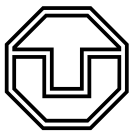


TECHNISCHE
UNIVERSITÄT
DRESDEN

Dep. of Computer Science Institute for System Architecture, Database Technology Group

Bachelor Thesis

APPLICABILITY AND PARAMETERIZATION OF MACHINE LEARNING APPROACHES FOR TIME SERIES MODELING

Onur Ekici
3668317

Supervised by:
Prof. Dr.-Ing. Wolfgang Lehner

and:
Dipl.-Inf. Claudio Hartmann

Submitted on 07 Mai 2014

CONFIRMATION

I confirm that I independently prepared the thesis and that I used only the references and auxiliary means indicated in the thesis.

Dresden, 07 Mai 2014

ABSTRACT

The forecasting of time series values plays an important role in many areas. The forecast values for time series are generally determined using statistical models, which require a long and consistent history. If this is not the case, machine learning approaches can be applied to create predictive model for time series. Regression tree and linear model are the most commonly used predictive models in machine learning community. Random forest offers competitive prediction performance as an ensemble of regression trees.

This thesis aims to provide a comprehensive analysis of the applicability of random forest and linear model to create a predictive model for time series. Here, random forest is presented as a novel way to build a predictive model. Moreover, the improvements to increase the accuracy of this model, with the help of linear model, are covered from three different perspectives:

- Selection of the relevant features
- The bias correction
- Ensemble of models

The accuracy of the random forest has improved by using the relevant features and assembling with linear model. The ensemble of linear model and random forest has a better accuracy than either the linear model, or the random forest alone.

CONTENTS

1 INTRODUCTION

The forecasting of time series values plays an important role in many areas such as for instance assistance in decision-making and investment planning. These are just two of many examples.

The forecast values for time series are generally determined using statistical models, which are based on time series model characteristics such as seasonal effects and trends. The statistical models require a long and consistent history. The aim of this thesis is to investigate the applicability of the machine learning approaches to time series, which are sparse and too short for the statistical models, and improvements to these approaches. For this purpose, two different approaches are used, linear model and random forest, which are based on two different techniques. Random forest is a tree based machine learning approach, which improves the decision tree by applying two methods bootstrap aggregation [Bre96] and random subspace method [Ho98] [AG97]. The predictive performance of random forest is confirmed in many studies from various fields such as chemistry [POGM07], biology [SSB+04] and finance [KT06]. Linear model is a mainstay of statistic for the past 30 years [HTF+09] and most known regression model with decision tree in the machine learning community. [KKZ06]

To understand the mechanism of random forest, decision tree learning is introduced in the second chapter. Additionally, the basis of machine learning, time series and linear model are given.

In Chapter 3 the way to build predictive models with random forest approach in time series is described. This chapter contain also the tuning and parametrization of random forest. Later on the improvement to the accuracy of random forest are studied. In the literature, several theories have been proposed to improve the accuracy of random forest. Three of them have been studied in this work: "Bagged averaging of regression models" (2006)[KKZ06], "Improvements to random forest methodology" (2013) [Xu13] and "Bias-corrected random forests in regression" (2012). [ZL12]

In Chapter 4 the following questions, which are studied theoretically in Chapter 3, are answered empirically with the help of experiment results and the cahpter 5 has drawn a conclusion:

- Which feature selection method should be used in random forest and is it necessary ?
- How effective are the bias correction methods in the time series ?
- Is it possible to improve accuracy with ensemble of linear model and random forest ?

2 BACKGROUND AND REVIEW OF LITERATURE

This chapter provides a basis for understanding of time series and two machine learning approaches, random forests and linear model.

2.1 MACHINE LEARNING

In this section the basic terms of machine learning are defined. After a short overview, classification and regression trees are introduced, which is the basis of the random forests. Subsequently, random forests and linear regression as machine learning approaches are studied. This section is mainly based on the book *The elements of statistical learning* [HTF+09].

2.1.1 Overview

Machine learning corresponds to the automatic learning techniques based on observational data, which aim to make predictions as precise as possible or identify patterns. There are many tasks which can be solved by using machine learning. In this work two examples will be given. One of them is based on recognizing spam mails for subsequent filtering, which nowadays every e-mail service does. The second one is the practical part of this thesis, which predicts the number of sold items in the coming month.

The machine learning systems function through applying the knowledge that they gained from examples on prediction. Most of these systems are algorithmically generated, which are contributed from the fields of probability and statistics, information theory and artificial intelligence. These algorithms are divided into a few categories such as *supervised and unsupervised learning* algorithms. In this thesis, *supervised learning* algorithms, which have an input-output pair to learn from, will be analyzed.[Mit97]

In the machine learning systems there are two major problems: classification and regression. Classification deals with identifying group membership such as spam or not spam mail. Regression involves predicting continuous values such as the number of sold items.

The supervised learning will be formalized and used in the following way:

Notation 1. *The problem consists of $n \in \mathbb{N}$ observation of p-dimensional feature vector $x = (x_1, x_2, x_3, ..., x_p)$ and response variable y. $\mathbb{X}$ represent all x and $\mathbb{Y}$ all y.*

Observation can be also called training sample or instance and each dimension is an attribute or a feature such as brand, color, size etc.

L is a data set contains instances. $L = (x_1, y_1), (x_2, y_2), (x_3, y_3).....(x_n, y_n)$

The aim is to find a function $f(x) = y$; $\forall(x, y)\ (x, y) \in \mathbb{L}$

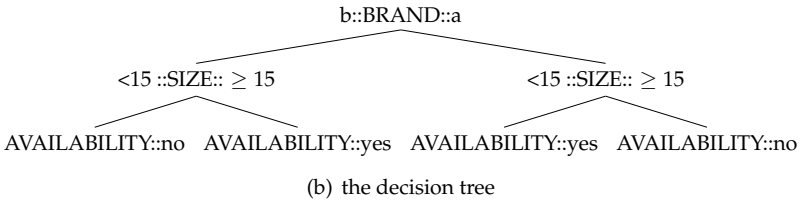
2.1.2 Classification and Regression Tree

Decision trees are special tree-like graphs which represent decisions and their possible consequences. The data is separated into several groups based on the certain rules. Usually decision trees are generated according to the top-down principle. To find best split at each step, the attribute according to which the data can be classified the best is searched. This means that the distribution of the data is chosen which promises the maximum gain of information.

In order to present the decision trees, a simple decision tree is shown in Figure 2.1 . The aim in this example is to predict the availability from brand and size. The response variable "availability" is categorical, thus a classification problem results.

Brand	Size	Availability
a	20	yes
a	20	yes
a	10	no
b	20	no
b	20	no
b	10	yes

(a) the data set



(b) the decision tree

Figure 2.1: A simple decision tree

In this example the difficulty in defining the splits is deciding the order of the features, brand and size. The information gain should be measured in order to find the "best" split. This can be done by entropy, Gini-index, deviance (for Classification problems) or mean-squared-error(for Regression problems).[HTF+09] In this example the splits are based on entropy.

Entropy is a measure of the purity of the data set. It is very useful for binary classifications. If entropy is 0 all members of the dataset belong to the same class. This is our aim to split data. If the number of instances for each group are equal, the entropy is 1. For unequal numbers, its between 0 and 1.

The binary entropy function for a data set L is defined:

$$H(L) = H_b(p) = -p * log_2(p) - (1-p) * log_2(1-p)$$

where p is the proportion for one class and 1-p for the other.

In the dataset the result is 3 times "yes" and 3 times "no" while the entropy is 1.

$H_{available}(L) = H_b(0.5) = 1$

The resultant entropy, after deciding all possible splits, should be calculated to determine the best split. First, the best of all possible splits for each feature is considered. There are 2 types of features in the data set namely categorical, such as brand, and continuous, such as size. A continuous feature with k distinct values has $k-1$ possible splits. The feature "size" has 2 distinct values: 10, 20. There is just one possible split for this feature, greater or equal than 15 or less than 15. The other type, categorical features with k categories have $2^{k-1}-1$ possible splits where as the feature "brand" has also just one possible split, a or b. After finding the best split for each feature the best split of all those results should be determined.

$$H_{brand}(L) = \frac{1}{2} * H_b(0.25) + \frac{1}{2} * H_b(0.25) = 0.81$$
$$H_{size}(L) = \frac{1}{2} * H_b(0.5) + \frac{1}{2} * H_b(0.5) = 1$$

The example in Figure 2.1 is simple and both features have just one split. Therefore, it is not required to search the best split for each feature. The question of which feature must be split remains. Splitting by "size" does not cause reduction of the entropy, because the proportion of group members stays equal after the splitting. But we can reduce the entropy by splitting based on "brand". In the following turn the best split is searched again with the same rules and all groups have one distinct class after splitting by "size", so the entropy is 0 in the end.

Classifications and Regression Trees

CART is a tree based machine learning method to build predictive models described by Breiman,Friedman, Olshen and Stone.[BFOS84] The main idea is to split the data recursively into two groups in order to improve the fit in each group. It can be represented as a binary decision tree.

In the book [HTF+09] the reason for the binary division is explained "The problem is that multi way splits fragment the data too quickly, leaving insufficient data at the next level down.[...] Since multi way splits can be achieved by a series of binary splits, the latter are preferred."

CART has a two major task. The first is when to stop splitting, it is called a stopping criteria which every recursive algorithm has and secondly how to split data in each step, also known as the selecting "best" split. The key word for the selecting best split is impurity of a node. The measure for impurity of a node was the entropy in Figure 2.1.

The search of the "best" is depending on local criteria, it is greedy algorithm, which promises the best result at the time of choice. In greedy search, tree is optimal at each split, but it is not

guaranteed globally optimal tree at the end.

Based on a simple regression problem the idea behind CART is illustrated in the above example. The ”price” for each instance is predicted from ”size” and ”brand”. The response variable y is continuous, thus it is a regression problem.

Brand	Size	Price
a	10	6
a	20	8
a	30	19
b	40	27

Figure 2.2: A simple data set

The data set in Figure 2.2 is similar to the data set in Figure 1.1 except two points: The number of distinct values of feature ”size” are 4. And the response variable y is continuous value, thus it is a regression problem and a regression tree is required. Entropy is not suitable for splitting criteria in regression tress. One of the splitting criteria for regression trees is the mean squared error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

For this dataset a regression tree will be grown. The classification trees have a class as a response variable in a terminal node such as ”yes” or ”no” in the data set in Figure 2.1. But a regression tree should have a constant k_i in a terminal node to build a function in the end [HTF+09]:

$$m(x) = \sum_{i=1}^{l}(k_i * I(x \in L_i))$$

where k_i are constants and I(.) is an indicator function returning 1 if its argument is true and 0 otherwise.

The constant k_i, which is the prediction for a leaf c is in CART defined:

$$k_i = \frac{1}{n_c}\sum_{i \in C} y_i$$

where n_c is the number of intances in the leaf c and y_i is the response value.

Before splitting data and growing tree it is started with a single node, which contains all instance. k_i for root is $\frac{60}{4} = 15$ and $MSE = \dfrac{9^2 + 7^2 + 4^2 + 12^2}{4} = 72.5$

Now, the aim of CART is minimizing MSE by recursive partitioning. First, the algorithm searchs best split for each feature. The feature ”brand” splits data and reduces MSE to 24.5 from 72.5.

Brand	Size	Price	Predicted price (k_i)	Squared error
a	10	6	11	25
a	20	8	11	25
a	30	19	11	64
b	40	27	27	64
				MSE: $\frac{98}{4} = 24.5$

Figure 2.3: Evaluation by splitting on ”brand”

The feature ”size” has 4 distinct values, that means there are 3 possible splits for feature ”size”:

- ≥ 15
- ≥ 25
- ≥ 35

The best split, which has minimal MSE, is following:

Brand	Size	Price	Predicted Price (k_i)	Squared error
a	10	6	7	1
a	20	8	7	1
a	30	19	23	16
b	40	27	23	16
				MSE: $\frac{34}{4} = 8.5$

Figure 2.4: Evaluation by splitting on ”size” ≥ 25

After splitting by feature ”size” MSE reduces dramatically from 74.5 to 8.5. It promises minimal MSE in all possible splits, thus CART picks the feature ”size” ≥ 25 as a ”best” split.

If the stopping criteria for recursive algorithm is not defined, CART applies all possible splits. The original random forests implementation by Breiman splits until less than 5 instances for regression (1 instance for classification) in a leaf remains. The stopping criteria is ignored in this example to maintain simplicity.

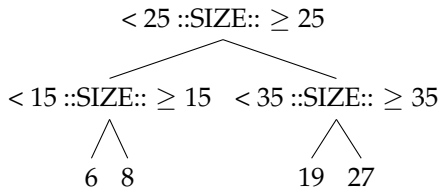


Figure 2.5: The Regression Tree

2.1.3 Random Forest

CART has a number of advantages, such as the simplicity in algorithm, which makes CART computationally fast and easy interpretable. On the other hand there are disadvantages. CART can

overfit the training data, which cause poor accuracy for unseen data. CART can be also too flexible to cause high variance. Small changes in training data lead big changes in the decision tree. To handle these problems through *ensemble theory* Breiman offers *random forest* as a solution.[Bre01]

Ensemble theory is combination of the predictions of poor learning algorithms by performing a lot of extra computation to build a general, robust single model. There are two commonly used ensemble learning algorithms *bootstrap aggregating* [Bre96] and *boosting* [HTF+09] .

Random forest algorithm can build a lot of different trees, while CART algorithm can do only one tree. Each tree in the forest votes for a class (classification problem) or build a function (regression problem) . At the end most voted class (*"The winner takes it all"* principle) or average of selected functions is given for prediction.

The trees in forest is built like CART algorithm. There is no point to build all trees with same input vectors. That give rise to randomness of random forests algorithm. Two methods provide randomness of algorithm, bootstrap aggregating and random subspace method.

Bootstrap Aggregating

Bootstrap aggregating is a machine learning ensemble method to combine from various regression or classification models. It was developed by Leo Breiman. [Bre96] The result of each predictive model is included in the forecast, so the stability and accuracy of machine learning algorithm improve.

Consider the data set L in Notation 1. Suppose that it should be $t \in \mathbb{N}$ number of tree in the random forest, then t data set will be created resulting in one data set for each tree. We resample randomly n times from original data set L with replacement. Therefore, we have $L_1, L_2, ..., L_t$, which has the same size as original. These is called a bootstrap data set. Through "with replacement" every bootstrap dataset can have duplicated data record and some records can be missing from original data set. It is called bootstrap aggregating or bagging.

Breiman shows the improvement after using bagging in "improvements for unstable procedures" [Bre96], which includes neural nets, CART and subset selection in linear regression.

Random Subspace Method

Ho (1995,1998)[Ho98] [Ho95] and Amit and Geman (1997)[AG97] proposed to build trees only for randomly selected subspace to improve the accuracy of individual trees. In CART by building tree in each step, feature for the best split is searched. Random forests applies random subspace method to provide randomness. Feature for best split is searched from $m_{try} \in \mathbb{N}$ random subfeatures out of m features. At each split m_{try} features are randomly selected from all features.

Out Of Bags Error

The random forest, which is studied in these work, is always created with the bootstrap aggregating and the random subspace method; without subsequent prune. An advantage of using the bootstrap aggregating is that one obtains the so-called OBB(out-of-bag) data. By bootstrap aggregating, new learning sets are generated and used for the growth of each tree. A new data set will not contain all the training data and the tree is generated without the knowledge of these data.

That makes testing how well the predictions of this one tree without any external - different from the training data - test data possible.

How many duplicate items are included in each bootstrap data set can be estimated as follows.

The probability that a fixed element of L is drawn at random, is $\frac{1}{n}$. The counter probability that an element is not taken is $1 - \frac{1}{n}$. Because of "with replacement" size of set is always same. The probability that an element is not pulled after n times drawing is $(1 - \frac{1}{n})^n$.

$$\lim_{n\to\infty}(1 - \frac{1}{n})^n = \frac{1}{e} \approx 0.37$$

If we choose the number of trees N sufficiently large, then 37 percent of the data set will be duplicated and because of constant size of data set 37 percent of the data set will be OBB data in each bootstrap dataset.

The OOB data can be used not only for the goodness of individual trees, also for goodness of the whole forest. The OBB estimator can test only OBB data over the tree and estimate OBB error rate. The generalization error can be estimated from the OOB error rate. Empirically, Breiman prove that in his paper 2001. [Bre01]

2.1.4 Linear Models

The linear model has been a mainstay of statistics for the past 30 years and remains one of our most important tools.[HTF+09] It is used to investigate the relationship between a number of independent variables (x1, x2, ...) and a dependent variable y. The idea behind the linear regression analysis is that the dependent variable y is expressed as a function of the independent variable, which is linear in the parameters.

Linear model is expressed by :

$$Y' = w_o + b_1 * x_1 + b_2 * x_2 + ... + b_p * x_p$$

where Y' is the estimated response variable, $b_1, ..., b_p$ are regression coefficients, $x_1 ... x_p$ are independent variables of instance and w_0 is the intercept, or so-called bias.

If p is one, it is called simple linear model, otherwise $p > 1$ it is multiple linear model.

There are different methods to fit the linear model into the data set, but the most popular method is *least squares*.[HTF+09] The least-squares method picks the coefficients to minimize the sum of squared residuals, which is the difference between the actual and the estimated value.

2.2 TIME SERIES

Formally, a time series is a set of values of a variable, which were measured over time and are ordered with respect to the time, such as number of customers, sales volume, population, temperature.

The time series analysis is used in the estimation of the values of these variables for future periods in addition to the description and explanation of the temporal evolution of these variables. It is clear that a forecast can be calculated only by the basis of the assumptions about the behavior of future values. Therefore, the existing data is analyzed in order to identify structures. The aim is to find regularities or patterns, which have not changed during the time and possibly is not going to change in the future, such as for instance seasonal effects and trends. The model, which identifies the structure of time series data precisely, can calculate forecasts according to this regularities and patterns.

The statistical models for the forecasting of time series data, which are based on time series model characteristics such as for instance seasonal effects and trends, require a long and consistent history in order to analyse the behavior of the time series.

Cross Sectional Forecasting

Under certain circumstances the statistical model is sparse and too short to estimate the initial optimization. In these cases the cross sectional forecasting can build a model for this time series. The forecasting models in these thesis are based on this cross sectional forecasting.

The cross sectional forecasting uses transitions between sequent months from a set of time series. If the number of sales units in nth month should be predicted, the algorithm learns from transitions between (n-1)th month to nth month within the last years and predicts nth month from (n-1)th month. $n \in \{1, ..., 12\}$

In the supervised learning algorithms two data sets are required. Therefore the historical data is divided into two parts :

- trainings data,which contains the features from (n-1)th months in the last years and the response variable from nth month within the last years.
- prediction data, which contains the features of (n-1)th month to estimate the response variable in nth month.

In Figure 2.6 cross sectional forecasting is illustrated. To forecast the first month in the third year, transitions between the 12th month and 1st month in the last years are analyzed to build a model. A model is built by using two transitions and reports to calculate 1st month from 12th month.

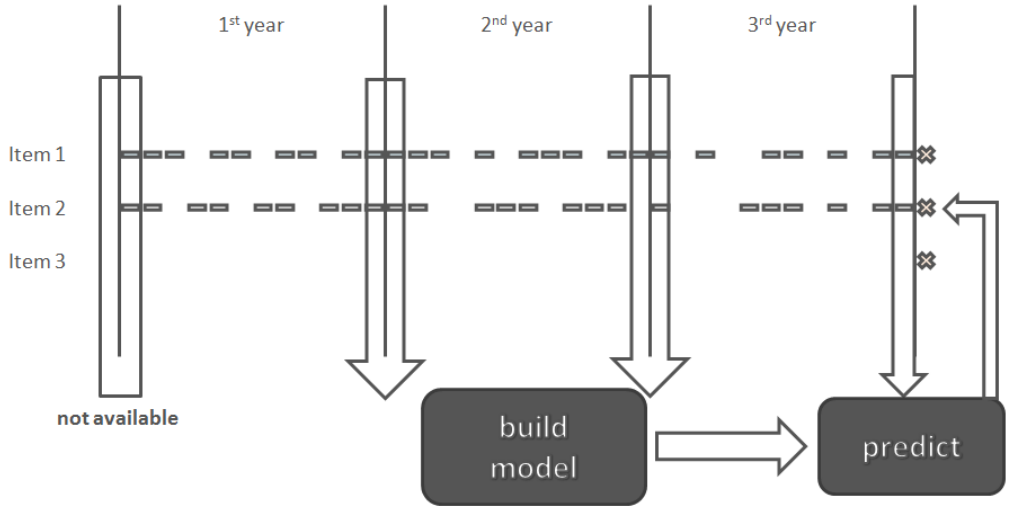


Figure 2.6: Cross sectional forecasting

3 BUILD AND ENSEMBLE MODELS

This Chapter describes the way how to build predictive models based on time series. Firstly, the task and the data will be introduced. Later on the analysis how to build models will be represented. To achieve a better predictive model, the potentiality of ensemble models will be discussed in the end.

3.1 REPRESENTING TASK AND DATA

The aim of this thesis is to investigate the applicability of predictive modeling methods from the field of machine learning in time series. There are market data for these case study, which can be seen as a proper example for sales data.

A simple sales data can be structured in the following way: There are 3 basic components:

- marketing information, which presents stock, sales and purchase numbers of units for each item in each outlet
- item information, which contains properties for each item such as height, volume and color
- outlet information, which describes the outlet such as state and turnover class

Part of the example data is shown in Figure 3.1.

The task is to predict the sales units for each item for the following month according to the cross sectional forecasting, which is introduced in Chapter 1. The data contains three years (which means 36 months in total) marketing information. For cross sectional forecasting, data of the first 13 months is required to start a prediction for the remaining months. In this work the data set, which is created according to the cross sectional forecasting, is labeled as initial data set. The model building process will be studied according to this initial data set in the next section.

period	outlet	item	brand	sales units	stock new units	purchase units
2009-01	outlet1	item 1	brand 1	1	0	0
2009-01	outlet1	item 2	brand 2	1	0	1
2009-01	outlet1	item 3	brand 3	1	4	2
2009-01	outlet2	item 3	brand 3	1	1	2
2010-01	outlet2	item 2	brand 2	2	0	2

outlet	channel	turnover class	state
outlet1	channel1	5 -7.49 MIO	NI
outlet2	channel1	10-14.9 MIO	NW

Figure 3.1: Sales data example

Subsequent to that, the models will be evaluated in the remaining 22 months, described in Chapter 4. The goal is to predict sales units number for each item as precisely as possible. In the end the monthly predictions will be evaluated in 3 different perspectives:

- sales number for each item in the month
- sales number for each brand in the month
- sales number in the month

3.2 DATA PREPROCESSING

In supervised learning, the algorithms learn from training data and predict the new data, which highlights the importance of training data. The training data should be suitable for the problem so that the algorithms can function in a precise way.

The problem is based on the sales numbers of each item so the aggregating of the data by item would be a suitable solution. The features should be aggregated for each item. The continuous features such as market informations can be summed. However, the categorical features such as brand can not. Each item has one constant value for the brand and categorical property features. Therefore, aggregation is not required for them.

In next section the relevance of features from the item and marketing information are analyzed.

3.3 BUILD MODELS

In this section the data structure will be analyzed and the importance of features for predictive models will be measured. To improve the accuracy of predictive models, tuning of parameters of the algorithms will be discussed.

3.3.1 Selecting Features for Random Forest

Identifying relevant predictor variables, rather than only predicting the response by means of some black-box model, is of interest in many applications. [SBK+08]

Due to the structure of a random forest, it is difficult to analyze the influence of input variables, which is easy to exercise in decision trees. For this particular reason, there is no clear way to determine the importance of the features. However, there are different methods to calculate.

A simple, but also naive method would obtain, how often each feature was used in the individual trees for the division. It is called selection frequency. Examples are given by [Sau99],[SRB07]. On the one hand this method is computationally fast and easy to interpret, on the other hand it has a big disadvantage: The count rate in each split has the same weight and the position of the split is not considered.

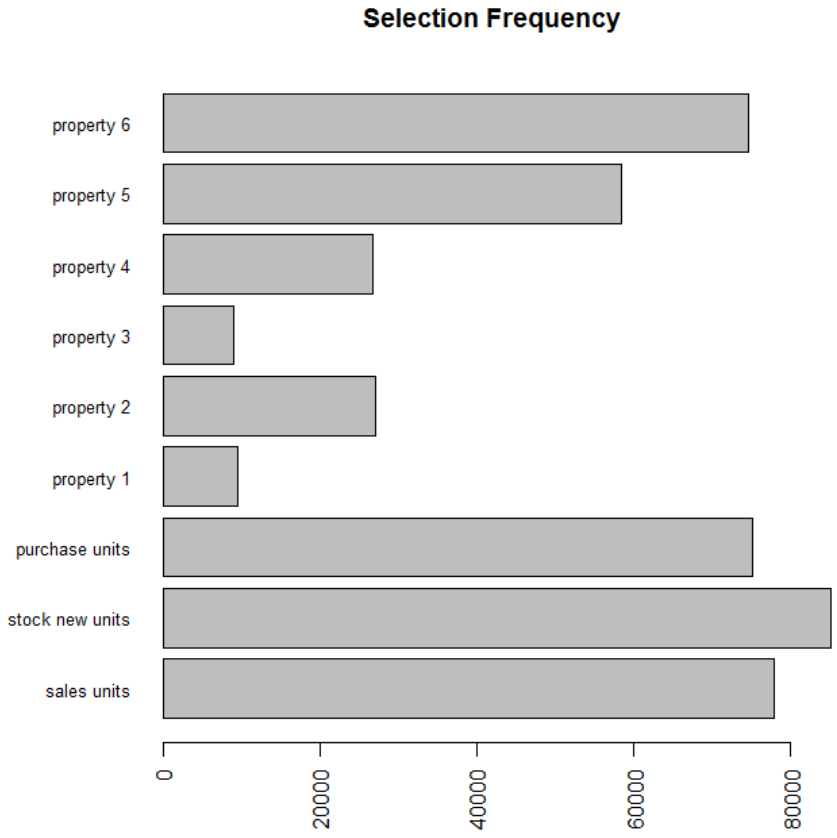


Figure 3.2: The averaged selection frequency of the features in the initial data set

The R implementation of random forests has the opportunity to measure the selection frequency of features, but randomForest package can not support more than 32 categories.[LW02] The feature ”brand” has 81 categories. Therefore, it is not used. The selection frequencies of other features are measured with randomForest package. In Figure 3.2 the averaged selection frequencies of the features in the initial data set are illustrated. The initial data set contains the marketing and item information of the first 13 months according to the cross sectional forecasting. Random forest

trains data of each month and measures the selection frequency. The averaged result of the first 13 months is given in Figure 3.2.

Figure 3.2 shows that ”sales units”, ”stock new units”, ”purchase units”, ”property 5” and ”property 6” is used significantly more than the others for division.

Another measure for feature importance is the Gini importance, which is available in R package randomForest.[LW02] It is using the criterion of the Gini index, which is utilized by the individual trees for their growth. As previously mentioned, this method is just applicable to classification problems. Therefore, it will be not considered any further in this thesis.

Probably the most widely used measure of variable importance is the permutation importance. The idea of the permuted importance is illustrated in Figure 3.3.

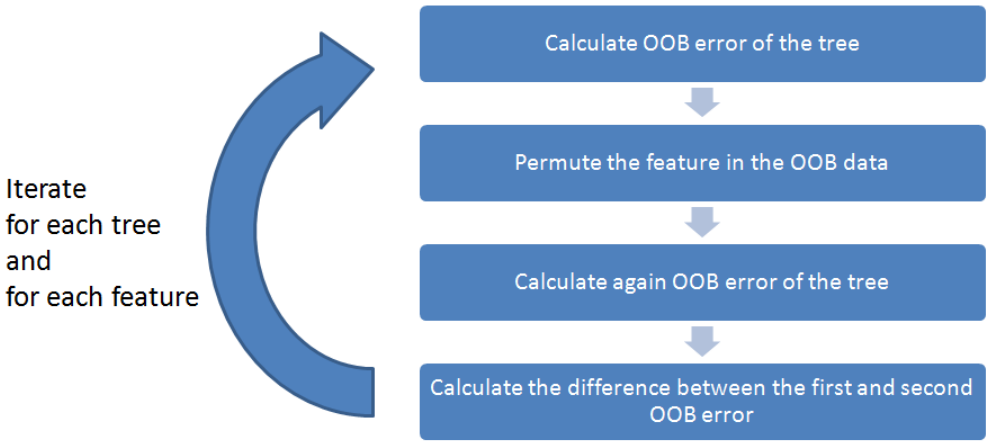


Figure 3.3: Workflow of the permutation importance method

The calculation starts after the creation of all trees in the forest. The OOB training examples of each tree are estimated and the OOB error is calculated. For each training example, the value of a feature is replaced by the value of a feature from another randomly selected training example. Afterwards the OOB calculates error again. The difference between the previous OOB error and the one after the change of the feature, is stored. This procedure is performed for each attribute and each tree. In other words, the value of the feature is artificially noised and the change the OOB error is measured. If the difference between the new and old value is low or even negative, the attribute has no great influence on the response variable. If the difference is large, the attribute corresponding to a high weight is assigned.

The randomForest package in R [LW02] supports the calculation of the permutation importance. Except for the feature ”brand”, the importance of all features are calculated in the same way like selection frequency. The permutation importances of features for each month in the initial data are calculated. The averaged result is illustrated in Figure 3.4. Figure 3.4 shows that ”sales units”, ”stock new units” and ”purchase units” have great influence on the response variable.

The features, ”property 5” and ”property 6”, which have high score in the selection frequency, have no great influence on the response variable according to the permutation accuracy. The prediction accuracies of both selection method are experimented in Chapter 4.

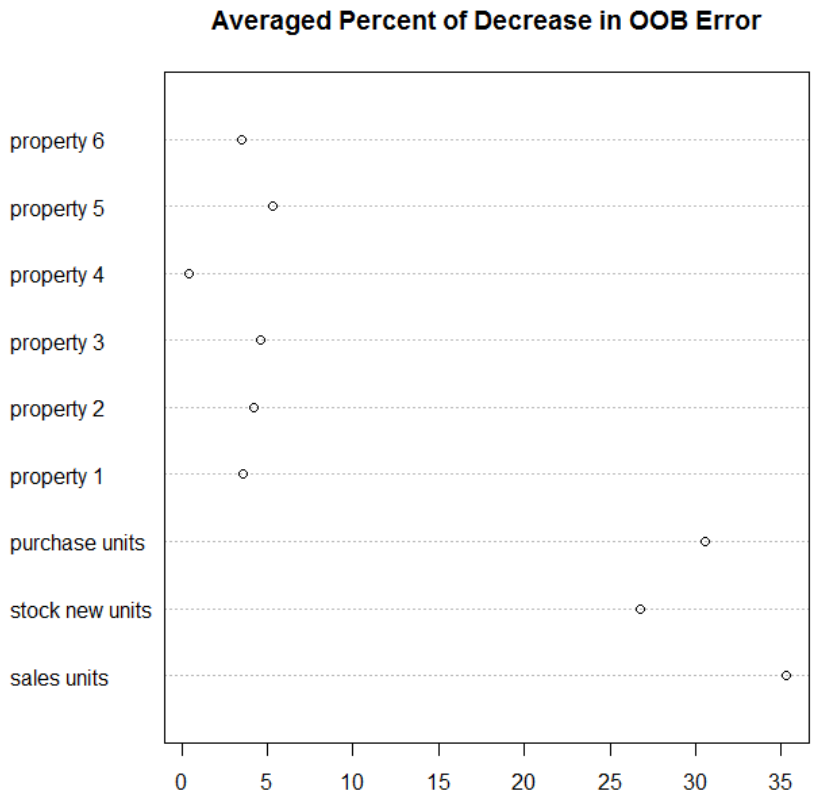


Figure 3.4: Averaged Percent of Decrease in OOB Error for each features

3.3.2 Tuning Random Forests

As it was mentioned before, the OOB error can be used to qualify the model. For the analysis in this section, the initial data set is used. The models with different parameters are compared with the OOB error rate. The argument in the R implementation of random forest is noticed in parentheses.

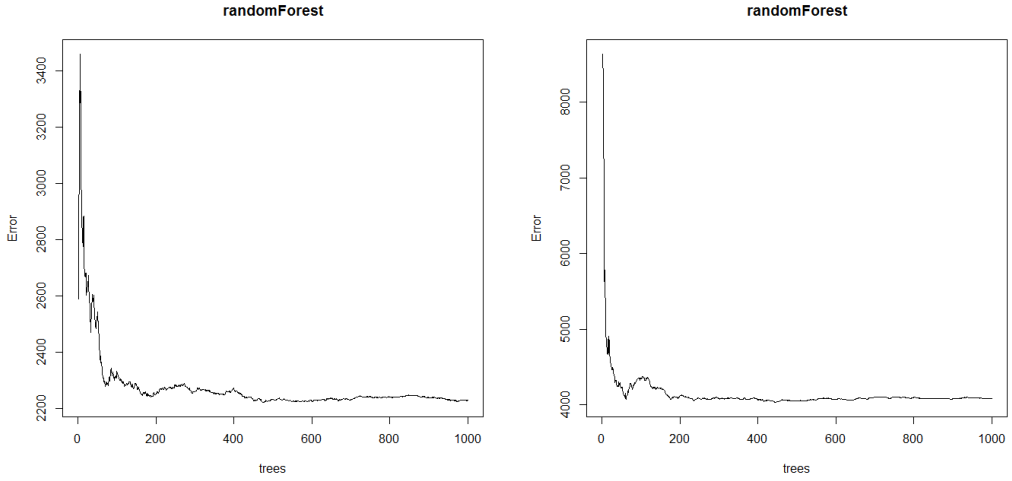
Number of Trees (ntree)

It shows the number of trees, which grow in the forest. It is used mainly for the optimization of the the performance rather than the optimization of the accuracy. It should not be set too small, hence the forest can be stabilized. It should be at least several hundred.[Ber08] In case it is too large, the computation takes more time, but the result does not change. Presumably, it should not be more than several thousand. After a certain number of trees the result does not get better, which proves it is stabilized.[Ber08]

In randomForest package of R the number of trees is set to 500 as a default value.[LW02] Regularly it is regarded as a good compromise. To determine cleverly proper value for this parameter, Richard A. Berk offers in his book to compare results with 3000 trees.[Ber08]

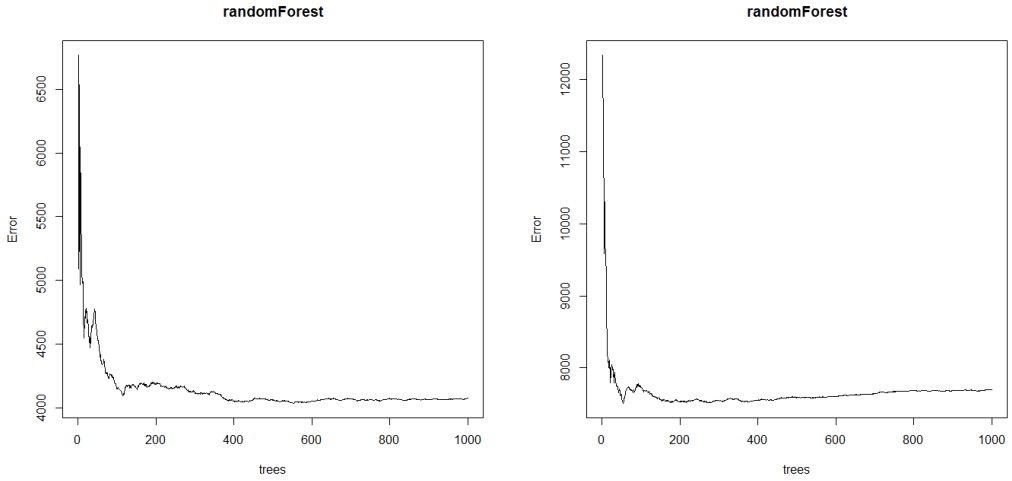
In the case study the initial data is analyzed with 1000 trees for each month. Figure 3.5 represents

OOB error during the forest growing in four different months. The Figure 3.5 shows that 1000 trees are enough to stabilize random forest for the initial data.



(a) between 1st month - 2nd month

(b) between 5th month - 6th month



(c) between 8th month - 9th month

(d) between 12th month - 13th month

Figure 3.5: OOB error during the growing forest with 1000 trees

Node size (nodesize)

Stopping criteria in CART is mentioned in Chapter 2. The node size parameter determines when the algorithm should stop. If the number of instances in the node is less than or equal to the node size, then the algorithm stops splitting and this node is called terminal node.

Node size is the important parameter for CART algorithm, but in random forest it has not so much effect on accuracy. Complexity of tree causes the high variance problem in CART, therefore it is important where the algorithm should stop growing tree to have optimal tree at the end. But this high variance problem is tolerated in random forest because of the averaging over a large number of trees.

Node size in random forest implementation of R is 1 for classification problems, 5 for regression

problems. It seems to work well in many applications.[Ber08] However, it could seen critical, if the data set has small number of instances. Trees in random forest would have only one node , if the data set has 5 or less instances. Therefore node size is set in this work to 2 for the data sets, which has 5 or less instances, otherwise 5.

Number of features sampled (m_{ytr})

In random forest the features are assembled before searching best split. m_{ytr} is the number of features to sample in random subspace method, which is mentioned in the second chapter. It is a key parameter to optimize accuracy of random forest.

Breiman suggested the square root of the number of features for classification problems and one third of the number of features for regression problems. They are also default values in random forest implementation in R. The result can be confirmed after trying a few more and a few less than default value.

There is a function in the R implementation of random forest, which tunes the parameter number of features sampled, which is called tuneRF. It creates random forest with default value of m_ytr and calculates the OOB error. In next step, the M_{ytr} increases and the OBB error is calculated again. If there is a improvement between the OOB errors, the recursive algorithm repeats the same steps again, until the improvement is not desirable. In this thesis, tuneRF function is used to optimize the parameter m_{ytr}. The optimal m_{ytr} is searched before each call of random forest.

3.4 ENSEMBLE MODELS

In the second chapter the decision trees are as and built to random forest to achieve better predictive performance. Random forests and linear model can be assembled to improve their results. In this section, two different perspective will be shown. First, the bias correction of random forest is discussed based on two papers, [ZL12] and [Xu13]. Secondly, ensemble of the results from two different models are discussed. The result will be presented in Chapter 4.

3.4.1 Bias Correction in Random Forest

Bias-variance trade off in supervised learning

The bias is the difference between the real and the estimated value of the model. It is a systematic error in the model estimation, which is caused by inaccurate data in the building of model component. A high bias means that the model does not have enough or proper information to learn from the the training data to predict the new case. It is called under fitting. For instance, linear regression for two sets, which have quadratic relationship will be quite off the mark.

A high variance occurs in the exact opposite case of overfitting, which takes place when a very complex model is too strongly adapted to the training sample. The Variance indicates how different the model act with different training data. Thus, if a small change in the training data leads to a completely different model, the model is unstable. The variance leads to a mistake because

the parameters from the training data are poorly estimated. For instance if a data point in linear regression is changed to the extreme outliers, linear regression produces completely different line.

The dilemma is that the bias decreases during increasing complexity of the model, while the variance increases.

Bias and Variance in Random Forests

In the second chapter the bootstrap aggregation has been explained. The bootstrap data sets have the same size as the original data set, but they may contain the same instance several times. It is shown in the second chapter, that ca. 37 percent of the data is missed in the bootstrap data set. Therefore, bootstrapping reduces the bias, but increases the variance. Aggregating method summarizes the models of several bootstrap data set together by averaging results. This reduces the variance.[Bre96]

Random forest can be interpreted as an adaptively k-nearest neighbor approach.[LJ02] The random forest uses bootstrap aggregating method, which creates different data set from the original data set for each tree, and average the results of trees in the forest at the end. By this averaging procedure random forests create a weighted neighborhood scheme from the training case.

Lin and Jeon prove this relationship between random forest and nearest neighbor. [LJ02] They showed that random forest finds the importance of the features with the help of neighborhood proximity. This explanation helps to understand the possibility of bias in random forests. Especially if the test case is totally different than the training cases, random forest uses the nearest neighbor to estimate the new value. That causes bias in random forest.[Xu13]

Linear models use the regression coefficients to determine the weight of each independent variable. It makes it possible to assign negative weights in some cases. But random forests can not use negative weights due to the tree structure. Therefore bias can be occurred in the random forests.[Xu13]

The random forests reduce the variance of the regression compared to a single decision tree, while leaving the bias unchanged.[ZL12] Bootstrap Aggregation reduces prediction variance but has a little impact on reducing prediction bias.[Xu13]

Two different methods to correct the bias in random forests are analyzed in this thesis.

First method is proposed by Zhang and Lu (2012) [ZL12]. They assume a linear relationship between the real value and estimated value. After random forest estimates the values in the training case, linear model estimates the linear relationship between the result of random forest and the real result in the OOB data.

$$Y = w_0 + b_0 * Y_0$$

where Y is the real value, Y_0 is the estimated value which results out of random forest and two linear model components : intercept w_0 and coefficient b_0

This method is implemented in the R package randomForest, but it is experimental in version

4.6.7. In Chapter 4 this method is analyzed and compared with second method, which is suggested by Ruo Xu [Xu13] to reduce bias.

Ruo Xu proposed that this bias correction method with linear model does not work often and suggests an alternative bias correction method. He suggests to use second random forest estimator to estimate bias in random forest. [Xu13]

Firstly, each tree in first random forest estimates the response value for bootstrap data set and the bias is calculated with OOB data. After this calculation, second random forest estimates the bias from the features in OOB data. In the end there are two random forest models, first model predicts response variable from features and second model estimates the bias of first random forest from features.

Ruo Xu compared both methods to correct bias in random forest in [Xu13] and achieved better accuracy compared to bias correction method with linear model. But he shows as well, that both bias correction methods predict better than standard random forest. In chapter 4 both approaches will be tested in time series and compared for the following tree cases:

- standard random forest
- random forest and bias correction with linear model
- random forest and bias correction with random forest

3.4.2 Ensemble of Models

Perlich proved that logistic regression and decision trees act as a complement to each other. [PPS03] It also makes possible to improve predictive performance to combine linear model and random forest. The estimation of decision tree and linear regression are combined with bagging method and analyzed in [KKZ06]. In this thesis the results of each model will be equal-weighted combined without bagging. The effect of bagging and this approach are presented in Chapter 4.

4 EXPERIMENTAL METHODOLOGY

In Chapter 3, the theories for the building models have been analyzed. In this chapter the effects of these theoretical decisions are presented. Three different experiments are performed to test the decisions. The theoretical analysis is done according to the initial data. The experiments in this chapter are performed in the remaining 22 months. The models are built according to cross sectional forecasting as before.

4.1 EVALUATION

In the second chapter the task is introduced. Task is to predict sales units for each item for the following month according to cross sectional forecasting. The monthly predictions will be evaluated in three different perspectives:

- sales number for each item in the month
- sales number for each brand in the month
- sales number in the month

Symmetric Absolute Percentage Error

For this three different perspectives, the quality of the forecasting models should be obtained to compare with each other. For this purpose symmetric absolute percentage error is used. The symmetric absolute percentage error (sAPE) is defined by:

$$sAPE = \frac{Y - \hat{Y}}{\frac{Y+\hat{Y}}{2}}$$

where Y is the real value and $\hat{Y}$ is the estimated value of the instance.

The difference between the real value and estimated value is divided by half of the sum of the real and estimated value. The SAPE is defined within the limits of $0 \geq SAPE \geq 200$. A smaller value is better than a large one.

In each model, the SAPE for each month are calculated and averaged. This measure summarizes the total deviation of the accuracy in the model.

4.2 EXPERIMENTAL SETUP

To perform the experiments, a standard home desktop computer is used, which has 4 core processor with 2.9 Ghz and 4GB memory.

For the experiments parallelization process is not used. However, for big data sets, parallelization reduces the execution time of experiments. Random forest has a big advantage for parallelization. Since all trees are independently grown, it it possible to grow trees in parallel.

In this work tree experiments are implemented in the R programming language. The R programming language is a freely available language and an interactive environment for statistical computing and graphics, which provides a wide variety of statistical and graphical techniques.[1] Comprehensive R Archive Network supports R with third party package. Here, randomForest package and data.table package are used. The data.table package can handle data operations such as ordered join, first aggregations, add, modify, delete etc. in a large data sets. The randomForest package provides an R support of the Fortran programs by Breiman and Cutler, which is avaible at http://www.stat.berkeley.edu/users/breiman/. [LW02] Random forest implementation is also available in the package party , which contains computational tools for recursive partitioning. The function cforest() is used for the the implementation of random forests.

Three years market data is available for the experiments. They are stored in a PostgreSQL database, to which R can connect easily with R package RPostgreSQL. The predicted values are stored as a simple comma-separated values format to make future analysis possible.

4.3 EXPERIMENTS

In this section, three experiments are implemented and performed. The results are discussed to check the theoretical decisions in the second chapter.

There are three different question from Chapter 3 :

- Which feature selection method should be used in random forest and is it necessary ?
- How effective are the bias correction methods in time series ?
- Is it possible to improve accuracy with ensemble of linear model and random forest ?

[1] http://cran.r-project.org/

Boxplot is used to represent the results of the experiments. The function of boxplot is giving a quick idea about the data set, i.e. how they are distributed over this range. Therefore, all the values of the so-called five-point summary, which is the median, the two quartiles and the two extreme values are shown. Here, the mean value of each data set is written as well in each box.

The parameters of random forest are discussed in Chapter 3 and in these experiments the followings parameters are used:

- Number of trees(ntree) :=1000
- Number of features sampled (m_{ytr}): the function tuneRF in the package randomForest is used to find optimal parameter.
- Node size (nodesize): The default value is used, 5.

The Figure 4.1 shows an overview of three experiments. Firstly, the effect of features selection will be determined in the experiment 1. In experiment 2, the improvement of the bias correction will be measured and in the end, experiment 3 will demonstrate the effect of the ensemble of estimations of random forest and linear model.

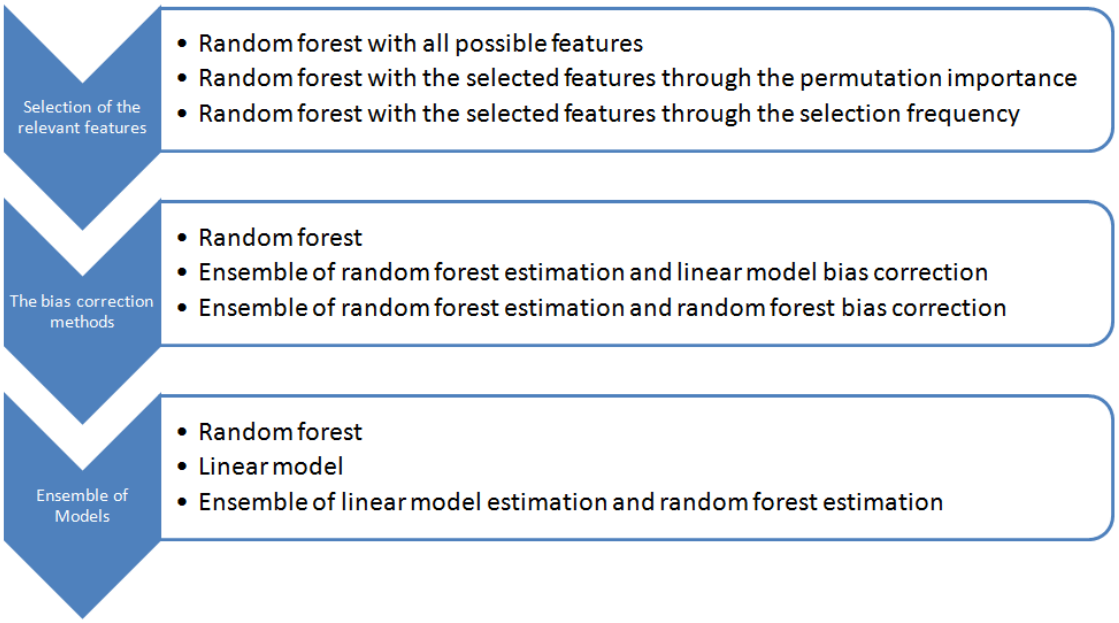


Figure 4.1: The overview of three experiments

4.3.1 Selection of the relevant features (Experiment 1)

In Chapter 3 two methods are used to determine the relevance of features. The methods are performed for each month in the initial data and the results are averaged.

The results are represented in Figure 3.2 and 3.5. The five features have a significantly higher selection frequency than the others: sales units, stock new units, purchase units, property 5 and property 6. However, only three of them are marked as relevant in the permutation accuracy method: sales units, stock new units and purchase units.

Random forest select the 'best' feature to split at each division with the greedy search algorithm. At first appearance, it is possible to think that the irrelevant features do not influence the accuracy. But the greedy search algorithm finds the best split at the time of choice, the globally best tree is not guaranteed. If more irrelevant features are searched in the algorithm, then the chance of "not globally optimal" tree is high. Random forests search in each division in each tree for the best value for each feature to split and compare the best values of all features and choose one. The execution time increases, if the number of features is high. Besides, identifying relevant features make the predictions more interpretable compared to some black-box model.

In this experiment there are three different random forest model:

- Model 1: Random forest with all possible features
- Model 2: Random forest with the selected features through the selection frequency
- Model 3: Random forest with the selected features through the permutation importance

These three models are evaluated in the case study data. The results are in Figure 4.2 illustrated. As shown Figure 4.2, the model 3 has achieved better accuracy in all three perspectives. The averaged sAPE of the Model 3 is the lowest in all three perspectives and the distribution of the sAPE is also better than the other models. The two quartiles for the first perspective is significantly better than the other models.

Random forest in the model 3 contains the three features, which are contained in other models. The Model 2 contains two more features and the model 1 contains 6 more features. These features are marked as irrelevant through the permutation importance, which considers the change of OBB error. Breiman shows empirically that the generalization error can be estimated from the OOB error. This experiment have confirmed and shown that the permutation importance is a more reliable method to determine the relevant features.

The model 1 is a black-box model here. The model 1 has a worse accuracy performance than the model 3, eventhough the model 1 has the same features too, which the model 3 contains. The chance of the fail to find globally optimal tree is high, if there are more options available for the greedy search algorithm. From the experiment 1, it can be seen that choosing relevant features improves the accuracy of random forests.

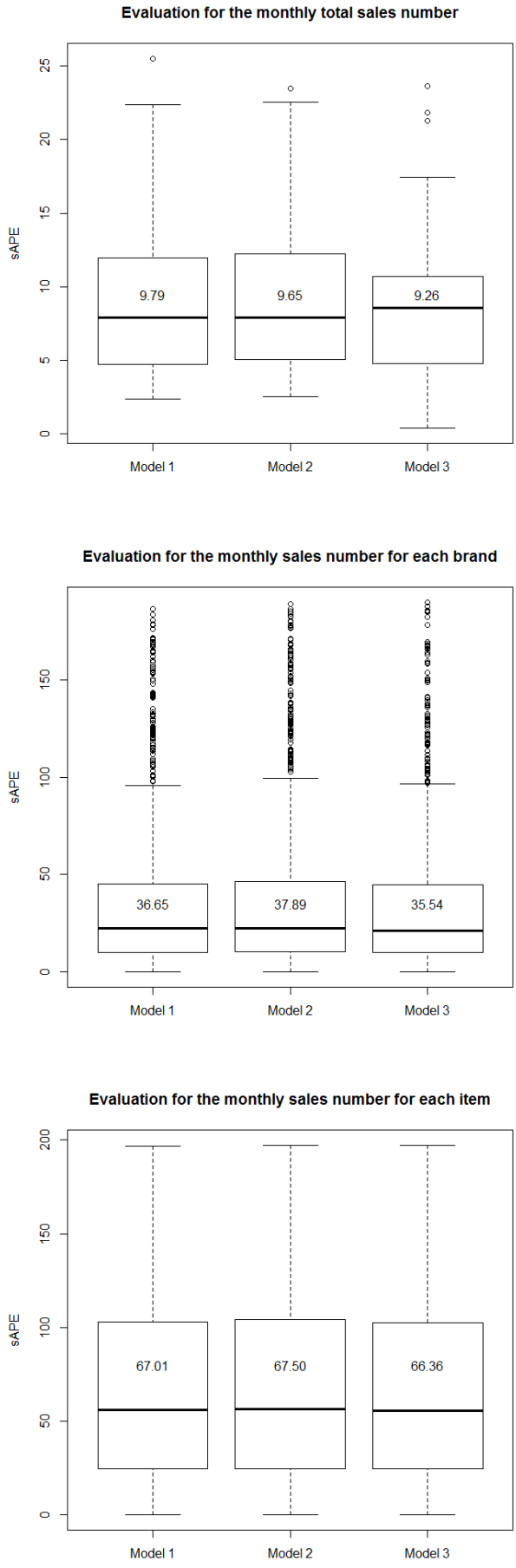


Figure 4.2: The evaluation of the three random forests with different features set in the three aspects

4.3.2 The Bias Correction Methods (Experiment 2)

In experiment 1 the relevance features are selected. In experiment 2, these features will be used and the effect of bias correction on the accuracy of random forest will be tested.

Two bias correction methods are introduced in Chapter 3. Zhang and Lu proposed a linear relationship between the estimated value and the real value. [ZL12] The bias is estimated in a linear model in their methods. This bias correction method is implemented in the package randomForest (corr.bias), but in this experiment it will not be used to compare with the bias correction method, which is developed by Ruo Xu.[Xu13] Ruo Xu proposed a second random forest to estimate bias of the first random forest.(see Chapter 3)

To apply a second bias correction in the data set, the OOB data set is required. In other words, the data, which random forest has no knowledge of, should be used to estimate the bias of random forest. For this purpose, the bootstrap aggregating method is used again. The data set is randomly sampled with replacement into two data sets, a training set and a test set. Random forest generates the model from the training set and bias correction algorithm uses the test case to estimate the bias of random forest. This procedure is iterated and the results are averaged. To find a proper iteration number, the process is repeated 10 times and 100 times.

There are three different random forest model in the experiment 2 :

- Model 4: random forest with additional bagging
- Model 5: Ensemble of random forest estimation and linear model bias correction
- Model 6: Ensemble of random forest estimation and random forest bias correction

Figure 4.3 represents the results of these three models with 100 iterations. Firstly, the model 3 from the experiment 1 and the model 4 can be compared. The only difference between both models is the bootstrap aggregating is applied before random forest in model 4. This approach have not improved the accuracy of random forest. To determine the influence of iteration, the bootstrapping is applied 10 times and 100 times. The increasing number of iteration have not changed the results significantly.

As can be seen from the Figure 4.3 both of the bias correction methods have not resulted in an improvement in the accuracy of random forest except for the evaluation for the monthly sales number for each brand. The improvement in this perspective is 0.8, which can be regarded as an insignificant change.

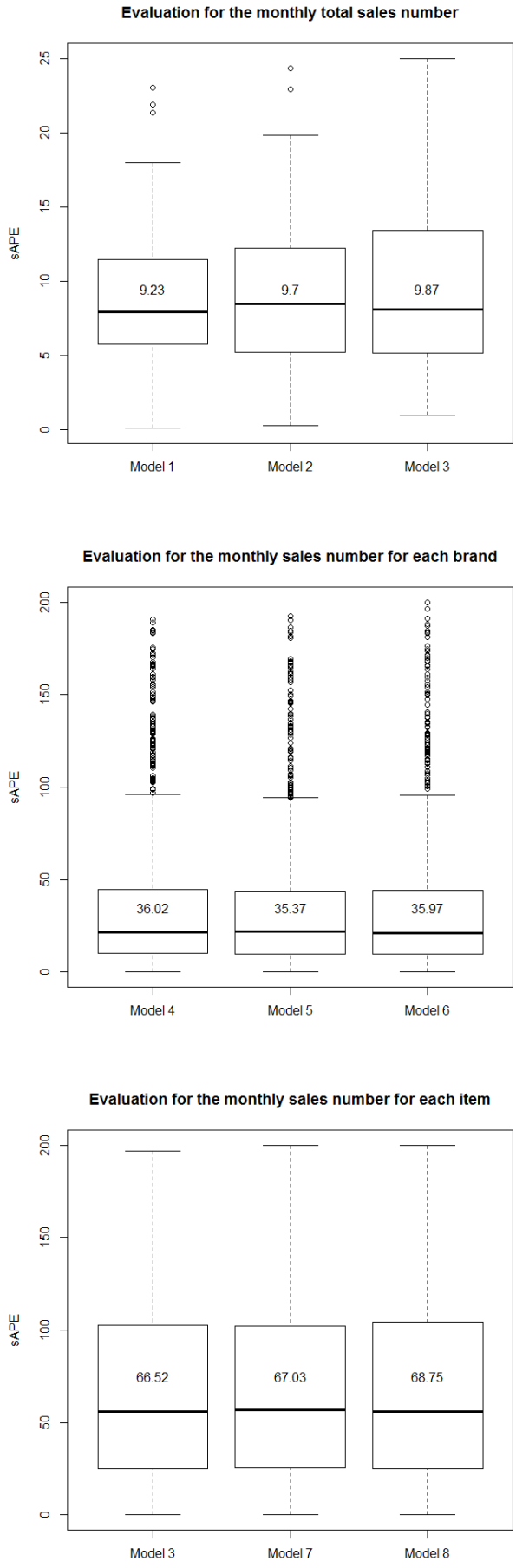


Figure 4.3: The evaluation of the three different random forest models in the three aspects(100 times)

4.3.3 Ensemble of Models (Experiment 3)

The bootstrap aggregating method and bias correction methods have not achieved to improve the accuracy of random forest. In the experiment 3 the bias correction methods are not used. Therefore, there is no need to use the bootstrap aggregating method. The random forest from model 3 can be used in this experiment to combine with the linear model.

Perlich proved that logistic regression and decision trees act as a complement to each other. [PPS03] And linear regression and regression tree [Bre01] with the bagging method [Bre96] are assembled in the article "Bagged Averaging of Regression Models"[KKZ06]. Here, the results of the experiment 2 have eliminated the bagging method. The estimations of the linear model and the random forest are equal weighted combined.

To sum up, the following models are considered in this experiment:

- Model 3: Random forest
- Model 8: Linear model
- Model 9: Ensemble of random forest estimation and linear method estimation

As shown in Figure 4.4, the ensemble of the random forest estimation and the linear model estimation improves the accuracy of the model. The averaged sAPE of the model 8 is the best in all evaluation perspective except for the evaluation for the monthly total sales number. In this evaluation perspective, the linear model has the best averaged accuracy with 8.93. And the model 8 is only 0.09 sAPE worse. The difference in this case is not notable compared to the difference in the evaluation for the monthly sales number for each item, which is 2.18 sAPE.

The experiment 3 shows that the ensemble of linear model and random forest estimates more precisely than either the linear model alone, or the random forest alone.

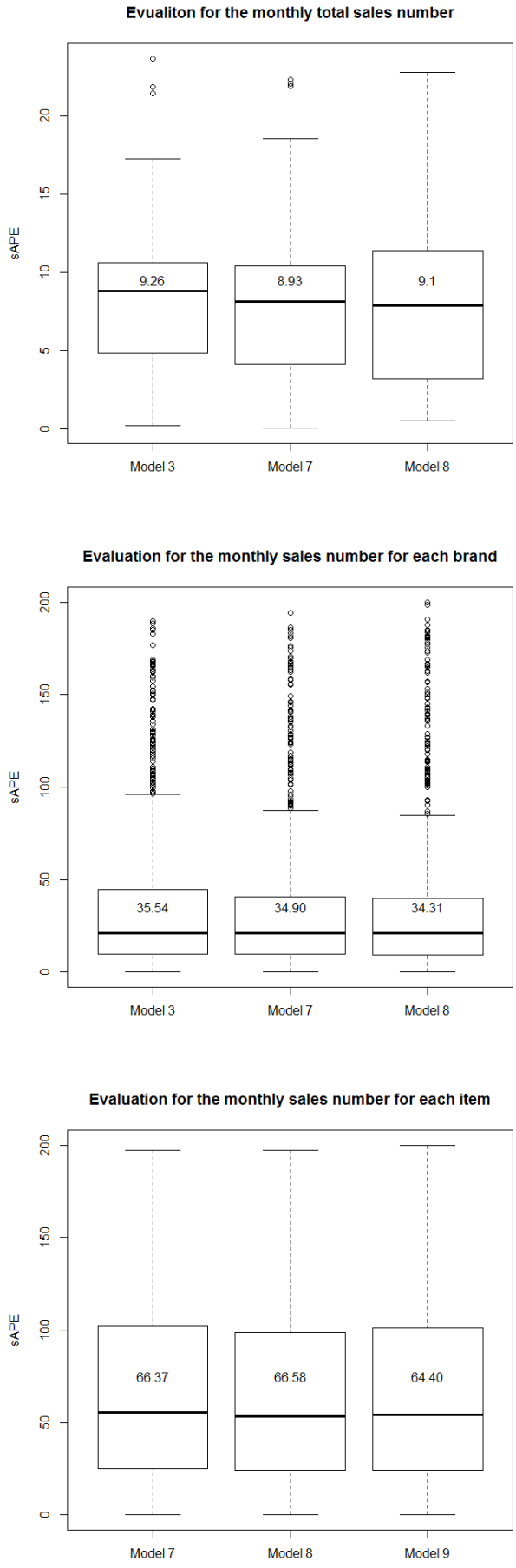


Figure 4.4: The evaluation of the three different models in the three aspects

5 CONCLUSION AND FUTURE WORK

With this work, eight different models based on two machine learning algorithms are studied and used to forecast time series. The best results were obtained using an ensemble of random forest and linear model. The ensemble of linear model and random forest has a better accuracy than either the linear model, or the random forest alone. This result is in a good agreement with the study "Bagged Averaging of Regression Models" [KKZ06], which has shown that the ensemble of a decision tree and a regression model with bootstrap aggregation performs better in most cases. Future work may involve to assembling different machine learning approaches such as neural networks and SVM in order to improve the accuracy of the model.

The results of the example 1 has shown that the selection of relevant features has a positive influence to the accuracy of random forest. Two different criteria are studied to determine the relevant features; the selection frequency and the permutation accuracy. From the outcome of our investigation it is possible to conclude that the permutation accuracy is a more reliable criterion to determine the relevant features for random forest compared to the selection frequency.

A further bootstrap aggregating and the bias correction are also conducted in this work. The further bootstrap aggregation has not improved the accuracy of the random forest. The results of random forest with the second bootstrap aggregating method have been worsened insignificantly. To estimate the bias, two different technique based on two different machine learning approaches, linear model and random forest, are used. The bias correction methods have not yielded any improvement to random forest. This analysis does not enable us to determine the general assertion for the bias correction, because the results are dependant only on the present data set. More experiments with different time series should be performed to verify the results.

The averaged symmetric absolute percentage error of the random forest black box model (the model 1) has improved ca. 0.8-2.6 points by using the relevant features and assembling with linear model. Based on these results, it can be concluded that the research into the improvements to random forest has been fairly successful.

The estimations of random forest and linear model are equal-weighted assembled, which has given the best result. The ensemble with different ratios can be seen as a future work to improve the accuracy of the predictive model. For this purpose, further research into characterizing the accuracies of both models is worthwhile.

LIST OF FIGURES

BIBLIOGRAPHY

[AG97] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[Ber08] Richard A Berk. *Statistical learning from a regression perspective*. Springer, 2008.

[BFOS84] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

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

[Ho95] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[Ho98] Tin Kam Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

[HTF+09] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[KKZ06] Sotiris B Kotsiantis, Dimitris Kanellopoulos, and Ioannis D Zaharakis. Bagged averaging of regression models. In *Artificial Intelligence Applications and Innovations*, pages 53–60. Springer, 2006.

[KT06] Manish Kumar and M Thenmozhi. Forecasting stock index movement: A comparison of support vector machines and random forest. 2006.

[LJ02] Yi Lin and Yongho Jeon. Random forest and adaptive nearest neighbors. Technical report, Citeseer, 2002.

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

[Mit97] Tom M Mitchell. Machine learning. wcb, 1997.

[POGM07] David S Palmer, Noel M O'Boyle, Robert C Glen, and John BO Mitchell. Random forest models to predict aqueous solubility. *Journal of chemical information and modeling*, 47(1):150–158, 2007.

[PPS03] Claudia Perlich, Foster Provost, and Jeffrey S Simonoff. Tree induction vs. logistic regression: a learning-curve analysis. *The Journal of Machine Learning Research*, 4:211–255, 2003.

[Sau99] Willi Sauerbrei. The use of resampling methods to simplify regression models in medical statistics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 48(3):313–329, 1999.

[SBK+08] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC bioinformatics*, 9(1):307, 2008.

[SRB07] Willi Sauerbrei, Patrick Royston, and Harald Binder. Selection of important variables and determination of functional form for continuous predictors in multivariable model building. *Statistics in medicine*, 26(30):5512–5528, 2007.

[SSB+04] Tao Shi, David Seligson, Arie S Belldegrun, Aarno Palotie, and Steve Horvath. Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma. *Modern Pathology*, 18(4):547–557, 2004.

[Xu13] Ruo Xu. Improvements to random forest methodology. 2013.

[ZL12] Guoyi Zhang and Yan Lu. Bias-corrected random forests in regression. *Journal of Applied Statistics*, 39(1):151–160, 2012.