Alma Mater Studiorum
Università degli Studi di Bologna

Facoltà di Scienze Statistiche

Corso di Laurea Magistrale in Sistemi Informativi per l'Azienda e la Finanza

Tesi di Laurea in Data Mining e Supporto alle Decisioni

Comparison of Data Mining Techniques for Insurance Claim Prediction

Candidato:
Andrea Dal Pozzolo

Relatore:
Prof. Gianluca Moro

Correlatori:
Prof. Gianluca Bontempi
Dott. Yann Aël Le Borgne

Anno Accademico 2010/2011 - Sessione II

Abstract

This thesis investigates how data mining algorithms can be used to predict Bodily Injury Liability Insurance claim payments based on the characteristics of the insured customer's vehicle. The algorithms are tested on real data provided by the organizer of the competition. The data present a number of challenges such as high dimensionality, heterogeneity and missing variables. The problem is addressed using a combination of regression, dimensionality reduction, and classification techniques.

Questa tesi si propone di esaminare come alcune tecniche di data mining possano essere usate per predirre l'ammontare dei danni che un' assicurazione dovrebbe risarcire alle persone lesionate a partire dalle caratteristiche del veicolo del cliente assicurato. I dati utilizzati sono reali e la loro analisi presenta diversi ostacoli dati dalle loro grandi dimensioni, dalla loro eterogeneitá e da alcune variabili mancanti. ll problema é stato affrontato utilizzando una combinazione di tecniche di regressione, di riduzione di dimensionalitá e di classificazione.

Contents

Chapter 1

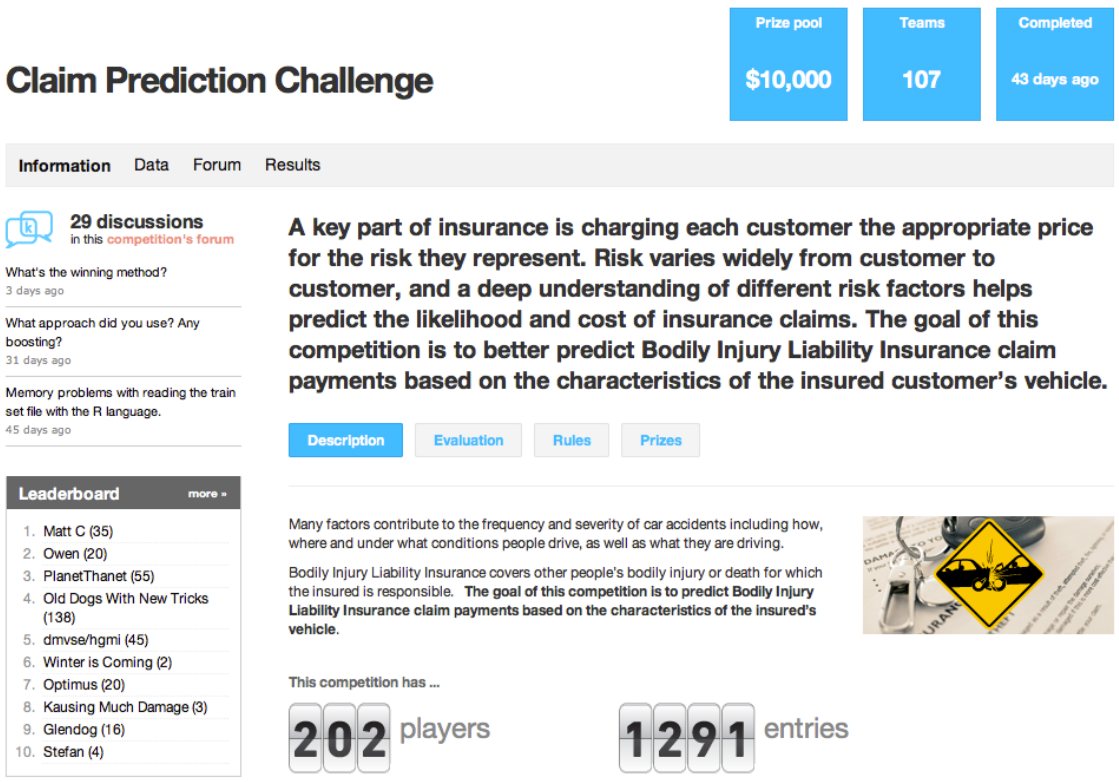
Introduction

With the increasing power of computer technology, companies and institutions can nowadays store large amounts of data at reduced cost. The amount of available data is increasing exponentially and cheap disk storage makes it easy to store data that previously was thrown away. There is a huge amount of information locked up in databases that is potentially important but has not yet been explored. The growing size and complexity of the databases makes it hard to analyze the data manually, so it is important to have automated systems to support the process. Hence there is the need of computational tools able to treat these large amounts of data and extract valuable information.

In this context, Data Mining provides automated systems capable of processing large amounts of data that are already present in databases. Data Mining is used to automatically extract important patterns and trends from databases seeking regularities or patterns that can reveal the structure of the data and answer business problems. Data Mining includes learning techniques that fall into the field of Machine learning. The growth of databases in recent years brings data mining at the forefront of new business technologies [WF05].

To apply and develop their new research ideas, data scientists need large quantities of data. Most of the time however business valuable data is not freely available, so it is not always possible for a data expert to have access to real data. Competitions are usually the occasion for data miners to access real business data and compete with other people to find the best technique to apply to the data. The Kaggle website (`http://www.kaggle.com/`) is a web platform where companies have the opportunity to post their data and have it scrutinized by data scientists. In this way data experts have the opportunity to access real dataset and solve problems with the opportunity to win a prize given by the company.

The competition used for this thesis was a *Claim Prediction Challenge* [http://www.kaggle.com/c/ClaimPredictionChallenge] proposed by Allstate Insurance. The goal of the competition was to predict the amount of money the insurance has to pay to its clients. This amount represents an important figure in order to determine the corresponding premium asked to their customers. Many factors contribute to the frequency and severity of car accidents including how, where and under what conditions people drive, as well as the type of vehicle they use. Bodily Injury Liability Insurance covers other people's bodily injury or death for which the insured is responsible. The company provided data from 2005 to 2007 to contestants, who analyzed correlations between vehicle characteristics and bodily injury claims payments to predict claims payment amounts in 2008. Using the data provided, modelers evaluated how factors such as horsepower, length and number of cylinders affect the likelihood of an insured being held responsible for injuring someone in a car accident.



Claim Prediction Challenge

Prize pool	Teams	Completed
$10,000	107	43 days ago

29 discussions in this competition's forum

What's the winning method?
3 days ago

What approach did you use? Any boosting?
31 days ago

Memory problems with reading the train set file with the R language.
45 days ago

Leaderboard

1. Matt C (35)
2. Owen (20)
3. PlanetThanet (55)
4. Old Dogs With New Tricks (138)
5. dmvse/hgmi (45)
6. Winter is Coming (2)
7. Optimus (20)
8. Kausing Much Damage (3)
9. Glendog (16)
10. Stefan (4)

A key part of insurance is charging each customer the appropriate price for the risk they represent. Risk varies widely from customer to customer, and a deep understanding of different risk factors helps predict the likelihood and cost of insurance claims. The goal of this competition is to better predict Bodily Injury Liability Insurance claim payments based on the characteristics of the insured customer's vehicle.

Many factors contribute to the frequency and severity of car accidents including how, where and under what conditions people drive, as well as what they are driving.

Bodily Injury Liability Insurance covers other people's bodily injury or death for which the insured is responsible. **The goal of this competition is to predict Bodily Injury Liability Insurance claim payments based on the characteristics of the insured's vehicle.**

This competition has ...

202 players   1291 entries

Figure 1.1: Claim Prediction competition

The contribution of this thesis is to analyze and discuss the advantages and disadvantages of the standard techniques used in data mining and machine learning. The thesis first presents a general overview of data mining in Chapter 2 and provides a state of the art of different learning algorithms. In Chapter 3, we detail the data and the problem addressed by this competition, and present the methodology that will be used. Chapter 4 presents and discusses the experimental results which were obtained. Chapter 5 finally concludes the thesis with a summary of the results and the methodology, together with future work which could be investigated in order to improve the work presented in this thesis.

Chapter 2

Data Mining algorithms: overview

2.1 Data Mining definition and notations

Data mining is a field of computer science that involves methods from statistics, artificial intelligence, machine learning and data base management. The main goal of data mining is to find hidden patterns in large data sets. This means performing automatic analysis of data in order to find clusters within the data, outliers, association rules and prediction models that can explain the data. The information discovered by data mining can be learnt by machine learning algorithms and applied to new data. What is important is that the patterns found by data mining are useful to explain the data and/or make predictions from it.

Tom Mitchell [Mit97] gives a nice definition of what 'learning for a computer" means:
A computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at task in T, as measured by P, improves with experience E

In learning program three things must be defined:

- the task
- the performance's measure
- the source of experience (training experience)

Machine learning is the subfield of computer science that deals with the design of algorithms and techniques that allow computers to learn [LB09].
The goal of Machine Learning is to find algorithms that can learn automatically from the past without the need of assistance. The algorithm is trained to learn from data and come up with a model that can be applied to new data.

These algorithms are data driven, so they are not guided by the impressions of a human expert. To see if an algorithm has effectively learnt something useful or not, it is usually tested with a new data set whose outcome is known in order to evaluate its outcome against the real one.

Let's introduce some data mining concepts that will be use in the thesis.

- **Dataset** A dataset is a collection of data of the same phenomenon given in a tabular form. The columns represent the attributes or variables. The rows, the instances/examples belonging to the dataset. Each instance has k values, one for each of the k attributes in the dataset.
- **Attribute / Feature** An attribute is one of the available variables present in the dataset. This means that the column of a dataset represent the possible values taken by an attribute.
- **Dimension** The dimension of a dataset is equivalent to the number of attributes present in the dataset. In case the dataset is composed of only two variables, its dimension is two.
- **Training and Testing sets** The training and test set are different collection of data of the same phenomenon used in supervised learning. The first is used to train an algorithm to learn, the second to evaluate the learning.
- **Distribution** A distribution is a functions that describe the frequencies of the values within the phenomenon.
- **Outlier** An outlier is an observation that appears to deviate markedly from other members of the sample in which it occurs [Gru69].

Data mining and Machine learning algorithms can be classified into supervised or unsupervised learning depending on the goal of the algorithm. Supervised methods are used when there is a variable whose value has to be predicted. Such a variable is referred to as response or output variable. In an unsupervised method, the data are not labeled and there is no value to predict or classify.

Supervised learning algorithms generate a function that is able to map the input/output values. In these algorithms the data provides examples about the kind of relationship between the input and output variables that has to be learnt. In unsupervised learning, there is no output value, but instead just a collection of input values.

Supervised learning algorithms can be further divided into classification and regression algorithms. In the case of classification, the goal is to predict qualitative or categorical outputs which assume values in a finite set of classes (e.g. True, False or Green , Red, Blue, ecc..) without an explicit order. Categorical variables are also called Factors. In regression the output to predict is a real-valued number.

2.2 Supervised learning

In Supervised learning, the *training set* consists of several input variables and an output variable which is a categorical variable in the case of a classification problem and a real-valued number in the case of regression. The goal of supervised algorithms is to uncover the relationships between input/output variables and to generalize the function so it can be applied to new data. The output variables are in general assumed to be dependent on the inputs. In this thesis input variables will be called X and the output Y.

Classification and regression learning algorithms fall into supervised methods because they work under the supervision of an output variable. This output variable is called the class or response value. The success of supervised learning is evaluated by applying the model on new set of data called *test data*, for which the values of the output are known but not used during the learning stage. The prediction accuracy on this test set can be used as a measure of a model's ability to predict the response value.

A supervised learning therefore requires to have a measure of how well or bad the model predicts the response variable. This is expressed with a loss function that is used to assess the adequacy and effectiveness of different methods. Before applying one model to a test set, the model accuracy is usually estimated using a cross-validation approach with the training data.

The elements of a supervised learning algorithm are:

- A target operator $y(x)$ that describes the relationship between input-output variables. This is a stochastic process that represents the distribution of $y(x)$ given x, $(P(y|x))$.
- A training set that contains the data where the relationship between the inputs-outputs variables can be learnt.
- A prediction model that estimates the input/output variables relationship by means of a functions and returns a predicted value for each examples.
- A loss function $L(y, \hat{y})$ that is used to measure the goodness of the model by means of the differences between the predicted output $\hat{y}$ and the actual output y.
- A learning procedure that tests different model parameter in order to minimize the loss function and finds the best model for the target function.

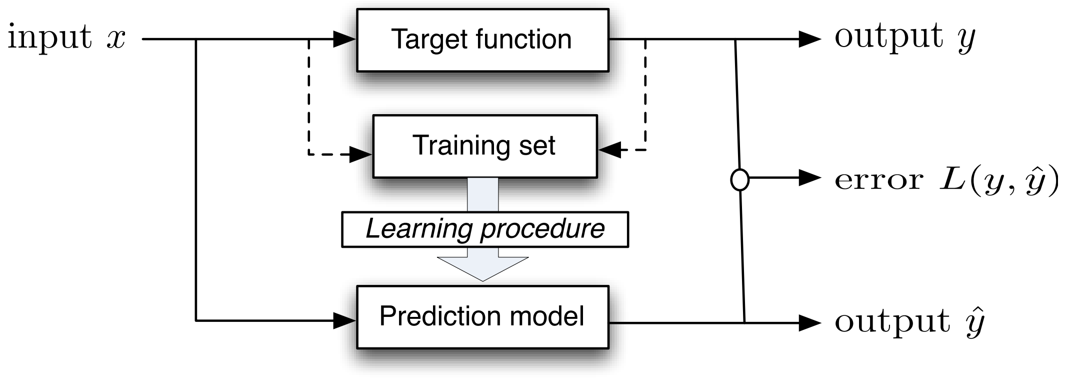


Figure 2.1: Supervised learning actors and procedure [LB09].

The goal of the learning procedure is to obtain a model which is able to find the dependency between the output and input variable. The model should be general enough to be applied to new data and return a small error. In general, complex models tend to overffit, i.e., they give good performances when applied to the training data, but poor results when applied to new data. On the contrary simple models tend to underffit the data, since they cannot represent important information in the model. Hence a model has to be general enough to return good performances in different datasets without losing discrimination power. The assessment of the generalization ability is called validation and this is done using techniques such as cross validation.

A k-fold cross validation consists in partitioning the training data into k smaller subsets of equal size. Then $k - 1$ subsets are used as a training set and the remaining subset as a test set. This allows to compute the model on a new training set and assess its generalization error on the subset used as the test set. This partition of the training set is repeated K times and at each time the subset used for testing is changed. At the end, the validation procedure provides an estimate of the generalization error and this can be used to select between different model's parameters.

In the case of a 10 fold cross validation the training data is divided into 10 different parts of equal size. Then one tenth of the instances present in the training set are used for testing and the remaining nine tenth for the training (figure 2.2).

Once the first round of validation is completed, another subset of equal size is used for testing, and the remaining 90% of the instances used for training as before (figure 2.3). The process is iterated 10 times to ensure the all instances become part of the training and test set.

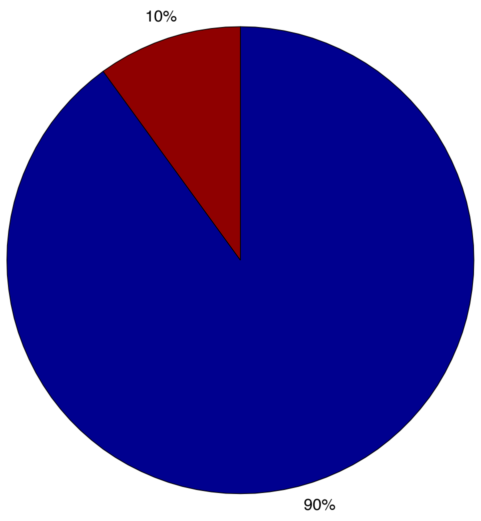


Figure 2.2: train (blue) and test set (red) used in the first round of a 10 cross validation

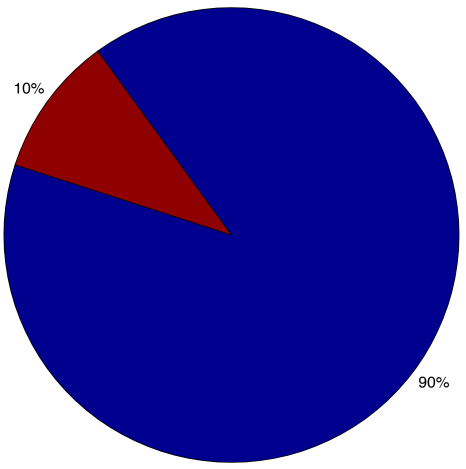


Figure 2.3: train (blue) and test set (red) used in the second round of a 10 cross validation

2.3 Classification

In a classification problem the output variable, also called class variable, is a factor taking two or more discrete values. The input variables can take either continuous or discrete values.

In the claim prediction competition the problem can be seen as a two-class prediction problem whereby instances are labeled as positives (P) if the *Claim Amount* is greater then zero and as negatives (N) in the opposite case. Therefore there are four different cases:

- an instance is predicted as P and the actual value is also P: this is said to be a true positive (TP)

- an instance is predicted as P but the actual value is N: this is called a false positive (FP)
- an instance is predicted as N and the actual value is also N: this is a true negative (TN)
- an instance is predicted as N but the actual value is T: this is a false negative (FN)

This information can be summarized into a “Confusion Matrix”, a matrix that makes it easy to visualize how precise the classification is. The columns of the matrix return the number of instances in a predicted class, while the rows give the number of instances in the actual/real class. The number of misclassified instances can be computed summing the FP and FN instances.

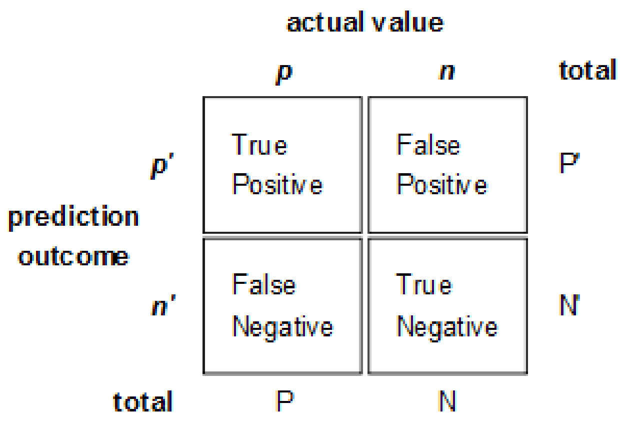


Figure 2.4: Confusion Matrix

A perfect classifier would have zero FP and zero FN meaning that all the instances are well classified. The entries of the matrix can also be combined in order to extract other measures. Typical measures obtained by the confusion matrix are:

- Precision = TP / (TP+FP)
- Recall/ Sensitivity = TP / P = TP / (TP+FN)
- Accuracy = (TP+TN) / (P+N)
- Specificity = TN / (FP+TN)

These other measures are useful for example when a dataset is unbalanced since the classifier can be biased towards the class that is more frequent affecting the accuracy.

For example, in the case where 95% of the instances are positives and only 5% negatives, the classifier could easily classify all the instances as positives leading to an accuracy of 95%. For this reason the accuracy is not a reliable measure of classifier's performance in unbalanced problems [WIK].

Another useful tool to assess a binary classifier is the Receiver Operating Characteristic (ROC) Curves. The ROC curve is a plot of the Recall, also called Sensitivity, versus (1 - Specificity), and is obtained by varying the discrimination threshold of the classifier. The ROC curve can also be represented by plotting the fraction of TP out of the positives (true positive rate or TPR) versus the fraction of FP out of the negatives (false positive rate or FPR). For this reason it is also known as a Relative Operating Characteristic curve, because it is a comparison of two operating characteristics (TPR & FPR) [WIK]. In the ROC curve the best possible prediction method would yield a point in the upper left corner representing no false negative and 100% specificity (no false positive). On the contrary a random classification would give a point along a diagonal line (line of no-discrimination) from the left bottom to the top right corner [WIK]. Different ROC curves from several models can be compared to find optimal models and eliminate suboptimal ones.

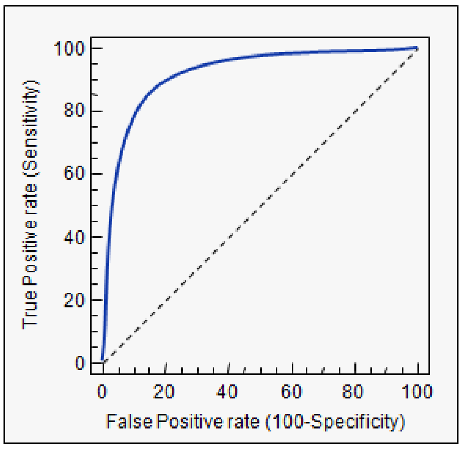


Figure 2.5: ROC curve

Confusion matrix and ROC curves do not take into consideration the cost of misclassification and the priori class distribution. The cost of having a high number of false negatives may be much higher than the cost of having an equivalent number of false positives.

2.3.1 Decision Tree

Decision tree algorithms, as the name suggests, are algorithms that can be represented graphically by a tree with many branches and leaves. Instances are classified going down from the root to the leaf nodes [WKRQ$^+$08]. The nodes in the tree are tests for specific attributes. The leaves contain the predictions for the response variable.

Trees can be used for both classification and regression problems. In a classification tree the predicted value is one of the possible levels of the response variable.

The Tree construction may be seen as a variable selection method [GE03]; at each node the issue is to select the variable to divide upon and how to perform the split. Decision tree algorithms differ in fact from the method of selecting the attribute used to test the instances at each node.

The attribute that alone classifies best the data is used at the root, then descending the tree the second best attribute is used to split further the data and so on.

The term *Entropy* is used to measure the "impurity" of the training dataset and can be computed for binary classification problem as well as for classification with the target attribute taking more levels. If for each node i of a classification tree, the probability of the instance to belong to the k class is p_{ik}, the entropy is then calculated as $\sum p_{ik} log\ p_{ik}$. The reduction of entropy can be itself a measure of the information gained partitioning on an attribute [VR02b].

Given a tree, to predict a new instances the classification begins at the root and the instance go down the tree testing different attributes until a leaf is reached. In general a decision tree model works well when:

- Each attribute of the dataset takes a small number of different levels.
- Learning rules do not overlap.

When the tree algorithm is applied, some common issues are the size of the tree, how to handle missing values/outliers and the appropriate attribute selection at each node. Noise in the data can easily lead to an over fitted tree on the training set giving a model that does not perform well in the test set. The size determines the complexity of the tree and has to be neither too simple or too big leading to overfitting.

In order to avoid overfitting classical approaches are:

- Stop growing approaches.
- Post-pruning approaches.

In general, post pruning approaches have been preferred since it may be difficult to detect when to stop the tree growing. Regardless of the method used to avoid overfitting, the most important thing is to the determine the correct final size of the tree. There are many ways to get the correct tree size, including the use of a separate set to determine when the further split improves the model and the use of a measure of complexity. The use of a separate set to compute the correct size consists in considering each node in the tree as a candidate for pruning. Whenever the tree applied in the test set performs worse than the original give by the training set, the node is removed and it becomes a leaf. The accuracy of the model applied to the test set increases with the proceeding of pruning.

A standard decision tree algorithm for classification problems is the C4.5 decision tree algorithm that was initially developed by Ross Quinlan [Qui93]. The C4.5 algorithm extends Quinlan's earlier ID3 tree algorithm to address certain practical issues such as orverfitting and the type of variables accepted in input.

Overfitting is a problem that affects all algorithms not only decision trees. It occurs when there is either noisy data or when the set of training data is not representative enough because it is too small. In general there is overfitting when there is a model that is excessively complex and it have poor predicting power when applied to unseen data. With noisy data the ID3 algorithm will create a more complex tree with inconsistencies.

C4.5 algorithm is able to use not only binary attribute, but also attributes with many levels and numerical ones. In the case of numerical attribute, the test is converted to a binary problem by identifying a threshold to see if the instances are greater or smaller than it. At each node, the criteria to select the attribute is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision [WIK].

Using the *normalized* information gain criteria avoids the problem to favor the selection of attributes with many outcomes that happens with the information gain criteria [WKRQ$^+$08]. The growth of the tree terminates when all instances in the training set can be classified. The class with more number of instances decides the class of the leaf. Pruning the tree is essential to avoid over fitting. The C4.5 algorithm uses a simplification of the *cost-complexity pruning* use in CART algorithm [DEO11], it is called the *reduced error pruning.* This method requires a different dataset for prediction. The test set permits evaluation of whether replacing a subtree to a leaf gives an equal or smaller number of error in the test set. If this is the case the subtree is pruned and replaced by a leaf, with the process of removing branches that do not help continuing until the classification error starts to increase.

2.3.2 Random Forest

Random forest is an algorithm that consists of many decision trees. It was first developed by Leo Breiman and Adele Cutler [Bre01]. The idea behind it is to build several trees, to have the instance classified by each tree, and to give a "vote" at each class [CZ01]. The model uses a "bagging" approach and the random selection of features to build a collection of decision trees with controlled variance. The instance's class is to the class with the highest number of votes, the class that occurs the most within the leaf in which the instance is placed. In this algorithm, all trees are grown in this way:

1. Let N be the number of instances in the training. Sample randomly N instances with replacement and use this sample as the training set for growing one tree.

2. Let M be the number of input variables. Select a number m smaller than M such that at each node, m variables are selected randomly from the M variables, and select the best split within the m variables. The value of m is held constant during the forest growing.

3. Each tree is grown unpruned to its largest extent possible.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.

- The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

The correlation and strength of the forest increases with the number m of variables selected. A smaller m returns a smaller correlation and strength. To improve the prediction's accuracy, a bootstrap method is used to create different trees. Every time a tree is created, one-third of the bootstrap sample is kept aside to estimate the test error. The subset that is not included in the tree construction is used as a test set for the same tree. The error of the tree on this test set is called the "out-of-bag" error.
This out-of bag (OOB) error is used to get a running unbiased estimate of the classification error as trees are added to the forest and to get estimates of variable importance [CZ01]. Once a tree is built, it is used to predict all data and this allows to compute proximities. Every time two instances fall into the same node, it increases the proximities. This computation is done for each tree and proximities can be used to replace missing values. One of the main advantages of this method is it that can handle thousands of input variables without variable deletion. On the other hand Random forests do not handle large numbers of irrelevant features, and are biased in favor of variables with high number of levels [WIK].

2.3.3 Naïve Bayes

The Naïve Bayes classifier receives the name from the well known Bayes' theorem [WIK]. We first define the Bayes' theorem before introducing the Naïve Bayes algorithm.
Let us suppose that the probability of a certain hypothesis h is $P(h)$, this is also called the *a priori probability* of h. When the probability of an event E is $P(E)$, the conditional probability of the event E given that the hypothesis h holds is $P(E|h)$. On the contrary $P(h|E)$ is the probability that given the event E, h will hold. The Bayes theorem is then defined by the following equation:

$$P(h|E) = \frac{P(E|h)P(h)}{P(E)}$$

In a learning scenario, given that the event E occurred, it is interesting to find which hypothesis is the most probable in a set of H possible hypotheses. This means to find the hypothesis with the higher $P(h|E)$. This probability is also called *a posteriori probability*. The maximum a posteriori probability h_{max} is found by:

$$h_{max} = arg \max_{h} P(h|E)$$

Using the notion of conditional probability this becomes

$$h_{max} = arg \max_{h} \frac{P(E|h)P(h)}{P(E)}$$

From this later equation, $P(E)$ can be removed since it is independent of h. Therefore we get:

$$h_{max} = arg \max_{h} P(E|h)P(h)$$

In a classification problem with class variable Y and input variables X, the Bayes optimal classifier can be defined as the classifier that maximizes the following:

$$arg \max_{y_j} Prob(y_j|x)$$

Using the Bayes theorem this becomes:

$$\hat{y} = arg \max_{y_j} P(y_i|x) = \frac{P(x|y_i)\,P(y_i)}{\sum\limits_{j} P(x|y_j)\,P(y_j)}$$

and that is equivalent to maximize:

$$arg \max_{y_j} P(y_i|x) = P(x|y_i)\,P(y_i)$$

where $P(y_i)$ is the a priori probability and can be estimate using the class frequency.

The Naïve Bayes uses the strong assumption that the input variables X are independent from each other conditionally to the distribution of Y so that the amount to maximize becomes:

$$\hat{y} = arg \max_{y_j} P(y_i|x) = P(y_i) \prod_{j=1}^{n} P(x_j|y_i)$$

Even if this algorithm may seems to simple at a first glance it has proved to perform well in several datasets [CNM06].

2.3.4 K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a *instance-based* learning algorithm, also known as a lazy learning algorithm [BBB99] since it lacks of a specific training model as eager learning does. This means that all computations are delayed until the classification. Such an algorithm stores the training data and retrieves a set of similar related instances when classification is required. The algorithm is based on the idea the similar instances tend to have similar solution [Zav97]. For each instance in the training set, it finds the k nearest instance and assign the original instance to the class which occurs the most within the k nearest objects.

In this algorithm the key elements are:

- the set of K-nearest instances used.
- the way to calculate the distances from the instance to be classified and the other K nearest neighbors.
- the number of K instances to include in the group.
- the way to classify an instance based on the distance and the classes of the K nearest neighbors.

The K number is an important parameter. If it is too small the result is sensitive to noisy data. On the other hand, if it is too high the K nearest neighbors can include instances from different classes. The best value of K can be estimated using a cross-validation approach, where the algorithm is repeated for different numbers of K.
Once the distances are computed, the class is defined by a voting function. A standard voting method is the *majority voting.* This method assigns equal weight to each neighbor and the class chosen is the one with the highest number of votes. This voting method gives equal importance to close and far neighbors. A weighting voting method can assign more importance to close neighbors instead. The voting system proposed by Dudani [Dud76] gives a weight of 1 to the nearest neighbor and 0 to the furthest, the other neighbors are weighted linearly between 0 and 1 with the following function:

$$w_j = \frac{d_K - d_j}{d_K - d_1}$$

where w_j is the weight of the j-th nearest neighbor and d_j is its distance. Then d_1 corresponds to the distance from the nearest neighbor and d_k the distance of the furthest.

Typically the Euclidian or the Manhattan distances are taken as distance measures. Euclidean distance is typical for continuous variables, but other metrics can be used for categorical data. However in the case of datasets with many attributes and where one attribute varies within a range that is much bigger compared to the others, the Euclidian distance becomes less discriminating, and the attribute with the much higher range influence the computation of distance. In n dimension, the Euclidian distance between two points p and q is defined as:

$$\mathrm{d}(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

With the majority voting method, the class vote v that occurs the most within the K nearest points is given by the following formula:

$$f(x_q) \leftarrow arg \max_v \sum_{j=1}^{k} \delta(v, f(x_i))$$

In this function $\delta(a,b) = 1$ if $a = b$ and $\delta(a,b) = 0$ otherwise.

2.3.5 Neural Network (NN)

The Neural Network at the beginning was studied to find an algorithm that mimics the human brain. The central idea of a NN is to extract linear combinations of the inputs as derived features, and then model the target variable as a nonlinear function of these features [HTF04]. The NN algorithm family is quite large. A frequent NN structure is the single hidden layer back-propagation network or single layer perceptron. A NN is a statistical model that can be represented by a *network diagram* as in Figure 2.6
Let us consider a K-class classification problem, with K binary class variables Y_i. Suppose there are p input variables defined by a vector X. M intermediate variables Z_m can be created as function of X in order to compute the Y_k class variables as a linear combinations of the Z_m.

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X) \quad m = 1, ..., M$$

$$T_k = \beta_{0k} + \beta_k^T Z \quad k = 1, ..., K$$

$$f_k(X) = g_k(T) \quad k = 1, ..., K$$

where $\sigma(v)$, called the activation function, is a nonlinear transformation such as logistic or sigmoid function $\sigma(v) = \frac{1}{(1+e^{-v})}$. The intercepts α_{0m}, β_{0k} represent an additional input that captures the bias in the hidden and output layers.

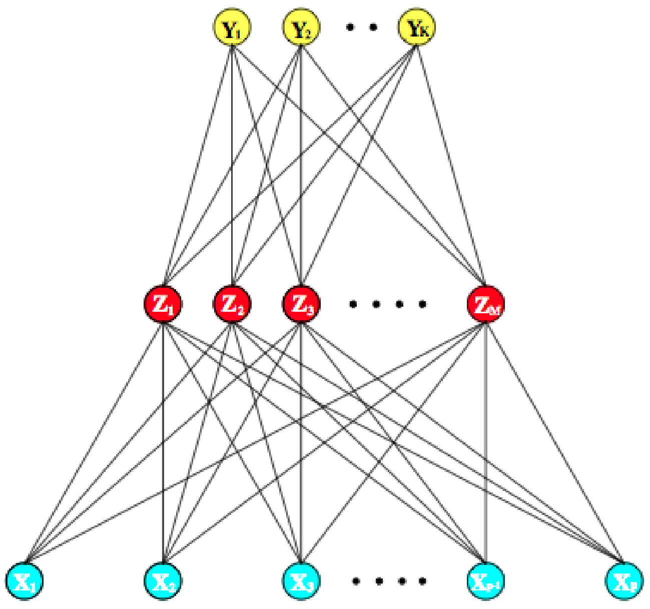


Figure 2.6: Schematic of a single hidden layer, feed-forward neural network

In the plot of Figure 2.7 the red curve correspond to the sigmoid function $\sigma(v) = \frac{1}{(1+e^{-v})}$. The sigmoid function can be multiplied by a scaling parameter s to have $\sigma(sv)$. This parameter s controls the activation rate at $v = 0$. A large value of s correspond to hard activation rate as in the purple curve, small value means a softer activation as in the blue curve. The values of s in this case are respectively 10 and 0,5.

The level between the input and response variable is called *hidden layer* and the Z_m variables that belong to it are called *hidden variables*, or units, since they are not directly observed. In more complex NN algorithms, there may be more hidden layers. Since it was first studied to model human brain, each hidden unit is also called a neuron, and the connections between the different layers represent the synapses. The NN can be thought as a linear model taking the transformation Z_m as input. If σ is the identity function the whole model becomes a linear model in the X.

When the model is applied to the data, the weights of the linear combinations are usually fitted by minimizing the sum-of-square errors $SSE = \sum_i \sum_k (y_{ik} - f_k(x_i))^2$, with a gradient descent approach called *back-propagation*. Back-propagation takes advantage

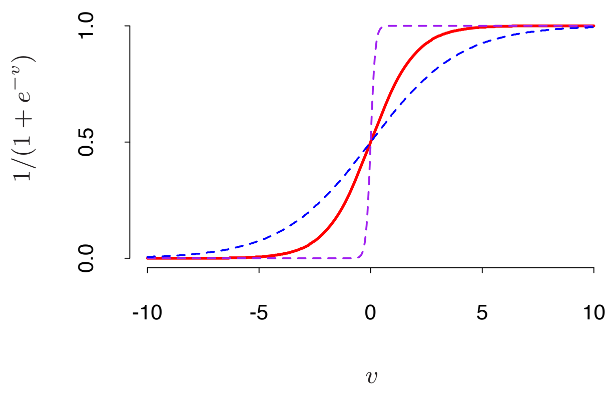


Figure 2.7: Plot of the sigmoid function

of the structure of the network to recursively compute the gradient. The weights are initialized with random values and are changed in a direction that will reduce the error [Bon08].

Let us define the set of weights by θ, which consists of

$$\{\alpha_{0m}, \alpha_m; \quad m = 1, 2, ..., M\} \quad M(p+1) \quad \textit{weights}$$

$$\{\beta_{0k}, \beta_k; \quad k = 1, 2, ..., K\} \quad K(M+1) \quad \textit{weights}$$

and the sum of square to minimize as

$$R(\theta) = \sum_{i=1}^{N} R_i(\theta) = \sum_{k=1}^{K}\sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2$$

Gradient descent minimizes $R(\theta)$ given that the derivatives of R with respect to parameter α and β are computed with the following functions:

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i))g'_k(\beta_K^T z_i)z_{mi}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g'_k(\beta_K^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il}$$

At each iteration, gradient descent update the weights using

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}}$$

where γ_r is the learning rate. Using the term δ_{ki} for the error of the model at the output level and s_{mi} as error at the hidden layer level

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = s_{mi} x_{il}$$

the errors have to satisfy the back-propagation equations:

$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki}$$

The procedure can be summarized in two step:

1. the *forward part* computes the initial weight using a linear combination of the input values.

2. the *backward part* propagates the error via the back-propagation equations.

In the NN algorithm a key factor is the number of neurons in the hidden layer. Increasing the number of neurons increase the computational time for the model.

2.3.6 Support Vector Machine (SVM)

In a classification problem the goal is to find a model that separates well instances belonging to two or more classes. In the case of Support Vector Machines, an instance is viewed as a p-dimensional vector, and the idea is to discover if it is possible to separate such instances with a (p - 1)-dimensional hyperplane [WIK]. In general there are many hyperplanes that can separate the data, therefore it is necessary to have criteria for selecting the best. In SVM the best hyperplane is chosen to get the largest distance, margin, between the closest point from either class. Hence the distance from the hyperplane and the closest instance in each class must be maximized and the hyperplane takes the name of *maximum margin hyperplane.* In Figure 2.8 three different hyperplanes separate the two class, but the red one is the one that maximizes the distance between the hyperplane and the closest point in each class. This hyperplane has support vector that coincide with the data point that are at the margin distance: none of the points far from the class boundary play a major role [Bon08].

Let us define a hyperplane by:

$$x : f(x) = x^T \beta + \beta_0 = 0$$

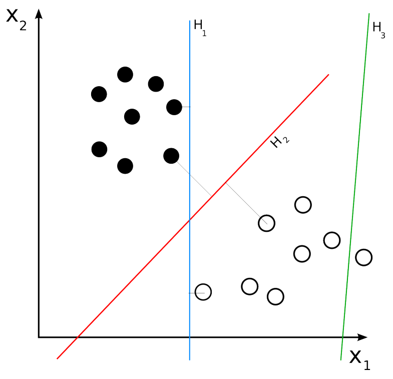


Figure 2.8: Separating Hyperplanes

where β is a unit vector: $\|\beta\| = 1$. If the classes are linearly separable it is possible to find the hyperplane that creates the biggest margin between the training data for each class. The computation of the optimal hyperplane is an optimization problem

$$\max_{\beta_0,\beta,\|\beta\|=1} M$$

$$under\ constraint\ y_i(x_i^T\beta + \beta_0) \geq M$$

where the constraint makes sure that the distance between all points and the optimal hyperplane given by β and β_0 is at least M.

The previous problem can be turned into a minimization problem:

$$\min_{\beta_0,\beta} \|\beta\| = 1$$

$$under\ constraint\ y_i(x_i^T\beta + \beta_0) \geq 1$$

where $M = \frac{1}{\|\beta\|}$.

Most of the time the data are not linearly separable so is not possible to use a linear classifier. To deal with this issue a soft margin approach can be used to relax the hypothesis of two well separated classes, allowing some points to be mislabeled. In this case the goal is still to maximize M, but allow for some points to be on the wrong side of the margin. Let's define the slack variables $\xi = (\xi_1, \xi_2, ..., \xi_N)$. The constraint can be then modified in the following way:

$$y_i(x_i^T\beta + \beta_0) \geq M(1 - \xi_i)$$

$\forall i$, $\xi_i > 0$, $\sum_{i=1}^{N} \xi_i \leq C$ where C is a constant. The value ξ_i is a measure of how much a prediction can fall in the wrong side of its margin, and $\sum_{i=1}^{N} \xi_i$ is a measure of the total cost of misclassification. With this constraint the minimization problem becomes:

$$\min_{\beta_0,\beta} \|\beta\| = 1$$

$$under\ constraint\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \quad \forall i$$

Another approach to deal with non-linearly separable classes is the use of a non-linear kernel function to linearize the problem and transform the maximum-margin into a new feature space with higher dimension that makes the problem linearly separable. The linear boundary in the higher dimension space can in general separate the data better and correspond to non linear boundary in the original space [HTF04].
The mapping performed by a SVM is designed to keep the computation of the dot products not too difficult in terms of original variables. The optimization problem here only involves the input features via inner products. The transformed feature vectors $h(x_i)$ makes the solution function $f(x)$ the following:

$$f(x) = h(x)^T\beta + \beta_0$$

$$f(x) = \sum_{i=1}^{N} \alpha_1 y_1 \langle h(x), h(x') \rangle + \beta_0$$

where $0 < \alpha_i < C$ and it is not necessary to know the transformation $h(x)$, but only the inner products given by the kernel function:

$$k(x, x') = \langle h(x), h(x') \rangle$$

Common kernel function are polynomial, radial basis and neural network (sigmoid).

$$dth - Degree\ polynomial: \ K(x, x') = (1 + \langle x, x' \rangle)d$$

$$Radial\ basis: \ K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

$$Neural\ network: K(x, x') = \tanh(k_1 \langle x, x' \rangle + k_2)$$

In Figure 2.9 there is an example of how a kernel function linearizes the problem.

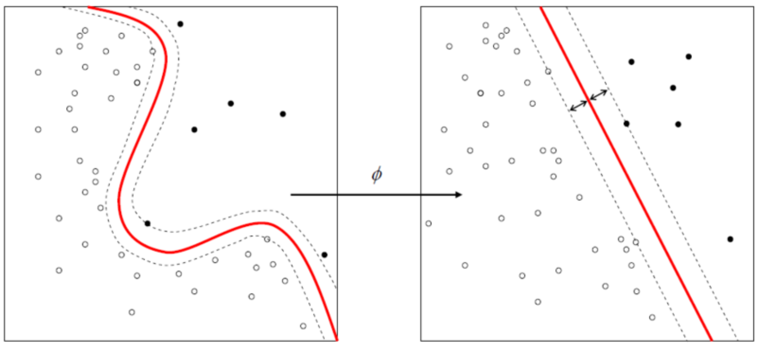


Figure 2.9: Example of a kernel function application [HTF04]

2.3.7 Linear discriminant Analysis (LDA)

Linear Discriminant analysis is a classification method that computes a linear combination of the training examples to extract a rule that is able to classify a new instance. Linear model and Analysis of Variance as well attempt to describe the input-output relationship as a linear combination of the input [MW92], however in LDA the output variable is not a numerical value, but a factor taking different levels. This algorithm is also known under the name of *Fisher's linear discriminant* but Fisher's original article describe a slightly different discriminant [Fis36].
Let us consider a two class problem. LDA makes the homoscedastic assumption that class covariances are equals and that conditional probability of the input variables X given the class Y

$$P(x|y=1), P(x|y=0)$$

are normally distributed with mean and covariance:

$$(\mu_1, \Sigma_{y=1}), (\mu_0, \Sigma_{y=0})$$

In this setting the Bayes optimal solution is given by the log-likelihoods. If it is below a certain value T the instance is predicted to belong to class two:

$$(x-\mu_0)^T \Sigma_{y=0}^{-1} (x-\mu_0) \ + \ \ln|\Sigma_{y=0}| \ - \ (x-\mu_1)^T \Sigma_{y=1}^{-1} (x-\mu_1) \ - \ \ln|\Sigma_{y=1}| \ < \ T$$

Under homoscedastic assumption and full rank hypothesis of the covariance this equation can be simplified to

$$w \cdot x < c$$

where

$$w = \Sigma^{-1}(\mu_1 - \mu_0)$$

Hence an instance belongs to a certain class y when the projection of x into the new direction w is within the class. With LDA the goal is to find a linear combination that maximizes the ratio of between-class variance to within-class variance.

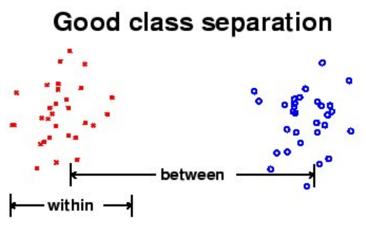


Figure 2.10: LDA separation

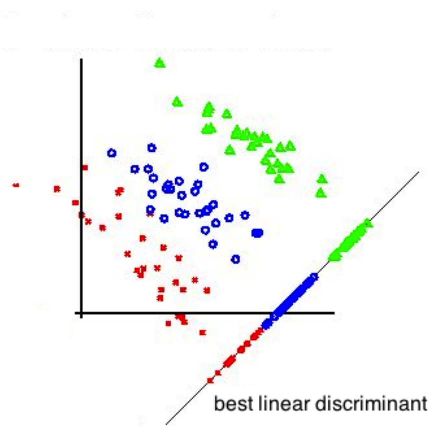


Figure 2.11: Best linear discriminant

The LDA determines a new axis so that when the instances are projected into this new direction, the differences between the groups are maximized.

LDA can be extended for non-linear classification problem via the kernel trick that is used in SVM as well. When the assumption of normally distributed input variables does not hold, other techniques such as Logistic regression may be preferable to LDA [WIK].

2.4 Regression

In a regression problem, the aim is to estimate the value of continuous output variable Y given the distribution of X independent input variables.

$$F(Y = y|X = x)$$

This dependency can be estimated using the expected value of the conditional distribution of the dependent variable Y given X. This dependency is described by a function called regression function.

$$E[Y|X]$$

In regression techniques, the most widely used loss function is the quadratic loss function such as *mean squared error* (MSE).

$$\text{MSE} = E[(\hat{Y} - Y)^2]$$

where $\hat{Y}$ is the value predicted by the regression model.

2.4.1 Linear model

A linear model is a standard approach to perform regression on data. In linear regression, the regression function is linear in its β parameters and is of the form:

$$f(X) = \beta_0 + \sum_j X_j \cdot \beta_j$$

so that the stochastic dependency between input and output is given by

$$Y = \beta_0 + \sum_j X_j \cdot \beta_j + \xi$$

where ξ is the prediction error or noise of the model. The parameters are usually estimated using least squares method that minimize the residual sum of squarers (RSS):

$$RSS(\beta) = \sum_i (y_i - f(x_i))^2$$

The least squared estimations are:

$$\hat{\beta} = (X'X)^{-1}X'y$$

For different estimations of the parameters β, the response value assumes different fitted values $\hat{y}$. The error as well becomes a stochastic variable that depends on the model parameters.

$$\hat{\xi}_i = y_i - \hat{y}_i = y_i - (\hat{\beta}_0 + \sum_j x_i j \cdot \hat{\beta}_j)$$

Using the Gauss Markov theorem it is possible to show that the least squared estimator are the one with smallest mean squared error within the class of linear and unbiased estimator [Ait35].

The Linear model has some underlying assumption:

- The error is a zero mean random variable conditionally to the independent variables.
- There is no multicollinearity, the predictors are linearly independent.
- The errors are not correlated.
- Homoscedasticity: constant variance of the errors.

2.5 Unsupervised learning

In unsupervised learning methods, there is no notion of class or value to predict. Instead, these methods are used to identify key features of the data and to explain its structure by grouping instances more alike or by transforming the data in smaller dimensions.

Unsupervised learning includes techniques such as clustering and dimensionality reduction via Principal Components.

Clustering is used to group instances that seem to share some similarity. The goal here is to find these clusters and to assigns the instances to them. The success of clustering is often measured subjectively in terms of how useful the result appears to be to a human user.

In the case of dimensionality reduction, the goal is to reduce the complexity of a problem and to view the data under few dimensions. These new dimension usually are able to discriminate better the data structure.

2.5.1 K-means

K-means is a standard clustering method in data mining and machine learning. The goal of this unsupervised technique is to divide the set of observations into k clusters by assigning the observations to the cluster with the closest center. For each cluster a center is found by minimizing a dissimilarity measure (distance) between each observations. Let us assume there are k groups S_k. When the euclidian distance is used as a dissimilarity measure, the goal is to partition the observation minimizing the following expression:

$$argmin \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

where μ_i is the mean of cluster i.

At first the centroids (or means) of k groups are selected at random from the training data, and the following two step are repeated:

1. Merge the observation to the cluster with the nearest mean.
2. Update the mean of the cluster and make it to be the centroid of the cluster.

Steps 1 and 2 are repeated until the assignment of an instance to a cluster does not change. The algorithm does not guarantee that running it twice will return the same cluster, because the results depend upon the initial centroids. The algorithm partitions the data to have k disjoint groups of instances. The method is really sensitive to the number of clusters used to perform the partition. This number has to be decided a priori so the number of clusters has to be known before the algorithm is applied. When the number of clusters is not know one possible approach is to try different numbers of clusters and select the number which minimizes the total squared distance of all points to their cluster center [WF05].

In clustering there is no measure of performance as we have in supervised learning. This explain the difficulty to asses the goodness of the model from the output of the algorithms. This makes effectiveness a matter of opinion and cannot be verify somehow.

Another distance measure used in clustering is the Mahalanobis distance introduced by Mahalanobis in 1936 [Mah36]. This measure takes into consideration the correlations between the variables, while the Euclidean distance does not. Another nice property of the Mahalanobis distance is that it does not change with the scale of the instances. The Mahalanobis distance between two points p and q having the same distribution and covariance matrix S is:

$$d(q,p) = \sqrt{(q-p)^T S^{-1}(q-p)}$$

The Euclidian distance can be seen as a special case of the Mahalanobis distance. If the covariance matrix S is equal to the identity matrix, the Mahalanobis distance becomes the Euclidian distance.

2.5.2 Principal Components Analysis (PCA)

Principal Components Analysis is an orthogonal transformation of the variables whose goal is to extract new uncorrelated variables called principal components [WIK]. The transformation is done in a way to have the first principal component to explain as much of the variability of the data as possible and to get each component uncorrelated with each other. The transformation projects the data into a new coordinate system where the first coordinate (principal component) explains most of the variance of the data, in other words PCA rotates the data such that the maximum variability is visible. In practice the goal of principal component analysis are:

- Discover or reduce the dimensionality of the data set.
- Identify new meaningful underlying variables.

Usually, the PCA is computed on the Covariance or Correlation matrix that can be derived by scaling the data. Then the eigenvalues and the eigenvectors are computed on the covariance matrix. The principal components are the eigenvectors of this matrix. The value of each eigenvalue is the amount of “variance” explained by each eigenvector.

The transformation performed by the PCA projects the data into a subspace of smaller dimension q. This subspace is defined by q vectors w_k called Principal Components and the projections of the input data x into this vectors are called *principal components scores.*

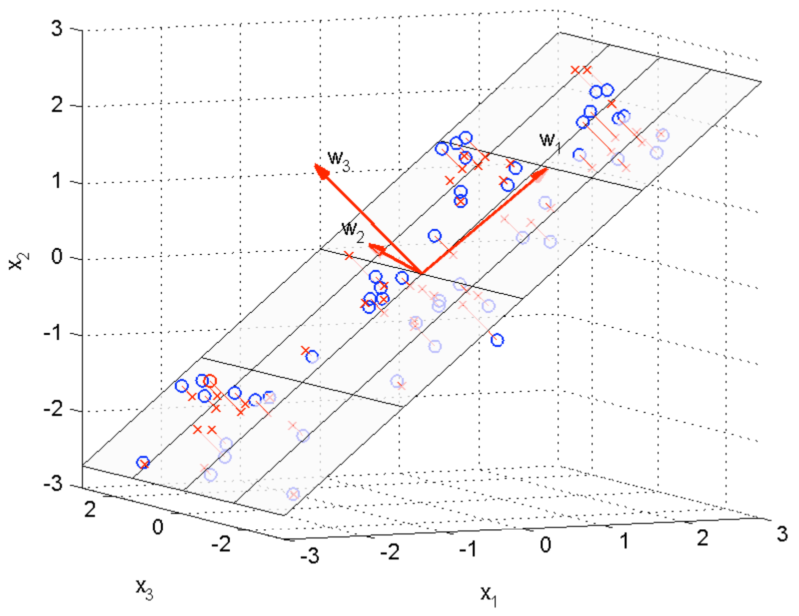


Figure 2.12: Illustration of the transformation obtained by principal component analysis. Circles denote the original observations while crosses denote their approximations obtained by projecting the original data on the two-dimensional subspace (w_1, w_2) spanned by the two first principal components [LB09].

The Principal Components are extracted to minimize the error given by the projections in the reduced space and it means minimizing the following optimization function:

$$J_q(w_k) = \frac{1}{N}\sum_{i=1}^{N}\left\|x_i - \sum_{k=1}^{q} w_k w_k^T x_i\right\|^2$$

where N is the number of examples, the observations x_i are centered and the q vectors w_k have unit norm. This equation represents the mean squared error between the observation x_i and their projections $\sum_{k=1}^{q} w_k w_k^T x_i$ on the Principal Components [LB09].

The vectors that minimizes $J_q(w_k)$ are calculated by computing the derivative of $J_q(w_k)$ with respect to w_k and setting it to zero. If the sample covariance matrix of the observation x_i is denoted by $\Sigma = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T$ then:

$$\Sigma w_k = \lambda_k w_k$$

where the Principal Components are the first q eigenvectors of the covariance matrix Σ. In the previous formula, λ_k is the eigenvalue of the eigenvector w_k and it is proportional to the amount of variance of the projected observations in the corresponding eigenvector. The sum of the eigenvalues is equal to the total variance of the original set of observations.

This makes possible the definition of the retained variance as the variance conserved by the first q principal components. To compute the retained variance, the eigenvectors are first ordered by decreasing order of the corresponding eigenvalues so that the amount P of retained variance is:

$$P(q) = \frac{\sum_{k=1}^{q} \lambda_k}{\sum_{k=1}^{n} \lambda_k}$$

It can be proven that the eigenvectors that minimize the projection errors are also the ones maximizing the retained variance.

Chapter 3

Problem characteristics and prediction strategy

This thesis uses some of the most famous data mining algorithms available in the open source world and implemented inside the software *R* [R D11], which is a free software environment for statistical computing. The software is a result of collaborative efforts with contributions from all over the world and can be easily extended via packages.

3.1 Data understanding

The Data provided by the organizer of the competition consists of a training set where actual claims are present and a test set for which the claims are not present and have to be predicted. The training dataset contains some missing values and has information from 2005-2007, while the test data has information from 2008. Each row contains one year' s worth information for insured vehicles.

The goal of the competition is to improve the ability to use vehicle characteristics to accurately predict insurance claim payments, which is the response variable (dollar amount of claims experienced for that vehicle in that year). Non-vehicle characteristics are included in the set of independent variables and labeled as NV. It is expected that no main contributions to the model will come from these non-vehicle variables, but there may be interesting interactions between these variables and the vehicle variables.
The variables of the dataset are shown in the following table:

Variable	**Type**	**Values**
RowID	Integer	1:13184290
HouseholdID	Integer	1:7542000
Vehicle	Integer	1:29
CalendarYear	Integer	2005:2007
ModelYear	Integer	1981:2007
Cat1	Factor	A,B,C,D,E,F,G,H,I,J
Cat2	Factor	A,B,C
Cat3	Factor	A,B,C,D,E,F,G
Cat4	Factor	A,B,C
Cat5	Factor	A,B,C
Cat6	Factor	A,B,C,D,E,F
Cat7	Factor	A,B,C,D
Cat8	Factor	A,B,C
Cat9	Factor	A,B
Cat10	Factor	A,B,C
Cat11	Factor	A,B,C,D,E,F
Cat12	Factor	A,B,C,D,E,F
OrdCat	Integer	1:7
Var1	Continuous	min:-2.578 mean:-0.010 max:5.143
Var2	Continuous	min:-2.493 mean:-0.065 max:7.829
Var3	Continuous	min:-2.790 mean:-0.025 max:5.563
Var4	Continuous	min:-2.508 mean:-0.054 max:7.589
Var5	Continuous	min:-3.350 mean:0.003 max:4.018
Var6	Continuous	min:-2.377 mean:-0.040 max:4.584
Var7	Continuous	min:-2.778 mean:-0.024 max:4.127
Var8	Continuous	min:-2.163 mean:-0.058 max:47.350
NVCat	Factor	A,B,C,D,E,F,G,H,I,J,K,L,M,N,O
NVVar1	Continuous	min:-0.231 mean:0.014 max:6.627
NVVar2	Continuous	min:-0.266 mean:0.017 max:8.883
NVVar3	Continuous	min:-0.272 mean:0.013 max:8.691
NVVar4	Continuous	min:-0.251 mean:0.018 max:6.389
ClaimAmount	Integer	0:11440

"CalendarYear" denotes the year when the vehicle was insured. "HouseholdID" is a household identification number that allows year-to-year tracking of each household. Since a customer may insure multiple vehicles in one household, there may be multiple vehicles associated with each household identification number. "Vehicle" identifies these vehicles (but the same Vehicle number may not apply to the same vehicle from year to year). The remaining columns contain miscellaneous vehicle characteristics, as well as other characteristics associated with the insurance policy. The data set consists of both categorical and continuous variables. Since the meaning of column Var1, Var2, etc.. is not given it is not clear their importance in the problem. In the dataset 16 out of the all 32 variables are categorical. The instances of the training set having Claim variable greater than zero are only 0,73%, therefore the dataset is highly unbalanced.

In this work the response variable “ClaimAmount” will be referred to Y and the input variables as X. Predicting the claim amount value may be seen as a regression problem because the outcome is a continuous variable. At the same time the problem could be transformed into a classification one by transforming the claim amount variable into a binary variable taking value one ($Y = 1$) if its value is greater than zero and zero otherwise ($Y = 0$).

3.2 The Gini Index (GI)

In this competition the metric to assess the results is the *normalized Gini coefficient* (named for the similar Gini coefficient/index used in Economics [WIK]). In this metric observations are sorted from “largest prediction” to “smallest prediction” and only the order determined by predictions matters.

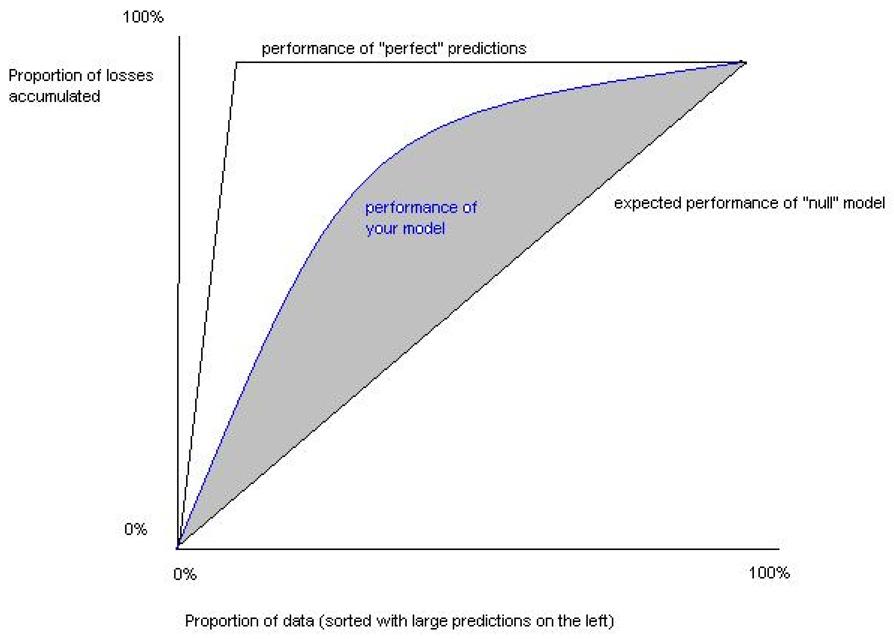

100%
Proportion of losses accumulated
performance of "perfect" predictions
performance of your model
expected performance of "null" model
0%
0%
100%
Proportion of data (sorted with large predictions on the left)

Figure 3.1: Gini Index

The abscissa axis of the chart in Figure 3.1 represents the proportion of data used for prediction where observations are sorted from largest claim to smallest claim. This means that the first 10% of data from the left are instances with the highest claim. Moving to the right, predictions with smaller claims are included. The ordinates axis represents the proportion of accumulate losses. The curve of the predictions tells how much of the actual observed loss is accumulated in the leftmost x% of the data. With no model, it is expected to accumulate 10% of the loss in 10% of the predictions, so a random model achieves a straight line.

The Gini coefficient corresponds to the area between the curve of the predictions and the straight line. Since the maximum achievable area is given by the *perfect* model, the normalized Gini coefficient is obtained by dividing the Gini coefficient of a predicted model by the Gini coefficient of the perfect model. With the normalized Gini index, it is important to find a correct rank of the car with claims, so to get a high score it is not important to get a precise estimate of the Claim Amount, but the correct order of cars having claims greater than zero.

With the normalized Gini metric the score of the null model is unknown. The first thing is to test the Null hypothesis where all the instances have the response variable *Claim Amount* equal to zero. The goal is to compute null model in order to have a internal benchmark. From now on the term *GI* will be used to refer to the normalized Gini index.

3.3 Methodology

The methodology proposed is to first see the problem as a regression task and exploit some dimensionality reduction techniques to reduce the problem size. First, the regression will be applied using a simple linear model and afterwards Principal Components Analysis (PCA) and clustering will be used to reduce the dimension of the dataset. The PCA allows to reduce the dimension using only the first q components that explaine much of the variability of the data. Another way to reduce the dimensionality is by grouping variables using clustering techniques.

In a second stage the problem will be treated using classification techniques and the results will be integrated with the regression. The idea in the classification is to transform the response variable as a binary variable and compute for each instance the probability of having a Claim Amount greater than zero. The probabilities can give a prediction order of the instances with claims. The greater the probability of being classified as one (Claim greater than zero), the bigger the expected original claim amount.

The classification and regression will be integrated in the following way:

1. Classification will be used to predict instances with no claim ($Y = 0$).
2. Regression will be applied on the remaining instances of the dataset that have been predicted as non zero in the classification to improve the order of the instances with claim.

Classification algorithms predict an instance as 1 if the probability of being 1 given the input variables $P(y = 1|x)$ is bigger than the probability of zero $P(y = 0|x)$. Since $P(y = 1|x) + P(y = 0|x) = 1$, the instances are predicted as 1 if $P(y = 1|x) > 0,5$ and predicted as 0 if $P(y = 1|x) < 0,5$.

The smaller $P(y = 1|x)$ the more likely is the instance to have no claim. If instead of using the classification threshold of 0,5 a smaller one is used, then the predictions will show a smaller number of instances with no claims because more will be predicted as 1. In this case the instances predicted as zero will be the ones with smaller $P(y = 1|x)$ that means with higher $P(y = 0|x)$. Hence reducing the threshold leads to a higher number of instances predicted as 0 having actual claim equal to zero (True Negative) and therefore a smaller number of claims predicted as 1 having actual value zero (False Negative).
In the classification task the instances will be predicted using different thresholds T that will return different subsets of the training set with which to apply the regression. The results will be compared to see which threshold can return a higher GI.

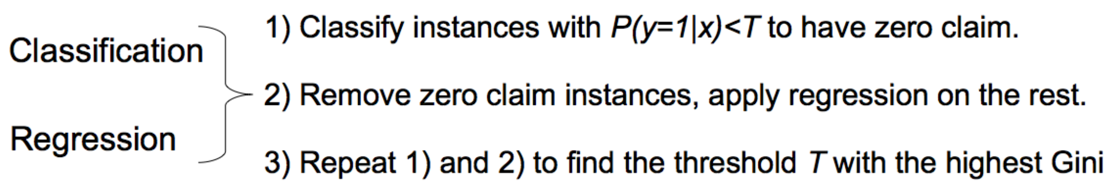

Classification
Regression
1) Classify instances with P(y=1|x)<T to have zero claim.
2) Remove zero claim instances, apply regression on the rest.
3) Repeat 1) and 2) to find the threshold T with the highest Gini

Figure 3.2: Combination of classification and regression.

At the extreme case when the threshold is zero, all the instances will be classified as 1 and the regression will be applied to the whole training set. On the contrary when the threshold is one, all the instances will be classified as 0 as in the null model.

Before combining classification and regression, different classification methods will be tested on the training data to find the best classifier. This will allow the tuning of the algorithms that will be evaluated using the cross validation accuracy and GI. Once the best classification algorithm is found it will be used in combination with the regression to predict the best results. In the case of classification, a balanced training set will be extracted from the original training set to have a dataset with the same proportion of examples with $y = 1$ and $y = 0$ in order to avoid the problem of high imbalanced data [L+10].

The assessment and evaluation of the different strategies use the following criteria. For classification, the confusion matrix is used to assess the accuracy of the algorithms together with the ROC curve. The classification accuracy resulting from the confusion matrix is computed on the training set with a 10 fold cross validation. From the confusion matrix the accuracy is obtained by summing up the number of instances correctly classified divided by the total number of instances, (TP+TN) / (P+N). In the case of a random prediction, it is expected that half of the predictions are correctly classified and half misclassified. This measure will give a first insight on how good a model applies to the data. The model will be then applied to the test set and the results compared using the GI score.

In the case of regression results are compared by looking at the GI first in the training data and then in the test data. The GI from the test set is estimated in the training set with a 10 fold cross validation. However this estimation is not really good in the case of classification. The problem is that the classification models are trained on a balanced dataset and then applied to the test set which is highly unbalanced. Therefore the resulting GI in the training set is really different from the score obtained in the test set used for the competition. In regression the dataset used for training is unbalanced as well so the cross validation GI in the training set is coherent with the score of the test set.

Chapter 4

Experimental Results

4.1 Data preprocessing

The first operation performed on the data consists in dropping columns *RowID* and *HouseholdID* since these variables can not give much information. The variable *CalendarYear* is removed as well since it takes different values in the training set but not in the test set used for the competition. Among all the 32 variables of the dataset 16 are categorical. To be sure that all the algorithms can work using the same data, all the categorical variable are transformed into L_j binary variables where L_j is the number of levels of each one. Having transformed the categorical variables into binary ones, the dataset has 101 variables.

Another problem of the data is the fact that the test set has categorical variables with new levels not present in the training set and not all the levels present in training set are in test set as well. In order to work with the same variables for model creation and prediction, only the intersection between all the possible levels present in the training and test set are taken into consideration.

The first step is the testing of the *Null* model whereby all the instances have the response variable *Claim Amount* equal to zero. GI is estimated using a 10 fold cross validation on the training data. The GI median is 0,0048, see the boxplot in Figure 4.1.

4.2 Regression

Linear regression can be performed in *R* using the classical method *lm()* from the *stat* package [R D11]. The entire training set contains more than 13 millions of examples and this size becomes a problem when the categorical variables have to be transformed into

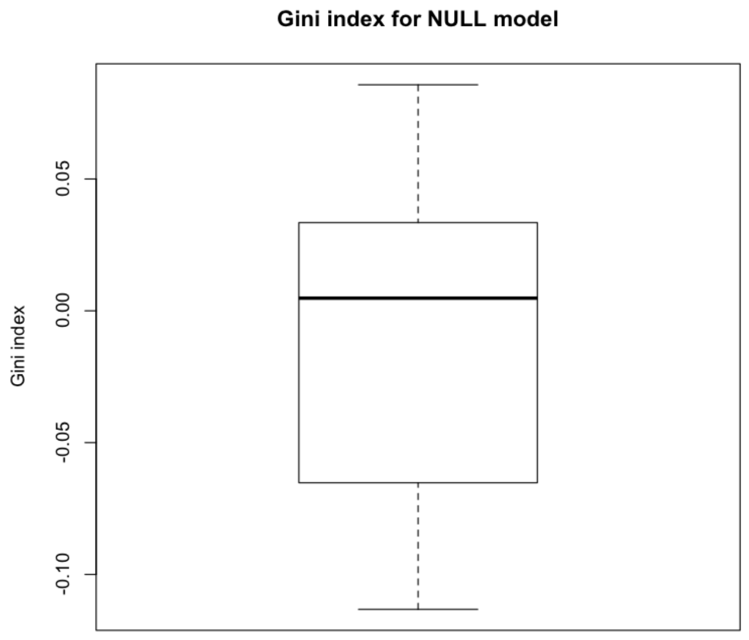


Figure 4.1: Gini index boxplot for Null Model.

binary ones. Using such a big training set pushes the computer to its computational limit.

To speed up the process it is considered to use a subset of the training set. In order to understand to what extent the set can be reduced, the same linear model is applied to training sets of different sizes and for each size a ten fold cross validation is performed to estimate the GI. From Figure 4.2 notices how the variance of the GI decreases but the improvement using a training set larger than 1 million is still not remarkable. The mean of the GI computed on the 10 fold cross validation for 1million examples is 0,0859. On the y-axis there is the predicted Gini index and on the x-axis the different size of the training set used starting from 500000 up to 2 million. In the plot it is shown that the Gini increases with the size of the training data until the size is 1 million (Figure 4.2). Therefore this reduced dataset of 1 million examples can be considered representative of the original training data and it will be used for all the following models. Applying the linear model to this training set, the Gini index in the test set is 0,0776.

As a second step, Principal Component Analysis is applied to reduce the dimensionality of the problem. To determine which number of components provides the higher Gini, the linear model is computed using 1 million of instances and different numbers of principal components: 10, 20, 30, 40 , 50, 60, 70, 80. Figure 4.3 gives the GI means and their confidence intervals for different numbers of components.

The 20 principal components returns the highest GI on the training set with a mean

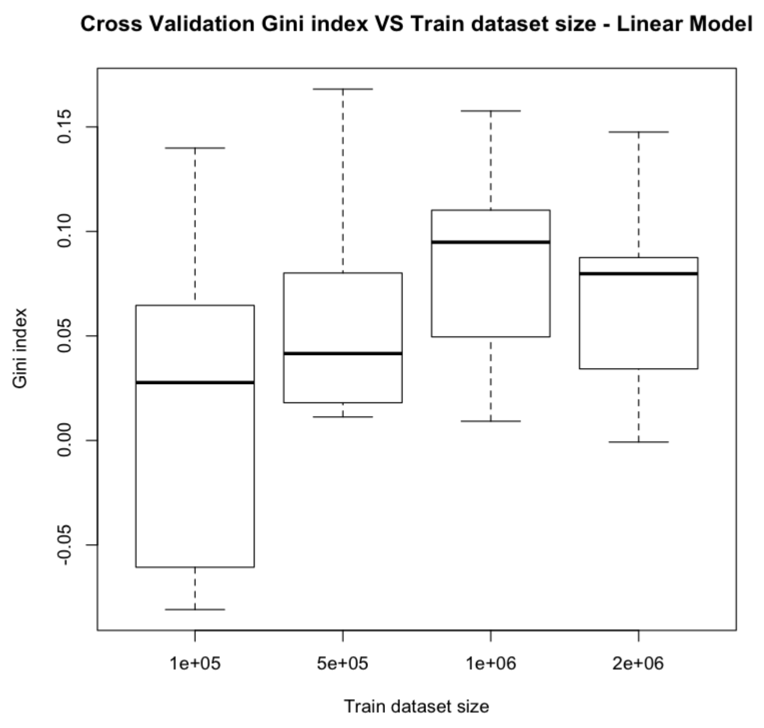


Figure 4.2: Cross Validation Gini Index for different training set size.

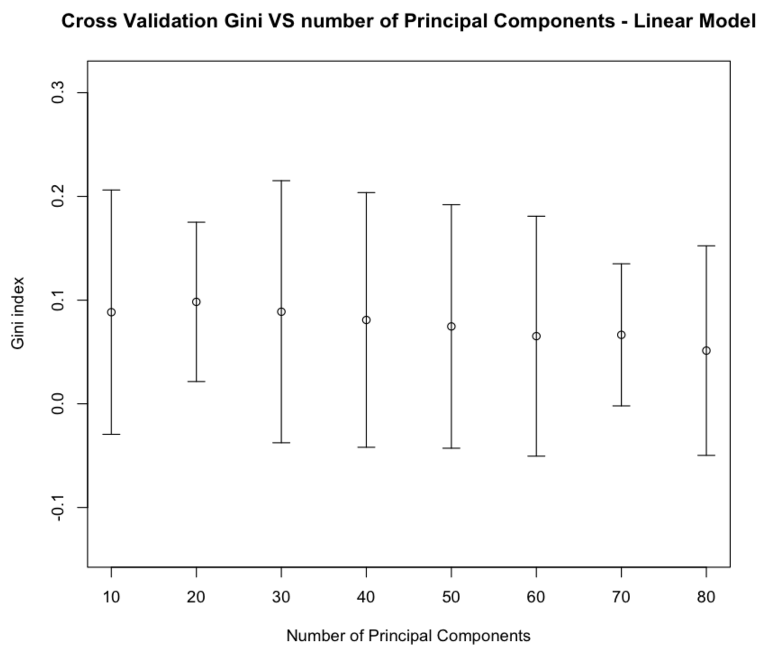


Figure 4.3: Cross Validation Gini Index for different numbers of Principal Components.

of 0,0983. The same model applied on the test set returns a GI of 0,0808. In both training and test data, the results obtained using 20 components is higher than the score reached with simple linear model so the PCA brings some improvements.

A clustering technique is then performed on the 101 variables to understand if it can achieve better results than the one obtained with the PCA. The idea here is to cluster groups of variables and replace the variables that fall into the same cluster with its cluster mean. The K-means algorithms is used for clustering and then the linear model is computed on the resulting clusters. To compare the results with the PCA, the model is computed using 1 million instances for different numbers of clusters: 10, 20, 30, 40, 50, 60, 70, 80.

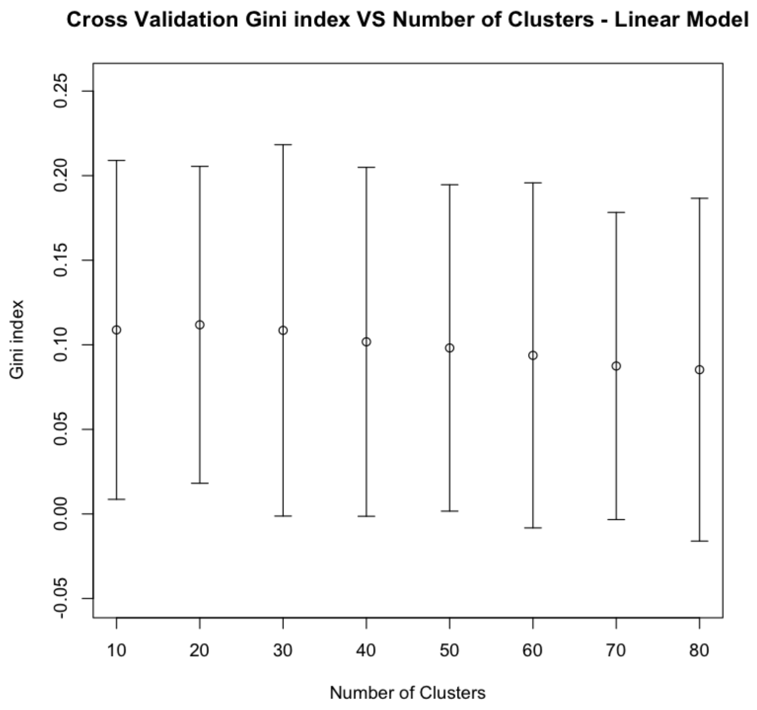


Figure 4.4: Cross Validation Gini Index for different numbers of clusters.

The highest score is achieved by using 20 clusters (GI = 0,1118), but the variance is not much smaller than the other clusters so it is not possible to be too confident with the fact that 20 cluster is the best choice. Moreover the GI on the test using 20 clusters is just 0,0750. K-means can return different results for the same dataset since this method depends on the choice of the first centroids. Even though the clustering is performing better than PCA in the cross validation results, the PCA shows better results in the test set and in this case the GI obtained using Kmeans is below the score of the simple linear model (0,0776).

4.3 Classification

To improve the results of the linear model, different classification methods are applied to predict the instances with Claim amount equal to zero. In a second step a regression is performed on the remaining non-zero instances. The idea we had is that the prediction of the linear model could return a better ordering of the examples with claims. All the examples are classified using a threshold T. The instances having $P(y = 1|x) < T$ are predicted as zero. This approach is repeated for different thresholds from 0 to 1 to find the threshold that returns the highest GI from the combinations of classification and regression.

The train dataset is highly unbalanced and this could have been a problem for some classifiers. Therefore a balanced data set is created from the training set by extracting all the instances having *Claim Amount* greater than zero and an equivalent number of instances with zero claim. Then different classification algorithms are applied to the data.

4.3.1 Decision Tree

The first classifier used is a decision tree, but before applying the combination of tree and the linear model it is necessary to explore the behavior of tree and to understand which parameters better fit the dataset. The decision tree classification in R is implemented by the *Rpart* and the *tree* packages [Rip11]. Trees models work well with categorical variables, but the algorithm is not able to deal with variables having more than 33 levels. To avoid this problem the dataset is preprocessed as said before to get dummy variables out of the original categorical one. The first thing tried, is to apply the *tree* algorithm with the default settings for classification.

The time taken to compute the model and make predictions is recorded to be able to compare the speed of different algorithms. In this algorithm a key figure is the *mindev* term that represents the within-node deviance. At each node the deviance is checked if it is at least this times that of the root node and if it is the case the node is split. In this way the parameter *mindev* defines the size of the tree. If this parameter is set to zero the resulting tree fits the training data perfectly, the greater the *mindev* the smaller the size of tree. To see how this term is affecting the results, several trees are trained with different *mindev* and the results compared. The values of *mindev* used are: 0,00005, 0,0001, 0,0005, 0,001 and 0.005.

The results are compared in terms of classification accuracy (computed with cross validation of the training set), GI on the test set, time to compute the model and time to get the predictions. The best result for the GI on the test set comes from the bigger tree, i.e. the one with the smallest *mindev*. Increasing this value reduces the GI. The maximum GI is 0,0895.

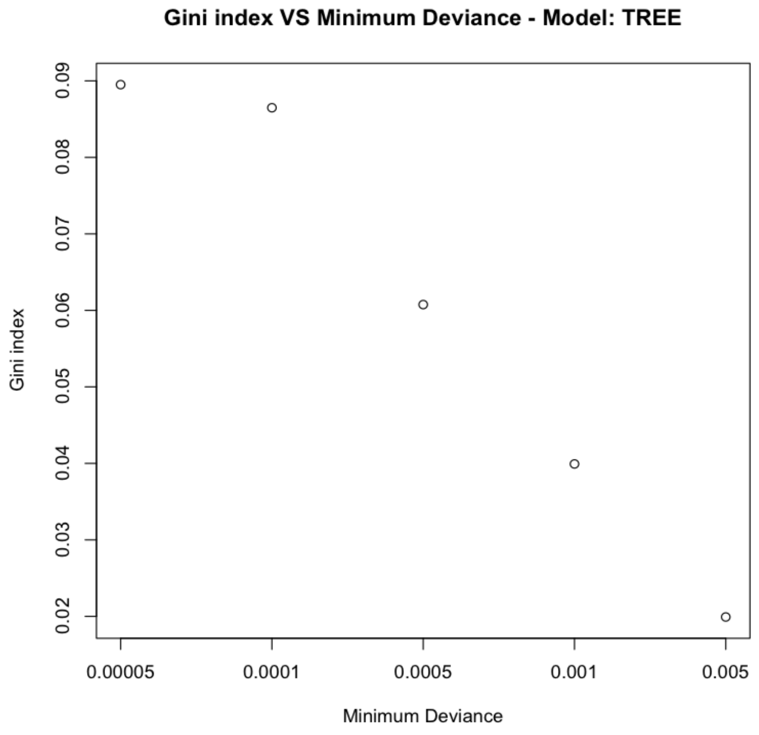


Figure 4.5: Cross Validation Gini Index for different number of minimum within node deviance.

The classification accuracy computed with the 10 fold cross validation is the highest when the minimum deviation is the smallest, with a maximum accuracy of 0,5657 obtained with *mindev*=0,0001.

This algorithm is quite fast to learn and predict. Increasing the size of the tree increases the time to compute the model and its predictions but the maximum time taken is below the minute. In Figure 4.7 the time is expressed in seconds and it is possible to see how it increases exponentially the smaller the *mindev*.

We now combine the decision tree with linear model using different thresholds. As said before, the instances predicted as zero with the decision tree are removed and then a linear model is applied to the remaining instances. The parameter chosen for the tree is *mindev=0,0001* since is the one with the highest GI and highest accuracy. For each threshold, a 10 fold cross validation on the training data is done to estimate the accuracy.

The plot of Figure 4.9 shows that a thresholds of 0,3 returns the highest GI (GI=0,0874). The threshold 0 corresponds to the linear model without classification

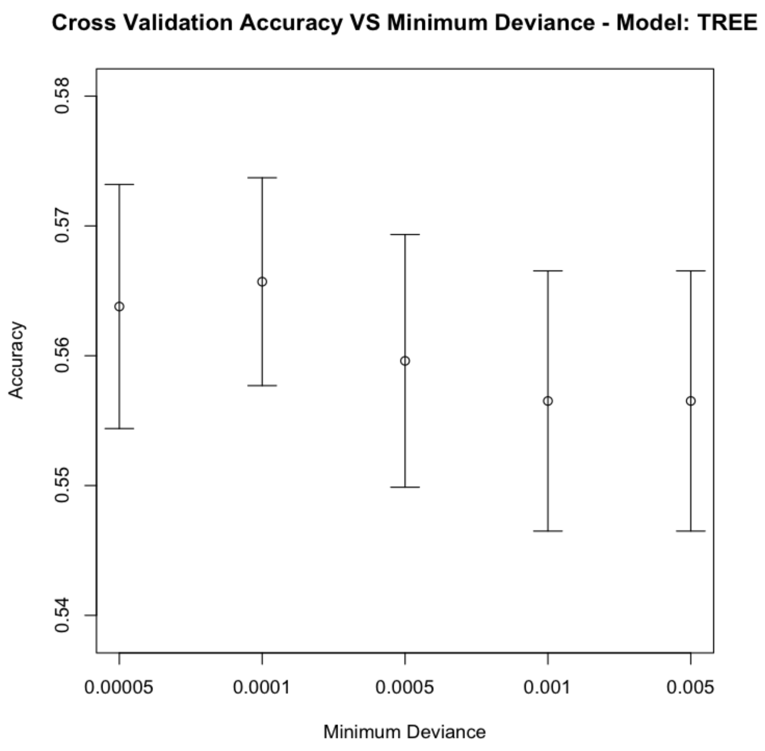


Figure 4.6: Cross Validation Accuracy for different number of minimum within node deviance.

since in this case the decision tree classifies all the instances as 1 leaving the entire training set to be predicted using regression. On the opposite side with a thresholds of 1 the model corresponds to the "Null" model since all the instances are predicted as zero from the tree leaving no subset to which regression is applied. The same combination of classification and regression is applied to the test set and the GI for threshold 0,3 is 0,07521.

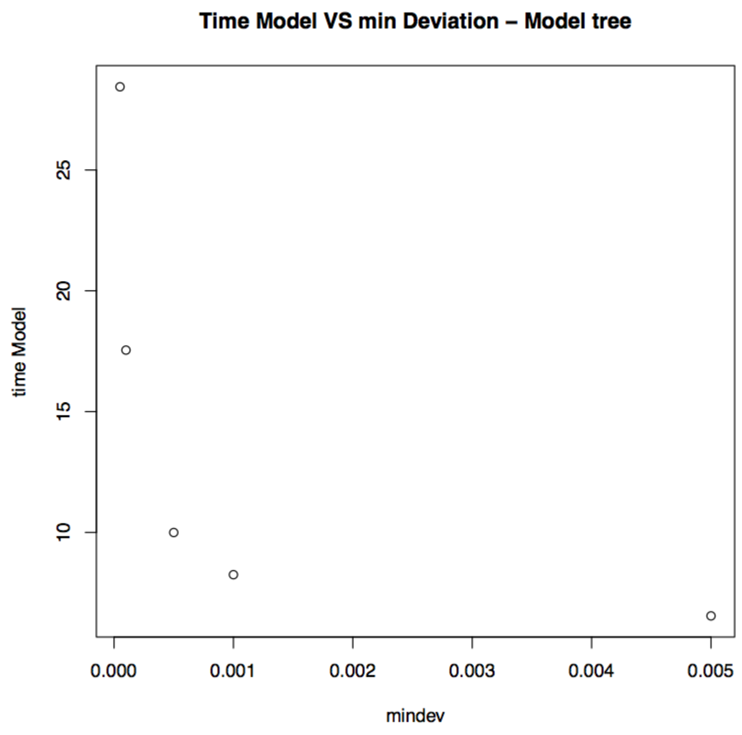


Figure 4.7: Seconds taken to create the model for different number of *mindev*.

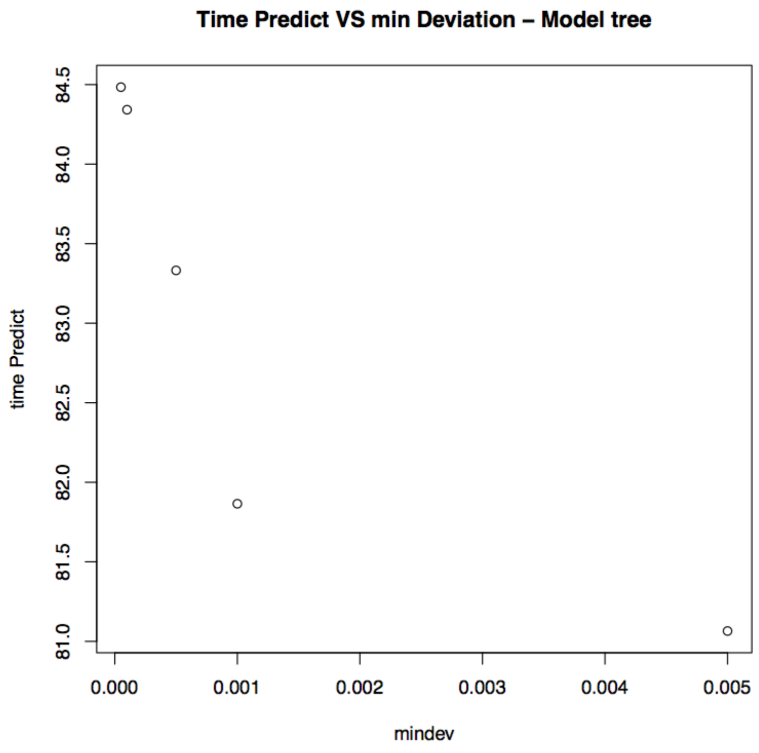


Figure 4.8: Seconds taken to predict for different number of *mindev*.

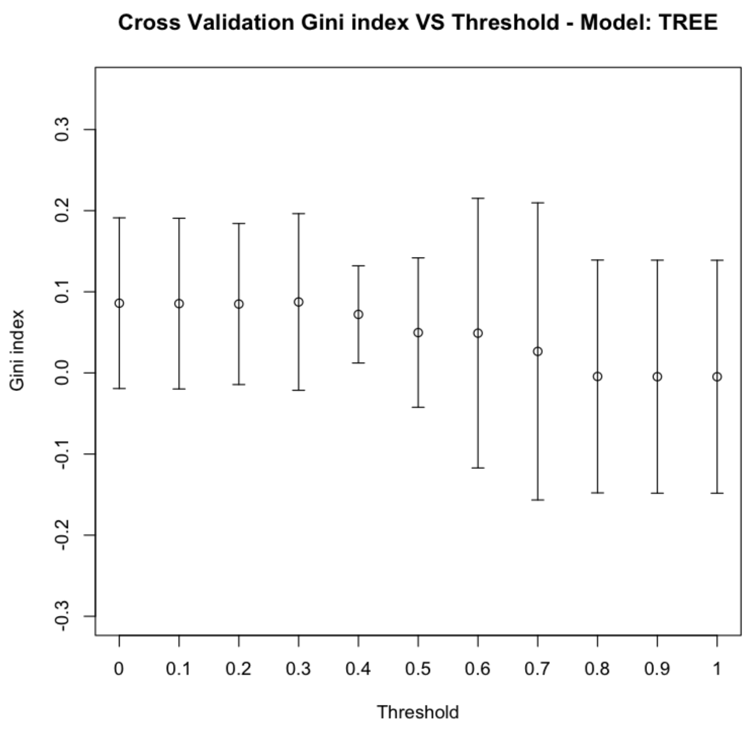


Figure 4.9: Cross Validation Gini Index for different thresholds with decision tree algorithm.

4.3.2 Random Forest

In the software *R*, random forest algorithm can be used via the *RandomForest* package [LW02]. In this implementation a key parameter is the number of trees to use in the forest.

Since random forest has problem with categorical variables having many levels, we decided to transform them into binary variables. The Out of Bag computation inside the algorithm could have avoided the use of a cross validation. However a 10 fold cross validation is computed to get an estimate of the accuracy in the training set as it has been done with other classifiers. Different forests are built to see how the number of trees could affect the results. The number of trees used ranges from 50 to 300 trees.

A higher accuracy on the training data is expected to come from the model with the highest number of trees. As expected, the classification accuracy computed via the cross validation increases with the number of trees. The plot however does not show a great variation between forests with different numbers of trees.

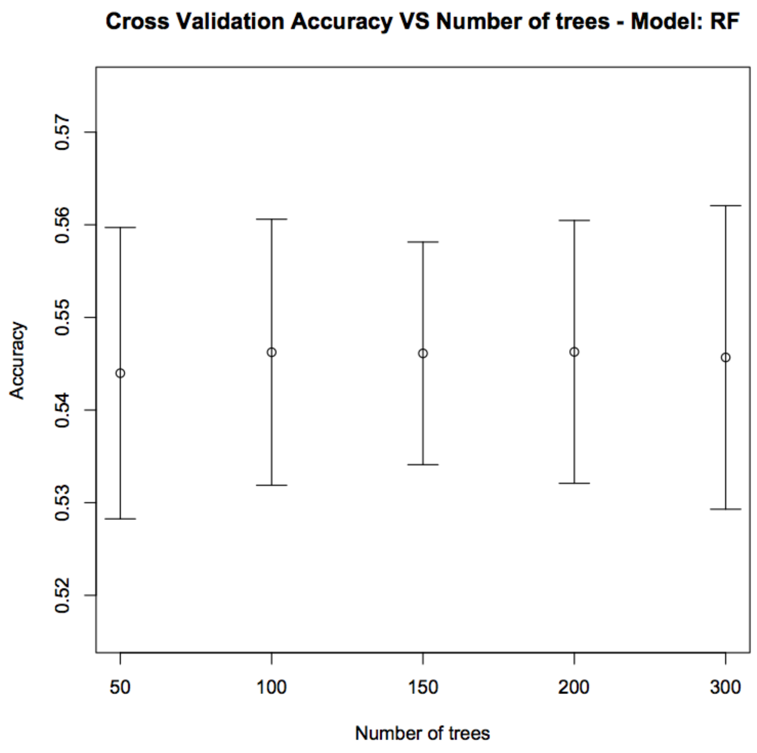


Figure 4.10: Cross Validation Accuracy for different number of tree in the forest.

The application of the model to the test set shows that the forest with 300 trees is the best model for this data with GI 0,062 (Figure 4.11).

What is nice about this algorithm is that the time taken for building the classifier appears to increase linearly with respect to the number of trees as it is possible to see in

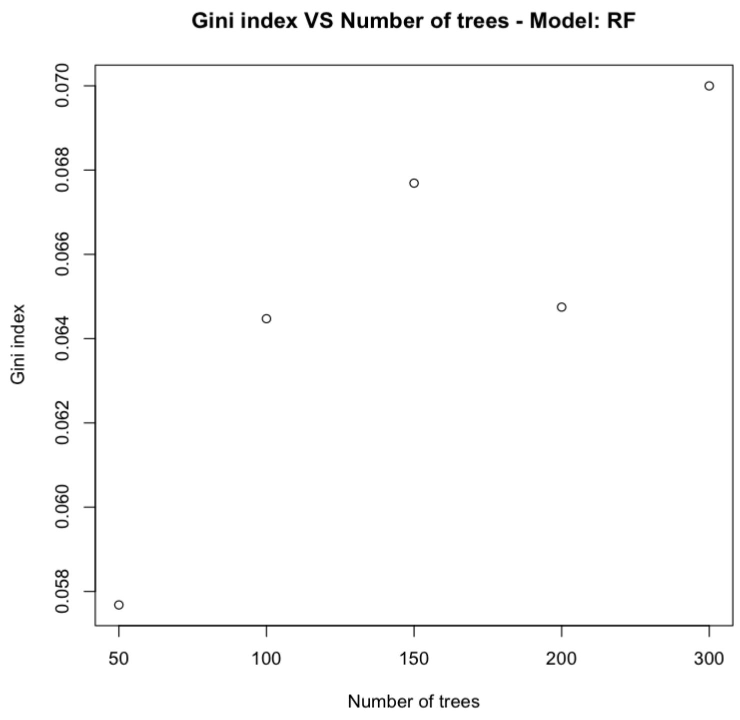


Figure 4.11: Gini Index for different number of trees in the forest.

Figure 4.12. The biggest forest with 300 trees takes about 14 min to be built.

As it has been done for decision tree, random forest is combined with the linear model. Here the forest used has 300 trees since the random forest model has the best GI for this size.

In Figure 4.13 the highest cross validation GI 0,0878 is obtained using threshold 0,1. Such a small threshold means that the forest is used to classify only a small part of the training set. With this model the GI on the test set is 0,0781. The combination of classification and regression improves the results reached with just the random forest from 0,062 to 0,078.

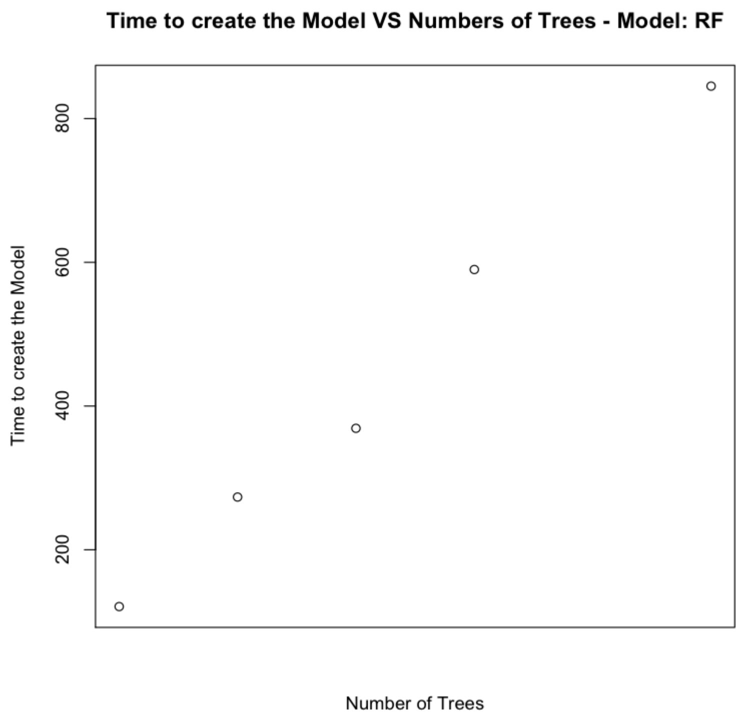


Figure 4.12: Time to create the model for different numbers of trees in the forest.

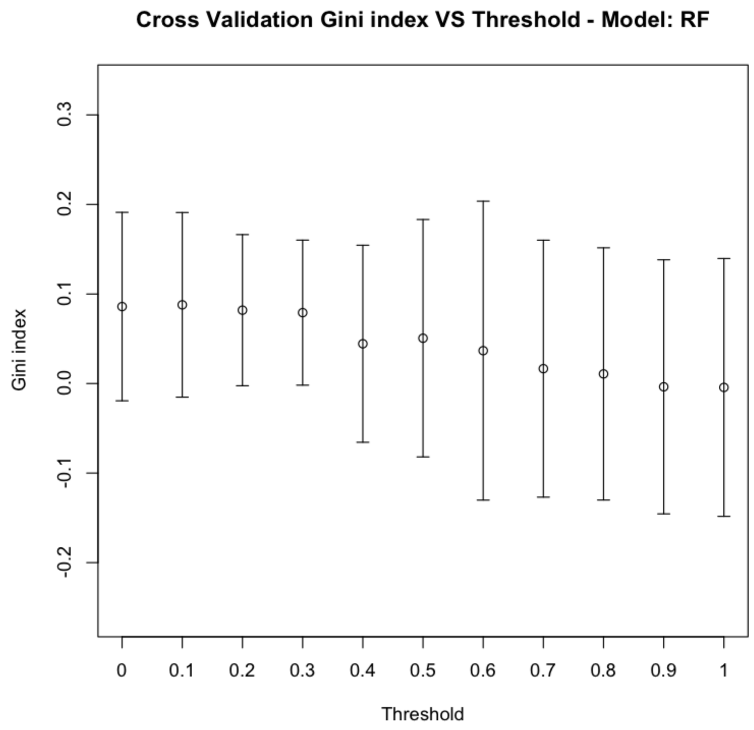


Figure 4.13: Cross Validation Gini Index for different thresholds.

4.3.3 Naïve Bayes

The Naïve Bayes algorithm is implemented using the *e1071* package [DHL+11]. Since no a priori probabilities are specified in the problem, these are estimated by the occurrences of each class in the data. The model is applied with the standard settings.

The mean of the cross-validation accuracy is 0,51 with a standard deviation of 0,0042. As it is possible to see from the boxplot (Figure 4.14) the highest accuracy obtained is 0,517. The result does not show a great performance since the accuracy is slightly above 0,50, that is the accuracy that can be reached by random predictions.

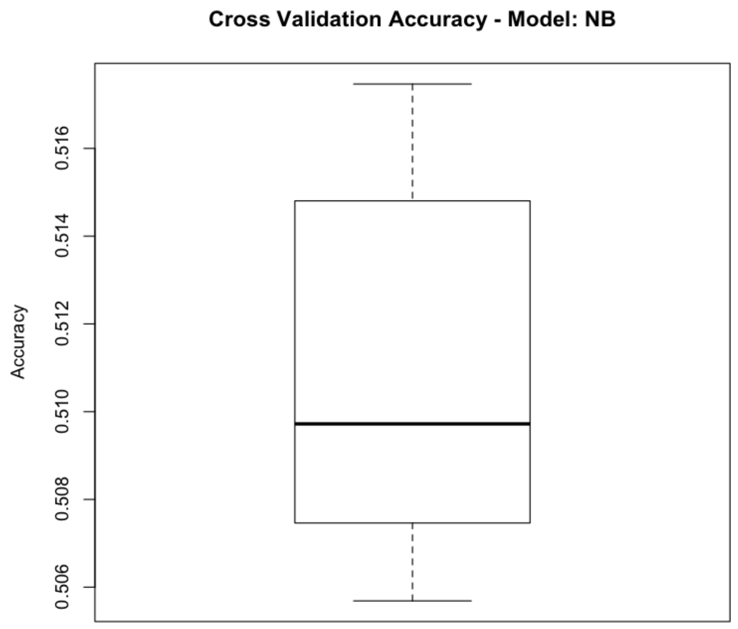


Figure 4.14: Cross Validation boxplot Accuracy for model Naïve Bayes.

To confirm the bad fit of the model, the GI index on the test is -0,02 suggesting that the Naïve Bayes is not able to predict well this type of data. For this reason the combination of the model with the regression is avoided, since the classification could not bring any improvements to the linear model.

4.3.4 K Nearest Neighbors (KNN)

The K Nearest Neighbors algorithm is made available in *R* by the *knn* function. This method makes possible to control the number of K nearest neighbors to use in the model. Another implementation of KNN is available through the *lazy* package [BB03]. This later algorithm implements a leave one out cross validation to select the best number of nearest neighbors that are selected according to either the Manhattan or the Euclidean distance.

With the first algorithms it is possible to set the number of neighbors manually and to have more control of the results. To understand which number of neighbors is best, a cross-fold validation is used to estimate the accuracy on the training data and then the same model applied in the test set. The algorithm is tested using a number of neighbors varying from 1 to 5. The classification accuracy for 5 and 3 neighbors is respectively 0,5332 and 0,5306.

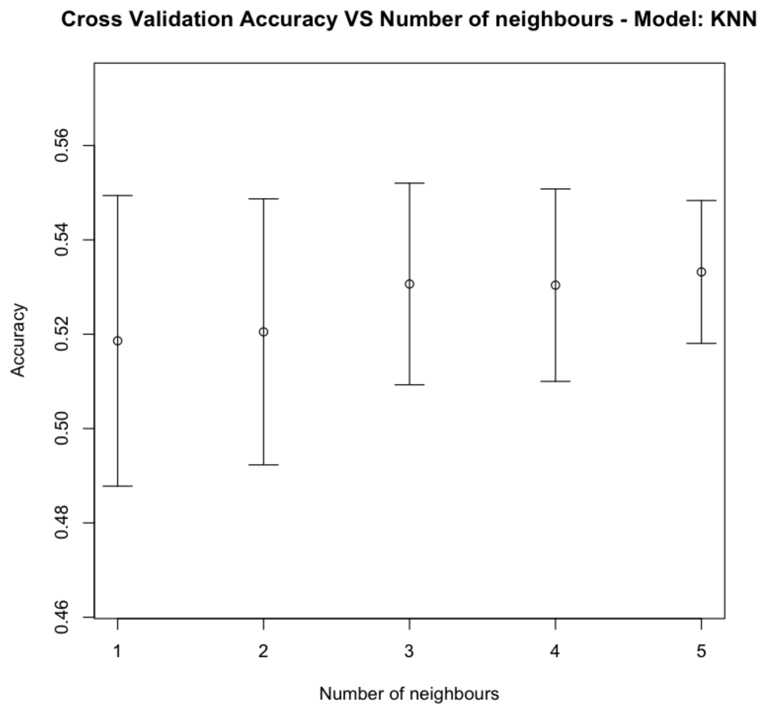


Figure 4.15: Cross validation accuracy mean for different numbers of neighbors.

The highest score on the test set is achieved for 3 neighbors with GI 0,0279.

Here the GI is not really high compared to the results obtained with others classifiers such as decision tree and random forest. Therefore the combination of KNN and linear model cannot improve the results obtained with the previous combinations of classification and regression. Notice that the linear model used for regression is always the same so when it is combined with a classifier the improvement can come only the classification part.

Since all the computation is done in the prediction step it makes sense to compare the time taken by the predictions rather than the time to build the model. The smallest time taken to predict 1 million examples is about 4 hours and half and corresponds to the model with 3 neighbors. The model with 5 neighbors is the slowest with almost 5 hours.

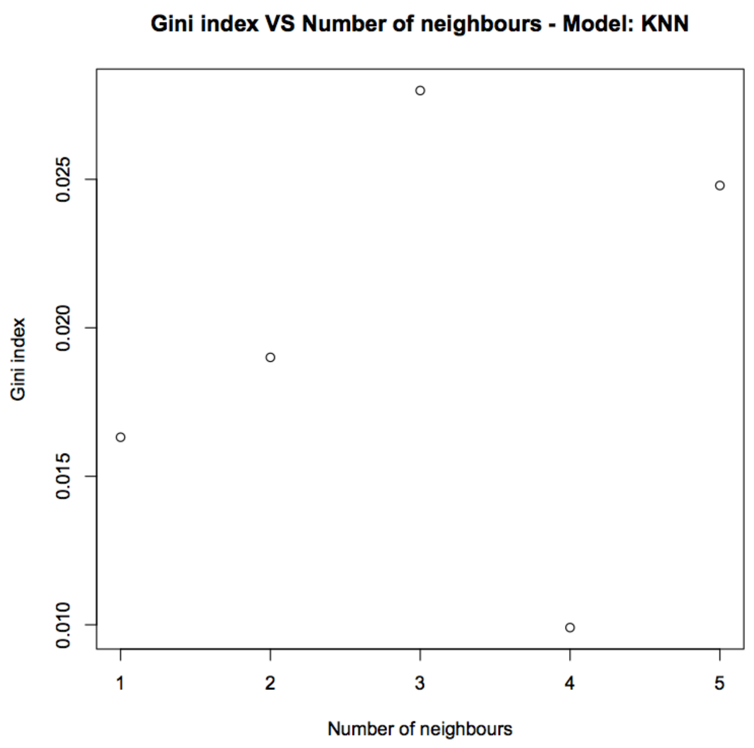


Figure 4.16: Gini Index on test set for different numbers of neighbors.

The *lazy* algorithm is applied as well to see if it can actually improve the results obtained by the *knn*. This algorithm selects automatically which is the best number of neighbors to keep into consideration at each time. The problem here is the computational time. To run one round of the cross validation the algorithm takes on average 49 hours (2 days). This is because it computes the k nearest neighbors for a large range of different *k*s. It is not computationally possible to build the model on the all data, the only metric that is possible to compute is the accuracy from the 10 fold cross validation. The accuracy boxplot shows a median of 0,527 but the mean is only 0,422. The classification accuracy in this case is smaller than the one obtained with the *knn* algorithm (0,5332).

4.3.5 Neural Networks (NN)

A well known implementation of the neural network algorithm in *R* comes with the *nnet* package [VR02a]. A NN model is not able to handle categorical variables, so once again the binary transformation of the categorical variables is needed. The *nnet* default settings use a single hidden layer, and the number of neurons can be specified using the parameter *size*. The accuracy estimated via cross validation is used here to test different models and to understand the number of neurons that best fit the data. Unfortunately the results do not change when using different numbers of neurons. The accuracy on the training set is

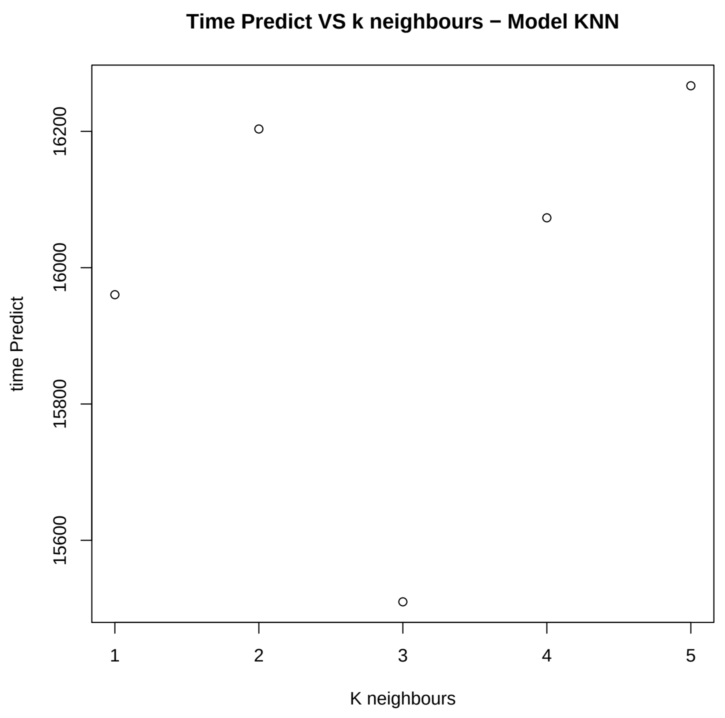


Figure 4.17: Time to predict for different numbers of K neighbors in *knn* algorithm. Time is expressed in seconds.

0,5006, just above the 0,5 threshold of a random prediction.

The GI on the test set is -0,0164 confirming a bad performance of the algorithm to the data. With these bad results, the algorithm is not taken into consideration for further investigation.

Looking at the time taken to create the networks (Figure 4.19) we observe that increasing the number of neurons, increases the computing time exponentially.

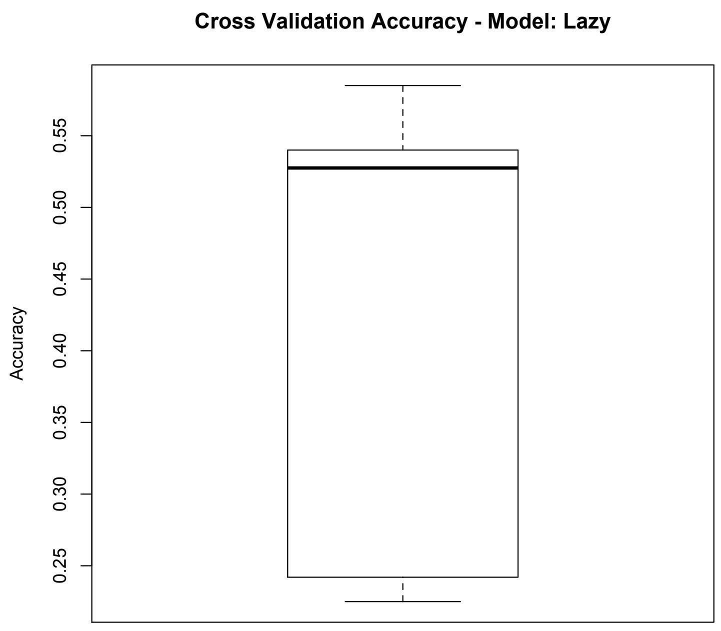


Figure 4.18: Cross validation accuracy for model *lazy*.

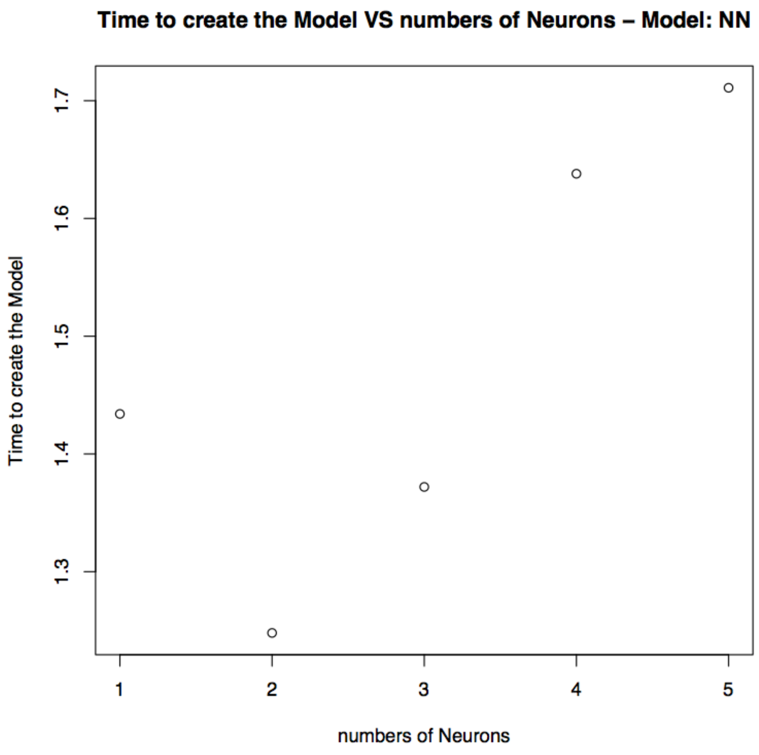


Figure 4.19: Time taken in seconds to build a networks with different numbers of neurons.

4.3.6 Support Vector Machine (SVM)

In *R* the Support Vector Machine algorithm is implemented in the package *e1071* [DHL+11].

The model allows the setting of different kernel functions: radial, polynomial, linear and sigmoid. To compare the different kernels a 10 fold cross validation in the training set is used to assess the accuracy and the GI metrics for the test set.

Unfortunately the results shows a bad performance of the model regardless of the type of kernel function used. The cross fold accuracy on the training set is always below 0,50 suggesting a performance worse than a random prediction.

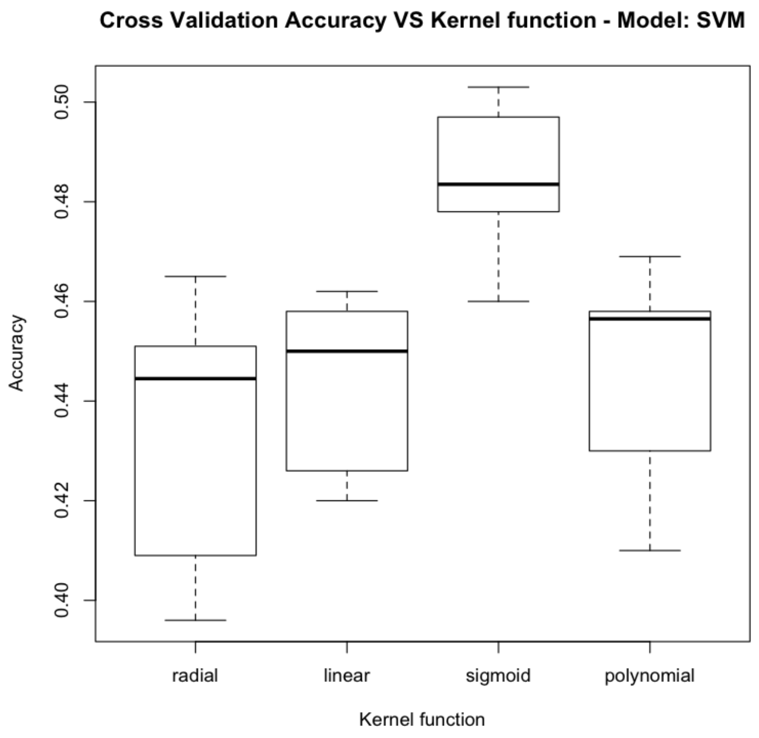


Figure 4.20: Cross validation Accuracy boxplot for different Kernel functions in SVM.

The GI on the test set are all below zero confirming that the model is not working well with the data selected.

The ROC curve in Figure 4.21 is below the diagonal and this is a sign of bad accuracy.

The linear kernel is the slowest to learn from data since it takes 30 minutes where the other kernels take on average 7 minutes (Figure 4.22). Since the results are not great, the SVM is not considered an eligible candidate to be combined with regression.

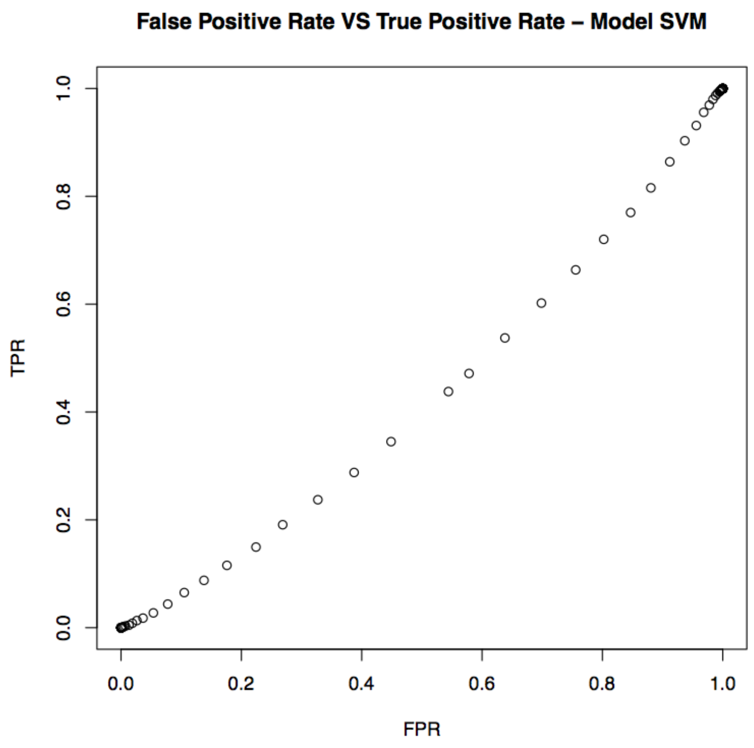


Figure 4.21: ROC curve for SVM.

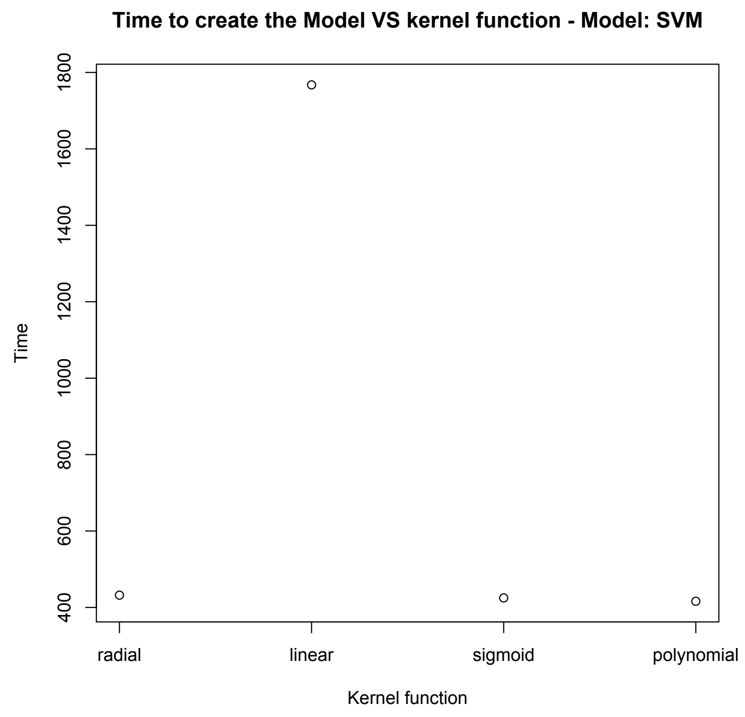


Figure 4.22: Time taken in seconds to build the model with different kernel functions.

4.3.7 Linear Discriminant Analysis (LDA)

In *R*, linear discriminant analysis is implemented by the *lda* function from the *mass* package.

Though the algorithm may seem too simple the results show a relative good performance of the model since the GI on the test set is 0,0789 and the cross validation accuracy in the training set 0,5703. However, as it is possible to see in Figure 4.23, plotting the data in the new direction given by LDA does not bring a great discriminanting power between the two groups of instances with class 0 or 1.

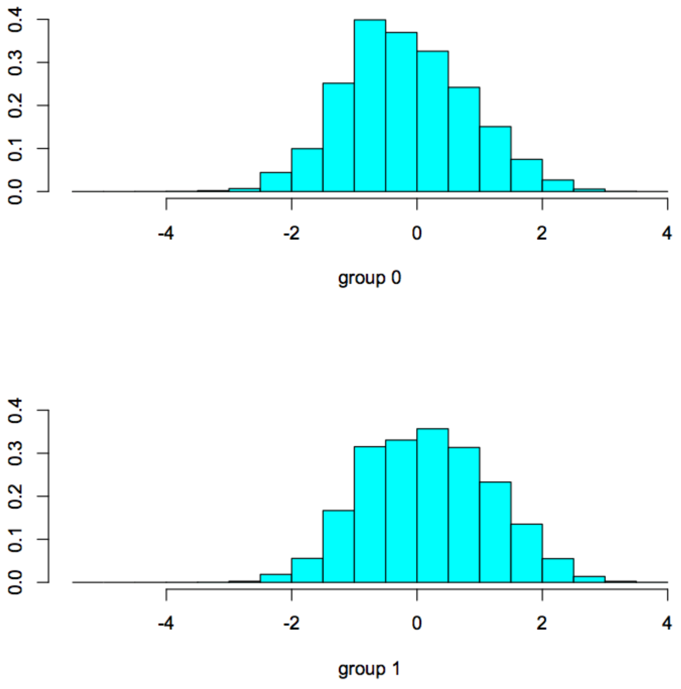


Figure 4.23: Plot of the group of instances with and without claims into the linear discriminator given by LDA

The LDA is combined with regression but this time even if the classification seems to work well it is not able to improve a lot the results of the linear model.

In Figure 4.24, the highest GI 0,0883 in the training cross validation is obtained for thresholds 0,3. Using this threshold the GI on the test set is 0,0786. The combination of LDA and linear model performs just a bit better than the simple linear model (GI 0,0776).

The ROC curve as well confirms that the LDA classification is doing well since the line is above the diagonal (Figure 4.25).

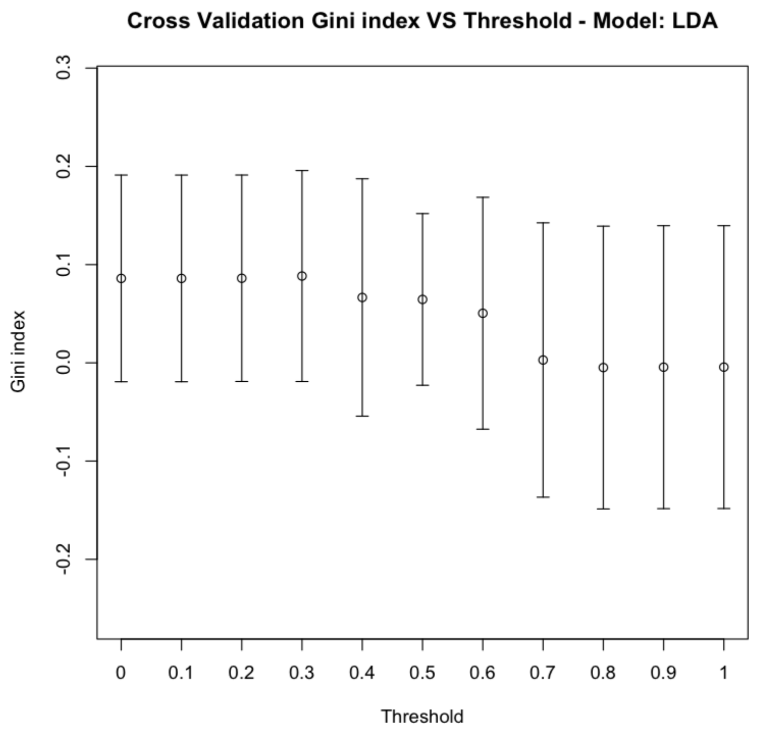


Figure 4.24: Cross Validation Gini Index for different thresholds.

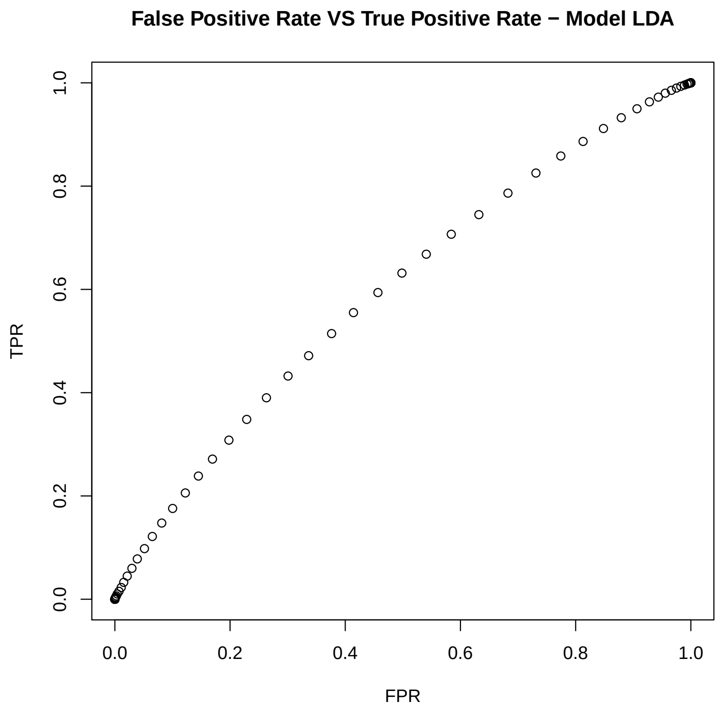


Figure 4.25: Roc curve for Linear discriminant analysis.

4.4 Unsupervised techniques

Between all the classifiers seen so far the ones performing better are: decision tree, random forest and LDA. However the classification techniques explored do not use any dimensionality reduction yet. We have seen previously that the K-means algorithm and the PCA are both able to improve the results of the regression so we now combine these techniques with our classifiers. With linear models clustering performed better than the PCA on the training data but worse on the test set. In this situation, is is not clear which dimensionality reduction strategy could return the best results. Therefore, in the following part both clusters and principal components are computed before the classification and regression to see which is the best.

4.4.1 K-means

In the previous application of K-means with the linear Model, 20 seems to be the best number of clusters. The same clusters are kept here in order to able to compare the results.

Using the K-means before the tree classifier and linear Model, the GI of the cross validation on the training data shows a great performance.

The highest GI is reached with threshold 0,1 (GI = 0,1144). Hence, using clustering, the optimal amount of data predicted as zero from the tree algorithm decreases since the threshold is smaller than before (0,3). With a small threshold the instances predicted as zero have a high probability of having actual values of zero. However the results do not improve a lot from the ones obtained using only K-means and then linear model (GI = 0.1118). This could be expected since with a threshold of 0,1 the contribution of the tree model is relatively small. Most of the data are predicted using a linear model. The same model on the test data returns a GI of just 0,07598.

Using K-means with random forest the highest GI is 0,1149 for threshold 0,2 but using this threshold on the test data the GI is 0,07356 (Figure 4.27).

The linear discriminant analysis as well is then combined with the linear model and K-means. The cross validation Gini on the training data for threshold 0,4 is the highest with a score of 0,1229 (Figure 4.28).

This later model, within all the models tested, is the one with the highest score in the cross-fold validation. This is applied then on the test data and using a threshold of 0,4 it returns a GI of 0,08593. Once again, using K-means, the results show a good performance in the cross-validation on the training set. However, when it is applied to the test set, the GI decreases a lot.

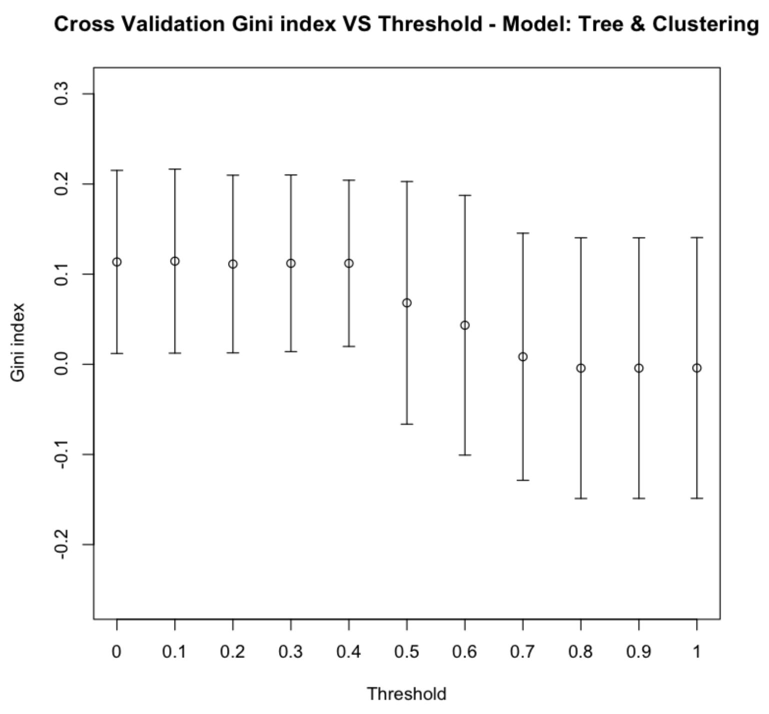


Figure 4.26: Cross Validation Gini Index for different thresholds using clusters and tree classifier.

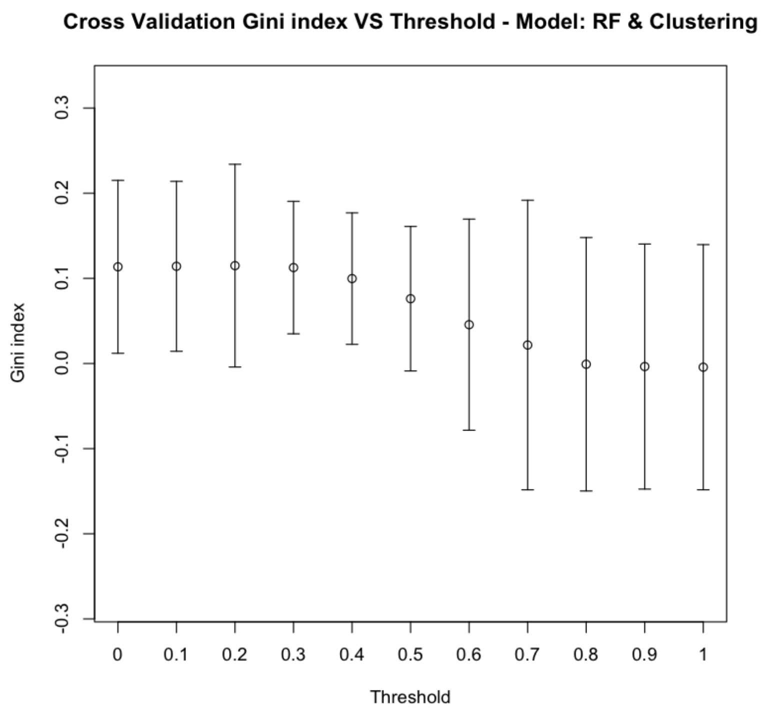


Figure 4.27: Cross Validation Gini Index for different thresholds using Cluster and RandomForest classifier.

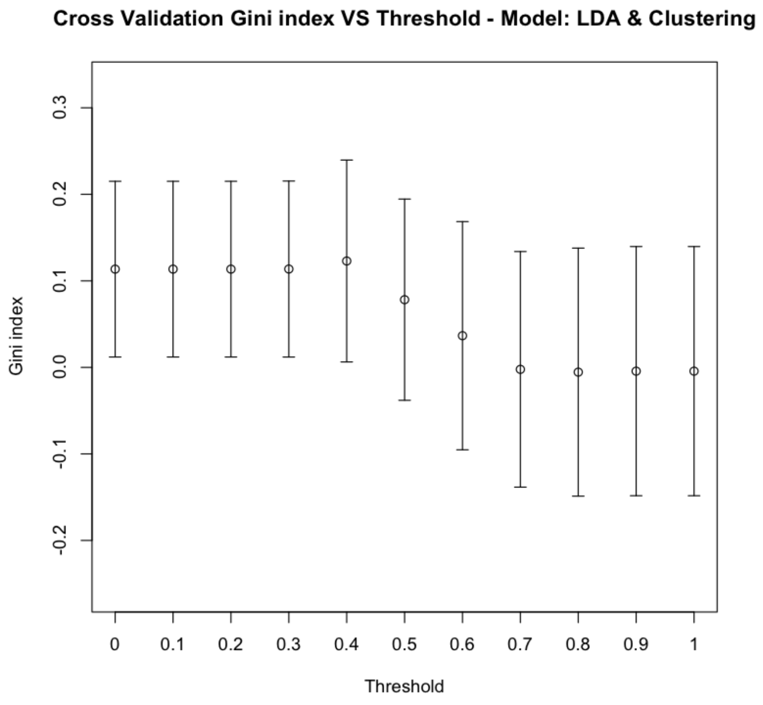


Figure 4.28: Cross Validation Gini Index for different thresholds using clusters and LDA classifier.

4.4.2 Principal Components Analysis (PCA)

To be able to compare the results with the linear model, 20 principal components are kept for both classification and regression.

Using the decision tree, the maximum GI on the training set is 0,10855 for a threshold of 0,1. The corresponding score on the test set is 0,07576.

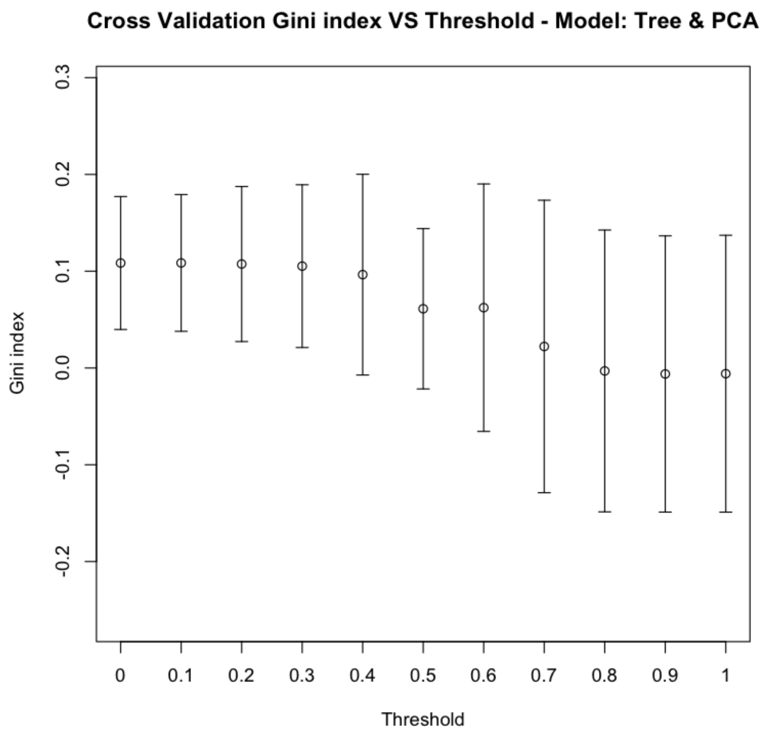


Figure 4.29: Cross Validation Gini Index for different thresholds using PCA and tree classifier.

For random forest the threshold 0,2 returns a GI of 0,10855 in the training data with and 0,078569 on the test set.

With the linear discriminant analysis the maximum GI in the train is 0,11009 for a threshold of 0,3 and 0,07667 in the test set.

All the results are summarized in the Table 4.32.

The first model with threshold 1 correspond to the "Null" model where all the instances are predicted as zero. The model that returns the highest GI is the simple decision tree algorithm without any regression or dimensionality reduction technique with a GI of 0,0895. Second is the combination of linear discriminant analysis with a linear model when the dataset is clustered to have 20 variables with a GI of 0,0859. Whilst, in third is the linear model with 20 clusters and a GI of 0,0808. In all cases the estimate of the GI in the training set using a 10 fold cross validation is always bigger than the score obtained

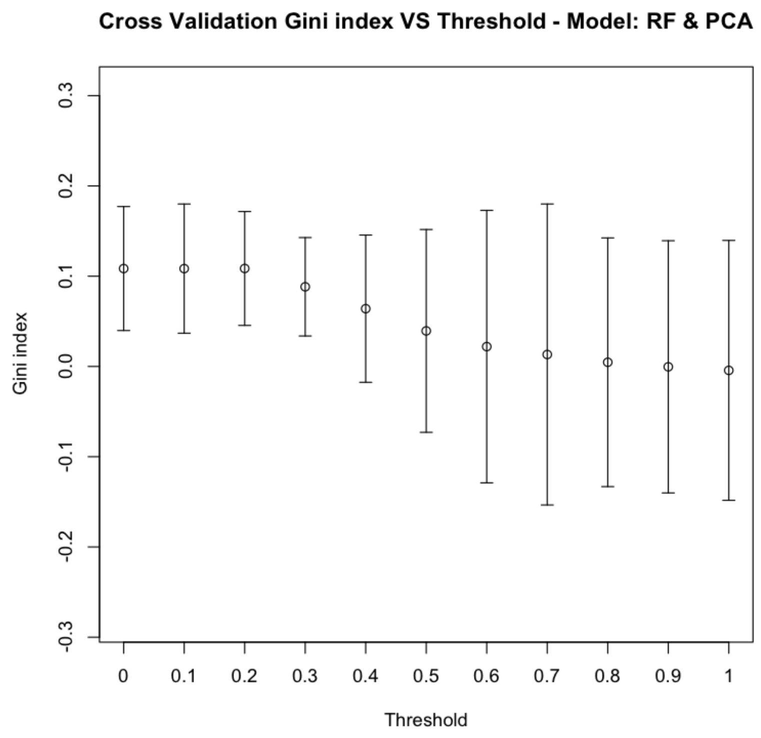


Figure 4.30: Cross Validation Gini Index for different thresholds using PCA and RandomForest classifier.

in the test set.

From Table 4.32 is possible to notice that in general using a strategy of combining a classification and regression improves the results obtained with just a linear regression. This is not the case for the tree classifier where actually the best score is obtained, since adding a linear model decreases the GI on the test set.

4.5 Submission and final position in the competition

The test set provided by the organizer of the competition contains information from 2008. However the leader board on the website is calculated using data from 2009. The GI on the Kaagle website is different from the one computed on the test set. The tree model is still the one with the greatest performance but the score is 0,06453. In the competition there are 202 players in 107 teams and with this score we are able to get the 43-rd position. It is likely that the dataset used for the final leader board contains new information such as new types of vehicles or new models of a car included in the data that could explain the difference between the scores obtained on the website and the one computed on the test set. Another reason could be that the data set used for the final scoring has different

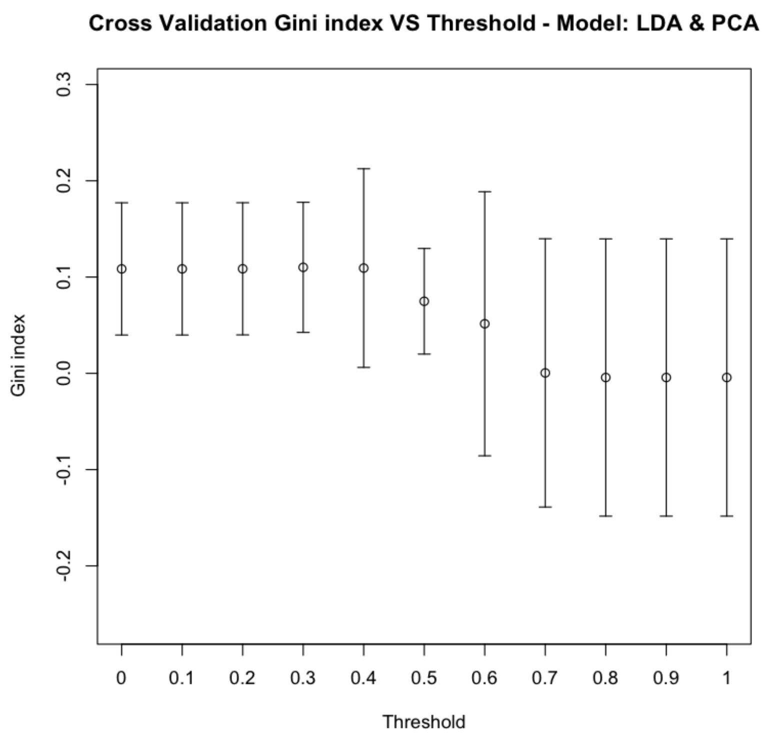
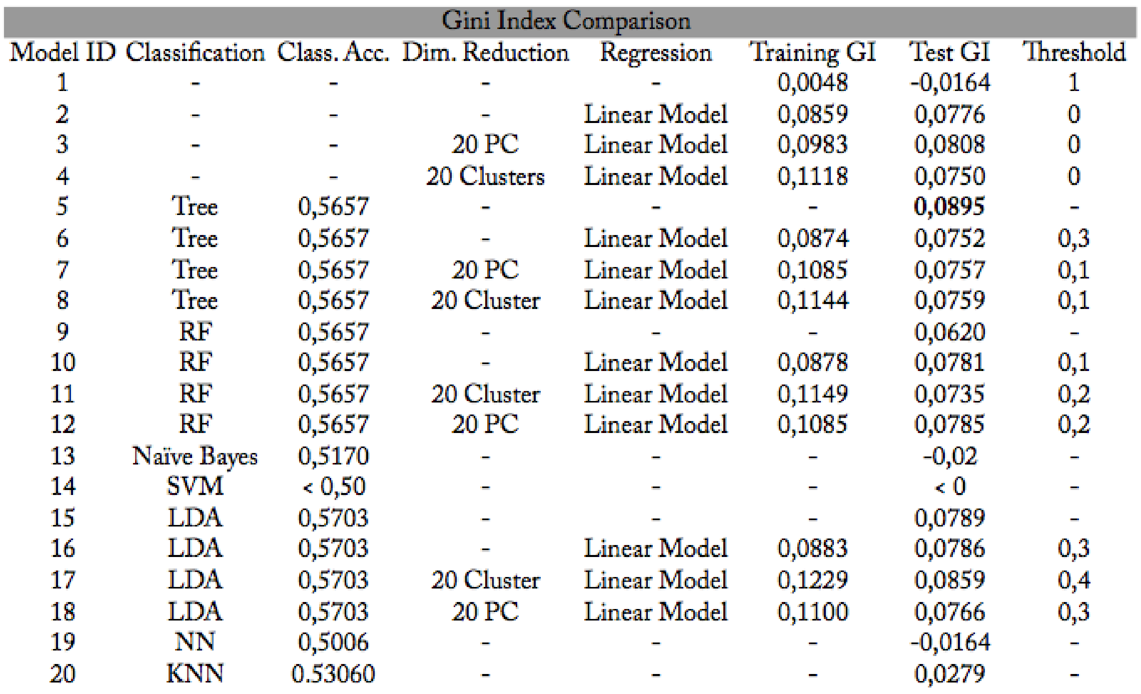


Figure 4.31: Cross Validation Gini Index for different thresholds using PCA and LDA classifier.

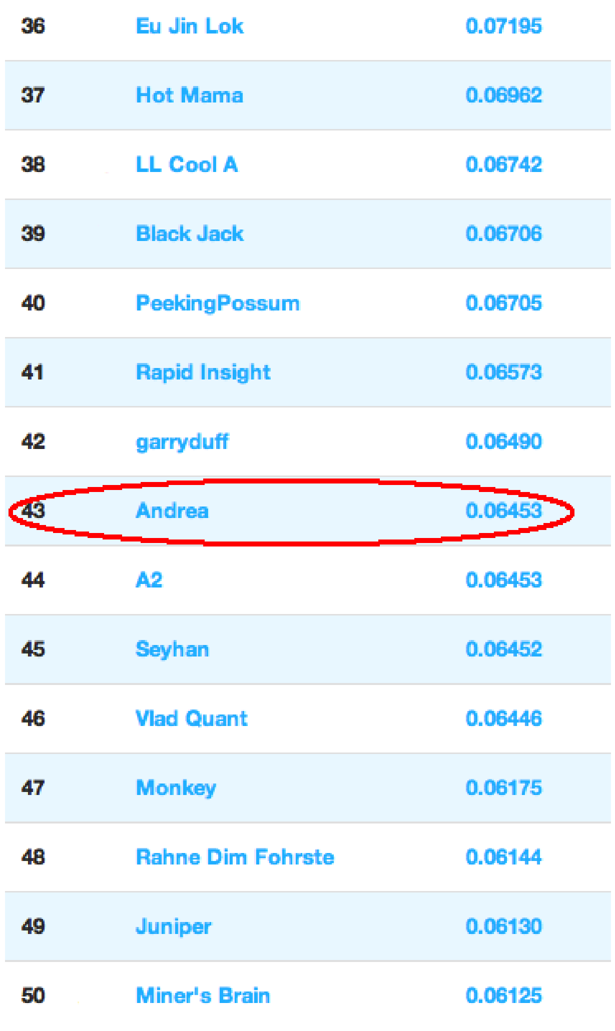
distribution than the one provided for the competition.

The maximum GI obtained by the winner of the competition was 0,2015. Only the first 15 were able to have a GI greater than 0,10.



Gini Index Comparison							
Model ID	Classification	Class. Acc.	Dim. Reduction	Regression	Training GI	Test GI	Threshold
1	-	-	-	-	0,0048	-0,0164	1
2	-	-	-	Linear Model	0,0859	0,0776	0
3	-	-	20 PC	Linear Model	0,0983	0,0808	0
4	-	-	20 Clusters	Linear Model	0,1118	0,0750	0
5	Tree	0,5657	-	-	-	**0,0895**	-
6	Tree	0,5657	-	Linear Model	0,0874	0,0752	0,3
7	Tree	0,5657	20 PC	Linear Model	0,1085	0,0757	0,1
8	Tree	0,5657	20 Cluster	Linear Model	0,1144	0,0759	0,1
9	RF	0,5657	-	-	-	0,0620	-
10	RF	0,5657	-	Linear Model	0,0878	0,0781	0,1
11	RF	0,5657	20 Cluster	Linear Model	0,1149	0,0735	0,2
12	RF	0,5657	20 PC	Linear Model	0,1085	0,0785	0,2
13	Naïve Bayes	0,5170	-	-	-	-0,02	-
14	SVM	< 0,50	-	-	-	< 0	-
15	LDA	0,5703	-	-	-	0,0789	-
16	LDA	0,5703	-	Linear Model	0,0883	0,0786	0,3
17	LDA	0,5703	20 Cluster	Linear Model	0,1229	0,0859	0,4
18	LDA	0,5703	20 PC	Linear Model	0,1100	0,0766	0,3
19	NN	0,5006	-	-	-	-0,0164	-
20	KNN	0.53060	-	-	-	0,0279	-

Figure 4.32: Table of Gini Index comparisons, Training GI is the Gini Index mean resulting from the cross validation on the training set. Test GI the Gini index in the test set. In the case of simple classification algorithms (models 5,9,13,14,15,19, 20) the GI is not estimated in the training set because the dataset is balanced while the test set is unbalanced.



36	Eu Jin Lok	0.07195
37	Hot Mama	0.06962
38	LL Cool A	0.06742
39	Black Jack	0.06706
40	PeekingPossum	0.06705
41	Rapid Insight	0.06573
42	garryduff	0.06490
43	Andrea	0.06453
44	A2	0.06453
45	Seyhan	0.06452
46	Vlad Quant	0.06446
47	Monkey	0.06175
48	Rahne Dim Fohrste	0.06144
49	Juniper	0.06130
50	Miner's Brain	0.06125

Figure 4.33: Final leaderboard of the competition.

Chapter 5

Conclusions

The most challenging part of the competition was to find a way to process the data and create a data set that could be used by different algorithms. Not all the algorithms were able to handle the presence of categorical variables so those had to be transformed into binary variables. Once these variables were transformed the dataset had about 100 variables so unsupervised techniques such as PCA and K-means were attempted to reduce the dataset dimensionality.

The company hosting the competition did not provide the meaning of all the variables included so it was not possible to asses if in reducing the dataset some meaningful variables were dropped by chance.

The dataset used to test the models contained only the intersection of the levels present in the train and test set so some relevant levels could have been omitted. The clusters were computed on the training and then variables falling in the same cluster grouped together. Since in the test set there are some new levels it may be the case that for the test set the best clusters may have different groups of variables than those found on the training set.

From the results obtained it is possible to say that in general clustering and PCA improved the model accuracy. In both regression and classification algorithms the use of principal components and clusters reduced the dimensionality of the problem and often increased the GI. This means that in the 101 variables some were redundant, not bringing useful information to the problem. The use of principal components made possible to transform the problem into a smaller dimension, while keeping most of variability in the data.

Using the normalized Gini as a metrics, it was not important to predict the claims with a good precision, only the order of the instances' claim was relevant. This allowed to use the probability of classification an instance with or without claim as a ranking for the predictions.

The tree algorithm resulted in the ability to give the best ordering of the claim. This algorithm was actually the fastest to build and to be applied, the time taken to create the model was only 30 seconds.

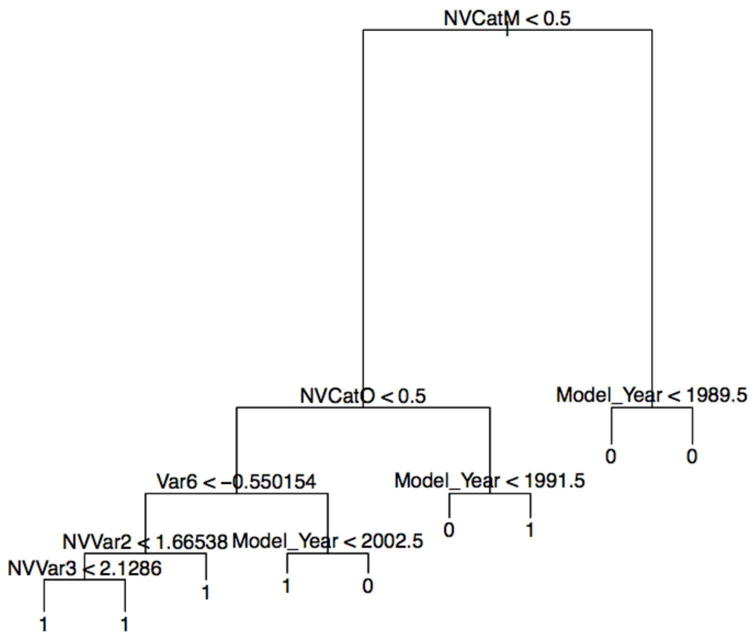


Figure 5.1: Tree model for *mindev* 0,0005.

Form the plot of the tree model in Figure 5.1 it is possible to see how the *ModelYear* variable is used three time down the tree so it is quite discriminant. A surprising fact is that at the first node there is the variables *NVCatM* that is the binary variable corresponding to the variable *NVCat* for level *M*. From the information given in the competition's website the *NV* variables are non-vehicle features and it was expected that those could not contribute a lot in the predictions.

The strategy proposed in this thesis was a combination of classification techniques and linear regression. This strategy was able to improve the results obtained by the simple linear regression model in most of the cases except for the tree classifier. The tree returned a higher GI than the linear regression. In the Table 4.32 is possible to notice that the application of regression in combination with tree (model 6) actually gave worse results than the one obtained with a single tree (model 5) or linear model (model 2). The best score resulting from the combination of tree and a linear model was with the threshold 0,1 which means that almost 90% of all the instances were predicted using the linear model. The higher GI for the tree-only model means that the tree algorithm was actually able to return a better order of the instances with claims than the linear model.

The time available for the competition was limited to three months and the competition was joined after three weeks it had started. Almost the first months was spent in understanding how to process the data and apply the algorithms. The limited time did not allow to explorer more algorithms.

With more time it could have been interesting to apply some feature selection techniques in order to select only the most relevant variables. In the case of *Backward elimination*, variables are tested one by one and the ones that are not "statistically significant" deleted from the dataset. A different approach is the *Forward selection* that starts without any variables in the dataset and then adds only the variables that are statistically significant. These approaches could have led to remove redundant features, enhancing generalization capability of the model [WIK].

It could have also been interesting to test different algorithms not only for classification, but even for regression. For example neural network, tree and support vector machines are able to predict continuous response variables as well. The thesis proposed only the K-means algorithm for clustering which is based on the idea of cluster centroids. However in literature there are many other clustering methods such as hierarchical clustering, distribution-based clustering and density-based clustering that do not define the number of clusters *a priori*. The use of these alternative clustering techniques could have also brought improvements to the results obtained in this thesis.

Chapter 6

Acknowledgments

In poche righe é difficile ricordare tutti coloro che in questi anni mi sono stati vicino.

Vorrei ringraziare in primis la mia famiglia che mi ha sostenuto non solo economicamente, ma anche moralmente durante tutto il percorso accademico.
Un grazie a Elisa che mi ha sempre supportato e spronato durante questi anni di studio e lavoro.
Un grazie alla compagnia di Isola Vicentina che mi ha sempre accolto anche quando non mi facevo vivo per tanto tempo.
Un grazie agli amici di Rimini che mi hanno fatto sentire meno il peso del lavoro e dello studio.
Un grazie ai colleghi di Traderlink e Directa che mi hanno fatto crescere non solo professionalmente ma anche umanamente.
Un grazie al Prof. Gianluca Moro che mi ha seguito durante la tesi e mi ha permesso di vincere la borsa di studio per la tesi all' estero.
Un grazie al Prof. Gianluca Bontempi e Dott. Yan Aël Le Borgne che mi hanno supervisionato durante il periodo di stage a Bruxelles.
Un grazie infine a tutti quelli che ho incontrato nel mio cammino.

Bibliography

[Ait35] A.C. Aitken. On least squares and linear combinations of observations. *Proceedings of the Royal Society of Edinburgh*, 55:42–48, 1935.

[BB03] Mauro Birattari and Gianluca Bontempi. *lazy: Lazy Learning for Local Regression*, 2003. R package version 1.2-14.

[BBB99] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for local modelling and control design. *International Journal of Control*, 72(7-8):643–658, 1999.

[Bon08] Gianluca Bontempi. Statistical foundations of machine learning. *Handbook*, 2008.

[Bre01] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[CNM06] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[CZ01] A. Cutler and G. Zhao. Pert-perfect random tree ensembles. *Computing Science and Statistics*, 33:490–497, 2001.

[DEO11] F. Diskaya, S. Emir, and N. Orhan. Determining the necessary criteria for the eu membership by using the machine learning method: 2005-2010 period analysis. *Procedia-Social and Behavioral Sciences*, 24:808–814, 2011.

[DHL$^+$11] Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer, , and Andreas Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2011. R package version 1.5-27.

[Dud76] S.A. Dudani. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, (4):325–327, 1976.

[Fis36] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[GE03] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[Gru69] F.E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, pages 1–21, 1969.

[HTF04] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: Data mining, inference, and prediction. *BeiJing: Publishing House of Electronics Industry*, 2004.

[L$^+$10] L. Lusa et al. Class prediction for high-dimensional class-imbalanced data. *BMC bioinformatics*, 11(1):523, 2010.

[LB09] Yann-Aël Le Borgne. *Learning in Wireless Sensor Networks for Energy-Efficient Environmental Monitoring.* PhD thesis, 2009.

[LW02] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[Mah36] P.C. Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Science, Calcutta*, volume 12, page 49, 1936.

[Mit97] T.M. Mitchell. Machine learning. *Mac Graw Hill*, 1997.

[MW92] G.J. McLachlan and J. Wiley. *Discriminant analysis and statistical pattern recognition.* Wiley Online Library, 1992.

[Qui93] J.R. Quinlan. *C4. 5: programs for machine learning.* Morgan Kaufmann, 1993.

[R D11] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[Rip11] Brian Ripley. *tree: Classification and regression trees*, 2011. R package version 1.0-29.

[VR02a] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S.* Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[VR02b] W.N. Venables and B.D. Ripley. *Modern applied statistics with S.* Springer verlag, 2002.

[WF05] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[WIK] WIKIPEDIA. Website: `http://www.wikipedia.org`.

[WKRQ$^+$08] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[Zav97] J. Zavrel. An empirical re-examination of weighted voting for k-nn. In *Proceedings of the 7th Belgian-Dutch Conference on Machine Learning*, pages 139–148. Citeseer, 1997.