Data Fusion for Consumer Behaviour

Goran Dizdarevic
gorandi@kth.se
Degree Project in Mathematical Statistics, Second Cycle
Royal Institute of Technology (KTH)
Supervisor: Prof. Henrik Hult

June 2017

Abstract

This thesis analyses different methods of data fusion by fitting a chosen number of statistical models to empirical consumer data and evaluating their performance in terms of a selection of performance measures. The main purpose of the models is to predict business related consumer variables. Conventional methods such as decision trees, linear model and K-nearest neighbor have been suggested as well as single-layered neural networks and the naive Bayesian classifier. Furthermore, ensemble methods for both classification and regression have been investigated by minimizing the cross-entropy and RMSE of predicted outcomes using the iterative non-linear BFGS optimization algorithm. Time consumption of the models and methods for feature selection is also discussed in this thesis. Data regarding consumer drinking habits, transaction and purchase history and social demographic background is provided by Nepa. Evaluation of the performance measures indicate that the naive Bayesian classifier predicts consumer drinking habits most accurately whereas the random forest, although the most time consuming, is preferred when classifying the Consumer Satisfaction Index (CSI). Regression of CSI yield similar performance to all models. Moreover, the ensemble methods increased the prediction accuracy slightly in addition to increasing the time consumption.

I den här uppsatsen undersöks olika metoder för data fussion genom att anpassa ett antal statistiska modeller till empirisk konsument data och evaluera modellernas prestations nivå med avseende på ett antal statistiska mått. Syftet för modellerna är att predicera affärsrelaterade konsumentvariabler. I denna rapport har konventionella metoder såsom beslutsträd, linjära modeller och metoden med de närmsta grannarna föreslagits samt enkelskiktade neurala nätverk och den naiva bayesianska klassificeraren. Vidare har även ensemble metoder för både klassificeringar och regressioner undersökts genom att minimera kors entropin och RMSE av predicerade utfall med den iterativa icke-linjära optimeringsalgoritmen BFGS. Tidskonsumption för modellerna och metoder för selektion av prediktorer har också diskuterats i rapporten. Data gällande konsumenternas alkoholvanor, transaktion- och köphistorik samt social demografiska bakgrund har försetts av Nepa. Evaluering av prestationsmåtten visar att den naiva bayesianska klassificeraren ger de mest precisa prediktionerna av konsumenternas driksvanor medan random forest, fastän den mest tidskrävande, är föredragen vid klassifiering av Nöjd Kund Index (NKI). Regression av NKI resulterade i likartad prestations nivå för samtliga modeller. Ensemble metoderna gav en lätt ökning av predicerings precision samt en ökad tidskonsumption.

Acknowledgements

I would like to thank Prof. Henrik Hult for supervising this project and for his valuable comments on the first draft of this paper. He contributed with instructive feedback on how to improve my research.

I would also like to acknowledge Markus Sällman Almén as the second supervisor of this project and thank him for his immense knowledge that made it possible for me to conduct this research and become more innovative. Furthermore, I would like to say a big thank you to the PEA team at Nepa for their support and for giving me this opportunity.

Finally, I would like to thank my parents for their unfailing support and continuous encouragement throughout my years of study.

Contents

Chapter 1

Introduction

The topic of data fusion has been discussed and investigated by data scientists for the past decades and is an exciting branch in applied statistics. From a business perspective, we want to see if the fusion of data enables us to make informative and complete deductions about consumers' behavior. It is desirable to utilize as much information about the consumers as possible to increase profit. Only collecting data from separate market surveys might yield data sets that are sparse and insufficient. There might be variables that are important to the company's business model but not necessarily considered by all surveys. Instead of repeating all surveys that do not include the missing questions, which is both time consuming and expensive, one can try to predict important consumer variables and then fuse the data using the *common features* of both data sets and obtain a more extensive data base. The idea is loosely depicted in Figure 1.1. Academically speaking, data fusion is about making insights from the joint distribution of multiple random variables where only knowledge of the marginal distributions is given. Papers covering this subject include efforts from McCulloch et al [8], Esteban et al [6] and Takama et al [17].

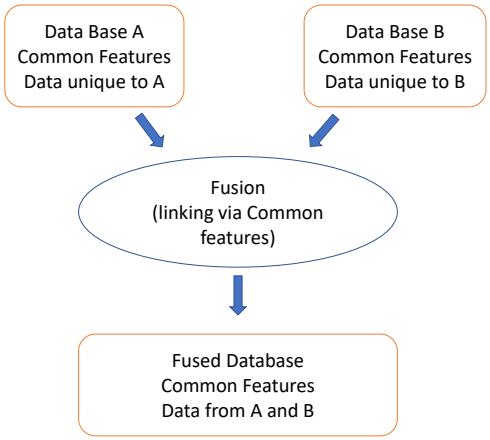

Figure 1.1: A sketch of data fusion.

The purpose of this thesis is to investigate different methods for data fusion to see if it is possible to find a preferable model to predict consumer attributes for empirical data provided by Nepa. The data contains valuable information about the consumer behavior, e.g. personal information such as salary, relationship, family and more business oriented variables such as how much they purchase and satisfaction with the products. The main focus is to predict specific variables that are important to the company's business model (called target or response variable) using sets of the other variables as predictors and hope that they mirror the values of the target variables accurately. This will be accomplished by applying a number of statistical models as well as various ensemble methods. The models considered are the following:

· Linear model

· K-nearest neighbour

· Decision Trees

· Random Forest

· Naive Bayesian classifier

· Neural Networks

The predictions of the response variables will be evaluated in terms of several statistical measures that indicate the precision of the predictions and the performance of the classifier. The goal here is not to find a model that performs well for arbitrary data since this is most likely not even plausible. However, it is in Nepa's interest to find models, in an automated way, that are suitable for the consumer data they have access to and this is the main objective of the thesis.

This thesis will, in addition to adding more detail to the process of data fusion, thoroughly elaborate on the theory behind the statistical models and performance measures used for imputation. There will be sections covering the feature selection and engineering done for all chosen data sets and the theory behind these concepts. The ensemble methods will be explained and evaluated. Finally, all results of the prediction performances and feature selections will be visualized in tables or plots.

Chapter 2

Methodology

This section will explain the selection of features, feature engineering and the imputation process, including the procedure of fitting statistical models and evaluating the results. First will a more detailed description of the data fusion process be presented.

2.1 The Process of Data Fusion

The data fusion consists of multiple steps that are illustrated in Figure 2.1, which represents a scheme of the fusion. The first part consists of a donor and a receiver set. The donor set is samples of data that contains observed values on a response variable that is particularly interesting to the company's business model. Examples of donors include surveys of consumer behavior or consumer transaction history etc. The receiver can also be data on consumer behavior but excluding the important business variables. The main principle here is to use the common features in both donor and receiver set to predict and impute values on the response variables that are missing in the receiver sets. The donor is naturally the key set of the fusion process.

Both donor and receiver go first through a pre-processing part where the data is cleaned and common features are identified. The cleaning of data includes removing missing or erroneous values (which are commonly occurring in real empirical data) as well as converting and renaming the variables to the right data types so they can be processed properly in R. Identifying common features include finding linking features in both sets, i.e. prediction variables that have been included in both sets. For example, the donor and receiver sets might both contain variables such as *age* and *gender* but variables such as *TV viewings* or *product purchases* are only considered in the donor set. The linking features in this case are *age* and *gender* whereas e.g. *TV viewings* is the response variable.

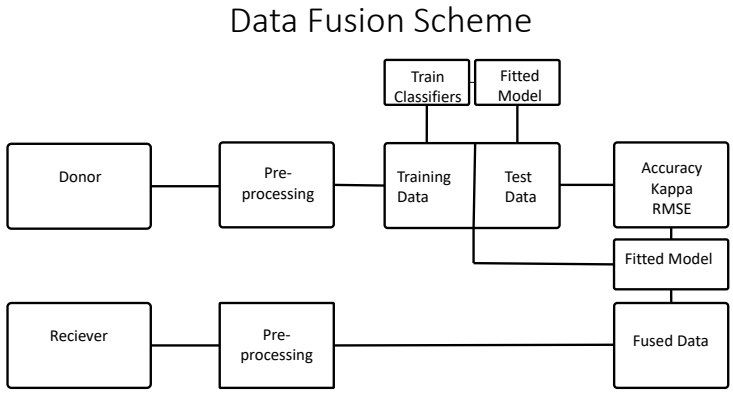

Figure 2.1: A sketch of data fusion.

The idea here is to use the linking features as predictors to the response variable so that we can predict the corresponding values in the receiver set. In Figure 2.1, we see that the donor set goes into a fitting part where the data is split into training and test data. This is explained more thoroughly in Section 2.3, where feature selection is also included. Fitting multiple statistical models allows us to choose the best performing one in terms of some performance measure. Once the type of the model has been chosen, it is fit to the entire sample size in the donor set. It is then possible to feed the model input from the linking features in the receiver set to obtain predicted values on the response variable. The aim is to obtain values that give a rough approximation of what the values should have been if the variable was considered in the receiver set. This completes the fusion which is the last part in Figure 2.1. The process described here have been implemented in R with the design to be as general as possible.

2.2 Feature Selection

The feature selection part of statistical modelling is quite important since we are dealing with large data sets. By selecting a specific subset of features that are relevant to our prediction models rather than using the complete feature space we can achieve a number of beneficial effects. In addition to making the model easier to interpret, training the model becomes less time consuming and it may also improve the performance of the model. Variables that are irrelevant and redundant might even decrease the accuracy of the model and therefore it

is better to discard them. For this project, a feature selection algorithm that works as a wrapper method around random forest has been utilized to select features in an automated way. The main principle here is to select features based on a feature importance measure that quantifies the significance of each feature.

2.2.1 MDA - Mean Decrease in Accuracy

The mean decrease accuracy (MDA) is computed when fitting data to a random forest model. When training classifiers that repeatedly fits decision trees to bootstrapped subsets of the training data, not all observation are considered by each tree. In fact, one third of the observations are not used for fitting and these are called out-of-bag samples. The out-of-bag error is the mean prediction error when using these samples as test data [12].
The random forest model, which is explained more thoroughly in Section 3.1.3, excludes features randomly in each bagged tree in order to decorrelate the trees and reduce variance. MDA is the contribution to decrease in prediction accuracy when excluding a specific feature. The higher MDA, the more significant was the feature to the response variable and is thus given a higher importance score.

2.2.2 Selecting the important features

The first step of the algorithm is to create permuted copies of the features and use them to extend the data set. These are called shadow features. Then a random forest model is fit to the data and the importance of each feature is measured in terms of the MDA. This process is repeated a chosen number of times and for each iteration it compares the score of the real feature to the score of its shadow features and continuously removes the feature with significantly lower importance than the shadow features. The algorithm is terminated when all features have been evaluated or when the number of iterations has been exceeded, i.e. if the algorithm has not converged by then. If this is the case, then a number of tentative features are returned by the algorithm. The tentative features are the features that have not successfully been determined to be included or removed. One reasonable solution to this problem is to assign the tentative feature as important or unimportant based on the median score of the feature compared to the median score of its best shadow features. Another approach is to simply increase the number of iterations until these features have either been confirmed as important or rejected. This will, however, make the algorithm more time consuming. The former approach will be used and implemented in R for this thesis.

The algorithm is convenient to use and more time efficient than other greedier methods such as recursive feature elimination. The results will be visualized by box plots of the scores of each feature, including the shadow features, which will

facilitate the interpretation of the results considerably.

Figure 2.2 displays an example of the results of this algorithm where it has been used to determine variable importance based on sales data for a company selling car seats. The target variable here is *Price*, i.e. the price of the car seats. The x and y axis shows the considered features and the importance score for each feature, respectively. The score of the shadow features are also present. The green boxes are the features that have been confirmed as important and the red boxes are the ones that have been rejected. Note that the median of all red boxes are below the median of the maximum score of the shadow features. Here we see that the prices of car seats charged by a competitor (*CompPrice*) and the amount of units sold in thousands (*Sales*) are highly significant for the price of the car seats which seems perfectly reasonable. The remaining variables were directly deemed as unimportant.

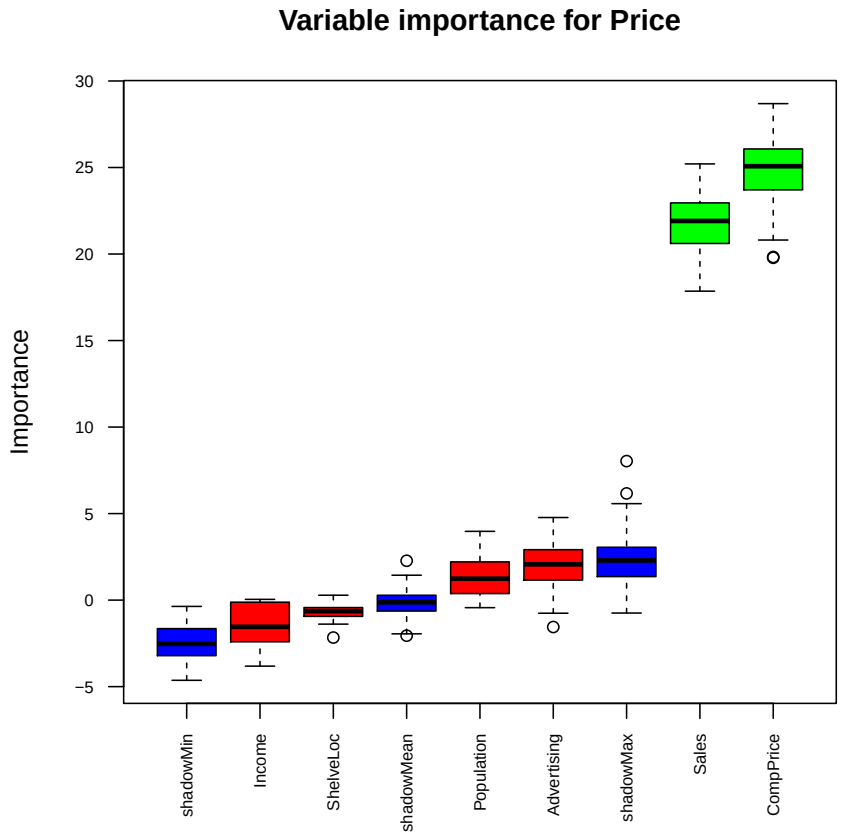

Figure 2.2: Variable importance for price on car seats using the decision tree based feature selection algorithm.

2.3 Fitting statistical models

When predicting values on the response variable, the process is similar for all statistical models. The donor data set is split into training and test data based on a rough 70/30 percent ratio. All the statistical methods and procedures are executed in R. Furthermore, to avoid getting biased results and increase performance, an implementation of 10-fold cross validation is done when training the classifiers. This means that the model is fit to ten separate sub sets of the training data and the model parameters are adjusted and estimated for all sub sets and then averaged for the final model. The parameters in the models, when such exists, are also optimized i.e. grid based optimization (unless otherwise stated e.g. neural networks) to produce the best possible fit in terms of the training

error. The quintessential R package used to achieve the grid parametrization and cross validation is the *caret* package by Kuhn [14].
Predictions are done for a variety of models which can either perform a classification or a regression. This relationship will ultimately be based on

$$y_i \sim f(\mathbf{x}_i) + \epsilon \tag{2.1}$$

where y_i is the observed response and $f(\mathbf{x})$ is the fitted model that takes input data $\mathbf{x}$ and produces predicted values on the response. The second term, ϵ, represents noise. The predicted values are evaluated using the statistical measures mentioned in this thesis. The performance of the classifiers in terms of these measures will be put in contrast to their respective time consumption.

2.3.1 Available Data

The data used for this thesis will consist of real consumer data based on consumer transaction history and social-demographic background provided by an international grocery chain and a well-established liquor company. There will also be data on consumer drinking habits, purchase power and earnings. More details about the structure of the data is given in Sections 4.1-4.3. The consumer data will in some cases in this report be encoded due to confidentiality. This means that details concerning variable names and interpretation will not be revealed, nor will any values or responses from the consumers. Only the results and conclusion of the performance measures will be presented as well as a description of how the data is structured but not the exact feature interpretation.

2.4 Computing Time Duration

When measuring the time consumption, a simple use of the *system.time* function in *R* will yield information about the CPU time needed to run the R sessions. The measurements include the time required to train the model with 10-fold cross validation, predict and impute the missing values in the data set as well as some additional negligible lesser operations. In order to obtain a more robust perception on the time consumption, the measurement is repeated four or five times for each model and then the mean is computed. Training the models is by far the most time consuming part of the code. Time is measured in seconds.

Chapter 3

Theory

This section will elaborate on the theoretical concepts behind the statistical models and give an instructive description of the models properties as well as the performance measures used to evaluate the models. The steps taken in Sections 3.1.2-3.1.4 are all outlined in Hastie et al [12].

3.1 Models and Classifiers

3.1.1 Multiple Linear Regression

Multiple linear regression is a common and useful way to model the relationship between a continuous response vector $\boldsymbol{Y}$ and one or more explanatory variables, denoted by the matrix $\boldsymbol{X} = (\mathbf{1}, \boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_p)$, where $\mathbf{1}$ and $\boldsymbol{X}_1, \dots \boldsymbol{X}_p$ are column vector of ones and column vectors of input data from p features respectively. The main assumption for linear models is that the dependence of $\boldsymbol{Y}$ on $\boldsymbol{X}$ is linear. We write this as $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ or component wise as

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip} + \epsilon_i, \tag{3.1}$$

where $X_{i1}, X_{i2}, \ldots, X_{ip}$ are also called covariates, features or predictors. Here, $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \ldots, \beta_p)$ are unknown coefficients that are the model parameters. The $\boldsymbol{\beta}$ coefficients are interpreted as the average effect on the response when increasing $\boldsymbol{X}_j, j = 1, ..., p$, with one unit if all other features are fixed. The ϵ_i in eq (3.1) is an error term which accounts for measurement error and potential missing variables and is assumed to be Gaussian distributed, i.e. $\boldsymbol{\epsilon} \in N_n(\mathbf{0}, \sigma^2\boldsymbol{I})$. We also assume that the observations Y_i are uncorrelated and that the sample data $\boldsymbol{x}_i$ is non-random. Additionally, the variance, σ^2 is assumed to be constant i.e. we say that the model is *homoscedastic*. The response vector is then distributed according to $\boldsymbol{Y} \in N_n\left(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}\right)$. Linear models are easy to interpret and work well when the data sample size is close to the number of

features, i.e. when p is approaching but not exceeding the data sample size n [5].

We estimate $\boldsymbol{\beta}$ using the training data and the strategy is to find a $\boldsymbol{\beta}$ that minimizes the residual sum of square, abbreviated RSS. One such method is ordinary least squares (OLS) where we let $\boldsymbol{X}$ have full rank. The idea behind OLS is to draw the regression line such that the distances from the data points to the line is as small as possible. If the sample size is n and the i:th observation of the j:th feature is x_{ij}, we want to minimize

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \ldots - \beta_p x_{ip}\right)^2 = \\ &= \left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right)^T \left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right). \end{aligned} \tag{3.2}$$

For equation (3.2) the optimal solution is given by

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}. \tag{3.3}$$

This approach is utilized when using linear models.

3.1.2 Classification and Regression Trees

Here we discuss tree-based methods for regression and classification. The most instructive route to build a decision tree is by stratifying the prediction space into a number of sub-spaces or regions by following certain splitting rules which results in tree nodes. This can be visualized in terms of a tree structure, hence the name decision tree methods.

We will first consider decision trees for regression problems. A decision tree is composed by typical aspects characterizing a tree, namely leafs and branches. The branches represents the connection between the tree nodes. At the bottom of the tree, we have the leafs which are node that do not split prediction regions any further. This is visualized in Figure 3.1.

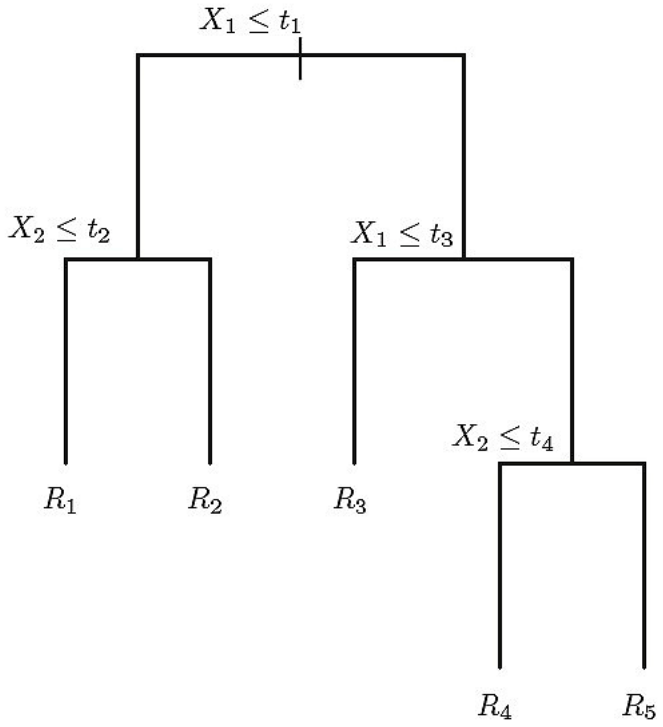

Figure 3.1: Decision tree that partitions the prediction space. This figure have been adopted from [12.]

We see in Figure 3.1 that leafs are denoted as $R_1, .., R_5$ and the internal nodes are given by the the predictor outcomes (also known as cut points) $t_1, ..., t_4$ which are determined by splits that yields the lowest RSS.

The main objective here is to obtain boxes $R_1, .., R_J$ that minimizes the RSS given by

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{3.4}$$

where $\hat{y}_{R_j}$ denotes mean response for the training data points that falls into the j:th leaf. It is impossible to evaluate every possible regional partition of the prediction space (computationally unfeasible) so therefore a regulated approach known as recursive binary splitting is taken. Among all predictors and all possible cut points t_i we choose the predictor and cut point that yields the tree with lowest RSS. This process is then repeated for all predictors until we only have terminal nodes left. To determine the response of a given test data point, the mean of the training data within the terminal node where the test

data belongs, is computed. The response is then simply the value of the mean.

Solely building a tree in this fashion might result in too complex trees which leads to over-fitting and poor test error rate. Therefore, so called tree pruning is utilized in order to reduce the tree into subtrees which might give more reasonable results. Subtrees means smaller trees with fewer terminal nodes, i.e. fewer splits in the prediction space. Reducing the trees this way can possibly result in a decrease in variance and a clearer interpretation of the tree. The question now is how do we know the optimal way to prune a tree? We do not check every possible subtree (there might be a lot of them) but instead resort to cost complexity pruning where a parameter α is regulated such that

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m}) + \alpha |T| \tag{3.5}$$

is minimized, where T is the total number of terminal nodes, y_i is the observed response and $\hat{y}_{R_m}$ is the prediction output associated with the m:th terminal node. The purpose of α is to tune the trade off between the complexity of the tree and the precision of the predicted responses. To avoid biased value on α, one can use k-fold cross validation.

A classification tree is based on the same structural principles as the regression tree. However, instead of predicting a quantitative data point, we try to predict qualitative observations. To predict the response of a classification tree, a majority vote is conducted within the terminal node where the test observation belongs. When we grow our tree, we do not consider RSS as a criterion for making binary splits. Instead, we look at the classification error rate which is the fraction of the training observation that do not belong to the most common class. In some cases alternative measures are preferred since classification is too insensitive for tree-growing. These are the Gini index and cross-entropy where the former is defined as

$$G = \sum_{k}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right), \tag{3.6}$$

where $\hat{p}_{mk}$ is the proportion of training observations that are from the k:th class. This measure quantifies the spread i.e. the variance across the K classes. The Gini index can be interpreted as the level of node purity where smaller values mean that nodes contains mostly observations from the same class. Cross entropy (more thoroughly investigated in section 3.3.2) is very similar to the Gini index and also measures node purity but is instead defined as

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{3.7}$$

3.1.3 Random Forest

The random forest is a robust algorithm that is based on bagged decision trees. The main idea behind random forest is to fit uncorrelated trees to bootstrapped data samples. If we split the training data at random and fit decision trees to each sample, the outcomes might become very different. There is a high variance in growing decision trees. In order to reduce the variance, one can utilize the idea of bootstrapping the data. Suppose that we have collected n observations, $Z = (z_1, ..., z_n)$, where $z_i = (\mathbf{x}, y_i)$ and we use this as training data. Suppose now that we randomly draw, with replacement, samples of data from Z and of equal size as Z. This is commonly known as bootstrapping. The bootstrapped data are treated as I.I.D samples of their empirical distribution. By repeating this procedure, say B amount of times, we can generate B sets of training data. This allows us to refit our model B times, i.e grow B decision trees and obtain B sets of predicted responses. These are then averaged to produce the final predicted value. More precisely, we have that

$$\hat{y}_{bag} = \frac{1}{B} \sum_{b=1}^{B} y^{*b}, \tag{3.8}$$

where y^{*b} is predicted value of the response variable for the b:th decision tree. This is called bagging and will thoroughly reduce the variance of the decision trees.
Random forest is built on this idea but when growing trees on the bootstrapped data samples, a majority of the predictors are not even considered. Why is this a good idea? Let us imagine we have case where a very dominating predictor is in the predictor space, i.e. it is very significant in predicting a certain response variable. Then each tree that is built for each bootstrapped data set will use this predictor as its top split (first predictor considered when partition the prediction space), making the trees rather indistinguishable. The predictions stemming from these trees will be highly correlated. This does not reduce the variance in the model as efficiently as if the predictions are uncorrelated. Random forest is a conceivable solution to this problem since the strongest predictor will not be considered in all trees which allows the other predictors to have a bigger impact on the response variable. We say that the random forest uncorrelates the trees.

3.1.4 *K*-nearest neighbour

The main principles outlining the K-nearest neighbour (KNN) algorithm is that one can estimate the probability that an observation belongs to a specific class by comparing it to its neighbouring data points and see what classes they belong to. In order to decide which points counts as the closest neighbours, similarity between the data points must be defined. It can be determined in various ways and differs depending on if the data is qualitative or quantitative. A plausible way to determine similarity when the observation features are quantitative is to

use the Euclidean distance between data points. A farther Euclidean distance indicate more discrepancies in the data. The KNN algorithm operates with training sets, denoted $\{(\mathbf{x}_i, \mathbf{y}_i)\}$. Let us now assume that we have observed a new observation, say $X = x_0$. First we want to form a set of the of the k nearest points, i.e lowest values on the distances $||\boldsymbol{x}_j - X||$ where $j = 1, \ldots, k$ and denote it $\mathcal{N}_0$. Then we follow this scheme:

1. Use $Pr(Y = c|X = x_0) = \frac{1}{k} \sum_{j \in \mathcal{N}_0} I(y_j = c)$ to determine the probability that observation X belongs to class c. If there are more occurrences of class c in $\mathcal{N}_0$, the estimated probability that response Y takes class c will become higher.

2. Finally, Y will be assigned to the class which yielded the highest estimated probability.

The last part is a majority vote of the different classes that are in the set $\mathcal{N}_0$. Regulating k severely impacts the outcome of the classifications. A consequence of a smaller value on k is that the KNN algorithm becomes increasingly sensitive. Then this classification procedure tends to yield more complicated decision boundaries. Most training points will be classified correctly since there are fewer neighbors that affect their classification. This comes at the cost of increased sensitivity to outliers, meaning that the accuracy for the test set can be low if there are many differences between the test- and training sets. For higher values on k, the method is more robust since more observations are taken into account which decreases the influence of outliers and the impact they have on the classification. On the other hand, it also decreases the impact the training points have on their own classification which might lead to training points being classified incorrectly. Tuning the value of k is a balance between variance and bias where high k yields a simple decision boundary that is too biased and low values mean complex decision boundaries with high variance. However, a condition that needs to be satisfied is that k is less than the sample size n, otherwise all points will have different classification.

3.1.5 Naive Bayesian Classifier

The naive Bayesian classifier is a simple yet surprisingly powerful classification technique well suited for large data sets. Like all Bayesian methods, it utilizes Bayes' rule but with a strong (and naive) assumption of independence among the predictors. Let us denote p features as $\mathbf{X} = (X_1, \ldots, X_p)$. Each feature takes a value from its domain Ω_j, $j = 1, \ldots, p$. The entire feature space is then given by $\Omega = \Omega_1 \times \ldots \times \Omega_p$. We can propose a classifier which assigns a class to any set of features based on the class discriminant function $f_c(\mathbf{x})$. The main purpose of the classifier is to, for a given set of features, maximize the discriminant function, i.e.

$$h^*(\mathbf{x}) = argmax \{f_c(x)\}. \tag{3.9}$$

A Bayesian classifier is based on these principles where the discriminant function is represented by the posterior probabilities. These probabilities are obtained by utilizing Bayes' theorem. In other words, the discriminant function is given by $f^*(\mathbf{x}) = P(C = c|X = x)$ and employing Bayes' theorem yields

$$P(C = c|\mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x}|C = c)\, P(C = c)}{P(\mathbf{X} = \mathbf{x})}, \tag{3.10}$$

where the denominator is the same for all classes and is thus ignored [13]. The Bayesian classifier is, for an observed data x, given by

$$h^*(\mathbf{x}) = argmax_c P(\mathbf{X} = \mathbf{x}|C = c)\, P(C = c) \tag{3.11}$$

In order to obtain the naïve Bayesian classifier, we must make the assumption that all features are independent in which case eq (3.11) becomes a product

$$f_c^{NB}(\mathbf{X}) = \prod_{i=1}^{p} P(X_j = x_j|C = c)\, P(C = c) \tag{3.12}$$

The outcome of the classification is given by the class with the highest posterior probability. The naive Bayes' classifier is easy to understand and is able to make fast predictions of classes in test data sets. However, the assumption of independence is incorrect for almost all real life empirical data and it might impair the precision of the predictions [16].

3.1.6 Neural networks

The idea of neural networks is originally inspired from how the neurons in the human brain communicate and interact with each other. This is a complex biological process, involving billions of neurons that continuously transmit electrochemical-signals which are received by other neurons through the synapse and dendrites of the nerve cell and eventually integrated in the cell body. Thousands of signals are processed in a single instant and if the aggregated effect of all signals exceeds some specific threshold, an impulse is created in the neuron and transmitted via the axon which is a long slender fiber that conducts impulses away from the cell body. Not all signals promote the generation of an impulse (so called excitatory signals) but some can result in an inhibitory reaction which suppresses the neuron from firing an impulse. The effects of the signals are changed as the human brain learns things, recognizes patterns, makes decisions etc [9].

It is impossible to construct a computational machine or software simulation that fully replicates the neurons in the brain. However, neural network models are loosely based on the structure and the style of processing in the brain. The neurons in this case are replaced by artificial neurons, more properly known

as units, that are arranged in three different types of layers. The first type is called the input layer which is composed by input units. It receives information from the outside world through observational data that the networks want to learn about and recognize its patterns. The third type is known as the output layer where the units are conveniently called output units which signal how the network responded to the data. Practically speaking the output layer will signal how the target variables were predicted. In between the first and third layers lies the hidden layer where the hidden units constitutes most of the artificial neural network. The connections between the units are called weights which is a number that determines if the "signals" are either excitatory or inhibitory. The weight number can be either positive or negative and the higher the weights the stronger the connection is between the units. A neural network can have multiple hidden layers in which case it is called multilayered neural network. Having more than one hidden layer can improve the performance of the network but the amount of parameters to determine increases significantly. Unless the problem is unusually complex, having only one hidden layer will suffice. This thesis will only focus on single-layered neural networks [15].

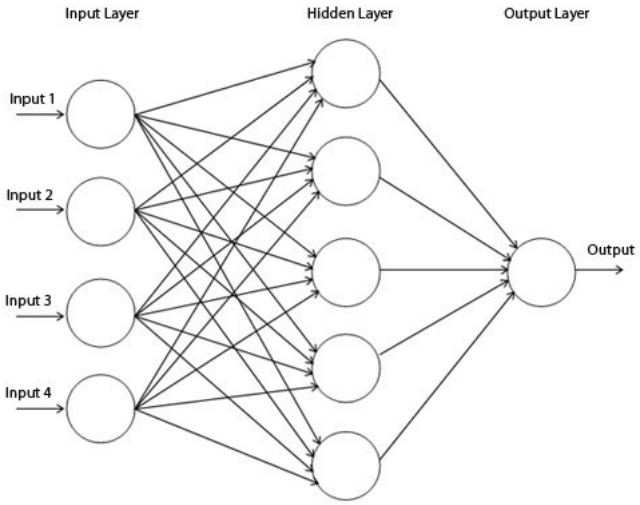

Figure 3.2: A sketch of the different layers in a neural network.

A visual presentation of the artificial neural network is given in figure 3.2.
All classifiers need to be trained before they can be used for prediction and the artificial neural network is no exception. Each artificial neuron in the layers use an activation function, denoted $\sigma(x)$, to convert the input data to an output which is then transmitted to the next layer. These activation functions take different forms depending on if it is a regression or classification and depending on which layer it is. For instance, multinomial classification is mostly done by

using the *softmax* function as $\sigma(x)$ whereas probabilistic regression uses the *sigmoid* function. To put this in a more formal context, we express a_j^i as the activation or output of $\sigma(x)$ of the j:th perceptron (artificial neuron) in the i:th layer. Each layer input can then be expressed by its preceding layers output in the following way

$$a_j^i = \sigma\left(\sum_k \left(w_{jk}^i \cdot a_k^{i-1}\right) + b_j^i\right) \tag{3.13}$$

In (3.13), w_{jk}^i is the weight of the k:th neuron in the $(i-1)$:th layer to the j:th neuron in the i:th layer. Furthermore, b_j^i is the bias of the j:th neuron in the i:th layer. During training phase, we want to optimize the weights such that the *cost* function, generally expressed as

$$C(W, B, I, O), \tag{3.14}$$

where W is the weights, B are the biases, I is the input from training data and O is the corresponding output, is minimized. The cost function can take different forms depending if it is a classification problem or a regression but the more commonly used costs for neural networks are mean squared error (regression) or cross-entropy (classification). Initially, the weights are randomly chosen [15]. Optimization of the weights is most frequently done by an algorithm known as gradient descent which is an iterative first-order optimization algortihm which takes steps that are porportional to the negative gradient. Another method used to find optimal points is the BFGS algorithm which is mentioned more in section 3.3.1. For this thesis, the BFGS algorithm is utilized for weight optimization.

3.2 Performance Measures

3.2.1 Accuracy

Accuracy is a simple intuitive measure that will be used to evaluate classifications. It is a measure of how close the predictions are to the true value of the data. In classification, this becomes the proportion of correctly predicted classes, i.e. the number of correct predictions, divided by the total number of predictions (sample size of test data). More precisley,

$$\text{Accuracy} = \frac{\#\text{correct predictions}}{\#\text{predictions}}. \tag{3.15}$$

This will yield a percentage number (between 0 and 1) where a higher value means more accurate prediction. Investigation of the confusion matrix in table 3.1 shows that accuracy also can be expressed as $\frac{\#TruePositive+\#TrueNegative}{P+N}$. Solely choosing a model on classification accuracy can be misleading and should not be done in all cases. When there is a high imbalance among the classes in the prediction domain, models only choosing the dominating class will yield a

very high accuracy and appear to be precise but they are in fact useless when it comes to modelling the minority classes. For instance, if we collect data on 200 adult females and try to predict how many will be diagnosed with breast cancer within five years, simply choosing a model which predicts everyone to not be diagnosed with breast cancer might give an accuracy of 95 % which seems to be very precise but this model says nothing about the females that actually get breast cancer which is what we want to model. When these kinds of imbalances are present in the data, many predictions might be correct by mere chance [3]. A conceivable alternative is to evaluate the classification in combination with the Kappa coefficient which is investigated in the next section.

3.2.2 Kappa

The Kappa statistics is a coefficient measuring the rate agreement for qualitative data. It is considered to be a more valid and robust measure compared to other statistics evaluating nominal data since it takes into account the possibility that agreements occur by chance. The Kappa value is based on the confusion matrix which is a table that visualizes performances of classifiers. A simple example of such a matrix is

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

Table 3.1: The confusion matrix.

The table above displays the result of a binary classification. Let us introduce the quantity *chance-agreement probability* denoted p_e. This is the hypothetical probability of chance agreement. It is computed by determining the *marginal frequencies* and then take the sum. Thus we obtain (looking at table 3.1)

$$p_e = \frac{\#\text{True Positive}}{P+N} * \frac{\#\text{False Positive}}{P+N} + \frac{\#\text{False negative}}{P+N} * \frac{\#\text{True negative}}{P+N} \tag{3.16}$$

Furthermore, let us denote the aforementioned accuracy as p_a. Then Kappa is given by

$$\kappa = \frac{p_a - p_e}{1 - p_e}. \tag{3.17}$$

From this expression we can obtain an estimated value on Kappa based on sample data. The denominator represents the percentage of data for one would not expect random agreement. The numerator represents the percentage for which actual agreements has occurred, i.e. not agreements that have occurred by chance. In order to evaluate the value on Kappa, one may refer to various benchmark scales presented in literature. One such scale, which is widely-used by statisticians, is the Landis and Koch Kappa's Benchmark Scale which is summarized in Table 3.2. Although its validity is sometimes questioned, it still provides a decent perception of how robust the prediction is. Regardless of benchmark scale however, we ultimately want Kappa to be as close to one as possible. Kappa ranges from -1 to 1 like most correlation statistics and negative values indicate poor classifiers and the number of agreements is what can be expected by chance [10].

Kappa Statistics	Strength of Agreement
<0	Poor
0.0 to 0.2	Slight
0.21 to 0.40	Fair
0.41 to 0.60	Moderate
0.61 to 0.8	Substantial
0.81 to 1.00	Almost Perfect

Table 3.2: Landis and Kochs' benchmark scale for Kappa.

3.2.3 RMSE and MAD

When doing regressions on data, the residuals are given by how much the regression line deviates from the measured data points. The further away from the points, the higher the values on the residuals become. The residuals can therefore be seen as prediction errors. This introduces the concept root mean square error (RMSE) which is a very common and standardized measure for regression analysis. It is the standard deviation of the prediction errors, i.e. the residuals. Intuitively, it shows how concentrated the data is around the line of fit. A smaller value on RMSE means that the residuals are less spread out and the regression line is more precise to the data. The mathematical formula is given by

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i}^{n}(\hat{y}_i - y_i)^2}, \tag{3.18}$$

where $\hat{y}_i$ and y_i are the predicted and observed values respectively and n is the sample size of test data [2].
At times, the mean absolute deviation (abbreviated MAD) can be used to evaluate regressions. It is simply the mean deviating absolute value of the predictions from the observed data. The lower value on MAD, the closer on average are the predicted values to the observed values. In terms of a formula, it is written (as outlined in [1])

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|. \tag{3.19}$$

3.2.4 Hellinger's Distance

If we assume P and Q denote two probability distributions that are absolutely continuous with respect to a third Lebesque measure denoted as λ, a measure of the distance between these is the metric Hellinger's distance. The derivatives of P and Q with respect to λ are probability density functions which allows us to express the squared Hellinger's distance (in L^2 sense) in terms of a regular calculus integral. More precisely, we obtain

$$H^2(P,Q) = \frac{1}{2}\int\left(\sqrt{\frac{dP}{d\lambda}} - \sqrt{\frac{dQ}{d\lambda}}\right)^2 d\lambda, \tag{3.20}$$

where we square for convenience. Hellinger's distance is a quantification of the similarity between probability distributions. For the definition above, the Hellinger's distance will take a value between 0 and 1, where 0 means that the two distributions are equal almost everywhere and a higher value indicates that the distributions are more separated. Thus will Hellinger's distance tell us how much the distributions are overlapping [4].
This is illustrated in Figure 3.3 where two one-dimensional distributions are displayed in different cases.

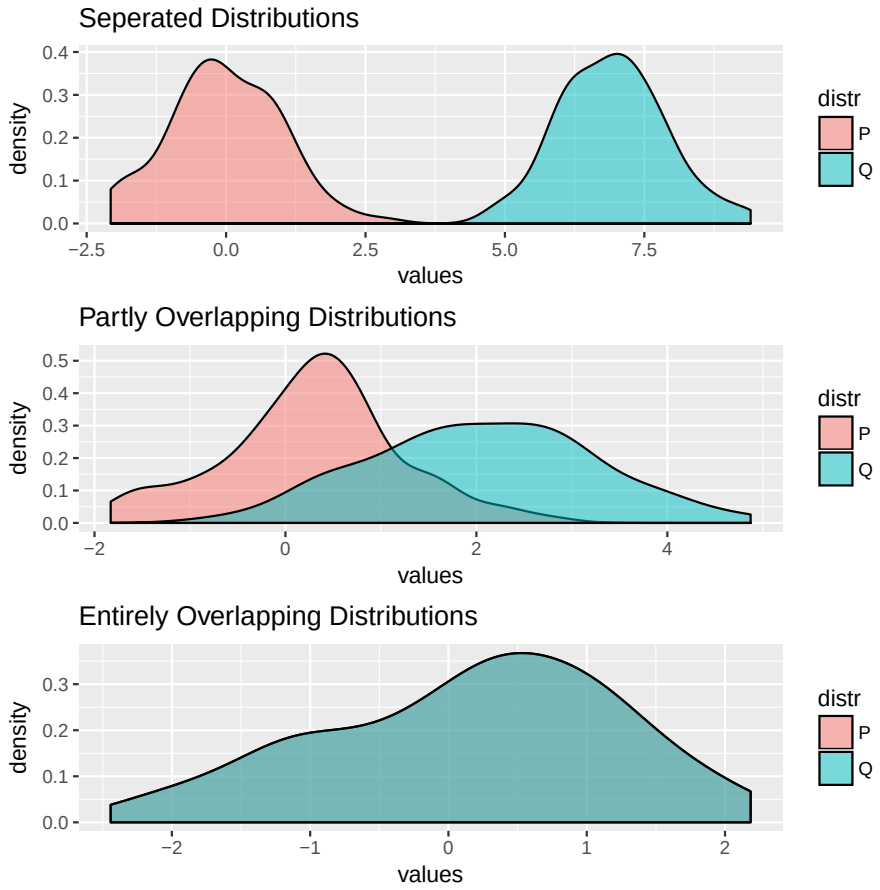

Figure 3.3: Three different cases of overlapping for two one-dimensional distributions:
upper plot: P and Q are completely separated: $H(P,Q) = 1$.
middle plot: P and Q are partly overlapping: $0 < H(P,Q) < 1$
lower plot: P and Q are completely overlapping: $H(P,Q) = 0$

For two discrete probability distributions, equation (3.20) become

$$H(P,Q) = \frac{1}{\sqrt{2}} \cdot ||\sqrt{P} - \sqrt{Q}||_2 = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{3.21}$$

3.2.5 Hellinger's Distance for Empirical Data

In reality, we can only work with discrete probability distributions since the number of observations is finite. We want to determine the probability of

each observation in our data set. Each observation can be viewed as an empirical outcome of some multidimensional probability distribution, i.e. if $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ is an outcome of p features or variables then the probability of observing $\mathbf{x}$ is $P(\mathbf{x})$ where P is some p-dimensional probability distribution representing the data. To determine $P(\mathbf{x})$ we can use the empirical probability distribution. For continuous data, the probability of each observation is infinitely small which in practice mean that the continuous features must be discretized into levels on an interval spanning the range of values of the continuous features. This will yield each continuous valued observation a discrete probability. In the multidimensional case, we can then group both continuous and nominal features to obtain a count for each type of observation, i.e. the number of occurrences in the data set for each type of observation. If we divide each count by the total sample size, we obtain the discrete probabilities for each observation. Hellinger's distance can then be computed using equation (3.21) where P will be the probability distribution of the set of observation when we include observed values on the response variable and Q is the probability distribution of the set of observation with predicted values on the response variable. When discretizing the continuous data, the grid will consist of homogeneous step sizes where each step corresponds to a discrete level in the continuous interval. The number of steps reflects the fineness of the grid and finer grid means a more extensive observation region. Hellinger's distance will be plotted against the total number of steps which varies from 2 to 100 where the step sizes are scaled towards the empirical quantiles of the continuous data meaning the entire continous interval is taken into account. These plots are presented in the result section of this paper.

3.3 Ensemble Methods

This section will outline the ideas behind the ensemble methods and explain the advantages and disadvantages of proposing these kind of classifiers as well as the differences between them.

3.3.1 RMSE based Ensemble Method

One way to optimize the use of multiple regression models is to implement an algorithm that tries to find the best weighted combination of these. Let us assume we want to find the best prediction model with respect to the RMSE value. By intuition, the simplest way would be to calculate the RMSE using training and test data for the different models and choose the model that yields the smallest value on the RMSE. It might be more accurate, however, to calculate the RMSE when considering a linear combination of the predicted values from each model, where the coefficient are weight parameters. More precisely,

we investigate the following problem,

$$\begin{aligned} \underset{RMSE}{\text{minimize}} \quad & \hat{y}_i^{(tot)} = \sum_{k=1}^{p} w_k \hat{y}_k^{(i)} \; i = 1, \ldots, n \\ \text{subject to} \quad & \sum_{k=1}^{p} w_k = 1, w_k \geq 0, \end{aligned} \tag{3.22}$$

where p and n are the number of classifiers and the sample size of test data respectivley. We want to find values on the parameters w_1,w_2, ..., w_p such that the RMSE for prediction y_i^{tot} is minimized. Incorporating the linear combination of classifiers into the formula of RMSE, here denoted $\widehat{\text{RMSE}}$, we obtain

$$\widehat{\text{RMSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i^{tot} - y_i)^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{p} w_k \hat{y}_k^{(i)} - y_k\right)^2} \tag{3.23}$$

Equations (3.22) and (3.23) compose an optimization problem that can be solved numerically using e.g. a multidimensional grid on the possible parameter values and then looping through each point in the grid until a minimum on the RMSE has been found. Then we choose that point as the optimal one. In this thesis however, a more sophisticated method known as the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) algorithm will be used for this purpose. It is a part of the quasi- Newton family of algorithms and is an iterative method mainly used to solve nonlinear optimization problems. Like many Newton-like algorithms, it seeks to find stationary points where the gradient of the of objective function has to be zero in order to satisfy the optimality condition. For a more mathematically rigorous motivation of the BFGS algorithm, see papers by Hongzhou Lin et al [11].

The advantages of using this method is that it will always yield an equivalent or smaller RMSE value than using only one prediction model. Suppose that the best prediction is given by only using a linear regression model. Then we will not lose any performance when using this approach since the weight parameters will adjust towards the linear model and yield a classifier that is essentially the same as a linear model. The lesser methods will be neglected or have very little impact. The disadvantages of this is the risk of getting biased results since the parameter values obtained for a specific set of training data might not be optimal in general. This risk is thoroughly reduced by implementing k-fold cross validation i.e. optimizing the weights over k-folds of the data and then averaging the results.

3.3.2 Cross-entropy based Ensemble Method

It is also desirable to find a corresponding method for multinomial dependent variables. As opposed to regression models, the prediction values from a clas-

sification cannot be weighted and summed. It is therefore not plausible to use mathematically convenient measures such as RMSE or mean absolute deviation. Instead we resort to cross-entropy which is somewhat more refined than the aforementioned Accuracy measure since the *log* part in equation (3.24) takes into account the closeness of a prediction which makes evaluation more consistent [12].
Suppose we have predicted the outcomes of a categorical target variable with j class labels using p different classification models. From each model, we can extract the class probabilities $p_{jp}^{(n)}$, which is the probability that model k has predicted observation i to be in class j. The class that is ultimately chosen as an outcome in each test data is the class with highest probability. Moreover, the cross-entropy for a predictor $p_{k*}^{(i)}$ is given by:

$$H\left(y^{(i)}|p_{k*}^{(i)}\right) = -\sum_{j=1}^{J} y_j^{(i)} \log\, p_{kj}^{(i)}, \text{ for } i = 1, \ldots, n. \tag{3.24}$$

If we consider the whole test data sample, we take the mean of the of all test observation, i.e.

$$H\left(\{y^{(i)}\}|\{p_{k*}^{(i)}\}\right) = \frac{1}{n}\sum_{i=1}^{n} H\left(y^{(i)}|p_{k*}^{(i)}\right). \tag{3.25}$$

Cross-entropy can intuitively be said to measure the randomness or the disorder of the prediction. A classification model is said to perform well if its entropy is small. We can propose a new classifier based on $\hat{p}_j^{(i)} = \sum_{k=1}^{p} w_k p_{kj}^{(i)}$ which is the weighted sum of the class probabilities for each method. More precisely

$$H\left(\{y^{(i)}\}|\{\hat{p}^{(i)}\}\right) = \frac{1}{n}\sum_{i=1}^{n} H\left(y^{(i)}|p_{k*}^{(i)}\right) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{J} y_j^{(i)} \log\, \left(\sum_{k=1}^{p} w_k p_{kj}^{(n)}\right). \tag{3.26}$$

In other words, for each predicted class probability in each test observation we seek to find the optimal value on the weights w_k so that the total sum of probabilities is weighted to the class probability yielding the smallest cross-entropy. The BFGS algorithm has been used for finding the optimal point. Similar to the weighted continuous case, including classifiers that perform poorly should not be a problem since the weights will lean towards superior classifiers. In order to reduce the risk of biased optimization, one can implement k-fold cross validation and average the result.

Chapter 4

Results

4.1 Classification of Consumer Drinking Habits

The first fused data set used to evaluate prediction and imputation of missing data points is a survey regarding the drinking habits of the consumers. It contains a total of 6718 observations from 14 different features where both social-demographic variables and alcohol consumption variables have been combined and matched to respondents. These are summarized in Table 4.1 where some details have been omitted due to confidentiality.

Features	Description
Age	The age of the respondent.
Gender	The gender of the respondent, 1 is male and 2 is female
Working Status	The current working status of respondent. The available classes here are employed full time, employed part time, retired, Housewife / househusband, Self-employed, unemployed (looking for work), Student /at school full time and other.
Level of Education	Respondent education and training. Levels here include e.g. high school graduate, college graduate etc.
Ethnicity	The race or ethnicity of the respondent. Levels are e.g. Asian alone, Black or African American alone etc.
Place of Residence	Where the respondent resides. Levels are Large city, Medium-sized city, Town or Village.
Life Stage	Current relationship status for respondent. Levels are Married / co-habiting children at home, married / co-habiting no children, Single, living alone with children (single parent), Single living alone, no children.
Household Income	House hold income of respondent. Levels are e.g. "less than 50000", "50000-74999", etc.
Segment	Social behavior of respondent. Confidential.
Vodka Consumption	The consumption of vodka for respondent. Levels are Less often, Never, Once per month and once per quarter.
Rum Consumption	The consumption of rum for respondent. Levels are Less often, Never, Once per month and once per quarter.
Cordials Liqueurs Consumption	The consumption of Cordials liqueurs for respondent. Levels are Less often, Never, Once per month and once per quarter.
Cordials Liqueurs Shots Consumption	The consumption of Cordials liqueurs shots for respondent. Levels are Less often, Never, Once per month and once per quarter.
Occasion	Last occasion of drinking for the respondent. Confidential.

Table 4.1: Explanation of features used in the first data set from the grocery store company.

The imputation variable here is the last feature in Table 4.1, labeled *Occasion*. The first step is to choose the most informative features according to the process described in Section 2.2. From the decision tree based algorithm, that took roughly 10 minutes to run, the following feature were deemed as significant: *Age, Gender, WorkingStatus, LevelOfEducation, Ethnicity, PlaceOfResidence, LifeStage, HouseholdIncome, Segment, VodkaConsumption and CordialsLiqueursShotsConsumption*. The difference in importance for the chosen set of features is most easily presented graphically as in Figure 4.1.

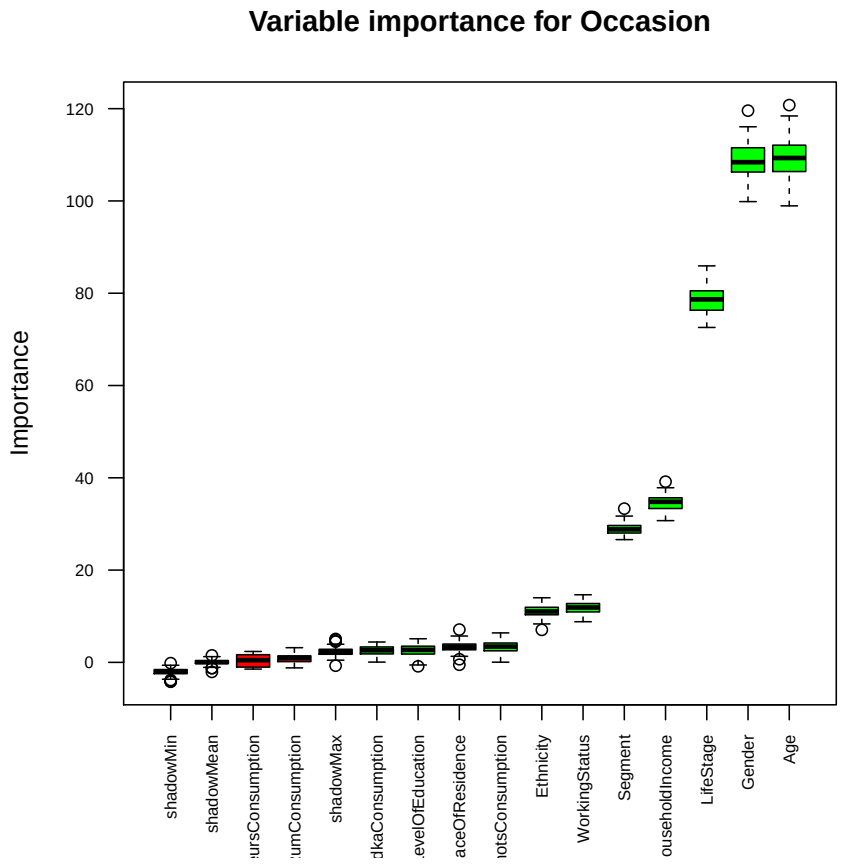

Figure 4.1: Importance of explanatory variables using decision tree based algorithm.

The imputation is then done according to Section 2.3 and the results are presented in Table 4.2.

Method	Accuracy	Kappa	Time [s]	Weight
KNN	0.33	0.27	6.16	$1.68 * 10^{-6}$
CART	0.30	0.21	2.11	$2.13 * 10^{-1}$
Naive Bayes	0.38	0.32	0.04	$5.89 * 10^{-1}$
Neural Networks	0.34	0.27	213	$1.75 * 10^{-1}$
Random Forest	0.37	0.30	994	$2.29 * 10^{-2}$

Table 4.2: Accuracy and Kappa values for each set of predictions as well as time consumption and the weight values from the ensemble optimization.

Furthermore, let us examine what happens when we incorporate the cross-entropy based ensemble classifier presented in Section 3.3.2. Here all models are considered in the optimization. This is a classification and thus we want to minimize the cross-entropy. The results are presented in Table 4.3.

Added Time [s]	Cross-entropy	Accuracy	Kappa
102	1.83	0.39	0.33

Table 4.3: Values on Accuracy, Kappa, Cross-entropy and added time consumption when using ensemble method.

It is also desirable to investigate Hellinger's distance as explained in Section 3.2.5. The results are presented in Figure 4.2 where it is also included a scatter plot visualization of the Accuracy and Kappa measures.

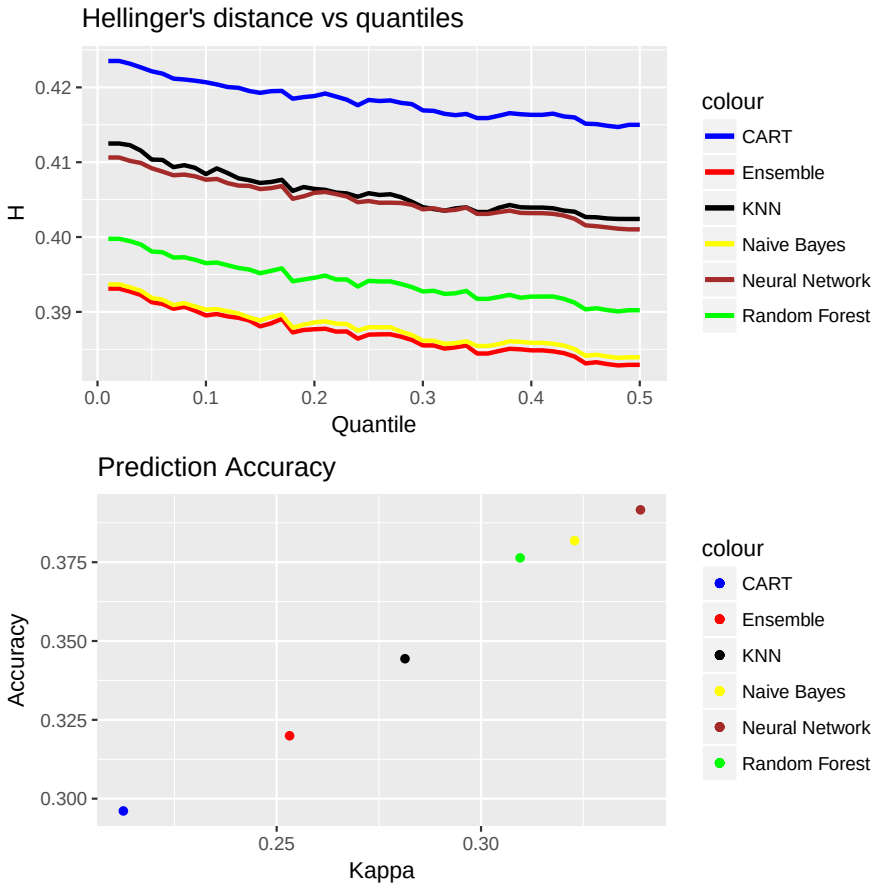

Figure 4.2: *Upper plot*: Hellinger's distance for some models
Lower plot: Graph of Kappa and Accuracy values.

4.2 Classification of Consumer Satisfaction Index

In this section, we will investigate the possibility to make precise predictions on consumers satisfaction with specific stores using data that is based on consumer attitude and purchase history. The goal here is to obtain a perception on how consumers generally rate their stores from a grocery chain in the cases where they have not responded to the question in the survey. Instead of repeating the survey multiple times, we try to fit a statistical model to the data and investigate how well we can predict respondents opinions. The first data set contains 12000 observations from 24 different features. These are described in tables 4.4 and 4.5.

Features	Description
Gender	The gender of the respondent, 1 is male and 2 is female
Purchase Power	A grouping of consumers purchasing power which is categorized from 1-8. The higher number a respondent has on this scale, the higher the income.
Mosaic Grouping	A categorization of the Social-demographic background of the respondent.
Total Price	The total amount of money spent on products half a year prior to march 2016.
Store Format	This is the most frequently visited store type in the grocery chain for the respondent.
Total Quantity	Total number of products the respondent has purchased during a one year period prior to march 2016.
Life Stage	The current life phase of respondent. Levels are e.g. "Young Single", "Singles and couples without kids", "Young family", "Established family", etc.
Kids at Home	Having kids at home. Binary variable for yes (1) and no (0).
Year of Birth	Birth of year of respondent.
Visits	How many visits the consumer has done to stores during a one year period prior to march 2016. Binary feature for regular visitor (1) and ordinary visitor (0).
Number of Weeks	How many different weeks the respondent has made purchases in the grocery chain one year prior to march 2016.

Table 4.4: Explanation of features used in the first data set from the grocery chain.

Features	Description
Membership points	How much points the consumer has collected during a one year period prior to march 2016.
Standard Food	Proportion spent on standard labeled food during a one year period prior to march 2016.
Ecologic Food	Proportion spent on ecologic food during a one year period prior to march 2016.
Campaigned Food	Proportion spent on campaigned food during a one year period prior to march 2016.
OB Food	Proportion spent on OB food during a one year period prior to march 2016.
OB Food, not from brand 1	Proportion spent on OB food but not from brand 1 (confidential brand) during a one year period prior to march 2016.
Premium Food	Proportion spent on premium labeled food during a one year period prior to march 2016.
Fair Trade Food	Proportion spent on fair trade labeled food during a one year period prior to march 2016.
Basis Food	Proportion spent on basis food during a one year period prior to march 2016.
Niche Food	Proportion spent on niche labeled food during a one year period prior to march 2016.
Category Medal	Represents the loyalty of the consumer during a one year period. A higher value indicates a more loyal customer in terms of amount of purchased products.
Share of Wallet	This represents the share of money the consumer spends on sale products in relation to the consumers purchasing power.
Needs Segment	The company's segmentation of consumer needs.
CSI	The respondents overall satisfaction with their most frequently visited store. Levels are the different grades on a scale from 1-10. Prediction variable.

Table 4.5: Explanation of features used in the first data set from grocery chain.

As stated in table above, the imputation variable here is "CSI" which is the consumer overall satisfaction of their most frequently visited store. The next step is to choose which of the features in Tables 4.4-4.5 are informative in terms of predicting "CSI". The rest of the features are discarded. The decision tree based algorithm is implemented and the results are plotted in Figure 4.3. The algorithm deemed variables that have to do with the proportion of money spent on different food products as well as the total money spent in the stores as the most relevant features for their store opinions. The variables concerning the respondents personal information such as gender, year of birth and kids at home were less informative. The running time of the tree based feature selection clocked in at roughly 15 minutes.

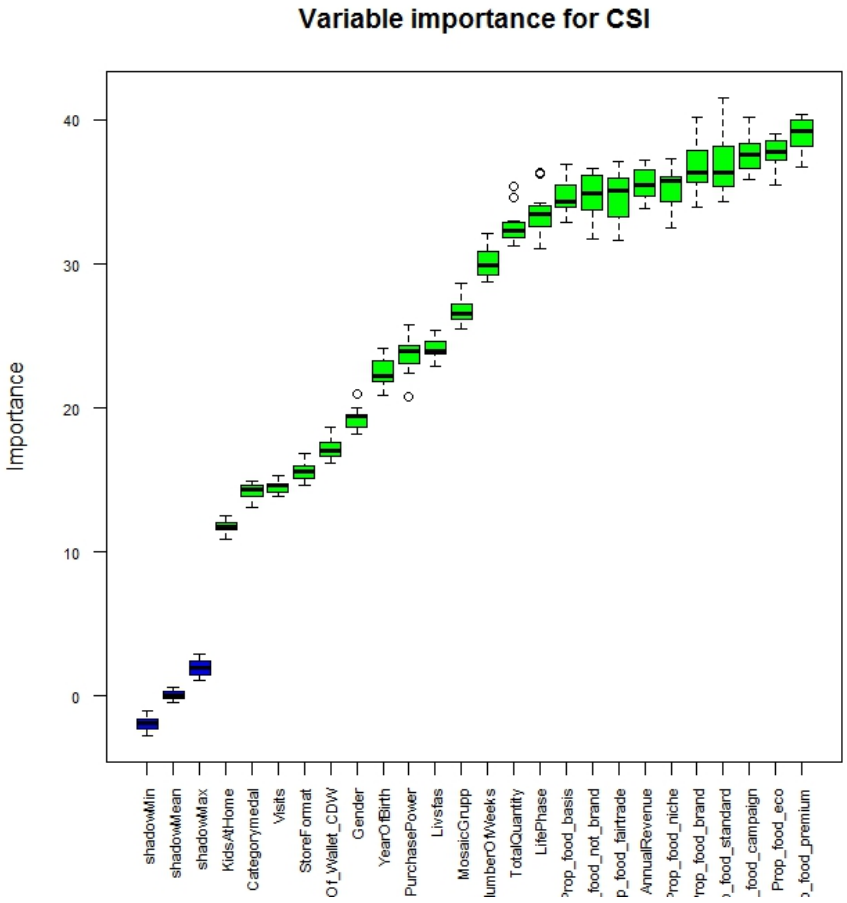

Figure 4.3: Feature importance from decision tree based algorithm

The results of the model predictions are summarized in Table 4.6.

Method	Accuracy	Kappa	Time	Weights
KNN	0.30	0.14	29.23	$1.4*10^{-3}$
CART	0.26	0.016	6.795	$1.0*10^{-6}$
Naive Bayesian Classifier	0.22	0.04	0.32	$1.0*10^{-6}$
Neural Networks	0.31	0.13	267	$1.4 * 10^{-3}$
Random Forest	0.57	0.46	2326	0.99

Table 4.6: Accuracy and Kappa values for each set of predictions.

The cross-enropy based ensemble method is applied once again and the results are presented in Table 4.7. The value on the weights are noted in the previous table.

Added Time [s]	Cross-entropy	Accuracy	Kappa
1233	1.25	0.57	0.47

Table 4.7: Values on Accuracy, Kappa, Cross-entropy and added time consumption when using cross-entropy based ensemble method.

Also, Hellinger's distance is investigated and presented in Figure 4.4 along with a scatter plot of Accuracy and Kappa.

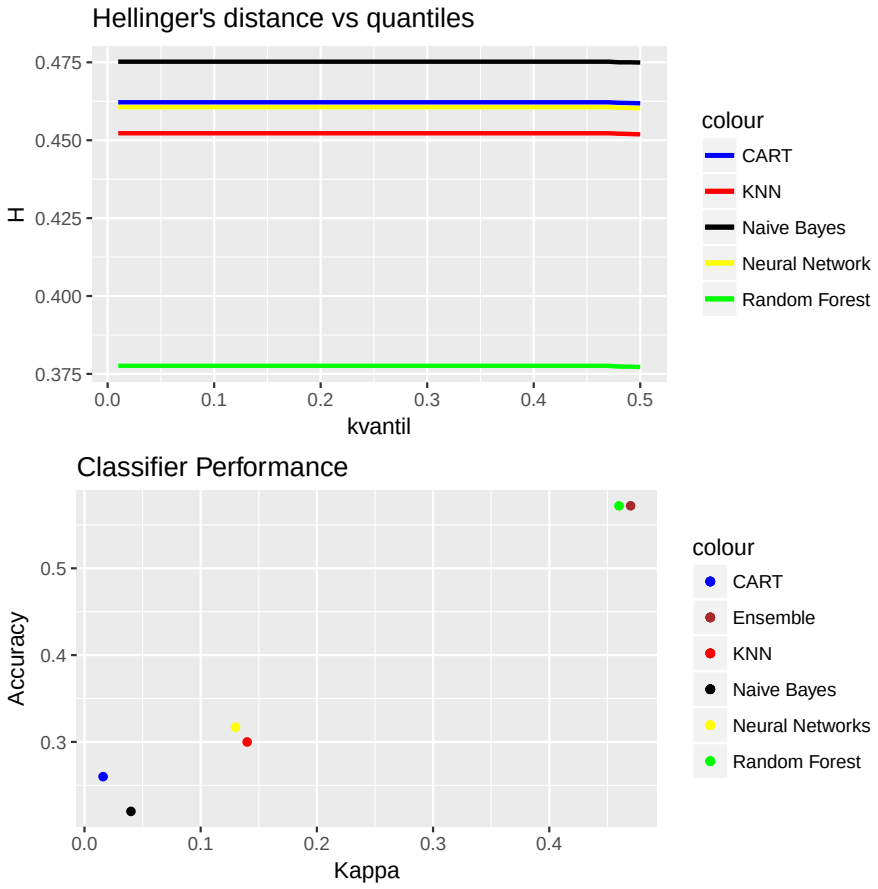

Figure 4.4: *Upper plot*: Hellinger's distance
Lower plot: Graph of Kappa and Accuracy values.

4.3 Regression of Consumer Satisfaction Index

In the second data set, there will be more emphasis on combining consumer attitude and transaction history from a time dependent perspective. Instead of considering data from a one year period prior to march 2016, we will consider half a year prior to march 2016. The response will once again be the consumers' grading of their most commonly visited store but the features will be fewer and somewhat modified. Moreover, the prediction will be a regression and thus differently evaluated. There will be 15000 observation from 13 features that directly connects transaction history during a specific time period with corresponding attitude. A more thorough description is given in Table 4.8.

Features	Description
Gender	The gender of the respondent, 1 is male and 2 is female
Purchase Power	A grouping of consumers purchasing power which is categorized from 1-8. The higher number a respondent has on this scale, the higher the income.
Mosaic Grouping	A categorization of the Social-demographic background of the respondent.
Total Price	The total amount of money spent on grocery products half a year prior to march 2016.
Category medal	This represents the share of money the consumer spends on sale products in relation to the consumers purchasing power. Value on levels ranges from "1" to "4".
Total Quantity	The total number of grocery products purchased.
Life Stage	The current life status of respondent. Levels are "Young Single", "Singles and couples without kids", "Young family", "Established family", "Parent(s) with kids that have moved out" and "Seniors and retired".
Kids at Home	Having kids at home. Levels are yes (1) and no (2).
Year of Birth	Birth of year of respondent. This feature, which originally took numerical values, has been decomposed into four levels represented by numbers from 1 to 4. The levels are: Child/Youth (age 1-20), Young Adult (age 21-40). Middle aged Adult (age 41-60) and Senior Adult (age 61+).
Needs Segment	Segmentation of respondent needs. Confidential. Three levels.
Visits	Total number of visits for a specific time period
Share of Wallet	This represents the share of money the consumer spends on sale products in relation to the consumers purchasing power.
CSI	The respondents overall satisfaction with their most commonly visited store. Prediction variable.

Table 4.8: The features used for the second data set from the grocery chain and a description of what they quantify.

Selection of significant variables is done with the decision tree based selection algorithm. The result of the algorithm is plotted in Figure 4.5. A quick review of this plot shows that features concerning the total amount of money spent on grocery products (*Total Price*) as well as the total quantity of products (*Total Quantity*) are most relevant for their satisfaction with their most frequently visited store. Feature concerning gender and needs segment are less prominent. The time consumption for the selection algorithm is roughly 9 min.

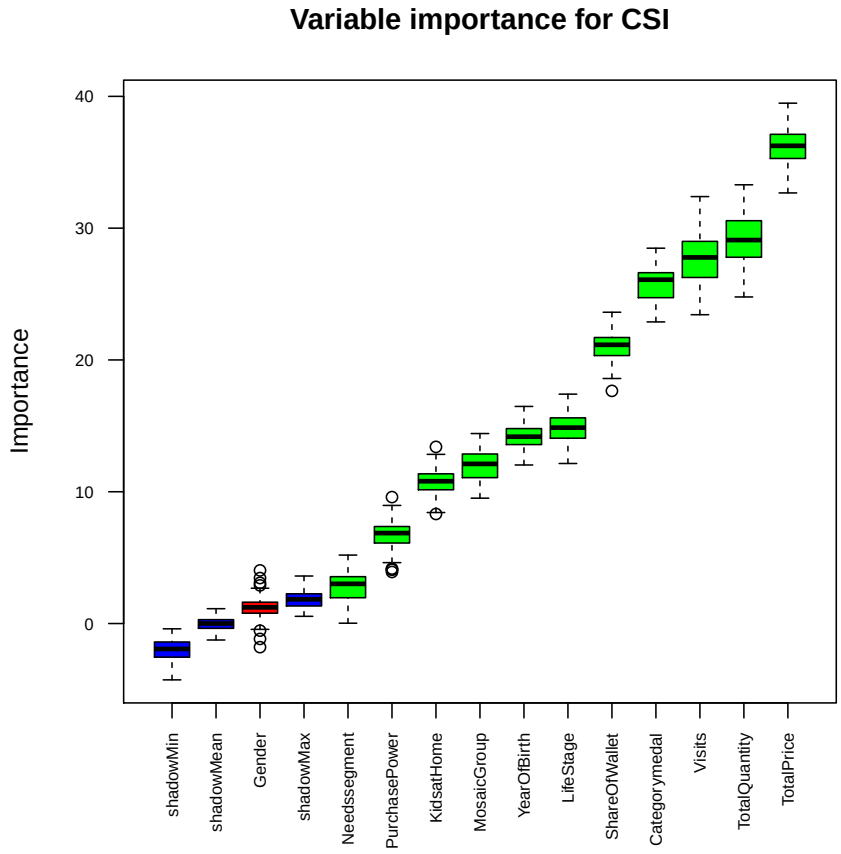

Figure 4.5: Feature importance using decision tree based algorithm.

The imputation of consumer satisfaction of stores is done by fitting a regression line to the data. The regression is evaluated using RMSE and mean absolute deviation (MAD). Results are summarized in Table 4.9.

Method	MAD	RMSE	Time [s]	Weights
KNN	1.65	2.09	6.2	0.069
CART	1.57	2.00	2.1	0
Linear Model	1.59	1.99	0.02	0.51
Neural Networks	1.58	2.01	213	0
Random Forest	1.58	1.99	1413	0.42

Table 4.9: MAD and RMSE for each model.

For this data set we want to apply the RMSE based ensemble method and investigate the performance of the proposed classifier. The weights are distributed in Table 4.9 and prediction performance is shown in Table 4.10.

Added Time [s]	MAD	RMSE
2	1.57	1.98

Table 4.10: Values on RMSE and MAD as well as added time consumption when using RMSE based ensemble method.

Finally, the Hellinger's distance is plotted for different discretizations as mentioned in Section 3.2.5. This is seen in Figure 4.6.

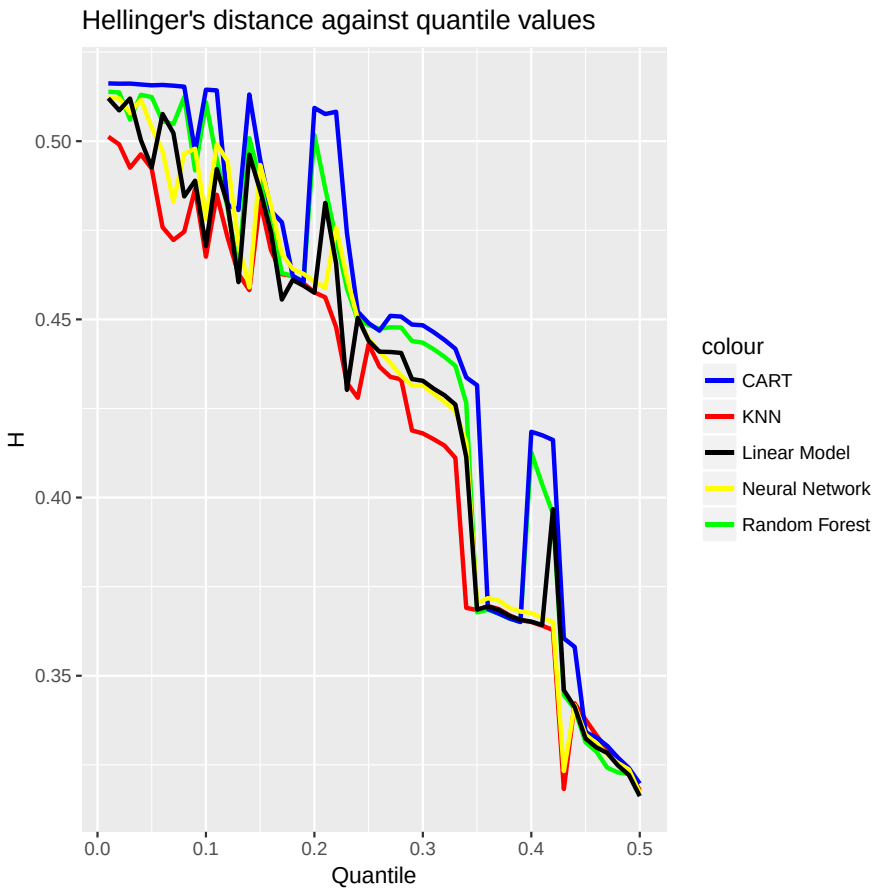

Figure 4.6: Hellinger's distance against different quantile discretizations.

Chapter 5

Discussion

5.1 Remarks on the First Classification

A direct review of Table 4.2 shows that, in terms of Kappa and Accuracy, the naive Bayesian classifier is the most superior one whereas ordinary (classification) decision tree perform the worst. Examining the time consumption shows that the naive Bayesian classifier is also the most efficient since it has by far the shortest running time. The random forest model gives a decent performance but the training phase takes a long time to complete. The method of nearest neighbours is also worth noting but there is no reason to consider anything else but the naive Bayesian classifier for this data set. The independence assumption of naive Bayes' does not seem to be as erroneous as in the data from the grocery chain. When this holds, even if only to a certain extent, the naive Bayesian classifier performs well on smaller sets of data. The sample size of the training data was limited to only 5000 observations which supposedly could be too sparse for classifiers such as random forest. Moreover, this data set only contained a single non-nominal feature which relaxes the very strong assumption of Gaussian distributed (bell curved) continues data.

Moving on to the ensemble classifier (Table 4.3), both Kappa and accuracy were increased as expected but not by a significant amount. It also added another 102 seconds to the time consumption in addition to the time required to train all the other classifiers. There is a trade off dilemma here, where one has to decide if it is worth training all the classifiers for a moderate increase in performance. The increase in time might be reasonable for this particular data set but in a more extensive context where the number of observation is large, the time duration might become too overwhelming for such a small increase in performance.

From Figure 4.2 it seems that the Hellinger's distance for each classifier agrees

well with its corresponding Accuracy and Kappa where the neural network is slightly preferred over KNN. The value on H does not seem to change much when making the discretization less fine and its shows a fairly stable behaviour for all values on the number of steps. The values ranges between 0.38 to 0.43 and the ensemble method gives the shortest distance which is expected since it produced the most precise set of predictions.

5.2 Remarks on the Second Classification

From Table 4.6 we can see that random forest is by far the most superior model in terms of prediction accuracy. The other model performances were quite poor and their Kappa indicates that the number of correctly predicted outcomes is close to what we can expect by chance. The Kappa of the random forest indicates a more robust performance. This particular data set seem well suited for the random forest model. Indeed, if we try to optimize an ensemble method based on cross entropy, the parameter values are weighted almost entirely towards random forest which is expected considering the prediction performance in terms of Accuracy and Kappa. The resulting classifier from the ensemble method will essentially be the same as random forest. The use of the ensemble method is therefore impractical since it will add a lot to time consumption but nothing to performance. However, it is a good example of how one does not loose any performance when using the ensemble method even though it might take longer time.

An inspection of the time consumption of all statistical models shows that the random forest model, although the most precise, is the one that consumes the most time. However, when the difference in performance is of this extent, a time analysis might become redundant unless the running times are of extreme sizes. We can take a note of the other models where neural networks and KNN performs decently but the measures show dominating advantage for random forest in this case. The complexity of this data (in comparison to other sets) seems to only be captured by the random forest whereas it makes the other models more crude. This palpable difference is noteworthy and somewhat suspicious. It might be instructive to investigate the structure of this data set more carefully to exclude any skewness in the data.

Hellinger's distance (Figure 4.4) changes very little and is stable for all values on the number of steps and it does not seem to matter which value we choose. This might stem from the fact that we have many features in the data set and the difference in probability distributions from different values on the response variable is small. It shows, however, a distinction between the classifiers where the random forest is giving the shortest distance by a large margin. The distances agrees well with the Kappa measure and the naive Bayesian classifier gives the farthest distance 0.47 whereas the random forest gives roughly 0.38.

5.3 Remarks on the Regression

In the previous data set, one classifier was vastly superior to the other but for this regression, the case seem to be the opposite. An investigation of Table 4.9 shows that the classifiers are giving very similar results. The differences in both RMSE and MAD are only of magnitude 10^{-2}. Considering that the regression variable takes values between 1-10, this small difference does not affect the predictions in a significant way. When evaluating only the performance, any model will do for this particular data. This is therefore a case where the consumption of time becomes the decisive factor. Fitting a linear model is by far the fastest way to produce new CSI output. The small inferiority in RMSE and mean absolute deviation is negligible in contrast to the much smaller consumption of time.

When can see that, on average, the prediction will deviate around 1.58 from the true observed value. This is an accurate enough prediction to distinguish satisfied consumers from the unsatisfied. If a response is predicted to be 9 then we can be comfortably sure that the respondent is at least not unhappy with the store. Same argument holds for predicted low values. The ensemble classifier barely makes any difference in terms of the performance measures. However, it does not consume a significant amount of additional time. It might be worth resorting to the ensemble method.

The Hellinger's distance shows a more fluctuating and unstable behaviour for this data set which most likely depends on the fact that the prediction variable is also discretized in order to give the outcomes distinct probabilities. The classifiers result gives both similar and more varied distances depending on the quantile values. The most stable behaviour seem to be between 0.2 and 0.3 where the KNN classifier gives the shortest distance and decision trees shows more separated probability distributions. When the other measures indicate indistinguishable performance levels, the Hellinger's distance can be an instructive way to choose a preferred classifier.

5.4 Future Work

This project lays the groundwork for future investigation of data fusion for consumer behavioural data. It took an automated approach for combining data sets where several models are used but not individually deeply investigated. Another plausible approach is to select only one or two of the listed models and put more emphasis on parameter estimation and model design. Significant features

were selected, in an automated way, using an iterative random forest algorithm but this result in a feature space which might be biased towards random forest and decrease the prediction power of the other models. Using different feature selection methods for different models can possibly give more instructive results.

Alternative ensemble methods can also be suggested where instead of minimizing the cross-entropy, we minimize Hellinger's distance. There are other measures that can be considered for this.

A more sophisticated research can be done for the trade off between model performance and time consumption where the goal is to find an optimization strategy for the gain in model performance versus the loss in time duration. This can be investigated in relation to delivery deadlines on client projects that use fused consumer data in order to find a model that is most appropriate for a specific delivery. This was only discussed on a speculative level in this thesis.

Chapter 6

References

[1] Adam, L. Absolute Deviation and Variance. (2013). [ONLINE] Avaible at: https:// statistics.laerd.com/statistical-guides/measures-of-spread-absolute-deviation-variance.php. [Accessed 15 April 2017].
[2] Barnston, A. (1992). *Correspondence among the Correlation [root mean square error] and Heidke Verification Measures; Refinement of the Heidke Score.* Notes and Correspondance. Climate Analysis Center, NMC/NWS/NOAA, Washington DC.
[3] Brownlee, J. (2014). Classification Accuracy is Not Enough: More Performance Measures You Can Use. Machine Learning Mastery. Preprint. Available at: http://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/ [Accessed 2017-04-05].
[4] Cao, X. (2007). *Model Selection Based on Expected Squared Hellinger's distance*, chapter 2, University Press of Colorado.
[5] Dickey, S.G., Pantula, D.A., Rawlings, J.O. (1998). *Applied Regression Analysis - A Research Tool*, 2ed, Springer, New York.
[6] Esteban, J., Bryanston-Cross, P., Hannah, P., Starr, A., Willets, R. (2005). A Review of Data Fusion Models and Architectures: Towards Engineering Guidelines. Neural Computing Applications, 14 (4) (2005), pp. 273-281.
[7] Fuller, R. (2010). *The Delta learning rule – Tutorial.* Lecture notes distributed in statistical learning theory. Available from http://uni-obuda.hu/users/fuller.robert/delta.pdf.
[8] Gilula, Zvi., McCulloch, E. Robert., Rossi, E. Peter. (2004). *A Direct Approach to Data Fusion.* University of Chicago.
[9] Gurney, K. (1997). *An Introduction to neural networks.* University of Sheffield. UCL Press.
[10] Gwet, K.L. (2014). *Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring the Extent of Agreement Among Multiple Raters.* Advanced Analytics, LLC, USA.
[11] Harchoui, Z., Lin, H., Mairal, J. (2016). A Generic Quasi-Newton Algorithm for Faster Gradient-Based Optimization. arXiv: 1610.00960v2 [stat.ML].

[12] Hastie, T., James, T., Tibshirani, R., Witten, D. (2013). *An Introduction to Statistical Learning with applications in R.* Springer. New York.
[13] Iba, W., Langley, P., Thompson, K. (1992). *An analysis of Bayesian Classifiers.* Proceedings of the Tenth National Conference on Artificial Intelligence (pp. 223-228). San Jose, CA: AAAI Press.
[14] Kuhn, M. (2017). Caret: Classification and Regression Training. R package version 6.0-76.
[15] Nielsen, M. (2015). *Neural Networks and Deep learning.* Determination Press.
[16] Ray, S. (2015). *6 Easy Steps to Learn Naïve Bayes Algorithm (with code in Python).* [ONLINE] Available at: https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/ [Accessed 2017-03-10].
[17] Takama, Y., Ursino, D. (2013). *A Review of Data Fusion Techniques.* The Scientific World Journal. Volume 2013. Article ID 704504.