CPSC- 531 Advanced Database Systems

Store Sales Estimation

By: Gurpreet Kaur (891517658)

Team: Purva, Ajinkya

Table of Contents

[ABSTRACT 1](#_Toc418780460)

[INTRODUCTION 2](#_Toc418780461)

[PROBLEM STATEMENT 3](#_Toc418780462)

[LITERATURE REVIEW 4](#_Toc418780463)

[METHODOLOGY 7](#_Toc418780464)

[DISCUSSION 15](#_Toc418780465)

[RESULTS 16](#_Toc418780466)

[IMPLICATIONS 17](#_Toc418780467)

[CONCLUSION 18](#_Toc418780468)

[APPENDIX 19](#_Toc418780469)

[REFERENCES 20](#_Toc418780470)

# ABSTRACT

In today’s competitive world it is really important for a business not just to relay on the blooming situation today but try to improve sales and save money based on planning that is supported by the facts and backed up by historical data. In order to help a multi departmental store to plan in advance for next year based on the data for previous year, me and my team has used the Store Sales Estimation Approach.

Store Sales Estimation Approach has its roots in the availability and accuracy of the historical data and thus based on the historical trend, to predict the sales for next year. This is definitely benefit the management of the organization to see different buying trends in different regions and popularity of different items during some specific times such as special holidays.

In our project, we are using the sample data from KDD website for a multi-departmental store for past three years i.e. 2010, 2011, and 2012 to predict the weekly sales of each department for each of the different stores provided in the sample data.

These predictions will definitely help the management to plan ahead for inventory placements and hiring of personnel during special times of the year. They will also be able to know the popularity of different departments in different regions and thus they can base their promotions based on these facts.

So, this is definitely a handy tool for the management to play around with numbers and make good decision that will help the business to survive in this competitive era.

# INTRODUCTION

The Store Sales Estimation project uses the sample data from KDD website for a multi-departmental store to predict the quantity of sales for each department in its each store for next year. The sample data is from the years 2010, 2011 and 2012 and the prediction of the weekly sales for each department of each store is for the year 2013 based on the historical data provided.

Algorithms such as TimeDate and RandonForest are being used for this prediction and procedures for Knowledge Discovery in Databases are used. The programming language used in this project is R Programming which is known as the language of Statistics and is really helpful when dealing with data inport, cleaning, exploration, statistics and analysis.

The sample data consist of 4 files in Comma Separated Format named stores, train, test, and features. The details of these files are explained in the following sections.

# PROBLEM STATEMENT

Planning for the future so that an organization makes profit and saves money while being ahead of its competitors has always proved to be a challenge for the management team. Using advances in technology and the vast availability of historical data to come up with some concrete facts to back up decision making has been an ideal path used my most of the organizations now a days.

The proper way to import this huge amount of data and then using it for decision making is a problem. We are using R programming language to import our sample data which is in comma separated format and then using the packages like TimeDate and RandomForest to read and apply logic on this data so that we can get weekly estimates of the quantity of sales in each department of each store for next year is our goal. This will definitely provide management with a better idea on items popular during special holidays like Super Bowl, Labor Day, Thanksgiving, and Christmas so that they can plan on the inventory and markdowns of those items etc.

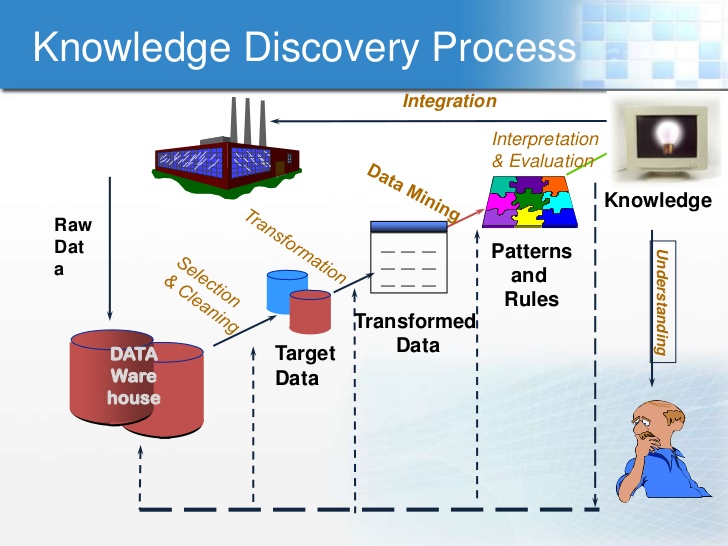
# LITERATURE REVIEW

Knowledge discovery in [databases](http://www.techopedia.com/definition/25827/knowledge-discovery-in-databases-kdd) (KDD) is the process of discovering useful knowledge from a collection of data. This widely used [data](http://www.techopedia.com/definition/25827/knowledge-discovery-in-databases-kdd) mining technique is a process that includes data preparation and selection, data cleansing, incorporating prior knowledge on data [sets](http://www.techopedia.com/definition/25827/knowledge-discovery-in-databases-kdd) and interpreting accurate solutions from the observed results.

Knowledge Discovery and Data Mining (KDD) is an interdisciplinary area focusing upon methodologies for extracting useful knowledge from data. The ongoing rapid growth of online data due to the Internet and the widespread use of databases have created an immense need for KDD methodologies. The challenge of extracting knowledge from data draws upon research in statistics, databases, pattern recognition, machine learning, data visualization, optimization, and high-performance computing, to deliver advanced business intelligence and web discovery solutions.

 Steps involved in the entire KDD process are:

1. Identify the goal of the KDD process from the customer’s perspective.
2. Understand application domains involved and the knowledge that's required
3. Select a target data set or subset of data samples on which discovery is be performed.
4. Cleanse and preprocess data by deciding strategies to handle missing fields and alter the data as per the requirements.
5. Simplify the data sets by removing unwanted variables. Then, analyze useful features that can be used to represent the data, depending on the goal or task.
6. Match KDD goals with data mining methods to suggest hidden patterns.
7. Choose data mining algorithms to discover hidden patterns. This process includes deciding which models and parameters might be appropriate for the overall KDD process.
8. Search for patterns of interest in a particular representational form, which include classification rules or trees, regression and clustering.
9. Interpret essential knowledge from the mined patterns.
10. Use the knowledge and incorporate it into another system for further action.
11. Document it and make reports for interested parties.



In our project we followed the same methodology which is explained below:

The first two steps require us to understand the problem and its scope and so we came up with Problem Statement as explained in the previous section.

The third step required us to choose our datasets and thus we chose the sample data provided on the KDD website for our datasets for a multi-departmental store.

The next two steps talk about cleaning and transforming data for proper use. Here after thoroughly analysing the datasets we figured out the missing information and replaced it with NA so that our program doesn’t has to deal with null values as they are always tricky to handle. Also, due to machine limitations we were not able to use the entire sample data provided on the KDD website and so we reduced the datasets to fifty percent of their contents for efficient computation on our machines.

Steps six to eight talk about choosing appropriate algorithms for the purpose of using this data to extract meaningful information. We chose two packages for this purpose: TimeDate and RandomForest which is actually based on the following three concepts:

* Decision tree: A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm.
* Tree bagging: This deals with averaging of the various decision trees that are produced in our case. Thus, it averages a given procedure over many samples, to reduce its variance.
* Random forest: This deals with the further refinement of bagged trees. Thus, it improves on bagging by “de-correlating” the trees.

For the implementation of the last steps, after using the algorithms discussed in the previous section, the weekly sales are calculated and the results are presented to the management for better decision making.

# METHODOLOGY

The language used for our purpose is R Programming Language which is free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.  R is a complete, interactive, object-oriented language: designed by statisticians, for statisticians. The language provides objects, operators and functions that make the process of exploring, modeling, and visualizing data a natural one. Complete data analyses can often be represented in just a few lines of code. The IDE used here is R Studio which is open source software as well. 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.

Our data is in CSV format which is an acronym for Comma Separated Values which stores [tabular](http://en.wikipedia.org/wiki/Tabular) data (numbers and text) in plain-text form. A CSV file consists of any number of [records](http://en.wikipedia.org/wiki/Record_(computer_science)), separated by line breaks of some kind; each record consists of [fields](http://en.wikipedia.org/wiki/Field_(computer_science)), separated by some other character or string, most commonly a literal comma or [tab](http://en.wikipedia.org/wiki/Tab_character#Tab_characters). Usually, all records have an identical sequence of fields. CSV files can be viewed in Microsoft Excel.

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 predicts the useful solution with the help of suitable logic.

The details of the sample files included in our project are:

**stores.csv**

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

**train.csv**

This is the historical training data, which covers to 2010-02-05 to 2012-11-01. Within this file you will find 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

**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.

**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):

SuperBowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13  
LaborDay: 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 = "")
* 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.

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
* 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, ...)

* 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 19
* 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.

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.

**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.

**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.

**OUR TAKE**

As we are quantitative approach, it is really important to maintain the accuracy of data so that the predictions are valid.

# DISCUSSION

The analysis of the results gave way to a number of facts such as:

* Popularity of certain departments in certain areas based on climate, kind and general wealth of the people.
* Considerable increase in the volume of sales of some departments throughout the stores during special holidays like Super Bowl, Labor Day, Thanksgiving, and Christmas.
* Nominal increase in the volume of sales of some departments during some special holidays mentioned above.
* Increase in sales of some departments due to holiday markdowns

These facts will definitely help the management to know about the:

* Buying trends of people in different regions.
* Their overall inventory requirements depending on the time of the year
* Their specific inventory requirements depending on the time of the year and location of the store
* Requirements for markdowns on special items
* Popularity of their products in different departments due to markdowns

The different ways that we analysed the data are discussed below:

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.

# RESULTS

Though the results are there for our analysis but the accuracy of these results to come to any concrete decision is questionable largely due to unavailability of enough data to show any significant variations in the trend.

It can be risky to predict future trends based on past three years especially when there is no enough clear variation to be found in the trends.

Though it can be used just as an indication combined with personal intelligence and experience of the management to make decisions related to popularity of products of different departments based on the location of the store.

# IMPLICATIONS

This project is an effort to help predict the quantity of sales for a multi departmental store for its various locations and departments within them. It shows the general trend and variation in trend during some special holidays:

* Super Bowl
* Labor Day
* Thanksgiving
* Christmas

Making good decisions that can save money for the store and enhance its sales based on the knowledge acquired through this project is one of the major advantages of Store Sales Estimation.

Though this projects has its limitations as well. The most important being the insufficiency of the dataset to make any concrete predictions on which important decisions can be based.

In future, we would definitely want to enhance the scope of our project and include more sample data so as so see some clear trends over a long period of time.

# CONCLUSION

The project results are 78% percent accurate and so we conclude that there is still a long way to go. The methodology that we implemented is good and can be applied to bigger set of data. This will definitely improve the results and thus we will be shown with some clear patterns to base our decisions on. The idea is definitely a handy tool for the management and definitely promises to be a good guide for important decision making for the management. This will help the organization attain competitive advantage by planning ahead and saving money where they see any scope

# APPENDIX

1. Download R-Studio which is an open source software
2. Download R-3.1.3. which is too an open source software
3. Copy the code folder with the csv files in the C:/ Documents folder for easy access.
4. After installation open R Studio and look for the code folder in the right down-corner under the “home”
5. Double click on the code folder and click on the project R project
6. Go to the R folder which is present inside the code folder and double click it.
7. Look for the makeSubmission.R file.
8. Click on it. It will appear in the left side editing window
9. Press CTRL+A Click on Code>Run region>Run all from the tool bar
10. The execution window will show it execution as started.
11. Wait for some time. Once the execution is completed look for the output in the final folder in the right down corner.
12. Click on the final.csv where you will get the output

# REFERENCES

1. <http://www.techopedia.com/definition/25827/knowledge-discovery-in-databases-kdd>
2. <http://researcher.watson.ibm.com/researcher/view_group.php?id=144>
3. <http://en.wikipedia.org/wiki/Decision_tree>
4. <http://www.inside-r.org/what-is-r>
5. <https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data>
6. Murray, C., & Gardiner. (n.d.). In *Inquiring Organizations: Moving from Knowledge Management to Wisdom.*