# Cost sensitive model selection for food sales prediction

Vasanthapriyan, S.

*Award date:*
2010

# COST SENSITIVE MODEL SELECTION FOR FOOD SALES PREDICTION

**S. Vasanthapriyan**
**(0641859)**

TECHNISCHE UNIVERSITEIT EINDHOVEN

# COST SENSITIVE MODEL SELECTION FOR FOOD SALES PREDICTION

**by**
**S. Vasanthapriyan**
**(0641859)**

**Supervisor:** dr. Mykola Pechenizkiy
**Tutor:** drs. Jorn Bakker

A thesis submitted in partial fulfillment for the
degree of Masters of Science

in
Computer Science and Engineering

February 2010

Examination committee:

dr. Mykola Pechenizkiy (TU Eindhoven)
drs. Jorn Bakker (TU Eindhoven)
Prof. Paul De Bra (TU Eindhoven)
dr. Jaap van der Woude (TU Eindhoven)

# *Abstract*

The ability to forecast consumer demand accurately is of great importance to companies like food sales retail and whole sales. These companies are very interested in the most accurate way to predict the sales of especially short shelf life products and seasonal products. Bad predictions often lead to a stock shortage at the start of the season and rest stock at the end of the season, causing additional costs. Thus, both shortage and surplus of goods can lead to loss of income for the company. In the current evaluation methods these costs are not taken into consideration. In this paper, we use a cost function to find these associated costs. We use these costs as an input to the ensemble of learners which employs dynamic integration of classifiers to select the best model for final prediction.

# *Acknowledgements*

I would like to thank Mykola Pechenizkiy, my supervisor for his many suggestions and guidance during this Master thesis work. I am also thankful to Jorn Bakker for his valuable input and motivation that encouraged me to always do better in this work. Also I thank the Database and Hypermedia research Group members for their valuable comments and suggestions on my research.

Especially I would like to thank Prof. Jaap van der Woude, the Master's coordinator and study advisor for Computer Science and Engineering program at TU/e, giving me valuable advices and encourages throughout my stay in Eindhoven.

I should also mention the Nuffic Fellowship which was awarded to me for the period 2007-2009 and was crucial to the successful completion of this work.

I would like to thank Maggy de Wert, who gave extra motivations and strengths on formatting this report.

Finally, I would like to express my gratitude to my mother for her constant support throughout my life.

*S. Vasanthapriyan*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Sales forecast is a prediction based on past sales and an analysis of expected market conditions. Moreover, it involves predicting the future outcome of various business decisions and also includes demand forecasting, trend forecasting, and sales and marketing plans. Accurate sales forecasting is certainly an expensive way to meet the aforementioned goals. But it leads to improved customer service, reduced loss of sales and reduced product returns and more efficient production planning. Accurate forecasts is therefore useful for food sales companies due to the short shelf-life of many food products and the dependency on seasonal changes[3].

In real life, there are seasonal influences on the products which make the prediction hard. Most of the time, promotions, religious activities (for example Christmas), sales of complementary products and cultural holidays cause rises and drops in the sales pattern. Seasonal changes are often difficult to predict. This might lead to a stock shortage at the start of the season and rest stock at the end of the season causing additional costs. Thus, both shortage and surplus of goods can lead to loss of income for the company. Predictions are also influenced by price, quality, location and economic conditions. In this research however we focus only on seasonal influences.

In sales management, forecasts of future demand are gathered to select an efficient inventory level, in order to reduce keeping cost and reduce keeping time. When these costs occur one can usually determine the cost of having made the error on prediction, and the amount of this cost will usually increase as the magnitude of the error increases. Moreover, this increased amount of costs will affect the company's food sales in future.

Because of all these reasons, sales forecasting has a crucial role in the decision support system of a commercial enterprise. Obtaining effective sales forecasts can help the decision-maker to determine the amount of products to be stored on shelf. Therefore the food sales companies are highly interested in sales forecasting.

To handle above mentioned costs in data mining and machine learning settings we introduce cost sensitive model selection [7][16], which will minimize the total costs. The goal of cost sensitive model selection is to build a classifier not only to minimize the expected number or proportion of mistakes, but also to minimize the expected misclassification costs.

## 1.2 Cost Sensitive Learning

Data Mining, also known as Knowledge Discovery in Databases (KDD)[6], is the process of discovering meaningful new correlations, patterns, and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques. Classification is one of the techniques in data mining settings and machine learning which is used to predict group membership for data instances.

Furthermore, the most of these classification algorithms such as naive bayes, decision trees and neural networks ignore different misclassification errors: for example they implicitly assume that all misclassification errors are equal. Conventional algorithms that operate with symmetric loss functions do not differentiate between under-prediction versus over-prediction or late-prediction versus early-prediction. Therefore each type of prediction error is weighted equally. However in the real world this assumption is not true. The differences between different misclassification errors can be quite large.

For simplicity, to motivate our primary goal of cost-sensitive learning, let's first consider the binary classification setting[16]. In traditional learning algorithms, which aim to maximize accuracy, positive and negative examples are treated as equally important and therefore do not always produce a satisfactory classifier under unequal cost conditions. Mathematically speaking, most classification algorithms attempt to minimize the expected number of misclassification errors. For example, in medical diagnosis[16] misclassification of a true patient to a false patient leads to lose his/her life because of the delay in the correct diagnosis and treatment. Usually, the rare class is the positive class and it is often more expensive to misclassify an actual positive class into negative rather than actual negative into positive. Therefore, the cost of False Negative (FN) is usually larger than that of False Positive (FP).

It is therefore imperative to implement effective cost-sensitive adjustments, which take this inherent cost associated with specific types of prediction errors into account during the learning process. Ultimately, cost-sensitive applications demand learning, with the ultimate goal of minimizing the expected cost of misclassifications.

## 1.3 Problem/Objectives

All the different stages of the KDD process are affected by economic utility (such an example is the cost). This economic utility impacts the assessment of decisions made, based on the learned knowledge. Utility-Based Data Mining (UBDM)[12] addresses this challenge by taking into account the complex economic environments in which data mining occurs. Moreover, it encompasses the research work in cost-sensitive learning (which considers the costs and benefits associated with using the learned knowledge, and how these costs and benefits should be factored into the data mining process), and active learning.

In our research too we addressed the challenges pertaining to the incorporation of such utility factors in real-world domains. This defines our research hypothesis which states that the utility optimization is better than error optimization in the context of food sales prediction(minimizing the expected cost is a mechanism that implements this idea).

Our focus in this research starts with studying how the food sales prediction is currently operated in a food sales company, and how we can improve predictions for a large number of products that are interesting for the company both from sales pattern perspective as well as regarding their relative importance based on the amount of sales. At the moment, the prediction is calculated by using a six weeks moving average.

We introduce the ensemble learning approach, which generates multiple base models using traditional machine learning algorithms and combine them into an ensemble model. This learning approach often demonstrated significantly better performance than a single model. In addition to this, we introduce sliding window approach which moves across the time series. Moreover, we apply dynamic integration methods at instance level and select the best model which is used for final prediction[14].

We try to show that the use of ensemble learning approach is better than the six weeks moving average approach. The latter approach works better in situations in which the pattern is with sudden raises and drops of the sales(also known as noisy data), due to some form of seasonal behavior. We also try to show that there are situations at the instance level where either ensembles of classifiers or six weeks moving average are employed across the series.

## 1.4 Related Work

Highly non-uniform misclassification costs are very common in a variety of challenging real-world data mining problems. This issue has been researched recently with the techniques known as cost-sensitive learning in the machine learning and data mining communities. Typical examples of the past research were medical diagnosis [8][10], fraud detection[5], and network troubleshooting.

The research work performed in the food sales prediction by Meulstee[9], presented ensemble learning approach with the dynamic integration of classifiers for handling seasonal changes in food sales. This technique will be used as our base idea to extend the use of ensemble approach for cost sensitive model selection. In addition, the experiments shown in[14] show that ensemble of classifiers with dynamic integration is the best way to increase the accuracy in the medical field.

## 1.5 Approach and Results

Our solution to cost sensitive model selection is based on a generic knowledge discovery framework with an ensemble of classifiers as a data mining component as depicted in Figure 1.1. We implement our framework using RapidMiner[1], Java and Matlab.

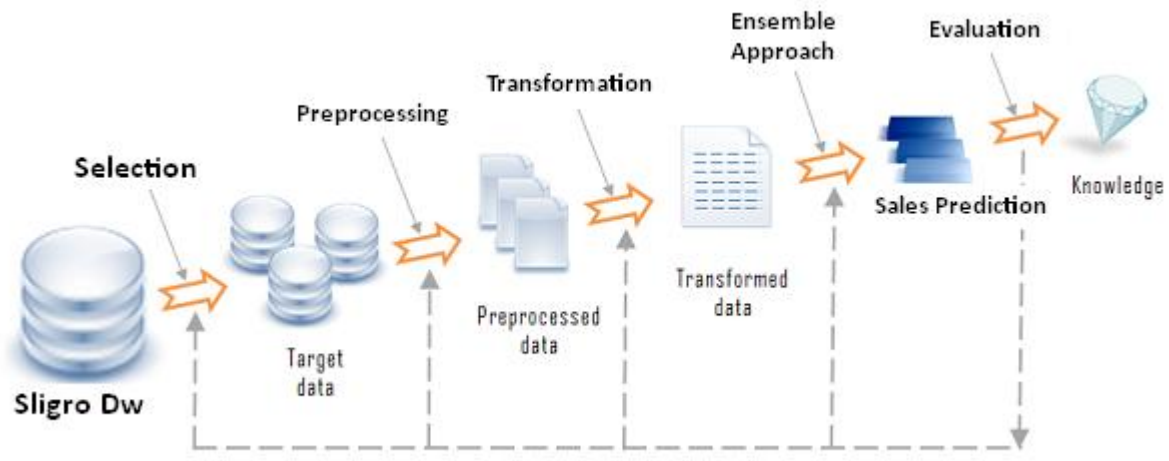


FIGURE 1.1: Data mining is the core of Knowledge Discovery Process [6]

The high level overview of the classification task is illustrated in Figure 1.2. The training data set consists of labeled training instances from Sligro Data Warehouse. For the learning algorithms we have selected nine algorithms (including moving average) which are available in RapidMiner repository. Then all models are learned. Next, the best model is selected and applied to the test instance.

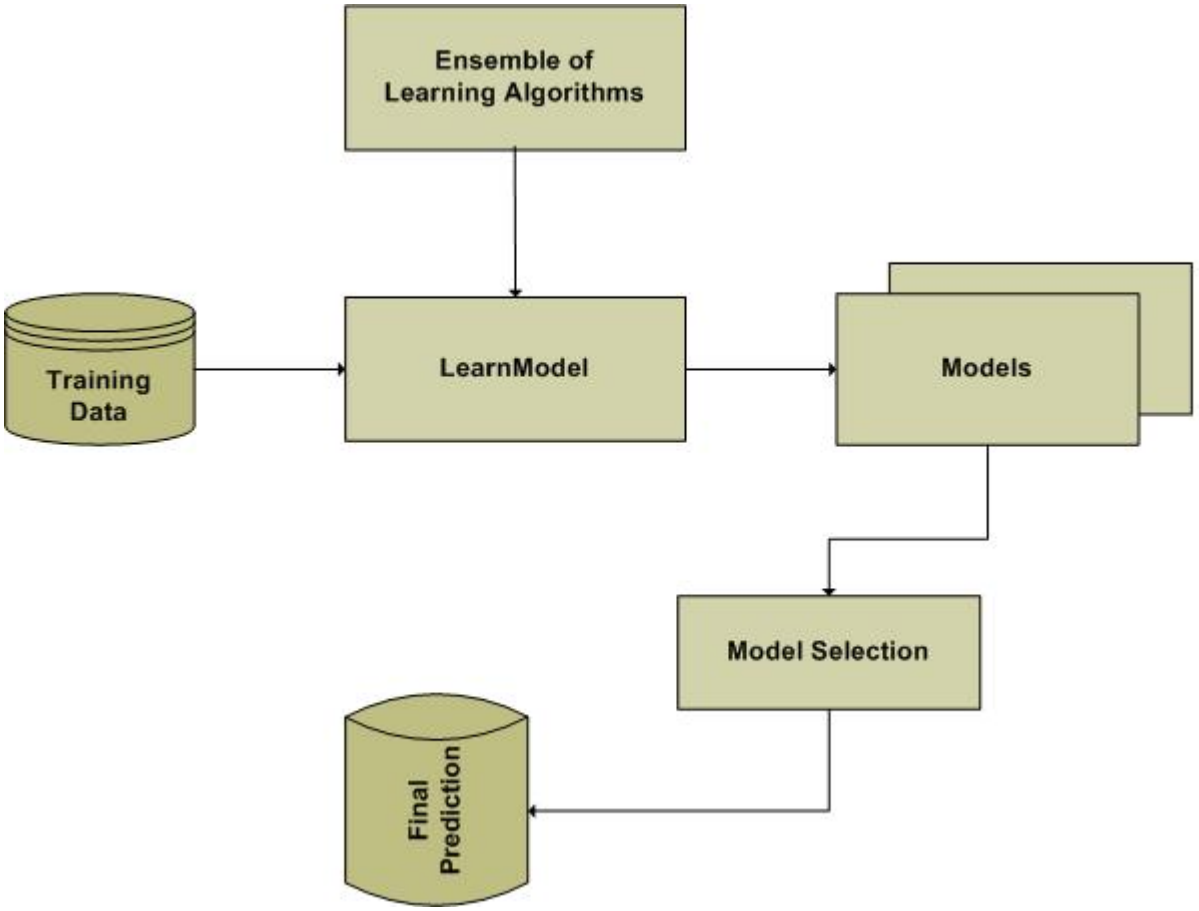


FIGURE 1.2: High level overview of the food sales prediction task

We validate our statement with the real world sales data from Sligro Food Group BV, a large wholesales seller of food and beverages in The Netherlands.

From the experimental study we suspect that model voting performs better than model weighting with respect to cost. That is, we saw that in 348 (66.14%) out of 524 experiments, cost sensitive method performs better than non cost sensitive method, when model voting is selected as the integration method.

## 1.6 Organization of Study

This thesis is organized as follows: in chapter 2, we introduce the problem description in detail. The cost function is described in chapter 3. In chapter 4 we discuss feature generation, feature selection, model learning and model selection. Chapter 5 describes the forecast experiments which includes data arrangements and experimental setup. Chapter 6 reports the forecasting results evaluations and discussions. Finally chapter 7 concludes with conclusions and recommendations.

# Chapter 2

# Problem Description

For a formal setting of the cost-sensitive classification problem, let us first start with some associated notations. Given a supervised learning problem, a set of labeled examples or training data $(X_i, y)$ is available to us, where $X_i$ is a vector of attributes and $y$ is the class label of $X_i$, belonging a set of class label $Y = \{1, 2, 3..., j\}$. We also assume that the training data are drawn from an unknown probability distribution $P(X, y)$. The objective of a learning algorithm is to find a model or a hypothesis $h$ which will be able to map correctly a higher proportion of unlabeled examples drawn from the same distribution. Alternatively, if an incorrect mapping is considered to be a cost (or a loss), the major goal of building a classifier can be treated as minimizing the total expected cost or loss $e$ of the hypothesis $h$.

$$e(h) = \sum_{X,y} P(X, y)C(h(X), y)$$

Where $C(h(X), y)$ is the cost function representing the loss caused by an instance $< X, y >$. $e(h)$ is calculated as the sum of the costs or losses of all the individual decisions. Here, $P(X, y)$ is the probability estimation of classifying an instance into class $y$.

Note that in traditional classification tasks, the loss (or cost) function $C(h(X), y)$ is 1 when $h(X) \neq y$ and 0 otherwise.

Therefore, these classifiers are also known as minimizing the expected 0/1 loss. Their underlying assumption is that misclassification errors have the same cost. However, in this research work we are merely interested in the cost-sensitive classification problems with non-uniform costs, i.e. different classification errors having unequal costs.

In inventory management of food sales outlets as shown in the example below, keeping units of consumer goods in stock or on shelf, in order to satisfy customer demand, the effect of overstocking a product may induce increased stock keeping costs for a single period, whereas the costs of under stocking will lead to lost sales revenue and dissatisfied customers.

More precisely, if we look at the four cases (early prediction, late prediction, under prediction and over prediction) as shown in the following graphs for a particular product, occurring costs are different which are explained below. Clear comparison of these costs

can be done in the section cost function evaluation in the next chapter. In depicted Figures 2.1 and 2.2, X-axis is meant by number of weeks and Y-axis is meant by number of units sold.

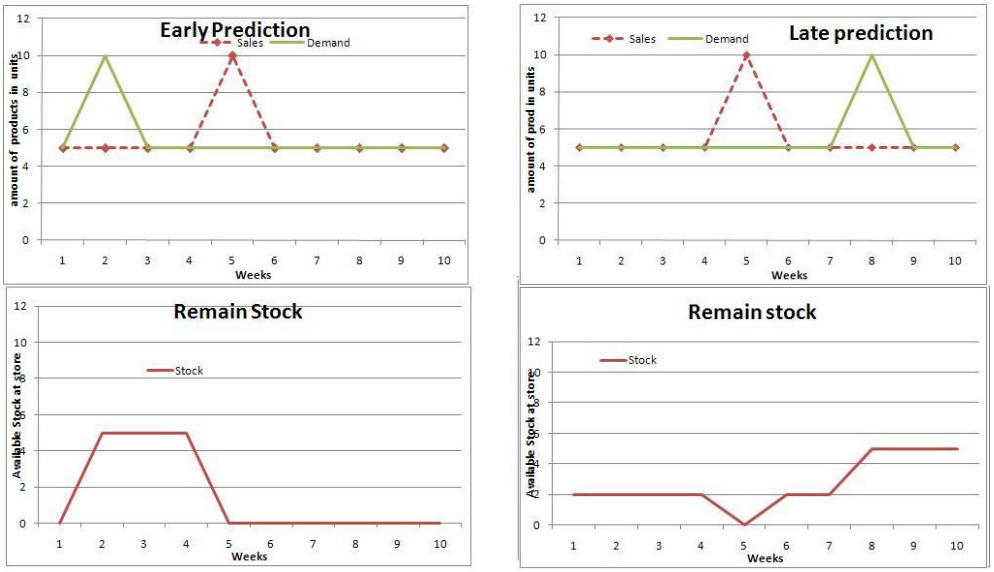


FIGURE 2.1: Early prediction versus late prediction

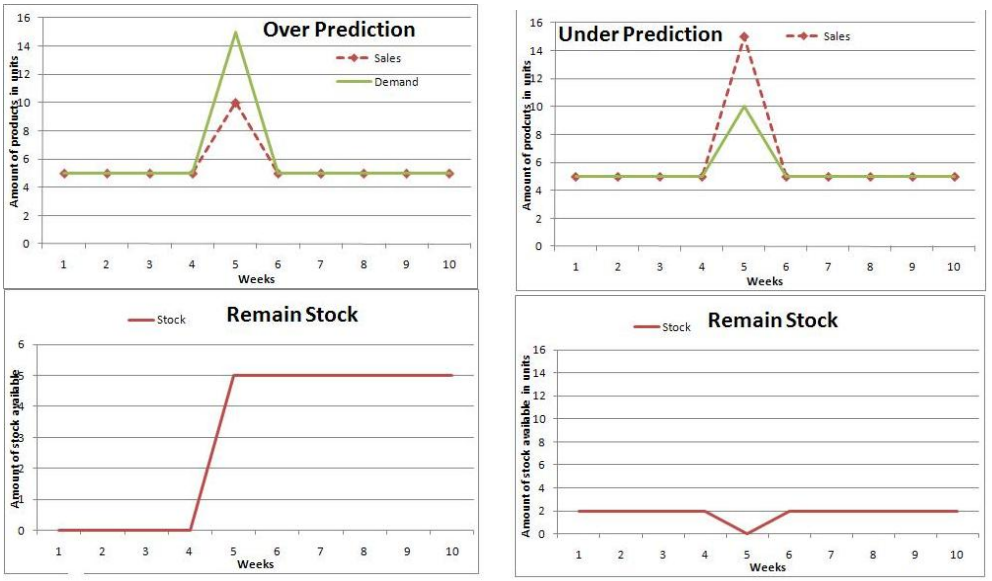


FIGURE 2.2: Over prediction versus under prediction

If you look at Figure 2.1, when early prediction occurs (drawn in green/thick line), there is a possibility to have extra stock (corresponding stock level is depicted in the below graph), which has to be stored and maintained in the store. As a result, the associated keeping cost is added. In addition, if the stored stock reaches it's perish date, then the obsolete cost will be added. Contrast to early prediction, in late prediction we will also see a case of remain stock, also causing keeping cost, and again there will be a possibility of perish items of those stored items. In these two cases, late prediction has the highest

possibility of storing it or perishing it, in comparison with the early prediction. Thus, we can see that the costs are different and therefore it must be considered.

Furthermore, Figure 2.2 depicts the over prediction versus under prediction. Whenever over prediction happens, we see more stock on shelves (look at the corresponding remain stock). Therefore we need to keep them, which causes two types of costs as we explained earlier: keeping cost and perish cost. But for under prediction, we don't have sufficient stock (look at the corresponding remain stock), and therefore the loss of sales cost is added. This clearly shows that the costs for both cases are different.

To reflect this in our research, an asymmetric cost function(in chapter 3), which is made up of stock dependent and storage-time dependent, is developed and employed as the objective function for the ensemble of classifiers approach for the model selection, deriving superior forecasts and a cost efficient decision.

# Chapter 3

# Cost Function

## 3.1 Generalized Cost Function

While academic research has continued to improve the misclassification error rate, in particular, one important problem is that there are different kinds of errors which have different costs. For example, the analysis we did in the problem description chapter (Figures 2.1 and 2.2), in the food sales perspective, the cost of under prediction always differ the over prediction of sales. Moreover, this is true for late prediction versus early prediction too. The relative importance of different kinds of errors can be represented by a cost function.

In this research, we consider the sales policy for the products causing stock dependent and storage-time dependent keeping cost. It's not a point of sale. It is dependent on the past sales. Moreover, at the end of the sales, if there are more stock remains at the stores, we need to store them and maintain them. This means there are some associated costs related, i.e. keeping cost for that product. Thus, all these have to be reflected in the cost function.

The generalized cost function can be given as

$$C_t = f(e_t, s_t)$$

$f(e_t, s_t)$ is a function that calculates the cost at time $t$. The error $e_t$ can be calculated by $e_t = y_t - p_t$ . That is the difference between the actual sales $y_t$ and the predicted value $p_t$ at time $t$. As was explained earlier, the stock is dependent on previous sales. Available stock at time $t$ can be given as $s_t = s_{t-1} - y_{t-1}$. Moreover, when there is a possibility to store the stock for a period of time, it might need a place and proper maintenance etc. In this case, the keeping cost has to be added. The keeping cost can be given as $C_k = S_n * K_c$. The total number of units stored of a particular product can be given as $S_n$, where $K_c$ is the keeping cost for that product. The profit cost is calculated when the sale is done and can be given as $C_{pr} = y_t * C_p * \Omega$. The number of products can be $y_t$ and the purchase price of the product can be given as $C_p$. The profit ratio is the ratio of a product's sales divided by the purchase price of that product and denoted by $\Omega$ ($\geq 1$). Whenever the stock is purchased, it can be given as $C_{pur} = n * C_p$.

The number of products that needs to be purchased is given by $n$, where the purchase price is indicated by $C_p$.

$(C_{pur} + C_{loss})$ is used in the situations when the demand is higher than the available stock in the shelves. Then available stock will be sold. The cost associated with the loss of sales cost (which is due to no stock at the store) is given by $C_{loss} = N_{loss} * C_p * \Omega$ . The finding of the value for $N_{loss}$ will be explained later in this chapter.

To reflect all these in the given generalized cost function, we came up with a final version of the cost function in the following form as shown below.

$$f(e_t, s_t) = \begin{cases} C_K + (-C_{pr}) & if s_t \neq 0 \\ (-C_{pr}) + (C_{pur} + C_{loss}) & if s_t = 0 \end{cases}$$

When there is no stock availability at the store, the needed stock will be purchased. $C_{pur}$ is the cost occurred by purchasing new items. $C_{loss}$ is the cost because of loss of sales due to the no stock availability in the store which was explained earlier.

The purchase price of the stock always has the value in between zero and one. It is because we cannot have absolute values as we do not have domain knowledge at the moment but we would like to compare the results. Therefore, it is better to have a relative measure which means all values are given as part of one dimension for this particular scale. This applies for the keeping cost of the products too.

How sales are done, how the stock is maintained, how new items are purchased and how obsolete items are discarded are explained below.

- In the first week available stock of any product is initialized to 1.
- At the beginning, the expiration date of any product is initialized to the duration of the product. Date of expiry of the product will be updated whenever the purchase is done.
- The stock is updated each time a sale is made, because the available stock is verified before the next sale.
- If the perish date exceeds, the existing stock is discarded and restocked to zero.
- If the stock is available at the end of the sale's year or for example at the end of a seasonal period, the remaining stock is considered as discarded.
- The restock due to a stock shortage, is referenced with $Needed\_Stock$. The associated cost occurred with the $Needed\_Stock$ is referenced with $stock\_cost$. These are shown in equation 3.1.

$$\begin{aligned} Needed\_Stock &= Demand - Get\_TotalStock \\ stock\_cost &= Needed\_Stock * Purchace\_Price \end{aligned} \tag{3.1}$$

Moreover, products cannot be purchased at the last day of the sales. This leads to an over stock, which adds additional costs.

- If the demand is higher than the available stock $s_t$ at time $t$, then the available stock will be sold. The amount of goods that are short, will be added as a loss of sales. We need to find a proper value for $N_{loss}$. This can be explained with the graph as depicted in Figure 3.1. In the graph depicted, X - axis is meant by the number of weeks and Y- axis is meant by the amount of products in units.

  When we look at week no.12, 13 or 14 in the depicted graph, the stock is not sufficient for sales. Therefore the available stock will be sold which can be given as in equation 3.2.

$$Sales\_Cost = Available\_Stock * Profit\_Ratio * Purchace\_Price \tag{3.2}$$

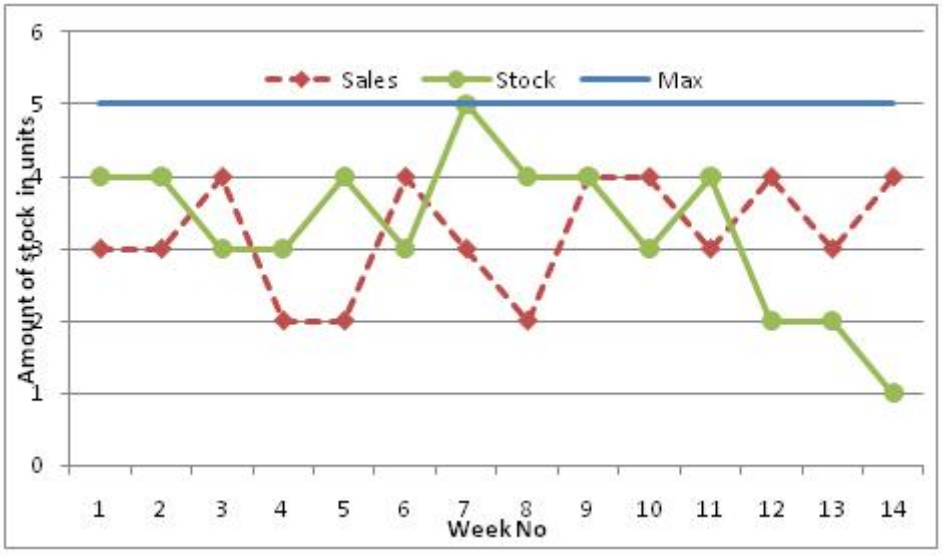


FIGURE 3.1: Available Stock versus actual sales

But there will be a loss of sales cost, because of no stock availability. The technique used to calculate this is explained hereunder.

The estimated demand can be either pessimistic or optimistic. If it is pessimistic then it is always bigger than the actual sales (i.e. worst case), which is always an over prediction. This causes remaining products to be stored. That is, the keeping cost will be added when time goes. If it is optimistic, actual sales and demand are the same. This will not happen all the time. Thus, we define an upper bound called maximum demand (max). The average of max demand and average stock is indicated by penalty stock, which is indicated by $N_{loss}$. This is given in equation 3.3.

$$Penalty = (max - Available\_Stock)/2 \tag{3.3}$$

Therefore the penalty cost ($Loss\_of\_Sales\_Cost$) can be given in equation 3.4.

$$Loss\_of\_Sales\_Cost = Penalty * Profit\_Ratio * Purchace\_Price \tag{3.4}$$

We here showed one way of calculating the cost of loss of sales. But there are several ways to achieve this. For example, we can define the safety margin for

each product (stock level- which reduces the loss of sales cost) and when the stock reaches to the stock level, the stock can be restocked as given in equation 3.5.

$$Updated\_Stock = demand + Stocklevel \tag{3.5}$$

## 3.2 Evaluation of the Cost Function

Once the cost function is derived, it is evaluated for some values to check whether it behaves well. To evaluate the given cost function we have set different values for the parameters of the cost function. This is because we don't know the domain knowledge, and would like to test whether these parameters make sense on the cost function. We would like to evaluate the four cases mentioned in the problem description chapter (Figures 2.1, and 2.2) against our values for the parameters. The obtained results are illustrated in Table 3.1. The parameter profit ratio is a constant value (in this experiment it takes the value 2). Purchase price is also constant (in this experiment it takes the value 1). Each column of the table represents the four cases discussed earlier. KP means keeping cost and DUR means perish date (number of weeks). The comparisons of these results are depicted in Figure 3.2. In the graph, discussed cases(X -axis) versus the associated cost in units(Y- axis) are drawn.

| Type | Cases | Early Prediction | Late Prediction | Over Prediction | Under Prediction |
|---|---|---|---|---|---|
| 1 | H(KP) and H(DUR) | -3.36 | -2.43 | -4.05 | -3.2 |
| 2 | H(KP) and L(DUR) | -3.36 | -2.43 | -4.05 | -3.2 |
| 3 | L(KP) and H(DUR) | -4.64 | -3.47 | -5.25 | -3.99 |
| 4 | L(KP) and L(DUR) | -4.64 | -3.47 | -5.25 | -3.99 |

TABLE 3.1: Type of prediction versus cases

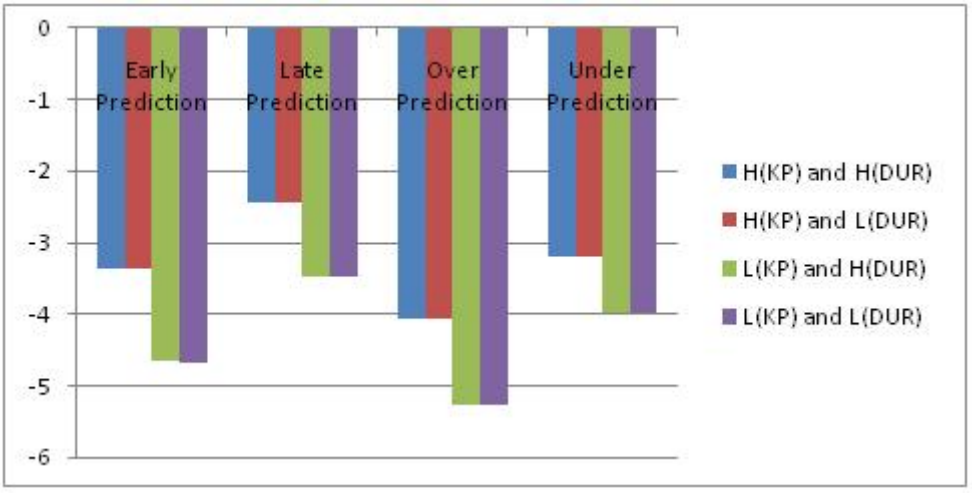


FIGURE 3.2: Type of prediction versus cases

From the graph illustrated in Figure 3.2, it can be observed that for any of the mentioned four types, the cost of late prediction is higher than the cost of early prediction. This is one of the objectives that we would like to see with our parameter settings because these calculated costs are entirely dependent on parameter settings.

Moreover, when we compare over prediction and under prediction, it is obvious that the costs are dependent on the parameter settings. For example, in our experiment the cost of under prediction is more expensive than the cost of over prediction. In addition to these investigations on the cost function and its parameters, we also experimented and concluded that keeping cost of any product should be always less than the purchase price of that product. Otherwise we will experience high additional costs whenever the products are stored for a period of time. This is taken into consideration when the parameter settings are defined in the experiment setup.

# Chapter 4

# Algorithm

This chapter is divided into three sub sections: feature generation [4.1], feature selection [4.2] and the ensemble approach [4.3]. The whole idea of the setup is depicted in Figure 4.1 and will be explained in the following sections.

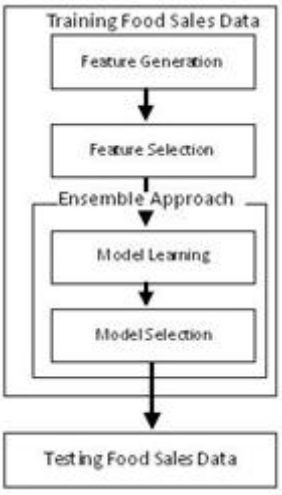


FIGURE 4.1: Step-wise approach of food sales classification

## 4.1 Feature Generation

Feature generation also known as feature construction, is a process of building new features based on those present in the examples supplied to the system, possibly using the domain theory (i.e., information about goals, constraints and operators of the domain). Feature construction techniques can be useful when the attributes supplied with the data are insufficient for concise concept learning. That is, feature generation algorithms search for new features that describe the target concepts better than the attributes supplied with the training instances [15].

We have extracted 335 sets of sales figures from the Sligro data warehouse. Each dataset consists of 120 examples of sales data and each example represents the number of items

sold per week. In our experiments we have generated the features according to two main categories: original features and external features. Original features are for example product id, week number, local max and moving average of the sales, as illustrated in Table 4.1.

| Feature Name | Description |
|---|---|
| Week no | Number of the week sales done |
| MA1 | One day moving average |
| MA6 | Six days moving average |
| Local Max | Lastly observed highest value |

TABLE 4.1: Original feature set

There are a lot of external features available. According to the recent research on Sligro DW with sales prediction[9] they have quoted two types of external features.

The first types are temperature and rain. These features have been added due to their seasonal influence on the consumer demand. For example, the consumer demand of ice cream will increase as temperature rises, where ice cream can't be stored for a long time. The descriptions of these features are illustrated in Table 4.2.

| Feature Name | Description | Values |
|---|---|---|
| Temp | Describes temperature | between 1-8 |
| Rain | Describes Rain | between 1-8 |

TABLE 4.2: External feature set consisting of rain and temperature

The Second type of external features that are recommended are holidays. As we have already mentioned earlier, it is important to add this information to our dataset because seasonal holidays affects the products sales a lot. For example, the sales of eggs during the Easter period are higher than the other days of the year. So it makes sense for a food sales company to store more eggs during this seasonal period and less during the other periods.

With holidays we refer to the cultural holidays and religious holidays. For example Koninginnedag (The Queen's day) and Bevrijdingsdag (Liberation day) are cultural holidays. Kerst (Christmas) and Pasen (Easter) are religious holidays. Features describing holidays have been implemented as binary features, having a value "one" if the holiday occurs in the working day of the week, and a value "zero" if it occurs in the weekend. The descriptions of these features are illustrated in Table 4.3.

Finally, the features such as promotions from Sligro are to be considered. This is another important feature which we have to add to our set because these retailers are spending more and more of their marketing money on sales promotions. Features describing promotions have been implemented as binary features, having a value "one" if the promotions occurs in the working day of the week, and a value "zero" if it occurs in the weekend.

| **Feature Name** | **Description** | **Values** |
|---|---|---|
| Holiday | Describes Holiday | binary values |
| Nieuw-Jaar | New Year | binary values |
| Drie Koningen | Three Kings | binary values |
| Valentijns dag | Valentine day | binary values |
| Carnaval | Carnival | binary values |
| Goede Vrijdag | Good Friday | binary values |
| Pasen | Easter | binary values |
| Koninginne dag | Queen's day | binary values |
| Bevrijdings dag | Liberations day | binary values |
| Hemelvaart | Ascension | binary values |
| Pinksteren | Pentecost | binary values |
| Moederdag | Mother day | binary values |
| Vaderdag | Father day | binary values |
| Sinterklaas | Saint Nicholas | binary values |
| Kerst | Christmas | binary values |
| Oud en Nieuw | Old and New years | binary values |

TABLE 4.3: External feature set consisting of cultural and religious holidays

## 4.2 Feature Selection

All features generated in the previous section can be used to predict the output for a set of inputs. But in real situations, there are some fundamental issues which may be invoked by the irrelevant features involved in the learning process in data mining. These issues have been studied by the statistics and machine learning communities in recent years. The first issue is identified, with irrelevant input features inducing greater computational cost. That is, when more features are available, the computational costs for prediction increases polynomially. Secondly, irrelevant features may lead to overfitting problems. Therefore, it is reasonable and important to ignore those input features with little effect on the output.

The goal of feature selection should be to choose a subset $X_s$ of the complete set of input features $X = \{x_1, x_2, x_3, ..., x_m\}$, so that the subset $X_s$ can predict the output $Y$ with accuracy comparable to the performance of the complete input set $X$, and with great reduction of the computational cost.

## 4.3 Ensemble Learning

In real life, singing a chorus with a set of people is always better than performing it with a single voice. This happens in machine learning too.

So far it is one of the leading areas of research in supervised learning, study methods for constructing good ensembles of classifiers. An ensemble of classifiers is a set of classifiers in which individual decisions are combined in a certain way (typically by averaging or voting) to make the final prediction. The main discovery of these ensembles is often much more accurate than the individual classifiers that produces the ensembles. A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members, are accurate and diverse classifiers[9][14].

Moreover, it has been shown that ensemble learning often increases the performance (e.g. bagging [4], boosting [13] and stacking [17]). This means, the generalization error of the ensemble is lower than the mean of the generalization error of the single ensemble members.

In the real world, concepts are often not stable but change with time. The models built with old data change when new data arrives. For example in food sales, the customer behavior of buying products or the seasonal behaviors always changes the current model or concept, therefore the regular update of the model is important. This problem is called concept drift and has been discussed in the past [6][2]. In order to make time-critical predictions, the model learned from the streaming data must be able to capture up-to-date trends and transient patterns in the stream. Therefore, if we use the ensemble of classifiers with sliding window approach across the series, we can easily adopt the new changes in the new data which might improve the accuracy.

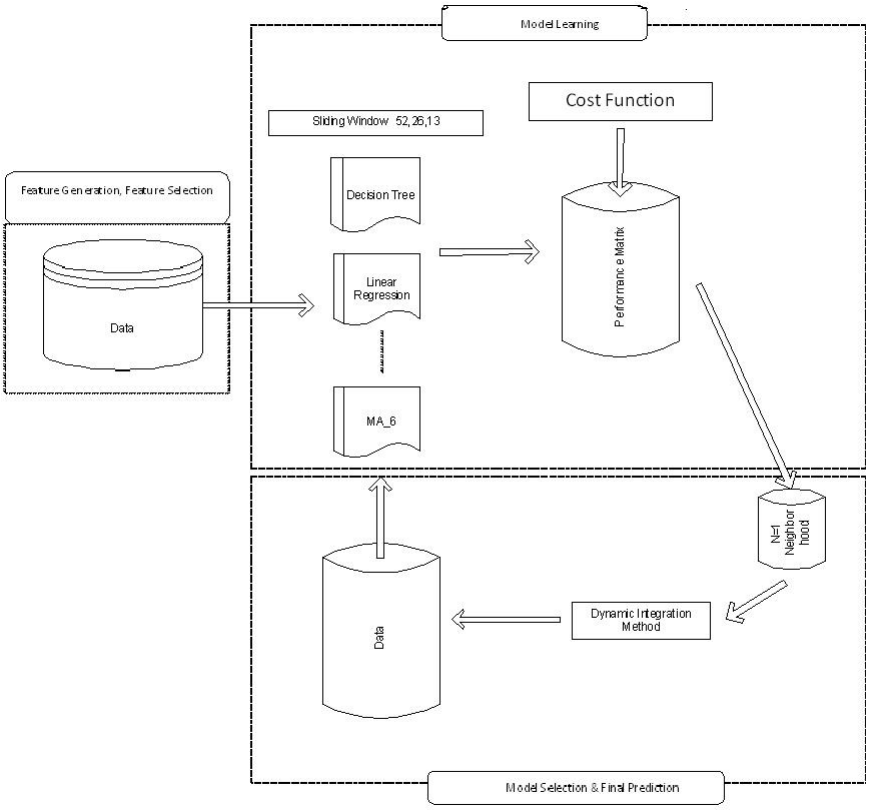


FIGURE 4.2: Ensemble set-up

The ensemble set-up we have used is graphically shown in Figure 4.2. Initially, as we described earlier, the first 77 data points will be used as train instances and the rest would be test instances. We have selected eight supervised learning methods (base learners) with default parameter settings from the RapidMiner repository. They are listed in Table 4.4.

Here, we have selected different types of base learners where each of them has a different learning style. That is they are independent to each other. If we have similar behavior of learners in the ensemble then we won't be able to see better results (they produce similar results) because all of them are related to each other with respect to the learning style. Therefore increasing diversity of an ensemble, by adding learners with different learning styles, is needed to ensure increased prediction accuracy.

| No | Base Learner | Learning Style |
|---|---|---|
| 1 | C4.5 | Decision Tree |
| 2 | CHAID | Decision Tree |
| 3 | Rule Learner | Rule Learning |
| 4 | Basic Rule Learner | Rule Learning |
| 5 | Attribute Based Vote | Lazy Learning |
| 6 | 1-Nearest Neighbors | Lazy Learning |
| 7 | Lib SVM | Functional Learning |
| 8 | Linear Regression | Functional Learning |
| 9 | Moving Average | Six Weeks |

TABLE 4.4: The different learning styles used in our ensemble

Short explanation of the base learners:

- C4.5 is an algorithm used to generate a decision tree. This operator learns decision trees from both nominal and numerical data. C4.5 builds decision trees from a set of training data using the concept of information entropy. Decision trees are powerful classification methods which often can also easily be understood.
- The CHAID decision tree learner works like the decision tree with one exception: it uses a chi squared based criterion instead of the information gain or gain ratio criteria. The learner has the learning capabilities of polynominal attributes, binominal attributes, numerical attributes, polynominal label, binominal label and weighted examples.
- The RuleLearner learns a pruned set of rules with respect to the information gain. In the growing phase, for each rule greedily conditions are added to the rule until the rule is perfect. In the prune phase, for each rule any final sequences of the antecedents are pruned with the pruning metric p/ (p+n).
- The BasicRuleLearner, in contrast to the RuleLearner, learns a set of rules minimizing the training error without pruning.
- AttributeBasedVoting learner is very lazy. Actually it does not learn at all but creates an AttributeBasedVoting Model. This model simply calculates the average

of the attributes as prediction (for regression) or the mode of all attribute values (for classification).

- LinearRegression learner is used to calculate a linear regression model.
- The K-nearest-neighbor (KNN) algorithm measures the distance between a query scenario and a set of scenarios in the data set. We can compute the distance between two scenarios using some distance function. Euclidean distance is one of the distance function used.
- LibSVMLearner in RapidMiner API encapsulates the Java libsvm which is an Support Vector Machine learner. The SVM is a powerful method for both classification and regression.
- Moving average creates a new series of attributes which contains the moving average of new series. The calculation of a series moving average uses a window of a fixed size (in our case six weeks) that is moved over the series data. At any position, the values that lie in the window are aggregated according a specified function. This aggregated value forms the moving average value which is put into the result series.

In this research, we have added the six weeks moving average too as a base learner to our ensemble setup which differs from earlier, because we wanted to know in which cases six weeks moving average is selected. We also wanted to have both ensemble of classifiers and six weeks moving average together, so our algorithm intelligently chooses one of these for prediction purposes. Finally, each of these base learners are trained on the window sizes of 52, 26 and 13. To validate the train data, we use the cross validation technique, where the local classification errors of each base classifier for each instance of the training set are estimated. The models are generated applying all of the base learners for the respective window sizes. The cost function which we derived is used with these cross validation results to find the costs for the cost sensitive method.

Basically our research is about cost sensitive model selection. The base learner's performance is stored in a matrix consisting of costs (is calculated using the cost function discussed in chapter 3) for each learner on each instance in the training set. In addition, we calculated the Mean Absolute Error (MAE) for each learner on each instance in the training set, and stored them in the performance matrix.

Once the performance matrix has been created, the algorithm finds the similar instances from the training set for the new instance using the Euclidean distance method. Once the neighborhood has been created, it uses one of the dynamic integration methods which selects the number of base learners (best models) to achieve final prediction.

The three different integration methods are used in our approach. They are model selection, model voting and model weighting. With model selection, the learner that has both the lowest distance to the testing instance and the lowest cost on the training set(if error based method then error), is selected. With model voting, for each neighbor the base learner with the lowest cost on the training set(if error based method then error), is selected. Then, final prediction is done by letting each selected classifier vote. The predicted label with the highest number of votes is selected. With model weighting, each vote has a weight proportional to the estimated generalization performance of the corresponding classifier. These steps are graphically explained in Figure 5.1.

# Chapter 5

# Experimental Setup

## 5.1 Implementation

RapidMiner (formerly known as Yale) is an environment for machine learning and data mining processes and is used in our experimental studies[11]. RapidMiner is the world-wide leading open-source data mining solution and is widely used by researchers and companies[1]. The modular operator concept of the RapidMiner allows the design of complex nested operator chains for a huge number of learning problems. RapidMiner uses XML(eXtensible Markup Language), a widely used language well suited for describing structured objects, to describe the operator trees modeling KD processes. All RapidMiner processes are described in an easy XML format.

We have implemented our ensemble approach in Java (six weeks moving average is also implemented). From our Java implementation we called the RapidMiner API for learning and applying the different base learners.

## 5.2 Data Setup

For our study we have selected around 335 best selling products, over a period of 120 weeks, provided by Sligro Food Group BV(Dutch food and beverages market)[14] . From the time series of aggregated transactional data, out of 120 data points, the first 77 instances are used as the training set and the last 43 instances are used for testing.

Though we have continuous data (time series data do not have the property of independent identically distributed), we need to make them segmented to use with the traditional machine learning algorithms explained earlier. Therefore, the raw data is converted into a pattern classification problem using a sliding window approach, and the respective target prediction was set as some discretised future value in the raw time series sequence.

Here we describe the time series segmentation approach in detail. Mostly time series segmentation can be done with the technique called sliding window. A simple sliding window can be described by a segment which is grown until it exceeds the bound.

The sliding window algorithm works by anchoring the left point of a potential segment at the first data point of a time series, and then with window size w (the width of the used windows which is smaller than the given time series), step size are moved across the series making new example sets. We choose a time shift of 1 of how far into the future we would like to predict a chosen attribute. This is illustrated in Figure 5.1 and it has the training window size of W. In our setup we have used three different window sizes.

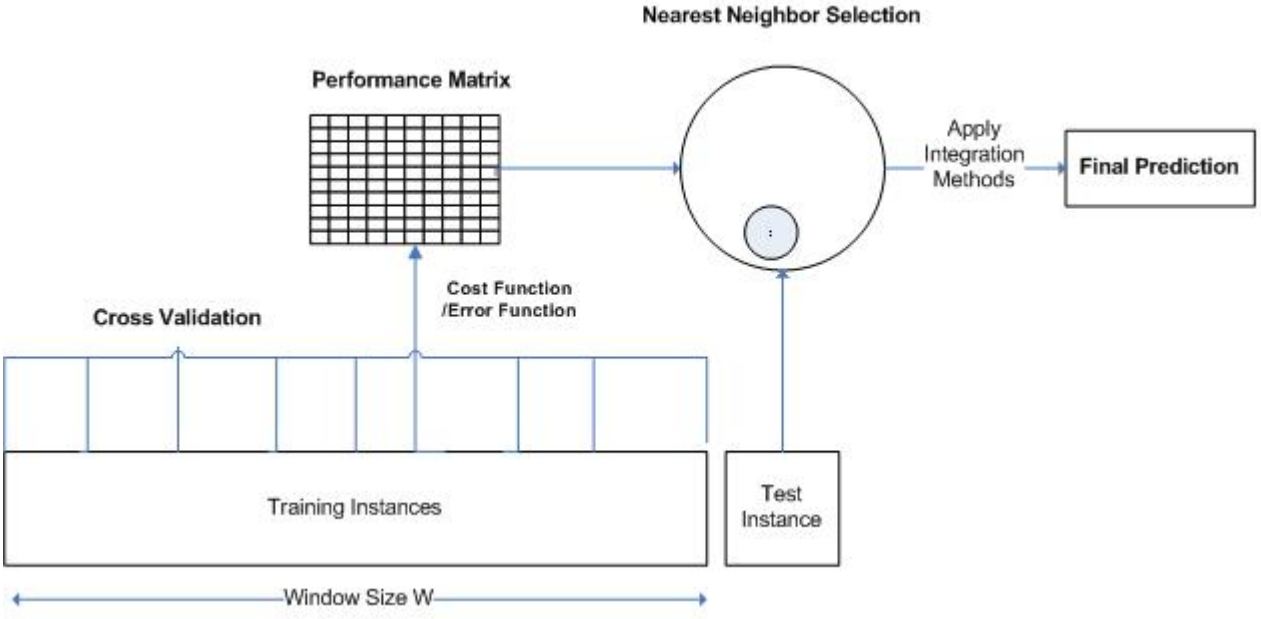


FIGURE 5.1: Sliding window approach

## 5.3 Structure of the Experiment Setup

First, the experiment starts with extracting the 335 datasets from the Sligro data warehouse. As we described earlier each of the data sets contains 120 points. We started our prediction of sales experiments with six weeks moving average for training instances for all window size of all products and continued to each set of test instances. Then we conducted the experiment with our ensemble approach for neighborhood size of 1 with model selection, model weighting and model voting as dynamic integration methods.

Moreover, we have come up with different possible and interesting values for the cost function parameters. This leads to different combinations of values for the parameters in all of our experimental setups. Our cost configuration file is shown in Table 5.1 for any product. First four experiments are with cost sensitive learning of that product. P is the product id. $P\_1\_1$, $P\_1\_2$ are denoted by the parameter settings one, parameter settings two respectively. The last experiment is without cost sensitive learning (i.e. error based learning). If any product is assigned with $P\_1\_6$, then it is for the error based method. The last column of the table determines which type of method is assigned (c-cost sensitive method/ e-error based method). Furthermore, the keeping cost cannot be greater than the purchase price. Keeping products with high keeping costs and a low purchase price

is useless (additional cost is generated). To simplify this, we considered that the short duration of life time products have high keeping cost and vice versa.

| **Exp Configuration for any product** | **KeepCost** | **Purchase Price** | **ProfitRatio** | **Duration** | **Cost Sensitive** |
|---|---|---|---|---|---|
| *P*_1_1 | 0.1 | 0.1 | 1 | 5 | c |
| *P*_1_2 | 0.1 | 0.9 | 2 | 5 | c |
| *P*_1_3 | 0.9 | 1 | 2 | 1 | c |
| *P*_1_4 | 0.1 | 1 | 2 | 10 | c |
| *P*_1_6 | | | | | e |

TABLE 5.1: Cost configuration file for any product

The parameter settings for the cost sensitive types of experiments are shown in Table 5.2.

| **Exp Configuration for any product** | **KeepCost** | **Purchase Price** | **ProfitRatio** | **Duration** | **Cost Sensitive** |
|---|---|---|---|---|---|
| *P*_1_1 | 0.1 | 0.1 | 1 | 5 | c |
| *P*_1_2 | 0.1 | 0.9 | 2 | 5 | c |
| *P*_1_3 | 0.9 | 1 | 2 | 1 | c |
| *P*_1_4 | 0.1 | 1 | 2 | 10 | c |

TABLE 5.2: Cost configuration file of the cost sensitive learning method for any product

In addition to above cost sensitive calculations, we calculated the cost for error based learning using the same parameters used in the earlier five experiments of the cost function. The cost configuration files used to calculate the costs for error based learning are depicted in Table 5.3 for any product.

| **Exp Configuration for any product** | **KeepCost** | **Purchase Price** | **ProfitRatio** | **Duration** | **Cost Sensitive** |
|---|---|---|---|---|---|
| *P*_1_6 | 0.1 | 0.1 | 1 | 5 | e |
| *P*_1_6 | 0.1 | 0.9 | 2 | 5 | e |
| *P*_1_6 | 0.9 | 1 | 2 | 1 | e |
| *P*_1_6 | 0.1 | 1 | 2 | 10 | e |

TABLE 5.3: Cost configuration file for error based learning for any product

# Chapter 6

# Evaluation

In this section, we are going to validate the hypothesis we defined in the earlier chapters. Our intention is to see that utility optimization is better than error optimization in the context of food sales prediction. Food sales companies are more concerned about the sales prediction of products having a short shelf-life and seasonal changes in demand. This might lead to no stock availability during the start of the season and might also cause storing more stock at the end of the season which might cause additional costs. Therefore, to minimize this cost we introduced cost sensitive model selection with ensemble approach in earlier chapters. We wanted to experiment, taking this cost into account versus not taking the cost into account and validate our statement.

## 6.1 Baseline predictions and error measures

In our research, we start by defining the baseline measures for the experiments. Moreover, our intention in this work is to show that the cost sensitive method performs better than the error optimization. Therefore we need to calculate the error for the food sales prediction.

Traditional measures are for example MAE (Mean Absolute Error) or MSE (Mean Squared Error), which is used in previous food sales research. But we recommend not to use them because the food sales data have different types of products with different types of behavior such as seasonal changes. The Mean Absolute Error(MAE) is simply given by the equation 6.1.

Let $True_t$ denotes the observation at time $t$. $Prediction_t$ denotes the forecast of $True_t$. Then we can define the forecast error as

$$e_t = |True_t - Prediction_t| \tag{6.1}$$

For example, Figure 6.1 illustrates two products with different behavior in demand across the series. The left one has periodic behavior where as the right one has variation of demand across the series. A learner (for example, naive predictor which chooses the last observed value as the prediction for the next point) is employed on both of the series. When we calculate the MSE for the left one then it always takes the popular value (in this case same as demand i.e. thick line drawn at y=1), and achieves a good performance. But the right one has noisy data (sudden rises and drops)and therefore likely produces bad results.

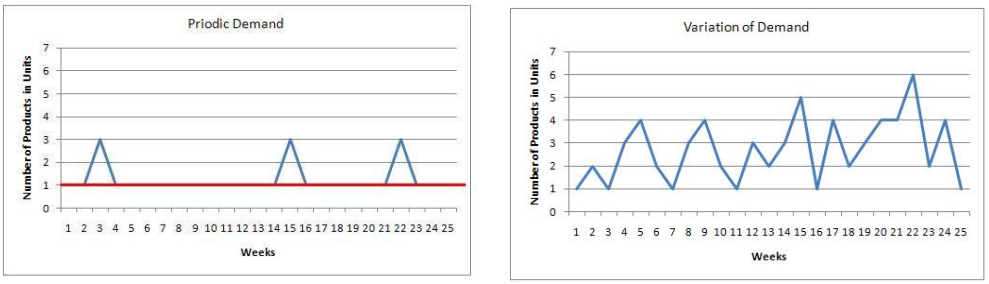


FIGURE 6.1: Periodic demand versus variation of demand

Therefore, we use the Mean Absolute Scaled Error (MASE) as the prediction error measure. MASE can be used to compare forecast methods on a single series and also to compare forecast accuracy between series. The Mean Absolute Scaled Error is simply given by equation 6.2.

MASE ($q_t$ ) can be given as

$$q_t = \frac{1/T \sum_{t=1}^{T} |True_t - Prediction_t|}{1/T \sum_{t=1}^{T} |True_t - True_{t-1}|} \tag{6.2}$$

## 6.2 Historic Sales data from related products as input

In this section we describe the basic statistics done for the different data sets for the cost sensitive and non cost sensitive experiments. Two different types of experiments are done for each integration method (model selection, model weighting and model voting). For each integration method we calculated the costs for cost sensitive method, and cost for non cost sensitive method. We used four different parameter settings discussed earlier in the experimental setup for each product. Table 6.1 shows the summary of the statistics. We did a total of 524 experiments.

In addition to these basic statistics, we are able to compare the costs calculated from the predictions for cost sensitive method and non cost sensitive method for all integration methods. Table 6.2 shows the comparisons of each method (model voting and model weighting) with respect to cost. The notation CCS is meant by cost of cost sensitive method and CNCS is meant by cost of non cost sensitive method.

We can see from the results obtained, that when model voting is selected as the integration method, cost sensitive (CS) method performs much better than other integration methods. That is, in model voting, in 348 out of 524 experiments, cost sensitive method

| Statistics | Model Weighting | | Model Voting | |
|---|---|---|---|---|
| | Exp CS(Cost Sensitive) | NCS(Non Cost Sensitive) | CS(Cost Sensitive) | NCS(Non Cost Sensitive) |
| Mean | -603.58 | -593.54 | -482.69 | -450.64 |
| Range | 3775.3 | 3703.7 | 4243.4 | 4037.8 |
| Minimum | -3166.8 | -3048.7 | -2997.3 | -2790.2 |
| Maximum | 608.5 | 655 | 1246.1 | 1247.6 |
| Sum | -316277 | -311013.8 | -252929.4 | -236134.2 |
| Count | 524 | 524 | 524 | 524 |

TABLE 6.1: Summary of the statistics

| **Performance** | **Model Weighting** | **Model Voting** |
|---|---|---|
| CCS < CNCS | 265 | 348 |
| CCS > CNCS | 259 | 176 |

TABLE 6.2: External feature set consisting of rain and temperature

performs better than non cost sensitive method (NCS) with respect to cost. That is 66.14% out of 524 experiments.

Moreover, the comparison of each of these integration methods with moving average (with respect to cost) is illustrated in Table 6.3.

| Performance based on cost | Model Weighting | | Model Voting | |
|---|---|---|---|---|
| | Exp CS | NCS | CS | NCS |
| Moving Average (better) | 231 | 235 | 313 | 315 |
| Moving Average (not better) | 293 | 289 | 211 | 209 |

TABLE 6.3: Moving average is compared with each integration method

From the results we can conclude that, when model voting is selected as an integration method, the base learner selected for the final prediction from the ensemble of classifiers is not the six weeks moving average. Otherwise the performance of the model voting is likely to be the same as moving average, if it is selected as an individual base learner.

In model voting, when we compare the cost sensitive method and non cost sensitive method with respect to costs. There will be two situations:

- Cost difference is high(cost sensitive is very expensive, i.e. worst case for CS method).
- Cost difference is low(non cost sensitive is very expensive, i.e best case for CS method).

With respect to the first case, the series is depicted in Figure 6.2. In the graph, by "Label" we mean the actual value and by "Model Voting CS" we mean the integration

method used to predict for cost sensitive method. It shows that the prediction is very low compared to true label. That is most of the time when under prediction occurs and this causes always additional costs (loss of sales cost) across the series. Also from the graph we can see there is a late prediction with a huge peak (same amplitude as true label),and it is obvious we can conclude that at this instance level the moving average is not the base learner. The reason is that moving average always responds with low amplitude. Moreover, there are a number of early, over, under, and late predictions happening(including high amplitude ones). As a result this leads to the worst case.

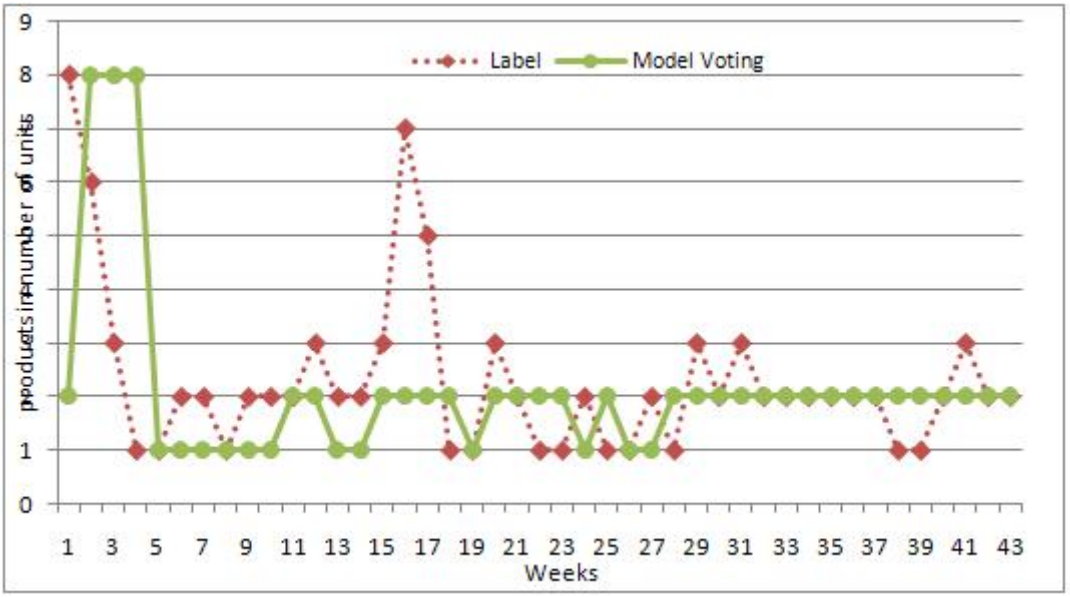


FIGURE 6.2: True label and worst case scenario prediction of cost sensitive method

In addition, from the graph illustrated we can see that there are many rises and drops, which the learner, selected for the final prediction, could not recognize. The reason is, there may be not enough positive instances in the training set to learn from, and to predict the rises and drops. We suspect that the training error is also too high.

For example, Figure 6.3 shows the training and testing part of a highly periodic series. If a learner is employed on these training instances then the probability of predicting the peaks in the training part are very low (i.e. high error) because the data is highly periodic. As a result, it could not predict the peaks in the testing instances.

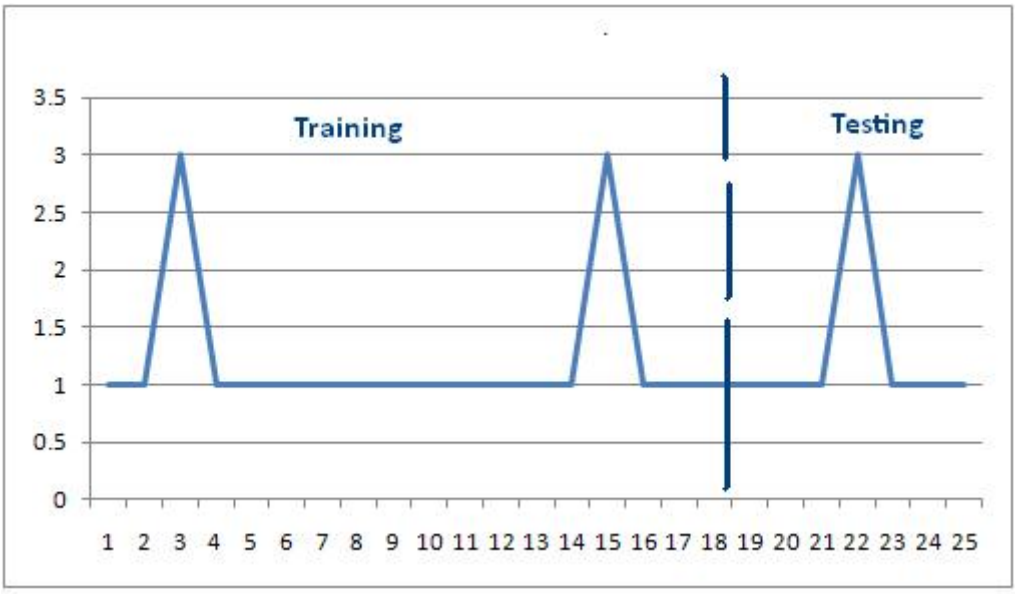


FIGURE 6.3: High periodic series

Another suspected reason is probably due to overfitting the data. It is obvious that food sales data consist of lots of seasonal and promotional information. The selected learner is assumed to reach a state where it will also be able to predict the correct output for unseen data (test data). However, especially in cases where learning was performed too long or where training examples are rare (for example in this case huge peaks and drops often occur in the test instances but not in the training instances), the learner may adjust to very specific random features of the training data which do not have any corresponding relation with the target function. This problem is called overfitting.

On the other hand, the best case is depicted in Figure 6.4 . From the graph we can infer that under, late and over predictions are happening here but in few cases the learner is able to identify the drops and raises of the true label. This causes the generated cost across the time series to be low when compared to non cost sensitive method. Still the reasons mentioned in the worst case are valid here too.

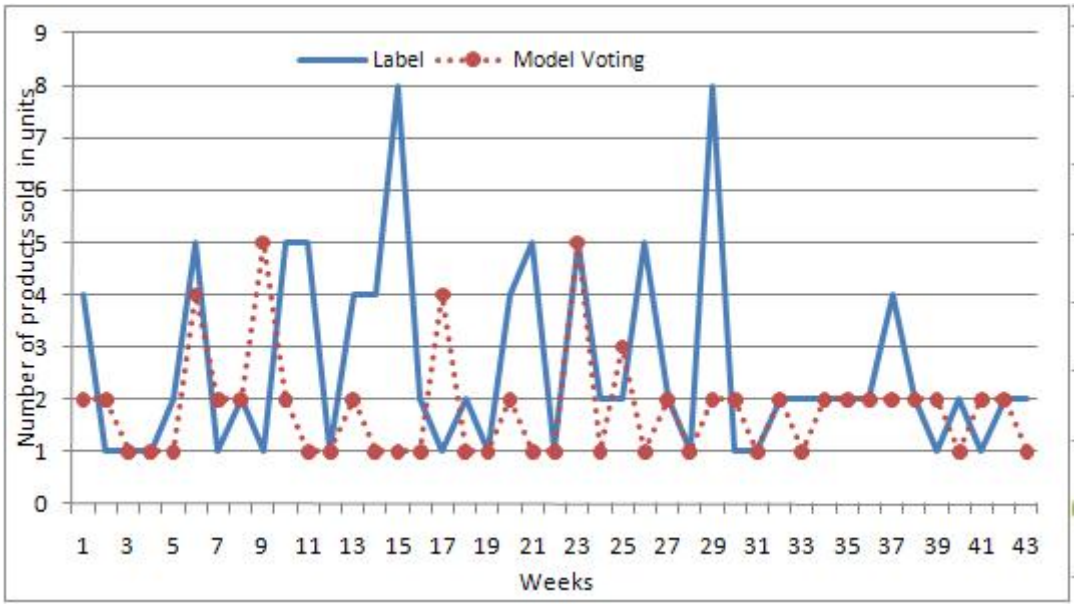


FIGURE 6.4: True label and best case scenario of cost sensitive method

Our intention in this part of the research is to show that any relationship holds between the cost sensitive method and the error based method in food sales prediction. That is to show if they have any correlation between them or not. The comparison of MASE error (Y-Axis), and the scaled cost(X- Axis) of cost sensitive method for the selected products for model voting, are shown in Figure 6.5. Here, the cost sensitive method's generated cost for a product is scaled by dividing the corresponding one day moving average cost of that product in order to compare with MASE of that product.

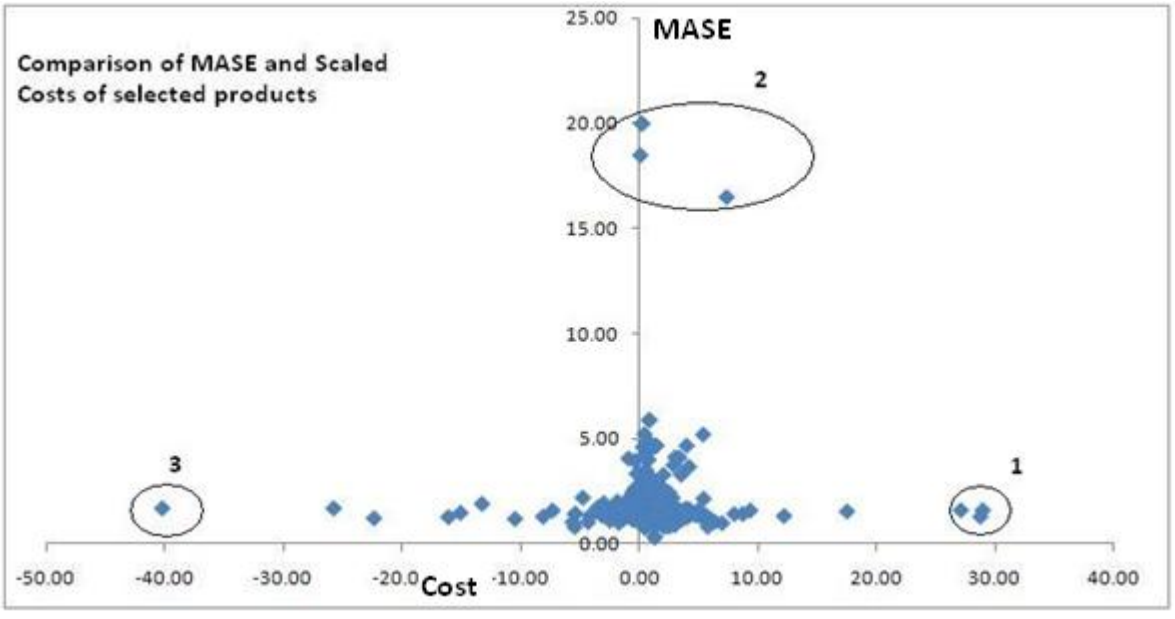


FIGURE 6.5: Comparison of scaled cost (CS) and MASE of selected products

By definition, if the data points make a straight line going from the origin out to high x- and y-values, then the variables are said to have a positive correlation and vice versa. From the graph we can see that there is no correlation between both axis (cost versus error). That is, there is no relationship between two of these variables.

Furthermore, from Figure 6.5, it can be observed that there are three sets of points (suspected as outliers). The point marked with number 1 has high cost and low error. The points marked by number 2 have high error but no cost. The set of points circled in number 3, have low costs and low error. Because of scaling, these points shown up in the graph as outliers. The MASE error is not bounded, there is no upper bound defined for it. Therefore, the error can take the value between 0 to infinitive.

## 6.3 Moving Average as the Prediction label

In this section we take the estimated demand as six weeks moving average and calculate costs for 524 experiments. We compared these calculated costs with respective costs of model voting and model weighting (both cost sensitive and non cost sensitive) of the products. The results are shown in Table 6.3.

From the table we can see that compared to model weighting in model voting moving average produces less cost in both cost sensitive (313 times) and non cost sensitive (315 times) method.

## 6.4 Results and Discussions

From the experiments we conducted to validate our hypothesis, we are able to summarize the following results and discussions.

From the experimental study we suspect that model voting performs better than model weighting with respect to cost. That is, we showed that in 348 (66.14%) out of 524 experiments cost sensitive method performs better than non cost sensitive method when model voting is selected as the integration method.

The reasons discussed below are the possible ways to improve the results and are useful to consider.

First, the sales policy that we have defined in our research with respect to stock can be discussed more. As we mentioned earlier that we can define the safety margin (stock level) for each product which might reduce the problem with no stock availability. In addition to this, we can think about the stock at the end of the sales' year (or may be for the seasonal products at the end of the season) by giving promotions on the last day which also might reduce the costs. This is rather considered than discarding the products.

Secondly, the cost function parameters are based on our experiments on food sales prediction. In food sales company's point of view they can add their own parameters or discard the parameters, and use them in the cost function to calculate the associated costs.

Thirdly, in our research we considered the neighbor size of 1. It is possible to experiment with different sizes of neighbourhoods (not looking at larger k's since accuracy normally decreases with the increase in size of neighborhoods), and compare the results. That is the experiments can be conducted with neighbourhood size k (for example k=1, 2, 5 and 10), and model selection, model weighting and model weighed voting as dynamic integration methods.

Finally, while the models are trained in our algorithm, we used only a few base learners ($\leq 10$). Generally, to get a good ensemble, the base learners should be as accurate, and diverse as possible. Therefore, it is advisable to have more base learners added to the ensemble. If there are more learners available we will see a diverse output, and therefore the one selected for the final prediction produces better results.

# Chapter 7

# Conclusions and Recomendations

The ability to forecast consumer demand accurately is of great importance to companies like food sales retail and whole sales. They are more concerned about the sales prediction of products having a short shelf-life and seasonal changes in demand. Because of the bad prediction in food sales, both shortage and surplus of goods can lead to additional costs and loss of income for the company. Therefore, accurate forecasts is useful for a food sales company to prevent customers from going to competitors.

The six weeks moving average works well when sales pattern is at an approximate level. Though the moving average is used in the current food sales prediction, it responds late to the sudden drops and raises (as we discussed earlier), that occur in the data set. An intelligent method might perform better than the moving average on these mentioned sudden drops and raises. Therefore, the Ensemble of classifiers method with dynamic integration of classifiers is proposed.

We discussed four possible types of predictions where the possible costs are generated. We made the sales policy which includes all these four cases and is reflected as a cost function. As a result, the cost function is able to calculate the related possible costs for the food sales prediction. We used these costs in our evaluation method.

We use these mentioned costs as an input to the ensemble of learners which is made up of different base learners. This ensemble generates multiple base models using traditional machine learning algorithms, and combines them into an ensemble model. In addition, we added the sliding window approach which uses a window that moves over recently arrived instances and uses the learnt concepts for prediction in the immediate future only. Finally, we apply dynamic integration of classifiers to select the best model for final prediction and for better handling of seasonal changes and fluctuations in consumer demands.

From the experimental study we suspect that model voting performs better than model weighting with respect to cost. That is, we saw that in 348 (66.14%) out of 524 experiments cost sensitive method performs better than non cost sensitive method when model voting is selected as the integration method.

Sligro can consider this research work with respect to the costs in future sales. First of all, we have shown that by introducing the cost sensitive model selection with ensemble approach, the accuracy increases. In this way, with less human intervention better forecasting is possible. Secondly, we have demonstrated that these calculated costs are

entirely dependent on parameter settings. So it is possible to use their own parameters, and calculate the costs, and produce better prediction results. In addition to this, the stock policy that we defined is based on our basic experience with the food sales. Therefore, they can modify or improve it according to their own policies.

# Appendix A

# Pseudo-code of the COST Function

**Listing A.1:Pseudo-code of the COST Function**
The following Pseudo-code code shows the COST Function.

```
public class CostEvaluator {

              //declare global variable

        public void SetKpCost(double _KpCost)
        {
          //  initialize keep_cost
        }

        public void SetPPrice(double _PPrice)
        {
                //  initialize purchase_price;
        }

        public void SetProfitRatio(double _ProfitRatio)
        {
                //  initialize Profit_Ratio;
        }

        public void SetStockLevel(int _StockLevel)
        {
                //  initialize Stock_Level;
        }
        public CostEvaluator(int _Duration)
        {
                // initialize the variables
       //stock is initialized to 1 at the beginning and
      //  duration too assigned for the product
                InitStock(1, Duration);
        }

        private void InitStock(int startval, int expdate)
        {
              //stock is initialized to 1 at the beginning and
          //  duration too assigned for the product
        }


        private double UpdateStock(int weeknr, int pred)
        {

           //  Throw away obsolete items

        For each week check the item's expire date
```

```
          If  date of expiry week  > current week
                obsolete items

             // Buy new items if needed

        //If   available stock less than prediction

                       Needed_stock = prediction - Get_stock()
                       Stock_cost = needed * Purchase_Price;
          // update the expiry date
                       int pdate = weeknr + Duration - 1;
                       // if the current week is last week then
           //  not allowed to purchase it

       return stockcost;
       }


       private void DeductFromStock(int amount)
       {

               //deduct the amount sold products

       }


       public int Get_Stock()
       {
                    // Calculate the available stock
       }


       private double sales(int act, int pred)
       {
        // if the stock is less than the Actual sales then
        //sell the available stock
           Profit_cost = stock * Profit_Ratio* Purchase_Price;
           DeductFromStock(stock);

      // find the penalty
          penalty = (max - stock)/2;
      // find the penalty cost
         Profit_cost = (penalty * Profit_Ratio * Purchase_Price);

     // if we have enough stock
               //sell the products and deduct it
        Profit_cost -= act * Profit_Ratio * Purchase_Price;
               DeductFromStock(act);

     return pcost;
       }


       public double CalcCost(int week ,int act, int pred)
       {
                // Cost for items stored
        cost = GetTotalStock() * Keep_Cost;
       // Cost for sales
        cost = UpdateStock(week, pred);
       // Cost for purchase
        cost = Purchase(act, pred);
     return cost;
       }

}
```

Listing A.1: Pseudo-code code shows the COST Function

# Appendix B

# Pseudo-code of the Performance Matrix

**Listing B.1:Create Performance Matrix**
This appendix shows the pseudo code for read the results of the cross validation process from file and calculate the cost/ error associated.

```
error(b,i)  = error/cost of base learner b on instance i
Label= Actual sales
Error= |Actual sales- Prediction |
methodname = cost sensitive or error based
prediction(b) = prediction done by base learner b
minicost= Minimum cost occurred (Lower bound of the cost)
Original cost= Cost by the sales
ErrorArray = stores the Error
Scaled Cost= Scale the original cost; otherwise cost will be negative.
             Always a positive value
MaximumCost = find the Maximum scaled cost; Upper bound of the cost
ArrayCost =Scaled cost are stored in an array
ScaledCostArray = final scaled cost( between 0 and 1) are stored in the array
For each model i=1 to 8
 While read the file which contains the cross validation results Until Last Line
  Calculate Minicost
  Calculate Error
  ErrorArray(each line)=Error
  Calculate Original cost
  Scaled Cost = Original cost  + Minicost

  For  each  element of the array ArrayCost
    IF   MaximumCost < Scaled Cost  THEN
        MaximumCost  = Scaled Cost
  end

  For  each  element of the array ArrayCost
     ScaledCostArray(element) = Scaled Cost/ MaximumCost
  end
 END of WHILE

 IF  methodname ==  cost sensitive  THEN
   For  each  element of the array ScaledCostArray
      error (b,i)= ScaledCostArray (element)
   end
 ELSE
   For  each  element of the array ScaledCostArray
      error (b,i)= ErrorArray(element)
 end
```

```
end
```

Listing B.1: Performance Matrix

# Bibliography


[1] Rapid miner, http://rapid-i.com/content/blogcategory/38/69/.

[2] P. Cunningham A. Tsymbal, M. Pechenizkiy and S. Puuronen. Dynamic integration of classifiers for handling concept drift. In *Inf. Fusion*, pages 56–68, 2008.

[3] D. Adebanjo and R. Mann. identifying problems in forecasting consumer demand in fast moving consumer goods sector. In *In Benchmarking: An International Journal*, pages 223–230, 2000.

[4] L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[5] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions. In *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, pages 64–168, New York, NY, 1998.

[6] A.Kadam G. Widmer D.Delen, G.Walker. Learning in the presence of concept drift and hidden contexts. In *In Machine Learning*, pages 69–101, 1996.

[7] Pedro Domingos. Metacost: A general method for making classifiers cost sensitive. In *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164, August 1999.

[8] Liangyuan Li, Mei Chen, Hanhu Wang, Wei Chen, and Zhiyong Guo. A cost sensitive ensemble method for medical prediction. In *Database Technology and Applications, International Workshop on*, pages 221–224, 2009.

[9] Pechenizkiy M. Meulstee, P. Food sales prediction: if only it knew what we know. In *ICDM Workshops,IEEE Computer Society*, pages 134–143, Los Alamitos, 2008.

[10] I. Mierswa, M. Wurst, R. Klinkenberg, and T Scholz, M.and Euler. Predicting breast cancer survivability: a comparison of three data mining methods. In *Artificial Intelligence in Medicine Volume 34, Issue 2*, 2006.

[11] I. Mierswa, M. Wurst, R. Klinkenberg, and T Scholz, M.and Euler. Yale (now: Rapidminer): Rapid prototyping for complex data mining tasks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006*, 2006.

[12] F. Provost. Toward economic machine learning and utility-based data mining. In *UBDM '05*, Chicago, Illinois, USA, August 2005.

[13] R. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[14] P. Cunningham Tsymbal, M. Pechenizkiy and S. Puuronen. Handling local concept drift with dynamic integration of classifiers:domain of antibiotic resistance in nosocomial infections. In *cbms*, pages 679–684, 2006.

[15] S. Puuronen Tsymbal and I. Skrypnyk. Ensemble feature selection with dynamic integration of classifiers. In *In in: Int. ICSC Congress on Computational Intelligence Methods and Applications CIMA 2001*, pages 558–564, 2001.

[16] P.D. Turney. Types of cost in inductive concept learning. In *Proc. Workshop Cost-Sensitive Learning at the 17th Int'l Conf*, 2000.

[17] D. H. Wolpert. Stacked generalization. In *Neural Networks*, pages 241–259, 1992.