**Sales Estimation:**

**Project Report**

**By:**

**Kunal Sontakke (802329466)**

**Sarthak Saxena (802325076)**

**CPSC 531-03 (20850)**

**Professor: Dr. Chun-l Philip Chen**

**Department of Computer Science**

**California State University, Fullerton**

**Spring, 2015**

**Table of Contents**

1. Introduction…………………………………………………………………..…………..4
2. Literature review…………………………………………………………………………5
3. Methodology .……………………………………………………………………………8
4. Results and Discussions……………………………………………………………….…21
5. Conclusion, Implications and Recommendation…………………………………….…..24
6. References……………………………………………………………………………..…26
7. Appendix…………………………………………………………………………………27

**Table of Figures**

1. [Figure 1 Needs and possibilities of KDD (Maimon & Rokach, n.d.) 5](#_Toc418774591)
2. [Figure 2 output 23](#_Toc418774592)

**Abstract**

Sales estimation means predicting the sales for a certain period depending on the historical data of the store. The sales prediction establishes the level of activity used in all the other forecasts and budgets for the business. If your sales forecast varies wildly from your actual results, your cash flow and profitability forecasts will similarly be inaccurate. The entire sales forecasting process is intended to generate better business results from better business decisions. So what really matters is whether or not your sales forecast is set up right and broken into factors you’ll be able to track and will lead you to better management decisions.

The main objective of this project is to analyse the historical sales data available for many departments of a store by exploring different data mining techniques. The datasets will undergo through various data mining tasks such as survey analysis, training the model using the dataset and prediction of knowledge using proper logic. Different data mining tasks, techniques and tools will be used for the sales estimation from the extracted data of the dataset.

1. **Introduction**

The most important thing in retail is to continue satisfying customer needs and their demands. As seen in today’s world competition is increasing and so is the expectations of customers. This project introduces the concept of data mining in this process. The concept behind this project is to increase the sales by analysing and keeping track of customer’s habits. This activity is known as “Knowledge Discovery in Database”. (Maimon & Rokach, n.d.) The motive of this project is to increase the sales by using data mining techniques. This includes algorithms which are used in order to provide such kind of techniques.

The sales of a particular store has gone bad. It depends on the quality, quantity and the prices which are available at the store. There are various discounts to be added in order to improve the sales. The store needs a proper plan about how the sales should be increased and improved. They need to find special occasions to implement, it may be its holidays or special days. For implementation of this, it is essential to look for the historical data in respective departments of the company.

The holiday markdown events are included. It is very important to note down the effects of holiday markdown events and particularly how they effect it. The effect and impact of every holiday is marked and analysed. The lack of awareness and prediction techniques may cause failure in the sales.

1. **Literature Review:**

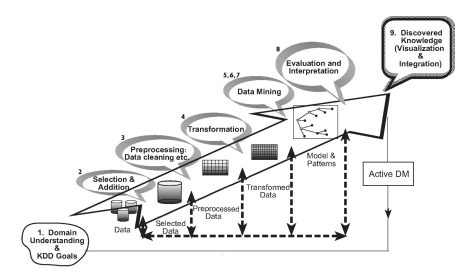
The knowledge data discovery process is processed in nine steps. This process is iterative and in each step going back might occur. There are different needs and possibilities 

Figure 1 Needs and possibilities of KDD (Maimon & Rokach, n.d.)

at each step of this process. The entire structure of the process as shown in the Figure1.

According to the first step of the Knowledge Data Discovery process we have to develop and understand the application domain. This is a very initial process which prepares the system for further steps. In this process the goals of the application are decided. So, we first came up with the problem statement and the techniques to solve it. After understanding the problem statement, the pre-processing of data is started. (Murray & Gardiner, n.d.)

The second step is to select and create the dataset on which the discovery is to be operated which we have obtained from KDD website for our project. It includes the training set, test set, features set. In this step, the availability of data is also decided, which type of data is required and then all the data is integrated into one data set which includes all the important attributes of this data. In this all the attributes should be present if not then the entire process can collapse. Here we created a “data” folder and kept all the files in it.

The third step is the pre-processing and cleansing. In this particular step the data mishandling is improved. If there are certain values which are missing they are cleared and the entire data set is to be cleaned. We have some missing data in the marked down events where we have mentioned NA. (Maimon & Rokach, n.d.)

The fourth step of this process is data transformation. This step includes cropping out some irrelevant data which is not useful. This step is essential because it is providing the specifications for the entire KDD process. We truncated the data we got from KDD site to 50% as it was very difficult to run such a big file which consists of 128mb size, having a RAM of 2GB.

The fifth step is to choose the appropriate data mining task. After so much of filtering the process is ready to choose its data mining task which should be appropriate for the particular process. The objectives of this step is prediction and description. For this we have chosen randomforest package and the timedate package which uses the Breiman’s algorithm for classification and regression. It is actually based on the following three concepts:

* Decision tree
* Tree bagging
* From Bagging to Random forest

The sixth step is to choose the data mining algorithm. Before this step the strategy has been made and now the task is to make tactics. This might include finding the searching method for process or may be the technique to search.

The Seventh step is to employ the data mining algorithm. This includes the implementation of the data mining algorithm which is decided in sixth step. This step might include the iteration of that algorithm just to make sure its validity and efficiency.

The eight step is to evaluate. In this step the evaluation of the pattern is done. It keeps a track of fulfilling the goals of the project and maintaining its efficiency. This step specifies the usefulness and the efficiency of the pattern used. The discovered knowledge is documented for the further use of the system.

The ninth and the last step of this process is to use the discovered knowledge. This step makes it useful for the knowledge discovered for the further purpose. This step also includes the changes made into data and noticing its effects in the system. Hence, this way the KDD process is completed in its nine stages. (Maimon & Rokach, n.d.)

1. **Methodology**

The problem solution will be written in R programming. R is a tool for statistics and data modelling. The R programming language is elegant, versatile, and has a highly expressive syntax designed around working with data. R is more than that, though — it also includes extremely powerful graphics capabilities. (code school, n.d.).R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. (The R foundation, n.d.)

The backend or the database proposed for the solution will be achieved by the use of CSV. CSV is a common file format that is widely supported by consumer, business, and scientific applications. Among its most common uses is moving tabular data between programs that natively operate on incompatible (often proprietary and/or undocumented) formats. (wikipedia, 2015).The so-called CSV) format is the most common import and export format for spreadsheets and databases. There is no “CSV standard”, so the format is operationally defined by the many applications which read and write it.

The dataset for this problem is acquired from KDD. KDD (Knowledge Discovery in Database) is modelling and analysis of the large data repositories. The data mining is the basis for KDD, involving construction of the algorithm, developing the model for unknown pattern and predict the useful solution with the help of suitable logic.

The editor for R programming of the graphical user interface to be used for R programming will be R Studio. R Studio is a free and open source integrated development environment (IDE) for R, a programming language for statistical computing and graphics. (The R foundation, n.d.)

R Studio Desktop, where the program is run locally as a regular desktop application; and R Studio Server, which allows accessing R Studio using a web browser while it is running on a remote Linux server. (Wikipedia, 2015)

CSV files can be viewed in Microsoft Excel or CSVed. CSVed 2.3.2 is the latest version of CSVed. It is an easy and powerful CSV file editor, you can manipulate any CSV file, separated with any separator. (wikipedia, 2015)

The following csv files will be included in the solution.

* + - * 1. **stores.csv**

This file will contain anonymized information about the 45 stores, indicating the type and size of store.

* + - * 1. **train.csv**

This will be the historical training data, which covers to 2010-02-05 to 2012-11-01. This file will contain the following fields:

* Store - the store number
* Dept - the department number
* Date - the week
* Weekly\_Sales -  sales for the given department in the given store
* IsHoliday - whether the week is a special holiday week
  + - * 1. **test.csv**

This file is identical to train.csv, except we have withheld the weekly sales. You must predict the sales for each triplet of store, department, and date in this file.

* + - * 1. **features.csv**

This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields:

* Store - the store number
* Date - the week
* Temperature - average temperature in the region
* Fuel\_Price - cost of fuel in the region
* MarkDown1-5 - anonymized data related to promotional markdowns that Walmart is running. MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA.
* CPI - the consumer price index
* Unemployment - the unemployment rate
* IsHoliday - whether the week is a special holiday week

For convenience, the four holidays fall within the following weeks in the dataset (not all holidays are in the data):

* Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
* Labor Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
* Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13
* Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

We have also imported two packages:

1. TimeDate:

* Description: Create a ’timeDate’ object from scratch using a character vector.
* Usage: timeDate(charvec, format = NULL, zone = "", FinCenter = "")

strptimeDate(x, format = whichFormat(x), tz = "") (Team, et al., 2015)

* Arguments:

Charvec: a character string or vector of dates and times.

Format: the format specification of the input character vector.

Tz: a character with the the location of the financial center named as "continent/city", or short "city".

Zone: the time zone or financial center where the data were recorded.

X : a character string or vector of dates and times.

FinCenter: a character with the the location of the financial center named as "continent/city".

* Value: returns an object of class timeDate. (Team, et al., 2015)

1. Randomforest:

* Description: Classification and regression based on a forest of trees
* Implementation: randomForest implements Breiman’s random forest algorithm (based on Breiman and Cutler’s original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data point (Breiman, Cutler, Liaw, & Wiener, 2015)
* Usage: ## S3 method for class 'formula'

randomForest(formula, data=NULL, ..., subset, na.action=na.fail)

## Default S3 method:

randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,

mtry=if (!is.null(y) && !is.factor(y))

max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),

replace=TRUE, classwt=NULL, cutoff, strata,

sampsize = if (replace) nrow(x) else ceiling(.632\*nrow(x)),

nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,

maxnodes = NULL,

importance=FALSE, localImp=FALSE, nPerm=1,

proximity, oob.prox=proximity,

norm.votes=TRUE, do.trace=FALSE,

keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE, keep.inbag=FALSE, ...) ## S3 method for class 'randomForest' print(x, ...) (Breiman, Cutler, Liaw, & Wiener, 2015)

Arguments:

Data: an optional data frame containing the variables in the model. By default the variables are taken from the environment which randomForest is called from.

Subset: an index vector indicating which rows should be used. (NOTE: If given, this argument must be named.)

na.action A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named.)

x, formula a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the print method, an randomForest object).

Y A response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, randomForest will run in unsupervised mode.

xtest a data frame or matrix (like x) containing predictors for the test set.

ytest response for the test set.

ntree Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.

Mtry Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification

(sqrt(p) where p is number of variables in x) and regression (p/3) replace Should sampling of cases be done with or without replacement?

classwt Priors of the classes. Need not add up to one. Ignored for regression.

Cutoff (Classification only) A vector of length equal to number of classes. The ‘winning’ class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).

strata A (factor) variable that is used for stratified sampling.

sampsize Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.

nodesize Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).

maxnodes Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by nodesize). If set larger than maximum possible, a warning is issued.

importance Should importance of predictors be assessed?

localImp Should casewise importance measure be computed? (Setting this to TRUE will override importance.)

nPerm Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.

proximity Should proximity measure among the rows be calculated?

oob.prox Should proximity be calculated only on “out-of-bag” data?

norm.votes If TRUE (default), the final result of votes are expressed as fractions. If FALSE, raw vote counts are returned (useful for combining results from different runs). Ignored for regression.

do.trace If set to TRUE, give a more verbose output as randomForest is run. If set to some integer, then running output is printed for every do.trace trees.

keep.forest If set to FALSE, the forest will not be retained in the output object. If xtest is given, defaults to FALSE.

corr.bias perform bias correction for regression? Note: Experimental. Use at your own risk.

keep.inbag Should an n by ntree matrix be returned that keeps track of which samples are “in-bag” in which trees (but not how many times, if sampling with replacement)

... optional parameters to be passed to the low level function randomForest.default. randomForest (Breiman, Cutler, Liaw, & Wiener, 2015)

**Value** An object of class randomForest, which is a list with the following components:

call the original call to randomForest

type one of regression, classification, or unsupervised.

predicted the predicted values of the input data based on out-of-bag samples.

importance a matrix with nclass + 2 (for classification) or two (for regression) columns. For classification, the first nclass columns are the class-specific measures computed as mean descrease in accuracy. The nclass + 1st column is the mean descrease in accuracy over all classes. The last column is the mean decrease in Gini index. For Regression, the first column is the mean decrease in accuracy and the second the mean decrease in MSE. If importance=FALSE, the last measure is still returned as a vector.

importanceSD The “standard errors” of the permutation-based importance measure. For classi-fication, a p by nclass + 1 matrix corresponding to the first nclass + 1 columns of the importance matrix. For regression, a length p vector.

localImp a p by n matrix containing the casewise importance measures, the [i,j] element of which is the importance of i-th variable on the j-th case. NULL if localImp=FALSE.

ntree number of trees grown.

mtry number of predictors sampled for spliting at each node.

forest (a list that contains the entire forest; NULL if randomForest is run in unsupervised mode or if keep.forest=FALSE.

err.rate (classification only) vector error rates of the prediction on the input data, the i-th element being the (OOB) error rate for all trees up to the i-th. confusion (classification only) the

confusion matrix of the prediction (based on OOB data).

votes (classification only) a matrix with one row for each input data point and one column for each class, giving the fraction or number of (OOB) ‘votes’ from the random forest.

oob.times number of times cases are ‘out-of-bag’ (and thus used in computing OOB error estimate)

proximity if proximity=TRUE when randomForest is called, a matrix of proximity measures among the input (based on the frequency that pairs of data points are in the same terminal nodes).

mse (regression only) vector of mean square errors: sum of squared residuals divided by n.

rsq (regression only) “pseudo R-squared”: 1 - mse / Var(y).

test if test set is given (through the xtest or additionally ytest arguments), this component is a list which contains the corresponding predicted, err.rate, confusion, votes (for classification) or predicted, mse and rsq (for regression) for the test set. If proximity=TRUE, there is also a component, proximity, which contains the proximity among the test set as well as proximity between test and training data. (Breiman, Cutler, Liaw, & Wiener, 2015)

* Working of RandomForest:

Algorithm

### Decision tree learning

Trees that are grown very deep tend to learn highly irregular patterns. They [overfit](http://en.wikipedia.org/wiki/Overfitting" \o "Overfitting) their training sets, because they have [low bias, but very high variance](http://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff). Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model. (Wikipedia, 2015)

### Tree bagging

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set *X* = *x1*, …, *xn* with responses *Y*= *y1*, …, *yn*, bagging repeatedly selects a random sample with replacement of the training set and fits trees to these samples:

For *b* = 1, …, *B*:

* Sample, with replacement, *n* training examples from *X*, *Y*; call these *Xb*, *Yb*.
* Train a decision or regression tree *fb* on *Xb*, *Yb*.

After training, predictions for unseen samples *x'* can be made by averaging the predictions from all the individual regression trees on *x'*:

\hat{f} = \frac{1}{B} \sum_{b=1}^B \hat{f}_b (x')

or by taking the majority vote in the case of decision trees. (Wikipedia, 2015)

From bagging to random forests

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the *B* trees, causing them to become correlated.

Typically, for a dataset with *p* features, √*p* features are used in each split. (Wikipedia, 2015)

1. **Results and Discussion**

While analysing the data set we found that holidays in the store should be used to create distribution of sales. For each holiday, based on their date, we found with 3 parameters namely, width, skew and location (relative to the actual date) are need to be considered. We made these distributions on a daily grid and then sum them up to weekly totals. Using skewed distributions was a key here for us. We largely fit the parameters by eye to begin with.

For each store/department combination, after calculating the trend in the data, we used a linear model with L1 regularization to fit the holidays to the distended data. Then after subtracting the holiday fit and trend, we took an average of value for the week over the years to find the residual weekly cycle that was not due to the holidays.

Then we also fit the trend + de-seasonalized data using the Unemployment, Fuel Price and CPI, using another linear model with L1 regularization. We also calculated the missing data using a simple AR model. This fit will give a small improvement over using the pure trend.

We used csv over the first two years / last 39 weeks split to pick whether the trend was constant or linear. For example something happened at store 5, which caused a dramatic drop in sales across all departments, so we applied a step function to account for this.

The dataset used here is acquired from KDD which provided us with a large volume of inputs to analyse and provide accurate estimates. As this is a solution to provide forecast or predictions we require a large dataset, getting a dataset from KDD was very useful.

We found that since we only have 1 year of markdown data in the dataset it is impossible to extract anything much useful from it, as we could not see the effect from one year to the next.

Steps to execute the code:

* Copy the code folder with the csv files in the C:/ Documents folder for easy access.
* After installation open R Studio and look for the code folder in the right down-corner under the “home”
* Double click on the code folder and click on the project.Rproject
* Go to the R folder present inside the code folder.
* Double click on it.
* Look for the makeSubmission.R file.
* Click on it. It will appear in the left side editing window
* Press CTRL+A Click on Code>Run region>Run all from the tool bar
* The execution window will show it execution as started.
* Wait for some time. Once the execution is completed look for the output in the final folder in the right down corner.
* Click on the final.csv where you will get the output

The output is visible in the final.csv file and can be seen as follows:

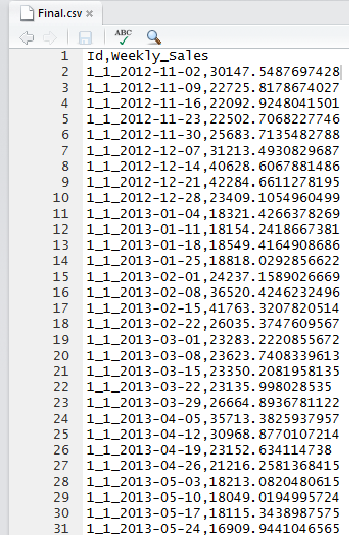


Figure 2 output

1. **Conclusion, Implications and Recommendation**

**5.1 Benefit**

The benefits of using Knowledge Discovery in Database would be the increment in sales, keeping the customers happy, efficiently obtaining the goals. This includes the design of a pattern on which the activity should be working. The store is to be analysed by considering different occasions and by considering different conditions like weather, fuel etc. This is beneficial in maintaining the inventory of the store and providing the customers the product they want. This is also helpful in maintaining the image and the brand value of the store. If the store needs some changes then it can even do it by following the pattern.

**5.2 Assumptions, Limitations, Future scope**

For example, if we consider animals they look for food with some kind of assumptions. Similarly a data analyst will also make assumptions for a task to be executed. (Murray & Gardiner, n.d.) This based on assumptions of sales using historical data and analysing it for mining purposes.

The limitations of this process may be considered as the availability of data. The data available is huge which affects the processing time. It takes more than a day to execute if we take the complete sets of data available. It is possible to execute fast such kind of programs if they can be implemented on the server and not in the personal laptops. It needs high processing CPU and storage capacity. 2GB RAM is not at all sufficient to run fast such kind of huge data.

The future scope of our project is we can implement it for huge data if we have such a big platform to work with.

We have implemented the project with an accuracy of 78 percentage. The project has executed successfully. We found that the sales estimation, for the year 2013, which our project provides is 78 percentage accurate in accordance to the reality.

# **References**

Breiman, L., Cutler, A., Liaw, A., & Wiener, M. (2015). Breiman and Cutler's random forests for classification and. *Package ‘randomForest’*.

code school. (n.d.). Retrieved from tryr.codeschool.com: http://tryr.codeschool.com/

Maimon, O., & Rokach, L. (n.d.). INTRODUCTION TO KNOWLEDGE DISCOVERY.

Murray, C., & Gardiner. (n.d.). In *Inquiring Organizations: Moving from Knowledge Management to Wisdom.*

Team, R. C., Wuertz, D., Setz, T., Chalabi, Y., Maechler, M., & Byers, J. W. (2015).

Rmetrics - Chronological and Calendar Objects. *Package ‘timeDate’*.

The R foundation. (n.d.). *The R Project for Statistical Computing*. Retrieved from r-project.org: http://www.r-project.org/

wikipedia. (2015, April 13). *Comma-separated values*. Retrieved from wikipedia.com: http://en.wikipedia.org/wiki/Comma-separated\_values

wikipedia. (2015, April 26). *Comma-separated values*. Retrieved from wikipedia.org: http://en.wikipedia.org/wiki/Comma-separated\_values

Wikipedia. (2015). *Random forest*. Retrieved from wikipedia.org: http://en.wikipedia.org/wiki/Random\_forest

wikipedia. (2015, March 5). *RStudio*. Retrieved from wikipedia.org: http://en.wikipedia.org/wiki/RStudio

Wikipedia. (2015, 5 3). *Rstudio*. Retrieved from wikipedia .org http://en.wikipedia.org/wiki/RStudio

1. **Appendix**