and $\epsilon_k \in [0, 1]$, such that $Y_k \in \{0, 1\}$, $k = 1, ..., K$, and $\sum_{k=1}^{K} f_k(\mathbf{X}) = 1$.

For a network with two hidden layers with M units in the first $\mathbf{Z} = (Z_1, ..., Z_M)$ and N units in the second $\mathbf{V} = (V_1, ..., V_N)$, (21) would be expanded to

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \boldsymbol{\alpha}_m^T \mathbf{X}), \quad m = 1, ..., M, \\ V_n &= \sigma(\beta_{0n} + \boldsymbol{\beta}_n^T \mathbf{Z}), \quad n = 1, ..., N, \\ T_k &= \gamma_{0k} + \boldsymbol{\gamma}_k^T \mathbf{V}, \quad k = 1, ..., K, \\ f_k(\mathbf{X}) &= g_k(\mathbf{T}), \quad k = 1, ..., K, \end{aligned} \tag{24}$$

where α_0, $\boldsymbol{\alpha}$, β_0, $\boldsymbol{\beta}$ and γ_0, $\boldsymbol{\gamma}$ are the unknown parameters to be estimated. In the same manner more hidden layers can be added to the model. The unknown parameters are called weights as they tell how much weight each variable from one node should have in the link to the next node in the network. Notice how the number of parameters to be estimated increases as another layer is added and therefore large networks require much computational power. With p input variables $\mathbf{X}$ there are $M(p + 1) + K(M + 1)$ weights to be estimated in (21) and the added layer in (24) increases the number to $M(p + 1) + N(M + 1) + K(N + 1)$ weights.

3.2.1 The Special Case of a Two-Categorical Target

Our classification problem where the target variable has only two categories is a special case for artificial neural networks. There are two possibilities to build a network for this particular case. Either one uses the general setup for K-class classification as described above and let $K = 2$ i.e. two output nodes. Each node is being coded as a "dummy" variable where the output of the first node corresponds to the probability of the class: H_0 is true, i.e. $Y = 1$, and the second the probability of the class: H_0 is false, i.e. $Y = 0$. Thus, the classifier would be

$$G(\mathbf{x}) = \underset{k}{\text{argmax}}(f_k(\mathbf{x})), \quad k = 1, 2. \tag{25}$$

The other possibility is using only one output node ($K = 1$), as in in the case of standard regression with neural networks, and introducing a threshold. We form

pairs under the hypothesis H_0, i.e. that $Y = 1$ for each pair, and this single node corresponds to the probability: $p(x) := P(Y = 1|\mathbf{X} = \mathbf{x})$. If the network predicts a value in this single output of $p(x) > 0.5$, the pair is classified as being true under H_0, i.e. $Y = 1$, otherwise as false, i.e. $Y = 0$. The classifier is thus

$$G(\mathbf{x}) = \begin{cases} 1, & p(x) \geq 0.5, \\ 0, & p(x) < 0.5. \end{cases} \tag{26}$$

Choice of output function is crucial for these two approaches. For the first approach with two output nodes the softmax function (refSmax) is clearly most suitable for the corresponding classifier as the outputs of the two nodes sum to one. The second approach requires an output function which strictly maps the outputs from the last hidden layer into the interval $[0, 1]$, just as in logistic regression. Thus the logistic function, see table 6, is the natural choice. These approaches are in fact equivalent but the second one is (mathematically) simpler [5] and therefore the choice of this thesis.

3.2.2 Fitting Neural Networks

Using the training set of n sample of pairs $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_n, y_n)$ constructed in (2) we want to find the optimal values on all the weights that makes the model fit the training set well. Let $\boldsymbol{\omega}$ denote the set of weights to be fitted, i.e. for a single hidden layer network $\boldsymbol{\omega} = \{\alpha_0, \boldsymbol{\alpha}, \beta_0, \boldsymbol{\beta}\}$. The fitting procedure thus aims to find the optimal $\boldsymbol{\omega}$. Introduce the cross-entropy loss function

$$R(\boldsymbol{\omega}) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log(f_k(\mathbf{x}_i)) \tag{27}$$

which is the common loss-function for K-classification. In the regression case we normally use MSE as loss function which is a possible choice for classification, however not as efficient since it tends to penalize outliers excessively [7]. In the article [8], Golik P, Doetsch P and Ney H presents a study comparing squared error and cross-entropy as loss function when training ANNs, concluding that cross-entropy is preferable. Thus, for a more thorough motivation of why cross-entropy is used, we refer to [8]. In the case the $softmax$ function (23) is used as activation function and cross-entropy (27) as error function, the neural network corresponds exactly to a linear logistic regression model in the hidden units. In that case all the parameters are suitably estimated by maximum likelihood ([1],p.395), see section 3.1.1. Otherwise the standard approach is by gradient decent methods and so called back-propagation described in the following section.

3.2.3 The Back Propagation Algorithm

Back-propagation is an algorithm to compute the gradient for minimizing the error function in a neural network and can be used for various gradient based optimization methods. In iteration j, for a fixed set of training observations $(\mathbf{x}_i, y_i)$, $i = 1, ...n$ and a fixed collection of weights $\boldsymbol{\omega}^j$, the output $f_k(\mathbf{x}_i)$, $i = 1, ..., n$ is calculated

from (21), (if we have one hidden layer, (24) if two etc.). This is the forward pass. Thereafter errors are calculated for the output layer by the chosen loss-function, comparing the computed output $f_k(\mathbf{x}_i)$ and the true output y_i for each of the observations $i = 1, ...n$. The error is then back propagated through the network in order to calculate the corresponding error on all the nodes, matching their contribution to the error difference between the computed output $f_k(\mathbf{x}_i)$ and the true output y_i, $i = 1, ..., n$. This is the so called backward pass. These errors are then used to calculate the gradient of the loss-function, fed into the gradient decent optimization method to the update of the collection of weights $\boldsymbol{\omega}^{j+1}$ for a new iteration in order to minimize the loss-function. The initial weight is often set at random, see section 3.2.4.

Here is a more detailed description for a network with one hidden layer, used for classification, with cross-entropy as loss-function and a single output node. For a training observation i, $\quad i = 1, ...n$, the cross-entropy loss-function (27) simplifies to

$$R_i(\omega) = -y_i \log(f_k(\mathbf{x}_i)). \tag{28}$$

By taking the derivative of R_i with respect to each parameter in $\boldsymbol{\omega}$ we may find the contribution each weight in the network has to the total error. Using (21) let $z_{mi} = \sigma(\alpha_{0m} + \boldsymbol{\alpha}_m^T \mathbf{x}_i)$ and $\mathbf{z} = (z_{1i}, ..., z_{Mi})$. The derivatives become

$$\begin{aligned} \frac{\partial R_i}{\partial \beta_m} &= -\frac{y_i}{f(\mathbf{x}_i)} g'(\beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i) z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{ml}} &= -\frac{y_i}{f(\mathbf{x}_i)} g'(\beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i) \beta_m \sigma'(\alpha_{0m} + \boldsymbol{\alpha}_m^T \mathbf{x}_i) x_{il}, \end{aligned} \tag{29}$$

for $i = 1, ..., n, \quad m = 1, ..., M, \quad l = 1, ..., p$. Now define

$$\begin{aligned} \delta_i &:= -\frac{y_i}{f(\mathbf{x}_i)} g'(\beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i), \\ s_{mi} &:= -\frac{y_i}{f(\mathbf{x}_i)} g'(\beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i) \beta_m \sigma'(\alpha_{0m} + \boldsymbol{\alpha}_m^T \mathbf{x}_i), \end{aligned} \tag{30}$$

where δ_i is the "error"/sensitivity at the output and s_{mi} is the "error"/sensitivity at the hidden layer for training observation i. The gradients (29) can then be written

$$\begin{aligned} \frac{\partial R_i}{\partial \beta_m} &= \delta_i z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{ml}} &= s_{mi} x_{il}. \end{aligned} \tag{31}$$

By their definitions the errors δ_i and s_{mi} satisfy

$$s_{mi} = \sigma'(\boldsymbol{\alpha}_m^T \mathbf{x}_i) \beta_m \delta_i, \tag{32}$$

called the back-propagation equations. For fixed weights $\boldsymbol{\omega}$ and every training observation i the prediction $\hat{f}(\mathbf{x}_i)$ is calculated by (21) in the forward pass. In

the backward pass the errors δ_i are first computed and then back-propagated by (32) which yields the errors s_{mi}. Through both these errors the gradients (31) may be computed for the updates in the gradient decent optimization algorithm. In a gradient descent, for iteration $(j+1)$ from (j) the weight updates would commonly have the form (over all training observations)

$$\begin{aligned} \beta_m^{j+1} &= \beta_m^j + \eta_j \sum_{i=1}^{n} \frac{\partial R_i}{\partial \beta_m^j}, \\ \alpha_{lm}^{j+1} &= \alpha_{lm}^j + \eta_j \sum_{i=1}^{n} \frac{\partial R_i}{\partial \alpha_{ml}^j}, \end{aligned} \tag{33}$$

where η_j is the learning rate, often a constant, determining how far to move in the gradients direction. If it is chosen to large the algorithm might miss the minimum and jump over it by taking to large iteration steps, whereas if it is to small it could require a very large amount of iterations to reach even near a minimum ([3],p.338). It should satisfy $\eta_j \to 0$ as the iteration $j \to \infty$ ([1],p.397). In *SAS* two alternatives of gradient decent algorithms can be chosen for the learning process, namely Stochastic Gradient Decent (SGD) and the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (LBFGS). Both of these methods are more refined gradient decent methods and use the calculated gradients in their iterations. See e.g. [9] and [10].

3.2.4 Training and Tuning Neural Networks

As seen in the sections above the theory behind neural networks is quite straightforward, however the possibilities in tuning the network are almost infinite. There are many aspects to consider and thus training neural network is far from a simple task. In practice it the tuning and training process that is determining the actual performance of the network. In this section we summarize some of these important aspects, how to approach this procedure and later in the thesis, in the result section, application on real data is presented.

Starting Values To start the optimization algorithm to find the proper weights for the network, suitable starting values must be provided. Usually they are chosen to be random values near zero ([1],p.397). The sigmoid/logistic function is nearly linear around zero and therefore, if this is the chosen activation function, and we set the starting weights near zero, the neural network starts as an approximately linear model. The network then gradually turns more nonlinear as non-linearities are introduce where it is needed by the optimization process ([1],p.397). If we would let the starting weights be exactly zero though, we would have zero derivatives and perfect symmetry in the gradient updates, causing the algorithm to never move. On the other hand too large weights at the start often leads to poor solutions ([1],p.398). Thus, we let the starting weights be random values near zero.

Overfitting A neural network is a rather complex model with a large amount of weights introduced to be fitted. Often there are more weights then needed. Thus the global minimum of the error function (27) will likely lead to an overfitted solution ([1],p.398), yielding a large test error even though the training error is small. Therefore we rather seek a local minimum of the error function ([1],p.398) and there are several tricks to achieve this. Firstly, if we use a network which is not larger than necessary, i.e. just large enough to provide an adequate fit, this automatically tends to avoid overfitting. A larger network can create more (unnecessary) complex functions whereas a smaller network does not have the power to overfit data [4]. However, to predict well the network must be complex enough to capture the various aspects of the data, where for example a linear model would unlikely predict well on non-linear data. Therefore, it is very difficult to set the correct size of the network from the start. Two other approaches is *earlystopping*, where the algorithm is stopped well before it reaches its global minimum, or *regularization*, where an penalty, penalizing a to complex network, is added to the error function [4]. However, in our case where we have access to a large amount of data, we may instead divide data into a training, validation and test set as described in section 2.4. The number of training data points is much larger the number of weights to be fitted and therefore there is minor risk of overfitting [4]. With such a large data set the model is forced to find general patterns and may not memorize small irregularities in the training data. With sufficient validation, necessary measures have been taken to avoid overfitting and ensuring that the network is constructed to predict well.

Scaling of the Inputs In many cases it may be suitable to standardize the inputs such that they have mean zero and standard deviation one[1]. Otherwise if the range of different inputs varies heavily, inputs with a larger range will dominate so much that those with a smaller range have little or no contribution to the prediction of the output [5]. Unless you know beforehand that some inputs should be given more weight, it is recommended to standardize the inputs such that they will be treated equally in the fitting process [5], ([1],p.400). It is also preferable to code binary inputs as $[-1, 1]$, instead of the classical $[0, 1]$, see [5]. It may seem that this would make no difference, however, this is due to geometrical reasons. For neural networks with a binary target Y, in geometrical perspective, a hyperplane is the separator of data into the two target classes, i.e. the decision boundary, compare to SVM's in section 3.3. Initially planes will be set close to the origin to pass through data in many different directions. In the optimization process we train the network by searching for the hyperplane separating data in the best way. If data is offset from the origin, as with $[0, 1]$ coding, there is a risk that the initial planes will completely miss data and also the range of directions initialized will be too limited. Thus, it will be hard for the algorithm to find an appropriate solution [5].

Number of Hidden Units and Layers How to construct a network of correct size the proper number of hidden layers and proper number of units in each layer is far from obvious. This was somewhat discussed in section (3.2) and in section

(3.2.4) in the aspect of overfitting. Here we discuss it a bit further. Even though one wants to avoid a too large network to decrease the risk of overfitting and decrease the computational cost, it is generally better to have too many hidden units than too few ([1],p.400). If the hidden units are too few the model might not have enough flexibility to capture the nonlinearities in the data whereas with too many hidden units, the extra weights will shrink to zero if proper regularization is used ([1], p.400) or if the number of observations for training are much larger the number than parameters to be fitted. As mentioned earlier we can create such a large training set and preform proper validation. Therefore, we may initially set a size which we believe is suitable and then both decrease and increase it and evaluate its effects on the validation data. The network size should be chosen so large that the validation error does not decrease notably if the size is increased further. The number of hidden units is typically chosen somewhere in the range of 5 to 100, where a higher number is chosen as the number of inputs and number of training cases increases ([1],p.400). I our case the network will be within a limited size, 1-5 hidden layers, and thus both the proper number of layers and the proper number of nodes in each layer can be tested individually through the validation process.

Multiple Minima Since the error function R() is non-convex, there may be multiple local minima and therefore the fitted network is much dependent on the starting values on the weights. To accommodate for this, it is recommended to try a couple of starting configurations to see which one yields lowest error. Then to one may then use an average of the predictions generated from a collection of networks as the final prediction ([1],p.401). One might also consider directly averaging the weights instead which only will yield one prediction. However, since the network is a non-linear model, it is much preferred to average the predictions rather than the weights. A network is fitted for every start configuration, where each fitted collection of weights together tries to capture a complex non-linear behavior. The weights are thus dependent on each other, making an average over these fitted collections will in some respect ignore this complexity and thus probably lead to quite a poor solution ([1],p.401).

3.3 The Support Vector Machine

This section is based on the layout in ([1], p.417-436).

The Support Vector Machine (SVM) is a supervised classifier for a target variable with exactly two categories. The model may be extended to include a target with multiple categories but that is of no interests of this thesis since we have a target on binary form $Y \in \{0, 1\}$. However, the SVM require a binary target variable with different coding based on sign. Therefore, when using the SVM we redefine the target with new coding as

$$Y = \begin{cases} 1, & \text{if } H_0 \text{ is true,} \\ -1, & \text{if } H_0 \text{ is false.} \end{cases} \tag{34}$$

With the new coding of the target, we have a training set of n observations, making up the sample $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_n, y_n$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$, $i = 1, ..., n$. The idea of the SVM is to use the training data to find a separator which separates all these training points into two classes, $y_i = 1$ and $y_i = -1$, in the space spanned by the predictors $\mathbf{X} = (X_1, ..., X_p)$. If the predictor space can be separated in this manner, the separator forms a simple classification rule and every pair of individuals i can be directly classified based on the predictors $\mathbf{x_i}$ for that pair. The separator can then be applied on new data as a classifier.

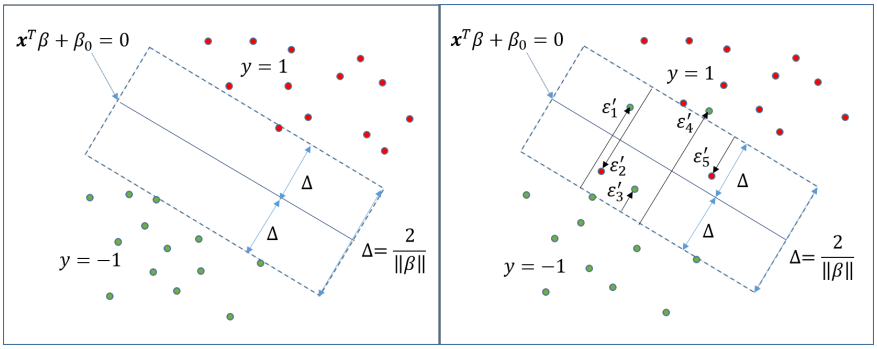

Figure 2: *Simplified illustration of separating hyperplanes. Note that the space is multidimensional and spanned by the attributes* $\mathbf{X}$ *as base vectors. The solid blue line is the decision boundary. Red dots symbolize data points with* $y = 1$*, green* $y = -1$*. To the left: perfectly separable data with no overlap. To the right: overlap due to the data points* $\xi_i^{'}$*,* $i = 1, ..., 5$ *fall on the wrong side of the margin with an amount* $\xi_i^{'} = \Delta\xi_i$*, points on the correct side have* $\xi_i^{'} = 0$*. The total amount is bounded by a constant as* $\sum \xi_i \leq$ *constant and the total distance of points on the wrong side of the margin is given by* $\sum \xi_i^{'}$*.*

We start with a linear separator in the form of a hyperplane and assume that the two classes are perfectly linearly separable. Define a hyperplane by:

$$\mathbf{x} : f(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0 = 0, \tag{35}$$

where $\boldsymbol{\beta}$ is the unit normal vector to the hyperplane, $\|\boldsymbol{\beta}\|_2 = 1$. $\|\boldsymbol{\beta}\|_2$ denotes the L^2-norm of $\boldsymbol{\beta}$ defined as

$$\|\boldsymbol{\beta}\|_2 := \left(\sum_{j=1}^{p} |\beta_j|^2\right)^{1/2}. \tag{36}$$

Naturally, a classifier would then be

$$G(\mathbf{x}) = sign(\mathbf{x}^T\boldsymbol{\beta} + \beta_0), \tag{37}$$

determining which side of the hyperplane the data point is located. We want to find the hyperplane $f(\mathbf{x})$ with the largest margin Δ between the training points in respective class and $f(\mathbf{x})$, such that all training points fall outside the margin on each side, see figure 2. Thus, the problem is to find the parameters $\beta_0, \boldsymbol{\beta}$, which defines the separating hyperplane $f(\mathbf{x})$, such that the margin Δ is maximized. It can be formulated as the following optimization problem

$$\begin{aligned} \max_{\boldsymbol{\beta},\beta_0,\|\boldsymbol{\beta}\|=1} \quad & \|\Delta\|_2 \\ \text{s.t.} \quad & \begin{cases} f(\mathbf{x}_i) = \mathbf{x}_i^T\boldsymbol{\beta} + \beta_0 \leq \Delta & \text{when} \quad y_i = 1, \\ f(\mathbf{x}_i) = \mathbf{x}_i^T\boldsymbol{\beta} + \beta_0 \geq \Delta & \text{when} \quad y_i = -1, \end{cases} \end{aligned} \tag{38}$$

for all training points $(\mathbf{x}_i, y_i), \quad i = 1, ..., n$. It may be simplified to

$$\begin{aligned} \max_{\boldsymbol{\beta},\beta_0,\|\boldsymbol{\beta}\|=1} \quad & \|\Delta\|_2 \\ \text{s.t.} \quad & y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq \Delta, \quad i = 1, ..., n \end{aligned} \tag{39}$$

We may drop the norm constraint $\|\boldsymbol{\beta}\| = 1$ by reformulating the constraints as

$$\begin{aligned} & \frac{1}{\|\boldsymbol{\beta}\|_2} y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq \Delta, \quad i = 1, ..., n \\ \Leftrightarrow \quad & y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq \Delta\|\boldsymbol{\beta}\|_2, \quad i = 1, ..., n, \end{aligned} \tag{40}$$

(redefining β_0 and making $\boldsymbol{\beta}/\|\boldsymbol{\beta}\|_2$ a unit vector). We may then arbitrarily set $\Delta = 1/\|\boldsymbol{\beta}\|_2$ (we may equally define the margin in terms of $\boldsymbol{\beta}$) and (40) can more conveniently be reformulated to the standard quadratic convex optimization problem

$$\begin{aligned} \min_{\boldsymbol{\beta},\beta_0} \quad & \|\boldsymbol{\beta}\|_2 \\ \text{s.t.} \quad & y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq 1, \quad i = 1, ..., n \end{aligned} \tag{41}$$

where finding the optimal β_0 and $\boldsymbol{\beta}$ is equivalent to maximize the margin Δ.

If the data is not perfectly separable, where some points fall on the wrong side

of the margin and the classes overlap in the predictor space, we may modify the model to accommodate for this. We still maximize Δ but allow for some points to fall on the wrong side of the margin. Define the slack variables $\boldsymbol{\xi} = (\xi_1, \xi_2, ..., \xi_n)$ and modify the constraint in (41) to

$$y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq \Delta(1 - \boldsymbol{\xi}), \quad i = 1, ..., n, \quad \xi_i \geq 0, \quad \sum_{i=1}^{n} \xi_i \leq \text{constant} \quad . \tag{42}$$

The idea is that every ξ_i corresponds to the amount, in proportion to the margin Δ, the predicted classification $f(x_i) = \boldsymbol{\beta}\mathbf{x}_i + \beta_0$ of a data point i, falls on the wrong side of the margin. By bounding the total sum of these slack variables, we also bound the total proportional amount the predicted classifications may fall on the wrong side of the margin. A data point is misclassified when $\xi_i > 1$, so by bounding the sum $\sum_{i=1}^{n} \xi_i$ at some constant value, we bound the total number of miss-classifications in the training set at that value ([1],p.419). With this formulation the optimization problem remains convex. Finally, by introducing the cost parameter C the optimization problem can more conveniently be re-expressed as

$$\begin{aligned} \min_{\boldsymbol{\beta},\beta_0} \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \xi_i \geq 0, \quad y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq \Delta(1 - \xi_i), \quad i = 1, ..., n. \end{aligned} \tag{43}$$

where C replaces the constant in (42).

3.3.1 Non-Linearity

The classification rule described above is based on a linear separator in form of a hyperplane. If data is not linearly separable this classifier will not preform sufficiently. The trick here is rather then abandoning the idea of a separating hyperplane, which is linear by construction, we make a transformation of the predictor space, in which the hyperplane is defined, to an enlarged predictor space by basis expansions. If data is non-linear in the space spanned by the predictors $\mathbf{X} = (X_1, ..., X_p)$, a new space may be constructed by making a non-linear transformation of the predictor base vectors by transformation functions, $\mathbf{h}(\mathbf{X}) = (h_1(\mathbf{X}), ..., h_p(\mathbf{X}))$, into a new space where the data is indeed (more) linear. In this new space we can define a linear hyperplane, forming the non-linear function $f(\mathbf{x}) = \mathbf{h}^T(\mathbf{x})\boldsymbol{\beta} + \beta_0$ in the original space. We use the same classifier $G(\mathbf{x}) = sign(f(\mathbf{x})) = sign(\mathbf{h}^T(\mathbf{x})\boldsymbol{\beta} + \beta_0)$ as before. Thus, the hyperplane technique stays intact, but the data setting is instead transformed. The only problem remaining is finding the transformation functions h_i.

To do this we would like to show the solution form of the estimation of the unknown parameter $\boldsymbol{\beta}$ and thus we must show the start of the solution process. The

minimization problem (43) is convex since it is quadratic with linear inequality constraints. To solve it we may use quadratic programming and Lagrange multipliers. Let the Lagrange primal function be defined as follows

$$L_P := \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{x}_i^T\boldsymbol{\beta}+\beta_0) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i,, \tag{44}$$

where P denotes primal. We wish to minimize (44) w.r.t. β_0, $\boldsymbol{\beta}$ and ξ_i. Thus setting the derivatives $\frac{\partial L_P}{\partial \beta_0}$, $\frac{\partial L_P}{\partial \boldsymbol{\beta}}$, $\frac{\partial L_P}{\partial \xi}$ to zero yields

$$\begin{aligned} \boldsymbol{\beta} &= \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \\ 0 &= \sum_{i=1}^{n}\alpha_i y_i \\ \alpha_i &= C - \mu_i, \quad \forall i \end{aligned} \quad . \tag{45}$$

and the positive constraints $\alpha_i,\ \mu_i,\ \xi_i \geq 0,\ \forall i$. Note in particular the solution form of $\boldsymbol{\beta}$. The original space is spanned by $\mathbf{X} = (X_1, ..., X_p)$ and the new transformed space is spanned by $\mathbf{h}(\mathbf{X}) = (h_1(\mathbf{X}), ..., h_p(\mathbf{X}))$, thus the solution form of $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{h}(\mathbf{x}_i) \tag{46}$$

in the new space. The solution function $f(\mathbf{x})$ can thus be written

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta} + \beta_0 = \sum_{i=1}^{n}\alpha_i y_i \langle \mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}_i)\rangle + \beta_0, \tag{47}$$

in the new space. $\langle\cdot,\cdot\rangle$ denotes the inner-product, defined as

$$\langle \mathbf{h}(\mathbf{x}_i), \mathbf{h}(\mathbf{x}_j)\rangle := \sum_{\ell=1}^{p} h_\ell(\mathbf{x}_i)h_\ell(\mathbf{x}_j), \tag{48}$$

for the two observations $\mathbf{x}_i$ and $\mathbf{x}_j$. We see that the solution to the minimization problem (43) can be represented in such a way that the transformation function $\mathbf{h}(\mathbf{x})$ only occur as an inner product. Therefore, it is not necessary to specify it at all. In fact we do not have to define the transformation properly, we only need to define $\mathbf{h}(\mathbf{x})$ in terms of inner products. We use the so-called kernel trick and introduce the kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{h}(\mathbf{x}_i), \mathbf{h}(\mathbf{x}_j)\rangle, \tag{49}$$

which computes the inner-product in the transformed space [1]. The inner-product is simply replaced with a chosen kernel-function in the calculations for the fitting

procedure. It should be a positive (semi-) definite function ([1],p.424), where some common kernel-functions can be seen in table 7.
Choice of kernel function depend on data available and the problem at hand. Remember that the purpose of them is to transform non-linear data into a new space such that it becomes (approximately) linear, so the choice should be made to achieve this goal, see more in section 3.3.3.

3.3.2 Fitting the Support Vector Machine

Given $f(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta} + \beta_0$, introduce the hinge-loss function

$$L(y, f) = [1 - yf(\mathbf{x}]_+, \tag{50}$$

where + denotes positive part, and consider the optimization problem

$$\min_{\boldsymbol{\beta},\beta_0} \quad \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \frac{1}{2C}\|\boldsymbol{\beta}\|_2^2. \tag{51}$$

In words, for an observation $(\mathbf{x}_i, y_i)$ the hinge-loss function $L(y_i, f(\mathbf{x}_i))$ takes the value zero for points well inside the margin and for points outside and far away from the margin it linearly increases as the point's distance to the margin increases. We may thus see that the solution to (51) is in fact equal to the solution to (43) when $L(y, f)$ given by (50), and we may equally solve (51) to find the separating hyperplane with the largest margin. In practice we thus solve (51) and hinge-loss is the loss function usually chosen for the SVM classifier.

In the previous non-linearity section we started the solution process of the optimization problem (43) by quadratic programming and Lagrangian multipliers to show the solution form of β. In practice this is however not the solution approach used to fit the SVM. In *SAS* primal and dual interior point methods are used to fit the SVM and thus the natural choice in this thesis, see e.g. [11] for more information about these methods.

3.3.3 Tuning the Support Vector Machine

The cost parameter C is the regularization parameter for the SVM classifier and have a considerable influence on how well the model predict. A high value on C limits the amount of data points in training data to fall on the wrong side of the

Kernel	$K(\mathbf{x}_i, \mathbf{x}_j)$	Parameters
Polynomial	$(1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^d$	d
Radial Basis Function	$exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$	γ
Sigmoid	$tanh(\kappa_1\langle \mathbf{x}_i, \mathbf{x}_j \rangle + \kappa_2)$	κ_1, κ_2

Table 7: *Table of common kernel functions for the support vector machine.*

margin by discourage positive ξ_i ([1],p.424) and thus imply a quite restrictive classifier. The complete separable case corresponds to $C = \infty$ [1]. Thus, a high value classifies training data very well but may instead lead to an overfitted solution when applied on new data. If a non-linear transformation of the attribute space is made, a large value on C will yield a very un-smooth boundary ([1],p.424). A lower value allows for more data points to fall on the wrong side of the margin and is not so restrictive. We get a smoother boundary and the risk of over-fitting decreases, but on the other hand when we allow for more misclassification in the training process, we will likely have several misclassifications when tested on new data as well. The general approach is to set C quite high even though this will cause the risk of overfitting ([1],p.432). Through validation we may tune a suitable value on C.

As mentioned above there are several choices of kernel-function. If data is linear we of course have no need of a non-linear transformation and no kernel-function have to be set i.e. we use a polynomial kernel function and let $d = 1$, see table (7). Otherwise we choose any of the options in table 7 and specify the corresponding parameter(s). The radial basis function is perhaps the most common choice for non-linear data and often works well. The parameter(s) for respective kernel function can be chosen through cross validation or by standard validation, as is our approach since we have been able to reserve enough data for validation due to the large size of the original sample.

4 Results

The objective of this thesis was to explore the three supervised learning methods, logistic regression (LR), artificial neural networks (ANN) and the support vector machine (SVM), ability to predict household relationships. This was done in two parts. The first part was to derive the target Y, predictors $\mathbf{X} = (X_1, ..., X_p)$ and generate a sample of observations $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$. The second part, using the constructed sample, was to train the LR-model, ANNs and the SVM, evaluate and validate their predictive performance for different settings on corresponding tuning parameters and finally test and compare the three models in their ability to identify household relationships. In this section we first conclude the solution to the first part and thereafter for the second part, the results of tuning and comparison the three supervised learning methods, LR, ANN and SVM are presented.

Regarding the first part of the objective. We have presented a technique to predict household links between individuals by forming pairs of individuals under the hypothesis H_0 that they share a household relationship, and then tests if this hypothesis H_0 holds true or not. This idea enable us to define a binary target $Y \in \{0, 1\}$, with coding, 1 if H_0 is true and 0 if H_0 is false, which transforms the initially vague and intangible problem to a much more tangible problem in the form of standard binary classification. Instead of the very difficult task of trying to find links between individuals in a vast population, tying them together according to their household affiliation, they are just tested two and two against each other and classical supervised learning methods are thereby applicable. Initially the attributes of the individuals are also on an intangible form for prediction, e.g. address. However, with this pairing solution, the predictors $\mathbf{X} = (X_1, ..., X_p)$ can easily be coded according to which attributes the paired individuals have in common or how the differ i.e. if address is the same or not, their age difference etc. Individuals, for which H_0 is not predicted to be true for any tested pair, would be predicted to live in a single household and not share a household relationship with anyone else in the data set. Hence single households can be predicted from this idea as well. Since the data records used in this study only contains individuals living in single households or households consisting of two individuals, only these cases are tested. However, this pairing idea holds for larger household constellations as well. This is discussed more in section 5. In section 2 a through description of this pairing idea was presented.

For the second part of the objective, two cases have been considered to train, validate and test the three supervised learning models. First the full sample of $n = 400000$ observations was used, divided such that 50% of the observations were used for training and 25% were used for validation and testing respectively. This is henceforth referred to as the full sample. Thereafter a smaller subsample of 10000 observations was extracted from the full sample. 300 observations were reserved for training and the rest were split equally for validation and testing of the models, i.e. 4850 observations respectively. This is henceforth referred to as the small sample.

In the sample generation algorithm described in section 2.2.1 the observations where finally reorder randomly by the $U(0,1)$ distribution. Hence the splitting of the sample into training, validation and testing and the extraction of the small sample is completely random. Also, the distribution of outcomes on Y is uniform and equal between the two sample sizes. This random extraction has thus ensured that there is sufficient distribution of outcomes also in the small sample. It should also be noted that a large majority of the observations have a low classification complexity, e.g. observed pairs with target, $y = 1$ and where most of the predictors $\mathbf{x}$ are 1 as well, i.e. real household pairs where the individuals have the same address, last name, etc. The pairs with high classification complexity are a smaller minority but also the ones most interesting to investigate. Using the full sample with so many observations, in combination of having considered multiple predictor attributes, the idea is to be able to sort out even those difficult cases when there is a subtler household link between two individuals. With the small sample the idea is to investigate how the sample size and kind of observations affects the prediction results. Also, we are interested to see if the difference in prediction ability increase between the three supervised models in this case.

The results using the two sample sizes are presented as follows. In Section 4.1 the validation results for LR are presented, in section 4.2 the validation results for ANN are presented and in section 4.3 the results from validating the support vector machine are presented. For LR and the SVM the results for the to sample sizes are presented simultaneously, whereas, due to the vast number of tuning parameters, for ANN the results for the two sample sizes are presented separately. First comes the validation on the full sample in section 4.2.1 and then follows the validation on the small sample with comparison of the results w.r.t. sample size in section 4.2.2. In the validation procedure the models are first fitted on the training set and the validation results are from validation on the validation set. Finally in section 4.4 optimal settings on the tuning parameters are determined and the test results from comparing the three supervised learning models are presented. The three tuned models are trained on the training set and the comparison results are from testing on the test set which has been unused in the validation analysis. Note that the misclassification rate, defined in (12), is used as error measure for both validation error E_{va} and test error E_{te}.

4.1 Validation of Logistic Regression

For logistic regression there are no tuning parameters to set. A comparison of validation error for the full large sample and the small sample is made and this can be seen in table 8.

	Full Sample	Small Sample
E_{va}	0.000766	0.00411

Table 8: *Logistic Regression - Table over comparison of validation error, E_{va}, for the two sample sizes.*

Most notable is the extremely low misclassification rate, suggesting that the classes are far away from each other in the space spanned by the predictors $\mathbf{X}$. This makes it easy to find the decision boundary, separating the classes, and thus also easy to classify the observations. The validation error is slightly larger, almost a factor 10, when only 300 observations are used for training. An increase is much expected. However, considering the overall low magnitude of the misclassification rate, the variation is actually very small and conclusively there is no significant difference when a smaller sample size is used for training.

In appendices tables over the estimated parameters $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)$ for the corresponding predictors $\mathbf{X} = (X_1, ..., X_p)$, can be seen. The table in Appendix A the estimates fitted on the full sample, the table in Appendix B the estimates have been fitted on the small sample. Note the sign on the estimates, negative sign indicates less likelihood of a pair of individuals sharing a household relationship and positive the opposite. Those predictors with p-value> 0.05, or equally, whose confidence limits do not cover zero, are considered insignificant for predicting Y at the 95% confidence level. In other words, not all predictors used in the LR-model are significant for predicting household relationships and could potentially be removed from the model without decreasing predictive performance. When the LR-model is fitted on the full sample we see that it is the predictors related to the individuals address and the non-official last name that are insignificant in the model. Official last name, gender, age difference etc. are all significant for predicting household relationships. It is generally the non-official address predictor or those not very specific such as city and country that are insignificant. Note however, that no estimate is zero when fitted on the full sample and thus all predictor variables have an influence on the prediction results. The results seen in table in Appendix B show that no predictor is considered significant when the model is fitted on the small sample. With a small training set the LR-model seems only capable to estimate very large confidence intervals, resulting in all predictors being insignificant. Also notable, a couple of the estimates are zero and the corresponding predictor has no influence on the prediction results. Thus, it seems that the model puts relatively more weight on a couple of predictors and fewer predictors are used for predicting household relationships when the LR-model is fitted on a small training sample.

4.2 Validation of Artificial Neural Networks

For ANN's there are multiple tuning parameters to set, the number of hidden layers, the number of hidden nodes on these layers and a suitable activation function. We

also have two different optimization algorithms to choose from, $LBFGS$ and SGD. In this section these tuning parameters effect on the validation error are evaluated for both the full sample and the small sample.

4.2.1 Validation with the Full Sample

First the results from when the full sample are presented. Figure 3 shows how the validation error varies with the number of hidden of layers. 50 hidden nodes on each hidden layer and the logistic activation function is used in this study, (on all layers). A comparison between the optimization algorithms is also made. Table 9 shows the number of iterations required for the two optimization algorithms to reach convergence versus number of hidden layers. Observed that the maximum number of iterations has been set to 500 and if this is reached, the algorithm may have never converged.

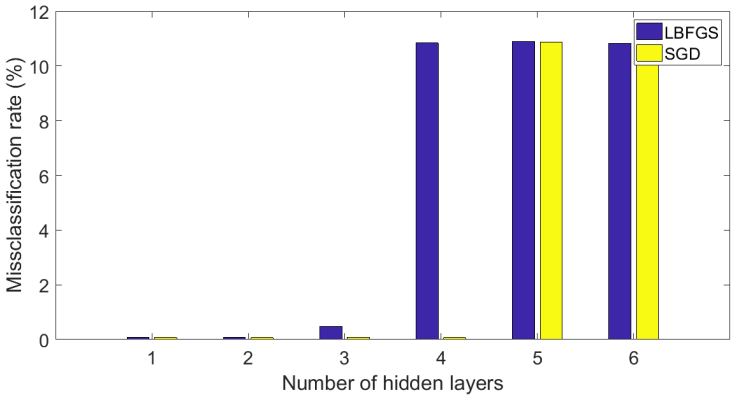

Figure 3: *Artificial Neural Network - Validation error, E_{va}, vs. number of hidden layers for the two optimization algorithms, LBFGS and SGD, fitted on the full sample. Blue is LBFGS and yellow is SGD.*

# of Hidden Layers	# of Iterations, $LBFGS$	# of Iterations, SGD
1	66	500
2	51	500
3	28	500
4	7	500
5	7	500
6	7	500

Table 9: *Artificial Neural Network - Table over the number of iterations required by the LBFGS-algorithm and the SGD-algorithm until convergence or reaching maximum allowed number of iterations versus number of hidden layers, fitted on the full sample.*

The results in figure 3 show that for an ANN there is no clear difference in validation error for $1, 2$ or 3 layers and that the misclassification rate is extremely small in this case, which also corresponds with the results obtained for LR. However, for $LBFGS$, from 4 layers and upwards, and for SGD, for 5 and 6 layers, there is a sudden significant increase in the validation error, suggesting that using multiple hidden layers for prediction of household relationships worsen the predictive performance. This increase is a very surprising and unexpected result since theoretically the error should decrease or at least remain approximately constant as more hidden layers are used. Also notable, it is not just an increase in validation error, but a significant jump, where the error goes from a low relatively constant level to a much higher but also relatively constant level. After the jump the error does not increase further as more layers are added. From table 9 it can be seen that for $LBFGS$ the number of iterations are significantly lower for 4 hidden layers and upwards, which might explain this strange jump in validation error. We discuss this more section 5.2. Conclusively we see that there seems to be no need for more than one hidden layer in our case.

The results in figure 3 show that the validation error is similar for $LBFGS$ and SGD, with the exception that the jump in validation error occurs one step later for SGD, i.e. at 4 versus 5 hidden layers. In this perspective one might consider SGD to be a better optimization algorithm than $LBFGS$. However, if studying table 9, one see that the number of iterations is significantly higher for SGD than $LBFGS$, and therefore SGD requires much longer computational time. It is also important to note that, no matter the number of hidden layers used, the number of iterations is constantly 500 for SGD. Remember that this is the set maximum allowed number of iterations and therefore this implies that the solution obtained by SGD has never really converged. Although, it should be empathized that for up to 4 hidden layers, the validation error is still very small for the obtained solution even if the SGD-algorithm has not converged. Conclusively though, due to significantly shorter computational time, effectiveness and SGD's lack of convergence, henceforth only $LBFGS$ is used when fitting ANNs in this thesis.

In figure 4, the results of how the validation error varies with the number of hidden nodes, on each hidden layer, are presented. We have seen that there is no need to use multiple hidden layers and thus only the results for 1 and 2 hidden layers are evaluated. In the case of two layers we use the same number of nodes on each layer, i.e. if 10 on the first then also 10 on the second. The logistic function serves as activation function also in this case. Table 10 shows the number of iterations used by the *LBFGS*-algorithm versus the number of hidden nodes, for 1 and 2 hidden layers respectively.

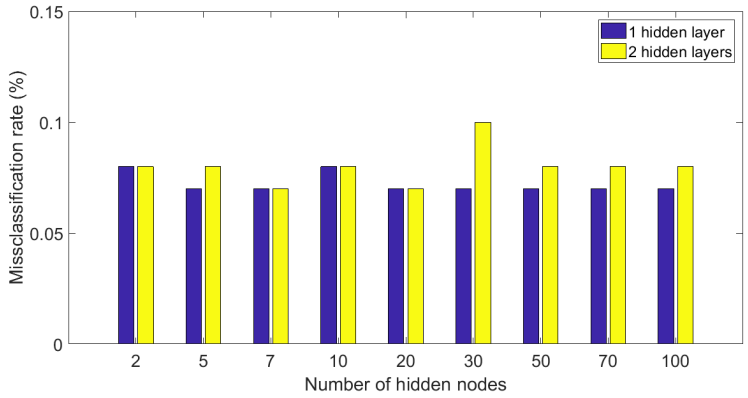

Figure 4: *Artificial Neural Network - Validation error, E_{va}, vs. number of hidden nodes for 1 respectively 2 hidden layers, fitted on the full sample. Blue is 1 layer, yellow is 2 layers.*

# of Hidden Nodes	# of Iterations, 1 Hidden Layer	# of Iterations, 2 Hidden Layers
2	247	151
5	136	189
7	88	78
10	86	73
20	76	83
30	103	56
50	66	51
70	70	54
100	61	51

Table 10: *Artificial Neural Network - Table over the number of iterations required by the LBFGS-algorithm until convergence versus number of hidden nodes for* 1 *and* 2 *hidden layers respectively, fitted on the full sample.*

The results in figure 4 show that the number of hidden nodes seems to have very little influence on the validation error. Considering the magnitude of the error the variation seen is insignificant. Probably, with such a small misclassification rate, it is sufficient with a small number of hidden nodes and the error does not decrease when increasing the number. Table 10 indicate that the number of iterations goes down as the number of hidden nodes increases. Particularly there is a larger number of iterations required for the algorithm to converge when there are very few hidden nodes, i.e. 2 and 5. For 7 and upwards the number of iterations is more constant. This suggests that it might be more computationally effective to use at least 7 nodes and there seems to be a certain risk that the optimization algorithm will have more difficulty to obtain a converged solution if too few nodes are used. However, a large number of nodes does not yield a lower misclassification rate and even though the number of iterations goes down slightly, each iteration probably requires longer computational time. Conclusively $7 - 20$ hidden nodes seems most appropriate.

In figure 5, the case of how the validation error varies with choice of activation function is presented. 1 hidden layer is used, and the effect of activation function is evaluated on 5, 10 and 50 hidden nodes respectively. The results show that the logistic, hyperbolic tangents and identity activation function almost perform equally well. However, the exponential function has a significantly larger validation error and seems not to be suitable for the prediction problem at hand. Note that there is much lower relative difference in misclassification rate between the exponential activation function and the other activation functions when 50 hidden nodes are used, compared to 5 and 10 nodes. Thus, a larger number of hidden nodes seems to in some sense compensate for the bad performance of the exponential activation function.

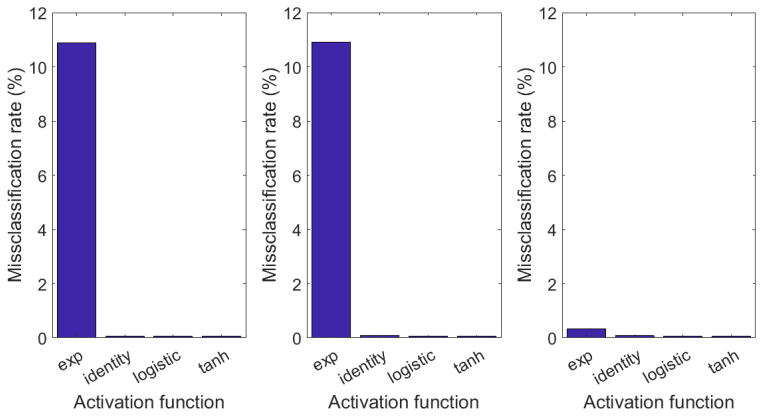

Figure 5: *Artificial Neural Network - Validation error, E_{va}, vs. activation function for 1 hidden layer with 5, 10 and 50 hidden nodes respectively, fitted on the full sample. Left panel: 5 hidden nodes. Middle panel: 10 hidden nodes. Right panel: 50 hidden nodes.*

4.2.2 Validation with the Small Sample

For the smaller sample, the study of evaluating validation error versus number hidden layers, number of hidden nodes and choice of activation function, was done in the same manner as for the full sample with the same setup. However, we do not care to evaluate effect of optimization algorithm for this subsample and $LBFGS$ is used in all cases. In figure 6 the case of evaluating the effect of number of hidden layers is presented and table 11 shows the corresponding number of iterations required by $LBFGS$ to converge. Just as for the full sample we use the logistic activation function and 50 hidden nodes on each hidden layer.

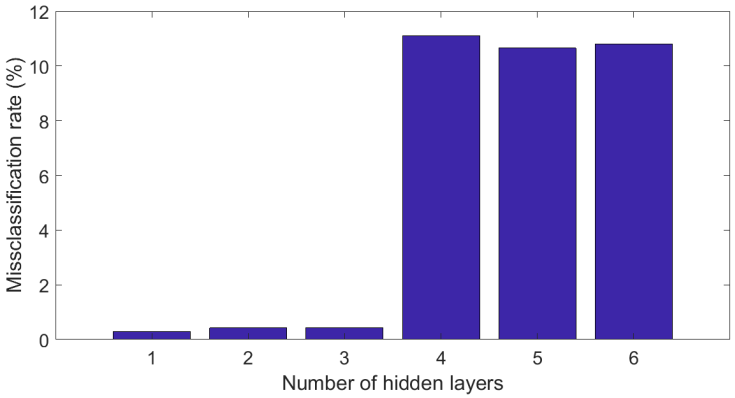

Figure 6: *Artificial Neural Network - Validation error, E_{va}, vs. number of hidden layers, fitted on the small sample.*

# of Hidden Layers	# of Iterations, $LBFGS$
1	19
2	27
3	28
4	8
5	7
6	8

Table 11: *Artificial Neural Network - Table over the number of iterations required by the LBFGS-algorithm until convergence versus number of hidden layers, fitted on the small sample.*

From figure 6 we see that when fitting ANN on the small sample, the same surprising results can be seen when it comes to number of layers as when fitting on the full sample. When increasing from 3 to 4 hidden layers the validation error significantly increases and remain on this higher level as the number of hidden layers is increased further. As expected, the magnitude of the error is larger for the small sample for 1 and 2 hidden layers, however, for 3 and upwards there is no clear difference. Table 11 shows that the number of iterations required by $LBFGS$ is significantly lower than before which is expected when the amount of training data is significantly lower as well. We can also note how the number of iterations go down when 4 or more hidden layers are used, just as was seen when the full sample was used for fitting. This strengthens the believe that is a part of explaining why the jump in validation error occurs. We discuss this more in section 5.2.

In figure 7 presents the significance of number of hidden nodes on each hidden layer and table 12 shows the corresponding number of iterations required by $LBFGS$ to

reach convergence. Just as for the full sample we use the logistic activation function.

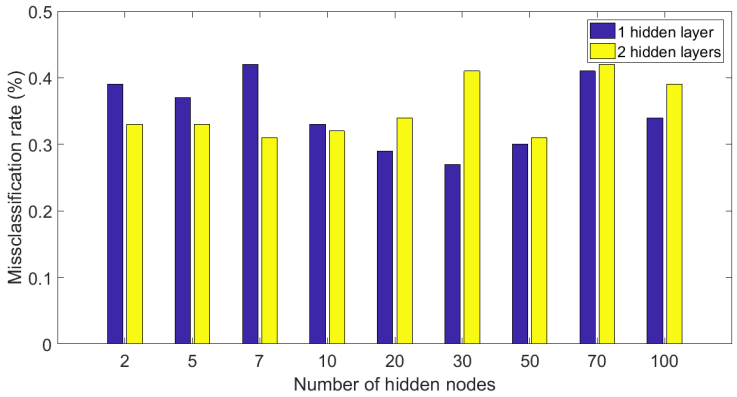

Figure 7: *Artificial Neural Network - Validation error, E_{va}, vs. number of hidden nodes for* 1 *and* 2 *hidden layers respectively, fitted on the small sample. Blue is* 1 *layer, yellow is* 2 *layers.*

# of Hidden Nodes	# of Iterations, 1 Hidden Layer	# of Iterations, 2 Hidden Layers
2	18	21
5	17	21
7	19	18
10	18	19
20	16	12
30	24	29
50	19	27
70	35	23
100	22	30

Table 12: *Artificial Neural Network - Table over the number of iterations required by the LBFGS-algorithm until convergence versus number of hidden nodes for* 1 *and* 2 *hidden layers respectively, fitted on the small sample.*

Figure 7 shows that there is a higher magnitude on the validation error and larger variations for different number of nodes for the small sample, compare to the full sample in figure 4. The larger variations probably depend on the small size of the training set, where the resulting validation error is more dependent on each single observation being used for training. A few observations close to the decision boundary can heavily impact the result. Observe though, with respect to the magnitude of the error, the variations are small. Considering how low the magnitude of the error is and the fact that the variations seems almost random due to the size of

the training set, we cannot draw any other conclusion than that the number of hidden nodes seems to have very little influence on our problem, no matter sample size. Table 12 show that that the number of iterations used by $LBFGS$ has overall decreased when a small sample of data is used for training compared to a large sample. The much smaller amount of data requires less iterations. We also see that there is minor variation in number of iterations w.r.t. to both number of hidden nodes and for both 1 and 2 hidden layers. Hence there is not the same indication of few nodes yielding more iterations as for the full sample. With a much smaller amount of data for training, it also become more random which observations are used to fit the model and the results might vary due to which observations that have been extracted from the full sample. In this case there are likely quite few training pairs which are difficult to classify, i.e. close to the decision boundary, and then few hidden nodes and few iterations are required to find an appropriate solution. It can be that if a new training sample of 300 is extracted, with a more complex set of observations, the results might be different. In section 6 we discuss more on how the sample size and the kind of observations obtained affect the results.

Finally figure 8 presents the case of validation error versus choice of activation function. Just as for the full sample, 1 hidden layer is used, and the effect of activation function is evaluated on 5, 10 and 50 hidden nodes. The results show that also for the small sample the exponential activation function has a larger validation error than the logistic and tangents hyperbolic activation function, even though the difference is not as dramatic. The error decreases as more nodes are used which also coincide with the results from the full sample. Here however, the identity function also has a slightly larger validation error than the logistic and tangent hyperbolic activation function which also decreases as more nodes are used. Thus, the results might be more sensitive to a linear activation function when a small amount of training data is used.

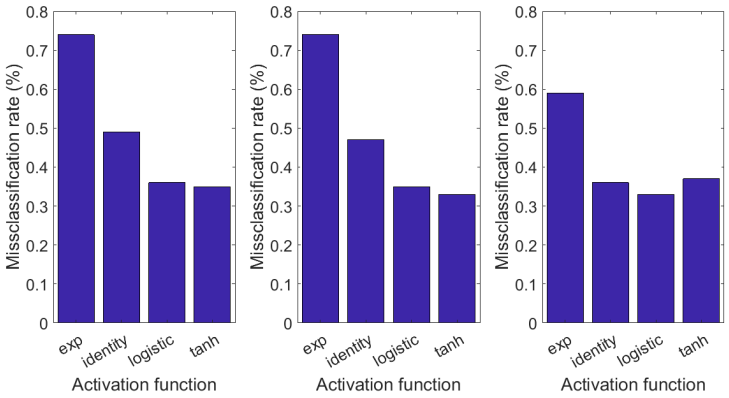

Figure 8: *Artificial Neural Network - Validation error,* E_{va}*, vs. activation function for* 1 *hidden layer with* 5, 10 *and* 50 *hidden nodes respectively, fitted on the small sample. Left panel:* 5 *hidden nodes. Middle panel:* 10 *hidden nodes. Right panel:* 50 *hidden nodes.*

Conclusively the validation error becomes larger as a significantly smaller sample size is used for training, but considering the low magnitude of the validation error the difference is in fact small. The same conclusions hold on suitable number of hidden layers and nodes, where a smaller number seems to yield best performance, for both sample sizes. We have also seen that the exponential activation function performs significantly worse in both cases but the otherwise there is no significant difference between the other activation functions.

4.3 Validation of the Support Vector Machine

The support vector machine has two main tuning parameters to set, penalty C and kernel function. Due to limitations in SAS only interior point methods are available as optimization algorithm and thus is the only optimization algorithm used. Also, only linear and polynomial kernels of $2nd$ and $3rd$ degree are available. In figure 9 validation error versus penalty C is presented for the three possible choices of kernel function for the full sample. In figure 10 the same results are presented but for the small sample. For SVM there actually a third tuning parameter to consider, namely, investigating how threshold affect the results. Unfortunately, this was not possible for LR and ANN due to limitations induced by SAS and that is the reason why this has not been done in those cases as well. Instead of using the *sign* function as classifier, as seen in 37, we model the hyperplane $f(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta} + \beta_0 = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j) + \beta_0$ such that the classifier becomes

$$G(\mathbf{x}) = \begin{cases} 1, & f(\mathbf{x}) \geq t, \\ 0, & f(\mathbf{x}) < t, \end{cases} \tag{52}$$

and vary the threshold t in the interval $[-1, 1]$. In figure 11, ROC curves (true

positive classification rate versus false positive classification rate) are presented for the three possible choices of kernel function, with threshold $t \in [-1, 1]$ as varying parameter. We show the results for both the full sample and for the small sample. Table 13 show the number of iterations required for the interior point method to fit the SVM. Results are presented for both the full and the small sample and for polynomial kernel functions of $1st$, $2nd$ and $3rd$ degree. $C = 50$. 500 is the set maximum allowed number of iterations and reaching this value thus not imply convergence.

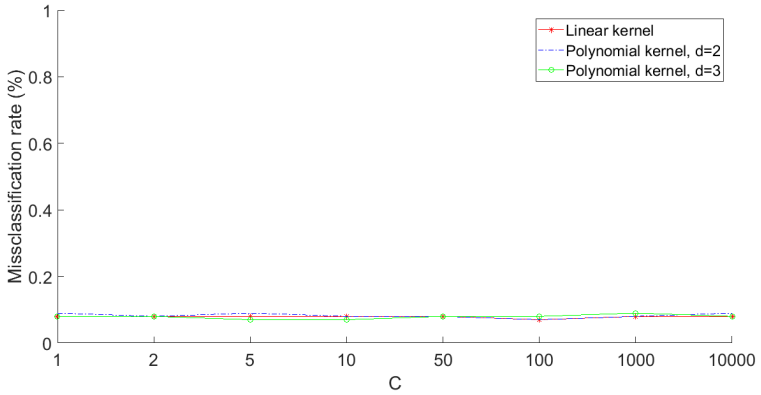

Figure 9: *Support Vector Machine - Validation error, E_{va}, vs. penalty parameter C for three different kernels, linear, 2nd degree and 3rd degree polynomial, fitted on the full sample.*

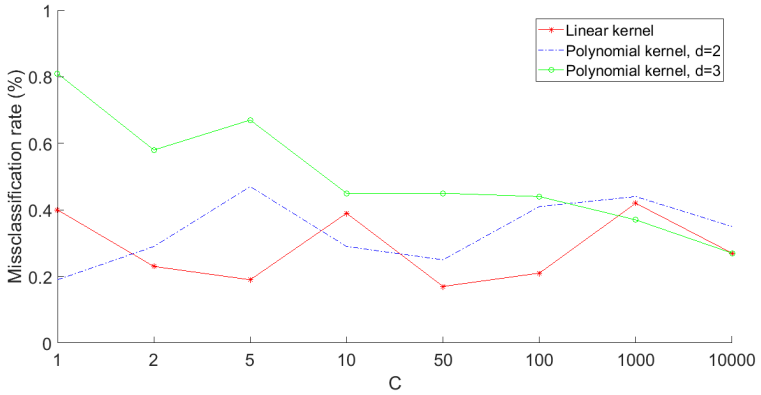

Figure 10: *Support Vector Machine - Validation error, E_{va}, vs. penalty parameter C for three different kernels, linear, 2nd degree and 3rd degree polynomial, fitted on the small sample.*

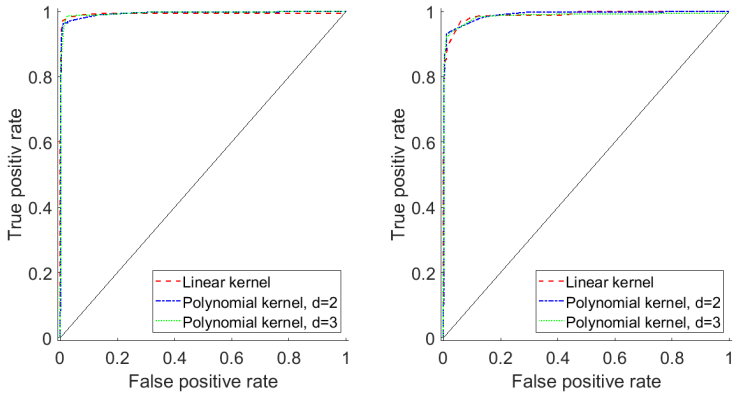

Figure 11: *Support Vector Machine - ROC curve for three different kernels, linear, 2nd degree and 3rd degree polynomial. Left: ROC curve fitted on the full sample. Right: ROC curve fitted on the small sample.*

Polynomial Kernel Degree	Full Sample	Small Sample
1	256	83
2	378	101
3	403	114

Table 13: *Support Vector Machine - Table over the number of iterations required to reach convergence when fitting the SVM using the interior point optimization method, on the full and small sample respectively, for linear, 2nd degree and 3rd degree polynomial kernels.*

The results in figure 9 and in figure 10 show that the validation error is, as for LR and ANN, very small but also slightly larger when a smaller sample is used for training. The results also indicate that kernel function have negligible effect on the validation error although, for the small sample, for a very low value on C, the $3rd$ degree polynomial kernel performs slightly worse than the others. However, even if there is a factor 4 in difference compared to the $2nd$ degree polynomial kernel, the magnitude on the validation error is so low that the difference must be considered insignificant. Conclusively this suggest the data is linear and there is no need to use a non-linear transformation. Studying figure 9 it seems that increasing penalty C has no effect on the validation error for the full sample. Figure 10 show that for the small sample one can argue that the error decreases slightly with a larger C for a $3rd$ degree polynomial kernel, but once again one must consider that the error is so small that the difference is insignificant. Conclusively a linear kernel is most suitable, and we may set C quite low. The ROC curves in figure 11 are very similar w.r.t. kernel function, i.e. the curves almost coincide. There is not a significant difference w.r.t. sample size either, even though the curves are slightly closer to the

top left corner, with a shaper turn, when the full sample is used for training. The ideal ROC curves should have a curvature as far up the top left corner as possible, with a near perpendicular turn in this corner. We see that this is almost the case when the full sample is used but not quite for the smaller sample. The top left corner corresponds to $t \approx 0$ in both cases and thus the *sign* function is suitable as classifier. The results in table 13 show that just as for the LR-model and ANNs the number of iterations is clearly higher as the full sample is used which corresponds to the significant larger amount of data used in the fitting process. It can also be noted that the number of iterations is much higher for the SVM compared tom ANN for both sample sizes, no matter settings on tuning parameters. Conclusively there is no significant difference in validation error due to different sample sizes for training, neither does tuning setting depend on sample size.

4.4 Comparison of the Supervised Learning Methods

Finally, the results from a comparison of the three supervised learning methods' performance on the test set, for corresponding chosen configuration on the tuning parameters, are presented.

For the ANN obviously 1 hidden layer is the optimal choice since there is no gain in predictive performance by adding more layers. 7 hidden nodes are set on this layer. We have seen that the validation error is extremely small and the setting on number of hidden nodes seems insignificant w.r.t. to the error. Thus, even though the validation error is almost equally as low for both 2 and 5 nodes, for both sample sizes, we have seen that there seems to be no significant increase in number of iterations required to fit the model and thus no increase in computational time if 7 nodes are used, arguably the opposite. Using the larger choice of 7 nodes, the model is potentially more robust to handle future more complex data, where perhaps observations are not so easily classifiable. However, it is not suitable to choose more hidden nodes than necessary and therefore there is no need for more than 7. Activation function is chosen to be the logistic function. Even though the simpler identity function performs well, the logistic function will handle non-linearities in the data better. Thus, if the model should predict well on new data, which potentially contains non-linearities, it seems more suitable to choose a non-linear activation function. We also saw how the relative predictive performance decreased slightly for the identity function when only 300 observations was used for training. The exponential function has does not perform well and should of course not be used. The non-linear tangents hyperbolic activation function performs equally as good as the logistic function and could be an alternative as well. However, the logistic function suitably maps the input in to the interval $(0, 1)$ and is also the standard activation function in ANNs.

For SVM then linear kernel seems to be the most natural choice as data used for validation seems linear. However, just as for ANN, if the model is applied on new data, which potentially contains non-linearities, a polynomial kernel of $2nd$ or $3rd$

degree could be considered as well. In this case though, we choose the linear kernel. A high C does not decrease the validation error significantly, so setting it high seems unnecessary, especially since the risk of over-fitting increases. We let $C = 50$, which also corresponds to the lowest validation error for the smaller sample. Threshold is set to $t = 0$, corresponding to the standard *sign*-function as classifier. For the linear kernel and for both sample sizes, this threshold approximately corresponds to the point in the top left corner on the ROC curve, which is the point yielding optimal predictive performance. The testing results are presented in figure 12 for both the full sample and the small sample.

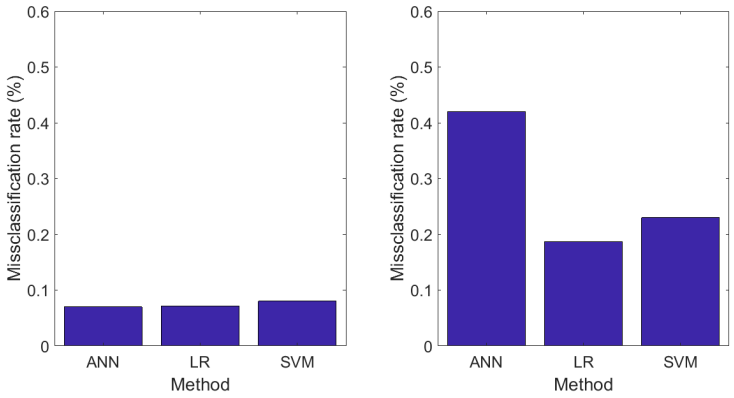

Figure 12: *Comparison - Artificial Neural Network (ANN), Logistic Regression (LR) and the Support Vector Machine (SVM). Testing error, E_{te}, vs. method, for chosen optimal configuration on the tuning parameters. Left figure: fitted on full sample. Right figure: fitted on small sample.*

The results in figure 12 show that the three supervised leaning methods have very similar predictive performance, with a very low test error, for both sample sizes. ANNs have a slightly higher test error when the smaller sample is used for fitting but considering the low magnitude of the error the difference is not significant. As has been discussed previously this is probably a random occurrence due to the small training sample and the increased significance of each single training observation and the randomness of how each gets classified. If only considering which model yields the lowest test error, it is not possible to conclude that one is better than the other. We discuss more about which one to choose in section 5.

5 Discussion

5.1 Derivation of Target and Predictors

The objective of this thesis was to explore the three supervised learning methods, logistic regression (LR), artificial neural networks (ANN) and the support vector machine (SVM), ability to predict household relationships To apply supervised learning methods, we have presented a technique which pairs individuals under the hypothesis H_0, and then construct a test of significance. The target Y and predictors $\mathbf{X} = (X_1, ..., X_p)$ could be coded, transforming the problem into a binary classification problem. Using this technique, a sample, for fitting, tuning, evaluation and comparison could be generated by random pairing of individuals. Since the available data only contain information about individuals living i single households and households consisting of two individuals, only these this case has been tested. However, this pairing idea holds for larger household constellations as well. All candidates that are suspected to share a household relationship should be paired with each other. All pairs predicted to be true, are also predicted to share a household relationship. For instance, if one individual is predicted to have a household link towards three other individuals they would together be predicted to form a household of four individuals. A potential problem when one individual can be predicted to share a household relationship with multiple other individuals, is that the risk of predicting false non-existing household relationships might increase. A solution to this can be to increase the threshold for when H_0 should be predicted true, however then the risk of missing household links increases as well.

The main difficulty of this pairing approach is that the number of possible pairs may rapidly become computational intractable when considering practical implementation of the method. If we have a large data set of individuals and wish to sort them into their household affiliation, the number of pairs to be tested would be very large. If households consisting of more than two individuals are considered the complexity of the problem increase even further. In this case, it would be suitable to first sort individuals into larger groups. This could for example be based on one or several common attributes considered necessary to share a household relationship i.e. first sorting individuals by e.g. zip-code and the apply the pairing method on those with same zip-code. However, those cases not fulfilling the requirements would be neglected and we would risk missing some less apparent relationships using this approach. Another alternative is to use the fact that for many individuals, the classification complexity is very low, and they are easily sorted into their household affiliation with high probability. Paired individuals fulfilling a few criteria, such as having the same address (street name, street number, city, zip-code, etc,) same last name, different gender, small age difference, common account ownership, would already be considered to have a very high probability of sharing a household relationship. Thus, they can be directly classified accordingly without applying any statistical machine learning models. The opposite holds as well, i.e. paired individuals sharing almost no attributes can be directly classified

as not sharing a household relationship. These two groups of individuals would be separated from the data set and only the much smaller remaining group would with much higher classification complexity would be classified by the pairing method. Clearly, a combination of both approaches can be used as well if necessary.

In all cases, no matter the size of the possible household constellations, a sample to train the models must be generated. The idea presented in section 2.2, where a mixture of random pairings and parings based on household affiliation forms one sample, would normally always be applicable. There could be other ways to generate a sample, but the resulting structure should be the same.

5.2 Performance of the Supervised Learning Methods

The results in section 4.4 have shown that both LR, ANNs and the SVM perform well in predicting household relationships. There are also relatively small variations in performance when modifying the settings on the corresponding tuning parameters. They all have extremely low misclassification rate and only considering this aspect it is not possible to conclude which one is preferable. Therefore, the focus of performance evaluation was shifted to the computational complexity aspect. The SVM is a complex method, requiring significantly longer computational time and more computational power than the others. Therefore, this method is considered least attractive to use and directly excluded as a method for predicting household relationships. ANN is also a complex method, but it does not require the same computational time and power as the SVM. It is theoretically a quite simple method but has the power to handle non-linear complex problem well. LR is the least complex method, significantly requiring the least computational time and power. This method also gives an exact form on f, the function relating predictors $\mathbf{X} = (X_1, ..X_p)$ and target Y, and also gives an interpretation on the predictors. Each predictor has an estimated weight and one can analyze the predictor variable's significance for predicting Y. This is a clear advantage compared to ANNs. The network structure and the estimated weights in an ANN is difficult to interpret and the form of f is unknown. With more layers and nodes, the network becomes even harder to understand. However, the idea with an ANN is only to give a good prediction, not to estimate the significance of each predictor. Given multiple input predictors and enough training data, the network has the power to optimally fit all the weights to find the most accurate prediction of the target, even if data is complex and non-linear. This yield good predictive power, but no concrete knowledge on how target and predictors are related. Conclusively, for data of the kind use in this thesis, logistic regression seems to be the best method of choice for predicting household relationships. However, for future data, on which the model should be applied, there might be a risk that it is more complex and has a non-linear behavior. In that case the artificial neural network should be considered as well. If possible, it would be suitable to test both methods on a smaller test set to see which performs best.

Regarding the small magnitude of the misclassification rate, for both sample sizes and for all three supervised learning methods. As mentioned earlier, this suggests that the classes are infinitely far away from each other and they are easily separable. A very large number of individuals has been used in the study, resulting in a very large sample of paired individuals. Of these pairs, a large majority are either individuals who have been randomly paired and share few attributes and no household relationship, or individuals sharing a household relationship and having multiple attributes in common. It is only a minority of the observed pairs who either share no household relationship but still have several attributes in common or share a household relationship but does not have multiple attribute in common. Conclusively the classification complexity is very low. Considering this, most observations are easily classifiable, and one might almost directly classify a large part of the observations by studying the predictors $\mathbf{X}$, without applying any statistical machine learning methods. Those observations that are much more difficult to classify are in such small minority and the total number of observations is so large, that the misclassification rate becomes extremely small. Since the small sample is extracted from the full large sample, the distribution of difficult and non-difficult observations to classify will be almost the same, i.e. the structural pattern of the underlying population distribution is preserved. Thus, the misclassification rate is very small in this case as well. To see any significant difference on the misclassification rate, one would have to manually identify pairs who would be considered difficult to classify and apply the methods on these.

One might also note that since the underlying population distribution of the sample sizes is equal, and the misclassification rate is so small, this leads to no difference in optimal settings on tuning parameters between the sample sizes. Even though there are more fluctuations in validation error as tuning configurations are modified in the small sample, compared to the full sample where tuning configurations in most cases have no effect at all (expect w.r.t. number of hidden layers in ANN), the variations seen are generally insignificant considering the small magnitude of the misclassification rate. For a small training set the fitting is much more dependent on each single observation and therefore the results might vary, and we would see fluctuations in misclassification rate depending on which observations are extracted. However, for ANNs there is a small difference for the sample sizes when the identity activation function is used which is interesting to discuss. Probably it is only a random occurrence. However, it could be that the smaller data set small non-linearities becomes more apparent and a non-linear activation function does not perform as well in that case. Conclusively though, the same configuration is suitable independently of sample size for training, neither is it significant for prediction performance for any of the supervised learning models.

Regarding the significance of the estimates for the LR-model, seen in Appendix A and Appendix B. It is quite expected that non-specific predictors such as city and country are not significant in the model. Almost all individuals are from the same country and many individuals may live in the same city no matter if they share

a household relationship or not. Hence knowledge of these attributes does not help to predict household relationships. Since for a large majority of the individuals in the data set, official and non-official attributes, such as last name, are equal and thus it is very natural that the corresponding predictors are not significant. The results have shown that more estimates are zero as the model is fitted on the small sample compared to the full sample. This give an indication that with less data for training, the model might consider fewer predictors and a couple of predictors have more influence in predicting household relationships. In the aspect of attempting to predict more subtle household relationships between individuals, where perhaps last name, address etc. are not equal, one would want to consider many predictors. In that case it would be necessary to use a sufficient size on the training sample including multiple observations of these subtler relationships. An important and interesting remark: removing insignificant predictors have been attempted as well but this does not notable affect the misclassification rate.

Regarding the significant jump in misclassification rate when expanding a ANN from 3 to 4 layers (4 to 5 for SGD), as is seen for both the full sample in and for the smaller subsample. This is very surprising since theoretically the error should decrease, or if the data is very linear and easy to classify, remain at a low relative constant level as the number of layers increase. One could consider overfitting as possible explanation, however the amount of data is very large and there exist no real difference between the training error and the validation error. Therefore, the possibility of over-fitting due to a too complex network with too many hidden layers is almost non-existing. Another explanation for this behavior might be the optimization algorithm or how the optimization in SAS is programmed. The convergence occurs extremely fast when there are several hidden layers, and this suggest that the convergence criteria in the optimization algorithm or defined by SAS is inappropriate, making the algorithm find a local minimum and not the sought global minimum corresponding to the best solution (not overfitted). When multiple hidden layers are used the optimization, algorithm seems unable to find it. How to avoid this premature convergence have not been investigated since clearly there is no need for multiple hidden layers, but could be an issue for future research.

For ANN's the number of hidden nodes seems to be almost insignificant for the accuracy of predicting household relationships w.r.t. the data set available. However, we have seen that when the exponential activation function is used, there is some indication that the number of nodes have some effect though. Primarily, the exponential activation function yields a much higher error than the other activation functions, but we can also see that the difference in misclassification rate compared to the other activation functions becomes much smaller when 50 hidden nodes are used compared to 5 or 10. Thus, with a larger number of hidden nodes, the negative effects of the exponential function significantly decrease, and this result indicates that perhaps a network with a larger number of hidden nodes has the power to compensate for a bad choice of activation function. This may also suggest that the risk of bad predictive performance decreases if a larger number of hidden nodes is

used.

Concerning the fact that both the value on penalty parameter C and choice of kernel function have very little influence on the validation error and thereby the predictive performance for the SVM. Theoretically, increasing C should decrease both the training error and the validation error up to a certain point, from which the training error continue to decrease whereas the validation error starts to increase. In fact, C regulates how much the decision boundary should adapt to the observations in the training set and thereby a high value on C should make the solution fit the training data very well, but also probable cause overfitting and yield a higher validation error. We should thus see a minimum of the validation error for a certain value on C. However, if data is approximately linear and we use low degree polynomial kernel functions ($d \in \{1, 2, 3\}$), C only regulates the slope or a small curvature of the hyperplane separating the data. On the other hand, for highly non-linear data, using a non-linear more complex kernel such as radial basis, C would have a much larger impact on how the hyperplane curves around the data (in the non-linear space). Conclusively value on C will have small impact on validation error when data is linear and when low-degree polynomials are used. Remember also that out data is very separable and thus most observations are not close to the decision boundary, making choice of kernel and C almost insignificant.

5.3 Suggestions For Future Research

The main interesting aspect which has not been investigated in this thesis is to investigate how well the models predict household relationships when on applied on new fresh data. In this thesis, the data records available contain detailed information about each individual, e.g. exact address, last name and account ownership. These are all factors, which for a large majority of individuals, can imply a strong household link, i.e. if this information is available, most individuals' household relationship is easily predictable. Therefore, for such a big population, the misclassification rate is very small as a large majority of the observations can be well separated into the two classes. It would be interesting to see what happens of some of this information is lacking, for instance if address in terms of street name and number is not available. How well would the models perform then? Also, one would like to increase the classification complexity by investigating a smaller population of individuals who belong to the minority of individuals whose household relationship is not as easy to predict. This would be individuals who share a household relationship but do not have the same address or last name etc. It might not be possible to predict their household relationships with accuracy if they do not share the same address or last name. The misclassification rate would most definitely increase in this case. Also, only households consisting of one or two individuals have been investigated in this thesis. Therefore, it would be interesting to use a data set where household relationships can consist of more than two individuals and the complexity of the problem increases. As previously mentioned, the paring idea, where individuals are tested against each other two and two, holds for larger

household constellations as well. However, from the results in this thesis, it is not known what happens with the predictions when H_0 can be true for several different pairs including the same individual.

Notable is that a household relationship s is in fact a dynamical constellation, changing over time. Most people do share living expenses with the same partner or remain in the same household constellation during their whole live span. In this thesis no time aspect has been considered, and these dynamical constellations have been modeled as static w.r.t time. However, if one truly wants to predict a household relationship a certain point in time i.e. ensure that the prediction is valid at the time it is made and not only at the time the data has been collected, the time aspect must be considered. Then we would have some kind of time series. However, the solution idea presented in this thesis is perhaps not applicable anymore or it may very well not be possible to consider this time aspect at all. This is something to consider though when attempting to predict household relationships.

Finally, it would be interesting to see what happens if more than one possible household relationship is allowed for an individual. We have only investigated individuals who belong to one household constellation. Usually people have one main partner who they share living expenses with. However, it is possible that an individual has separate household relationships with separate partners. For instance, an individual could be married with one partner and they jointly own an apartment. The same individual might also own a separate summer house together with a friend and the married partner is not part of this ownership. In this case the individual would belong to two different household relationships in some sense and it would be interesting to see if it possible to predict both by using the pairing idea presented in this thesis. Perhaps only the strongest relationship can be identified.

6 Conclusions

This thesis objective was to explore the ability of logistic regression (LR), artificial neural networks (ANN) and the support vector machine (SVM) to predict household relationships between individuals. Data records over a limited population of individuals, where household affiliation and several attributes were registered, were available for this task. In order to apply these methods for predicting household relationships, a target Y and a vector of p input predictor variables $\mathbf{X} = (X_1, ..., X_p)$, based on the set of p attributes registered on each individual, had to be defined. We have presented a solution idea which forms pairs of individuals that are hypothesized to share a household relationship, and then tests if this hypothesis H_0 holds true or not. This idea enable us to define a binary target $Y \in \{0, 1\}$, with coding, 1 if H_0 is true and 0 if H_0 is false, which transforms the initially vague and intangible problem to a much more tangible problem in the form of standard binary classification. A sample for fitting the methods could be generated by randomly pair individuals and using the information form the data records to code the corresponding outcome on Y and $\mathbf{X}$ for the random pairs. For evaluation and tuning of the three supervised learning methods, the sample was split into a training set, a validation set and a test set.

The validation error has proven to be very small and the two classes, H_0 is true and H_0 is false, are far away from each other and thereby the observations are very easy to classify. For LR there is no tuning to be made. For the ANN we have seen that there is no need for multiple hidden layers and performance decrease as the more are used. We have also seen that the number of hidden nodes have insignificant effect on the validation error. All activation functions have proven to perform well, except the exponential function. In the end we have concluded that 1 hidden layer, with 7 hidden nodes and the logistic activation function is most suitable for the prediction task. Data has proven to be linear and thus for the SVM a linear kernel is most suitable. The penalty C has shown to have no significant impact on the validation error and the most suitable threshold is $t = 0$. We have has thus set a linear kernel, with $C = 50$ and the *sign*-function as classifier as optimal tuning.

Comparing the three supervised learning methods we have seen that w.r.t. test error, in term of misclassification rate, there is no significant difference between the methods performance and the all predict very well. However, due to complexity, computational time and power, we have concluded that SVM is the least suitable method to use whereas LR most suitable. LR also have the advantage to give an interpretation on the significance each input predictor/attribute has for predicting household relationships. However, the ANN handles complex and non-linear data better than LR, therefore, for future application of the model, we find it suitable to consider ANN as well. If possible, we recommend using a small test set and test both the ANN model and the LR model and compare their performance. Then one can see the complexity of the data and see if there is non-linear behavior and thereafter chose between the two methods accordingly.

References

[1] Hastie T, Tibshirani R, Friedman J, *The elements of Statistical Learning (2nd edition)*, Springer-Verlag, 2009, 12th printing.

[2] James G, Witten D, Hastie T, Tibshirani R, *Introduction to Statistical Learning (2nd edition)*, Springer-Verlag, 2015, 6th printing.

[3] Izenman A.J, *Modern Multivariate Statistical Techniques*, Springer-Verlag, Temple University, 2008.

[4] Improve Neural Network Generalization and Avoid Overfitting
https://se.mathworks.com/help/nnet/ug/improve-neural-network-generalization-and-avoid-overfitting.html

[5] Sarle, W.S., ed. (1997), *Neural Network FAQ, part 2 of 7: Learning, periodic posting to the Usenet newsgroup comp.ai.neural-nets*
ftp://ftp.sas.com/pub/neural/FAQ2.html

[6] The Newton-Raphson Method
http://web.mit.edu/10.001/Web/Course_Notes/NLAE/node6.html

[7] Why You Should Use Cross-Entropy Error Instead Of Classification Error Or Mean Squared Error For Neural Network Classifier Training
https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/

[8] Golik P, Doetsch P and Ney H, *Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison*, 2013
https://www-i6.informatik.rwth-aachen.de/publications/download/861/GolikPavelDoetschPatrickNeyHermann–CrossEntropyvs.SquaredErrorTrainingaTheoreticalExperimentalComparison–2013.pdf

[9] SGD
terminus.sdsu.edu/SDSU/Math693a/Lectures/18/lecture.pdf

[10] LBFGS
ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent

[11] Wright.J.S, *Primal-Dual Interior-Point Methods*, SIAM, 1997

Appendices

Appendix A

Parameter Estimates							
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**	**95% Confidence Limits**	
Intercept	1	77.493380	12.167984	40.5594	<.0001	53.64457	101.34219
hh_lastname_off_du -1	1	-6.721325	1.013014	44.0230	<.0001	-8.70680	-4.73586
hh_lastname_off_du 1	0	0	.	.	.	.	.
hh_lastname_kv_du -1	1	1.623665	1.014935	2.5593	0.1096	-0.36557	3.61290
hh_lastname_kv_du 1	0	0	.	.	.	.	.
hh_gen_du -1	1	-1.596383	0.088391	326.1826	<.0001	-1.76963	-1.42314
hh_gen_du 1	0	0	.	.	.	.	.
hh_marstat_du -1	1	0.706992	0.092362	58.5930	<.0001	0.52597	0.88802
hh_marstat_du 1	0	0	.	.	.	.	.
abs_diff_years5710 0	1	14.150301	3.824433	13.6898	0.0002	6.65455	21.64605
abs_diff_years5710 1	1	1.649120	0.083775	387.5021	<.0001	1.48492	1.81332
abs_diff_years5710 2	1	0.934162	0.135849	47.2860	<.0001	0.66790	1.20042
abs_diff_years5710 3	1	0.639797	0.129886	24.2639	<.0001	0.38523	0.89437
abs_diff_years5710 4	0	0	.	.	.	.	.
hh_gemkont_du -1	1	-10.832642	0.270366	1605.3326	<.0001	-11.36255	-10.30273
hh_gemkont_du 1	0	0	.	.	.	.	.
hh_adress_off_du -1	1	-6.613104	1.014357	42.5040	<.0001	-8.60121	-4.62500
hh_adress_off_du 1	0	0	.	.	.	.	.
hh_adress_kv_du -1	1	1.890845	1.029008	3.3766	0.0661	-0.12597	3.90766
hh_adress_kv_du 1	0	0	.	.	.	.	.
hh_adress_an_du -1	1	-8.149974	6.501298	1.5715	0.2100	-20.89228	4.59233
hh_adress_an_du 1	0	0	.	.	.	.	.
hh_city_off_du -1	1	-1.984301	0.354317	31.3640	<.0001	-2.67875	-1.28985
hh_city_off_du 1	0	0	.	.	.	.	.
hh_city_kv_du -1	1	-1.333793	0.356241	14.0181	0.0002	-2.03201	-0.63557
hh_city_kv_du 1	0	0	.	.	.	.	.
hh_city_an_du -1	1	-0.598391	0.718235	0.6941	0.4048	-2.00611	0.80932
hh_city_an_du 1	0	0	.	.	.	.	.
hh_pstaddrcntry_off_du -1	1	0.610688	2.100087	0.0846	0.7712	-3.50541	4.72678
hh_pstaddrcntry_off_du 1	0	0	.	.	.	.	.
hh_pstaddrcntry_kv_du -1	1	-0.996398	0.411510	5.8628	0.0155	-1.80294	-0.18985
hh_pstaddrcntry_kv_du 1	0	0	.	.	.	.	.
hh_pstaddrcntry_an_du -1	1	0.474284	0.270755	3.0685	0.0798	-0.05639	1.00496
hh_pstaddrcntry_an_du 1	0	0	.	.	.	.	.

Parameter Estimates							
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**	**95% Confidence Limits**	
hh_gatunamn_off_du -1	1	0.988338	0.859004	1.3238	0.2499	-0.69528	2.67195
hh_gatunamn_off_du 1	0	0	.	.	.	.	.
hh_gatunamn_kv_du -1	1	-2.234225	0.835187	7.1563	0.0075	-3.87116	-0.59729
hh_gatunamn_kv_du 1	0	0	.	.	.	.	.
hh_gatunamn_an_du -1	1	2.553180	6.454308	0.1565	0.6924	-10.09703	15.20339
hh_gatunamn_an_du 1	0	0	.	.	.	.	.
hh_gatunr_off_du -1	1	-0.895530	0.300599	8.8754	0.0029	-1.48469	-0.30637
hh_gatunr_off_du 1	0	0	.	.	.	.	.
hh_gatunr_kv_du -1	1	-0.303941	0.310747	0.9567	0.3280	-0.91299	0.30511
hh_gatunr_kv_du 1	0	0	.	.	.	.	.
hh_gatunr_an_du -1	1	-1.288239	0.943165	1.8656	0.1720	-3.13681	0.56033
hh_gatunr_an_du 1	0	0	.	.	.	.	.
hh_ext_off_du -1	1	-1.867739	0.268350	48.4427	<.0001	-2.39370	-1.34178
hh_ext_off_du 1	0	0	.	.	.	.	.
hh_ext_kv_du -1	1	-2.041528	0.268485	57.8189	<.0001	-2.56775	-1.51531
hh_ext_kv_du 1	0	0	.	.	.	.	.
hh_ext_an_du -1	1	-2.871550	0.808552	12.6130	0.0004	-4.45628	-1.28682
hh_ext_an_du 1	0	0	.	.	.	.	.
hh_tr_off_du -1	1	-2.278927	8.901179	0.0655	0.7979	-19.72492	15.16706
hh_tr_off_du 1	0	0	.	.	.	.	.
hh_tr_kv_du -1	1	-5.586905	8.856348	0.3980	0.5281	-22.94503	11.77122
hh_tr_kv_du 1	0	0	.	.	.	.	.
hh_tr_an_du -1	1	-1.842376	1.666114	1.2228	0.2688	-5.10790	1.42315
hh_tr_an_du 1	0	0	.	.	.	.	.
hh_lghnr_off_du -1	1	-2.711649	0.222416	148.6400	<.0001	-3.14758	-2.27572
hh_lghnr_off_du 1	0	0	.	.	.	.	.
hh_lghnr_kv_du -1	1	-2.224175	0.224202	98.4143	<.0001	-2.66360	-1.78475
hh_lghnr_kv_du 1	0	0	.	.	.	.	.
hh_lghnr_an_du -1	1	-4.314290	11.518584	0.1403	0.7080	-26.89030	18.26172
hh_lghnr_an_du 1	0	0	.	.	.	.	.
hh_other_off_du -1	1	-7.074034	2.824109	6.2744	0.0122	-12.60919	-1.53888
hh_other_off_du 1	0	0	.	.	.	.	.
hh_other_kv_du -1	1	-5.193101	2.817899	3.3963	0.0653	-10.71608	0.32988
hh_other_kv_du 1	0	0	.	.	.	.	.
hh_other_an_du -1	1	-7.618487	1.771582	18.4933	<.0001	-11.09072	-4.14625
hh_other_an_du 1	0	0	.	.	.	.	.
hh_zipcode_off_du -1	1	-2.301737	0.578793	15.8148	<.0001	-3.43615	-1.16732
hh_zipcode_off_du 1	0	0	.	.	.	.	.
hh_zipcode_kv_du -1	1	-0.651084	0.578136	1.2683	0.2601	-1.78421	0.48204

Parameter Estimates							
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**	**95% Confidence Limits**	
hh_zipcode_kv_du 1	0	0	.	.	.	.	.
hh_zipcode_an_du -1	1	-4.620985	1.084771	18.1465	<.0001	-6.74710	-2.49487
hh_zipcode_an_du 1	0	0	.	.	.	.	.
abs_diff_years	1	0.001968	0.000964	4.1686	0.0412	0.00007879	0.00386

Appendix B

Parameter Estimates							
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq	95% Confidence Limits	
Intercept	1	6.191875	13202	0.0000	0.9996	-25869	25881
hh_lastname_off_du -1	1	0.146219	10957	0.0000	1.0000	-21475	21475
hh_lastname_off_du 1	0	0	.	.	.	.	.
hh_lastname_kv_du -1	0	0	.	.	.	.	.
hh_lastname_kv_du 1	0	0	.	.	.	.	.
hh_gen_du -1	1	-1.692266	2445.052979	0.0000	0.9994	-4793.90804	4790.52351
hh_gen_du 1	0	0	.	.	.	.	.
hh_marstat_du -1	1	0.421587	886.992515	0.0000	0.9996	-1738.05180	1738.89497
hh_marstat_du 1	0	0	.	.	.	.	.
abs_diff_years5710 0	1	0.612042	5053.946731	0.0000	0.9999	-9904.94153	9906.16561
abs_diff_years5710 1	1	2.753906	5506.623846	0.0000	0.9996	-10790	10796
abs_diff_years5710 2	1	-0.156267	11780	0.0000	1.0000	-23089	23089
abs_diff_years5710 3	1	0.302061	3929.824700	0.0000	0.9999	-7702.01282	7702.61694
abs_diff_years5710 4	0	0	.	.	.	.	.
hh_gemkont_du -1	1	-2.568581	10982	0.0000	0.9998	-21528	21523
hh_gemkont_du 1	0	0	.	.	.	.	.
hh_adress_off_du -1	1	-4.183846	12632	0.0000	0.9997	-24763	24755
hh_adress_off_du 1	0	0	.	.	.	.	.
hh_adress_kv_du -1	1	-1.685291	12619	0.0000	0.9999	-24734	24731
hh_adress_kv_du 1	0	0	.	.	.	.	.
hh_adress_an_du -1	1	0.360632	13748	0.0000	1.0000	-26945	26946
hh_adress_an_du 1	0	0	.	.	.	.	.
hh_city_off_du -1	1	-4.458890	33234	0.0000	0.9999	-65142	65133
hh_city_off_du 1	0	0	.	.	.	.	.
hh_city_kv_du -1	1	-4.036512	33659	0.0000	0.9999	-65974	65966
hh_city_kv_du 1	0	0	.	.	.	.	.
hh_city_an_du -1	1	0.260048	15511	0.0000	1.0000	-30400	30401
hh_city_an_du 1	0	0	.	.	.	.	.
hh_pstaddrcntry_off_du -1	0	0	.	.	.	.	.
hh_pstaddrcntry_off_du 1	0	0	.	.	.	.	.
hh_pstaddrcntry_kv_du -1	1	0.391862	31010	0.0000	1.0000	-60778	60779
hh_pstaddrcntry_kv_du 1	0	0	.	.	.	.	.
hh_pstaddrcntry_an_du -1	1	0.258148	8333.936644	0.0000	1.0000	-16334	16334
hh_pstaddrcntry_an_du 1	0	0	.	.	.	.	.

Parameter Estimates							
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq	95% Confidence Limits	
hh_gatunamn_off_du -1	1	-12.751648	3524.931812	0.0000	0.9971	-6921.49105	6895.98775
hh_gatunamn_off_du 1	0	0	.	.	.	.	.
hh_gatunamn_kv_du -1	0	0	.	.	.	.	.
hh_gatunamn_kv_du 1	0	0	.	.	.	.	.
hh_gatunamn_an_du -1	0	0	.	.	.	.	.
hh_gatunamn_an_du 1	0	0	.	.	.	.	.
hh_gatunr_off_du -1	1	-5.673315	3936.211596	0.0000	0.9988	-7720.50628	7709.15965
hh_gatunr_off_du 1	0	0	.	.	.	.	.
hh_gatunr_kv_du -1	1	-3.519155	4129.025081	0.0000	0.9993	-8096.25960	8089.22129
hh_gatunr_kv_du 1	0	0	.	.	.	.	.
hh_gatunr_an_du -1	1	0.403459	10707	0.0000	1.0000	-20985	20986
hh_gatunr_an_du 1	0	0	.	.	.	.	.
hh_ext_off_du -1	1	2.886740	4053.357311	0.0000	0.9994	-7941.54761	7947.32109
hh_ext_off_du 1	0	0	.	.	.	.	.
hh_ext_kv_du -1	0	0	.	.	.	.	.
hh_ext_kv_du 1	0	0	.	.	.	.	.
hh_ext_an_du -1	1	0.959177	11551	0.0000	0.9999	-22639	22641
hh_ext_an_du 1	0	0	.	.	.	.	.
hh_tr_off_du -1	0	0	.	.	.	.	.
hh_tr_off_du 1	0	0	.	.	.	.	.
hh_tr_kv_du -1	0	0	.	.	.	.	.
hh_tr_kv_du 1	0	0	.	.	.	.	.
hh_tr_an_du -1	1	3.122658	11713	0.0000	0.9998	-22954	22960
hh_tr_an_du 1	0	0	.	.	.	.	.
hh_lghnr_off_du -1	1	1.559298	3247.480602	0.0000	0.9996	-6363.38572	6366.50432
hh_lghnr_off_du 1	0	0	.	.	.	.	.
hh_lghnr_kv_du -1	0	0	.	.	.	.	.
hh_lghnr_kv_du 1	0	0	.	.	.	.	.
hh_lghnr_an_du -1	0	0	.	.	.	.	.
hh_lghnr_an_du 1	0	0	.	.	.	.	.
hh_other_off_du -1	0	0	.	.	.	.	.
hh_other_off_du 1	0	0	.	.	.	.	.
hh_other_kv_du -1	0	0	.	.	.	.	.
hh_other_kv_du 1	0	0	.	.	.	.	.
hh_other_an_du -1	0	0	.	.	.	.	.
hh_other_an_du 1	0	0	.	.	.	.	.
hh_zipcode_off_du -1	0	0	.	.	.	.	.
hh_zipcode_off_du 1	0	0	.	.	.	.	.
hh_zipcode_kv_du -1	0	0	.	.	.	.	.

Parameter Estimates							
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**	**95% Confidence Limits**	
hh_zipcode_kv_du 1	0	0	.	.	.	.	.
hh_zipcode_an_du -1	0	0	.	.	.	.	.
hh_zipcode_an_du 1	0	0	.	.	.	.	.
abs_diff_years	1	0.026528	205.366051	0.0000	0.9999	-402.48354	402.53659