Walmart Store Sales Forecast

Aparna Kumar
aparna.kumar@sv.cmu.edu

Felix Amoruwa
felix.amoruwa@sv.cmu.edu

Feng Liang
feng.liang@sv.cmu.edu

Lloyd D'Silva
lloyd.dsilva@sv.cmu.edu

Mrikondu Barkakati
mirkondu.barkakati@sv.cmu.edu

Neha Arora
neha.arora@sv.cmu.edu

Vignan Uppugandla
vignan.uppugandla@sv.cmu.edu

ABSTRACT

A survey of time series and machine learning methods is presented and applied to predict the future values of weekly demand of specific store and department of a departmental retail chain. The effect of influencing parameters in weekly Consumer Price Index (CPI), holidays, unemployment, temperature, fuel price, and markdowns is considered. In setting up for evaluating the methods, candidate design and implementation of a data mining workflow is discussed. The performance of the methods is evaluated by comparing the results on the method based model on the Kaggle test data set and it's Weighted Mean Absolute Error (WMAE) with an emphasis on the holiday and markdown features. The comparison shows that that ensemble learning with combination of clustering and support vector based regression yields better prediction quality as measured by the WMAE.

1. INTRODUCTION

Forecasting is a crucial data mining problem, especially in context of business. It essentially means to predict future values of a temporal sequences. In this paper, we present the problem of predicting future weekly sales for a departmental retail chain. A number of complex, possibly mutually dependent factors can influence time series data. Given these factors we need to evaluate the appropriate contributing features, and data exploration methods. To this end, we discuss setting up of data workflow to enable experimentation and result gathering. Further, we need to look at appropriate data analysis methods in time series analysis, and machine learning; and to how to make them work appropriately for our end goal in prediction.

Hence, the objective of this paper is to provide a sample data analysis workflow and a discussion of time series and machine learning methods to predict future values of weekly sales of a retail store chain.

2. DATA DESCRIPTION

Dataset provided by Walmart contains historical sales data of Walmart for 45 stores from February 2010 to July 2013.

Three files: stores.csv, features.csv and train.csv contains the above mentioned data and test.csv has the data required for the prediction. sampleSubmission.csv contains the format required for submission to Kaggle. Below table mentions all the data provided by Walmart.

Table 1: Data Description

stores.csv	features.csv	train.csv
• # of stores (1 - 45) • Type of stores (A, B, C) • Size of the store	• Temperature • Fuel Price • Promotional Markdown (5 Types) • CPI • Unemployment rate • Holiday Week or not	• # of stores (1-45) • # of departments • Weekly sales • Holiday week or not

In the above mentioned files, stores.csv and features.csv has data available at store level. But in train.csv, the data is further drilled down to department level and has the most important Weekly sales data. features.csv and train.csv has data for weeks from February, 2010 to July, 2013. Overall, there are 45 stores, with each store belonging to one of the types: A, B or C and each store has departments varying from 1-99.

3. PRE-PROCESSING STEPS

Initial Data Set provided by Walmart has 4 .csv files apart from the sampleSubmission.csv in normalized format. We had to denormalize the data to have a better view of the whole data set and the dependency between the variables provided in the data set.

Following are the pre-processing steps that the data went through:

1. Import the data onto a MySQL database.
   Imported all the .csv files including the sampleSubmission.csv file which has the final set of data in the format required for Kaggle submission.
2. Identify all the primary-foreign key relationships between the tables.
   Data in stores and features are related through the store (number) value and the data in both these tables are at store level. Data in train table are available at department level are related with the store (number) value. Weekly Sales values for the stores at department level are available in the train table. Test data is related to the other tables with store and department value.
3. Using the stores, features and train tables are denormalized into a single table for further pre-processing. The same done with the stores, features and the test tables.
4. Clean the date variable with a proper date format across the whole data set.

Above pre-processing steps involved the provided variables from Walmart data set. For optimal weekly forecast, we also calculated more variables from the existing variables which can influence the prediction models.

Following are the few variables we introduced which have more influence on the models:

1. We divided the existing date variable into year, month and day variables. Month and Day had more influence on the models and year the least as the forecast has more dependency on the month and day of the sales than the year.
2. We identified the names of the holidays for the respective day and assigned them to a new variable.
3. We also calculated the last_holiday, days_from_last_holiday, next_holiday and days_to_next_holiday.
4. We created a new holiday binary variable for IsHoliday.
5. Another important variable which influenced the model is the last_year_weekly_sales.

Apart from the above changes, we had to convert the denormalized data to fit the format required by prediction models in WEKA. Models in WEKA only allowed single time series at a time. So we had to convert the department level weekly sales per store into additional variables for that week. This is one of the important pre-processing steps we had to perform for fitting models in WEKA.

4. EVALUATION CRITERIA

4.1. WMAE

The Kaggle submission results are based on Weighted Mean Absolute Error (WMAE).

$$\text{WMAE} = \frac{1}{\sum w_i} + \sum_{i=1}^{n} w_i |y_i - \hat{y}_i|$$

1. n is the number of rows
2. $\hat{y}_i$ is the predicted sales
3. y_i is the actual sales
4. w_i are weights. w = 5 if the week is a holiday week, 1 otherwise

4.2. SIC (SBC)

SAS provides selection criterion to specify what measurement should be used for the model fit for the time series. In this case, using the Schwarz Bayesian Information Criterion (SBC) value was optimal.

AIC selects a more highly parameterized model. SIC selects a model with fewer parameters. For forecasts SIC generally gives more accurate confidence intervals. But it can sometimes overstate the level of uncertainty, i.e. the confidence intervals will be too large. For model fitting to understand the dynamics of a system the AIC is better.

5. INFRASTRUCTURE SETUP

5.1. MySQL – Java/Weka

We used MySQL for the flexibility in data processing of the Kaggle input csv format and in creating feature sets for processing by WEKA GUI and API. WEKA provides a proven set of machine learning algorithms for data mining via Java APIs.

The MySQL - Java - WEKA - Java pipeline was constructed to accomplish the following steps:

1. Prepared a normalized data set in MySQL database instance to represent dates, features, and weekly sales data.
2. This database fed into an unnormalised data view in a java implementation. The first layer of the experiment set up enabled flexibility in fetching the selected data features into WEKA understandable data instance. This instance was set up to run on different architectures of different regression models.
3. Run experiments on the feature subset instances via WEKA API and derive training related scores.
4. Given such a tested regression model, the final test data set was prepared by combining with relevant features and then exporting to a Kaggle readable csv format.

5.2. R

One of the tools that they team decided to explore and use was R. R is a free open source “software environment for statistical computing and graphics”.[1] The basic setup was done by following the steps given in CRAN R official page.[2]

After installing R, the next step was to install R Studio by downloading the installer.[3]

R Studio is an IDE for R, which makes the interface more graphical and has several functionalities that ease programming in R such as an embedded text editor, import dataset button, help pane etc.[4]

Post installation completion, the denormalized data set and test data set were loaded in R Studio. R Studio was mainly used for time series forecasting. R script was written by utilizing the ‘forecast’ library package. This package contains the “tools and methods for displaying and analyzing univariate time series forecasts including **exponential smoothing** via **state space models** and automatic **ARIMA** modelling”.[5]

The R Script composed of 3 parts:

1. The first part created time series for weekly sales of each store and its department’s combination. The time series was used as the input to create the required model (ARIMA, Holt Winters etc). The model thus created was used to forecast 39 weeks into the future(max forecast weeks desired by Kaggle in test data)
2. The second part is used to generate the forecasted output in the format that Kaggle requires i.e. [id = Store_Department_Date, Weekly Sales = Forecast]
3. The last part makes sure only the dates that Kaggle test file has, is included in creating the final file to be uploaded.

6. METHOD CATEGORIES

6.6.1. Regression

Regression is a statistical technique to determine the linear relationship between two or more variables. Regression is primarily used for prediction and causal inference. In its simplest (bivariate) form, regression shows the relationship between one independent variable (X) and a dependent variable (Y), as in the formula below:

$$Y=\beta_0+\beta_1X+u$$

The magnitude and direction of that relation are given by the slope parameter (β_1), and the status of the dependent variable when the independent variable is absent is given by the intercept parameter (β_0). An error term (u) captures the amount of variation not predicted by the slope and intercept terms. The regression coefficient (R^2) shows how well the values fit the data.

Regression thus shows us how variation in one variable co-occurs with variation in another. What regression cannot show is causation; causation is only demonstrated analytically, through substantive theory.

The linear regression model

The simple (or bivariate) linear regression model is designed to study the relationship between a pair of variables that appear in a data set. The multiple LRM is designed to study the relationship between one variable and several of other variables.[6]

Multiple linear regression examines the linear relationships between one continuous response and two or more predictors.

In the simplest multiple regression with three variables, regression shows the relationship between two independent variables (X_1 and X_2) and a dependent variable (Y), as in the formula below:

$$Y=\beta_0+\beta_1X_1+\beta_2X_2+u$$

If the number of predictors is large, then before fitting a regression model with all the predictors, you should use stepwise or best subsets model-selection techniques to screen out predictors not associated with the responses.[7]

Nonlinear regression models
If the relationship between the variables being analyzed is not linear in parameters, a number of nonlinear regression techniques may be used to obtain a more accurate regression.

Of the variety of regression models available, we analyzed and tried few models on the given dataset. Regression methods such as Linear Regression, SMOReg, IBk, gbm etc that we tried and forecasted the weekly sales for the given Walmart store sales data set are further elaborated in the next section.[8]

6.2. Time-Series

Time-Series as the name suggests utilizes time as an independent variable and forecasts the future based on historical patterns in the data. Time series methods work on the principle that if data has "shown a consistent pattern in the past that is expected to recur in the future".[9] The manner in which time series makes use of these patterns are in the form of classifying them into four components: trend component, cyclical component, seasonal component, and irregular component. The trend component suggests if there is an upward or downward trend in data over the years. The cyclicity suggests if there is a pattern in the trend year over year. The seasonal component looks into the data within a year and tries to find a pattern. The three previous components compose the systematic pattern. The last component called irregular component represents the error or the random noise in the data and can be attributed to widespread events or just random human actions.

Based on the four components and how the methods identify and project these four patterns, time series can be divided into two broad categories: open-model time series techniques and fixed model time series techniques.[10]

A. **FMTS:** These methods assume that the time series have certain characteristics and then based on the classification of the data type, a fixed formula is applied. There are methods for data with no trend and/or seasonality, to varying weightage depending on how recent the data is, to methods that account for both trend and seasonality for forecasting. Some of methods under Fixed Method are moving average, exponential smoothing, adaptive smoothing, and Smoothing with trend and seasonality. Below table summarizes the methods:[11]

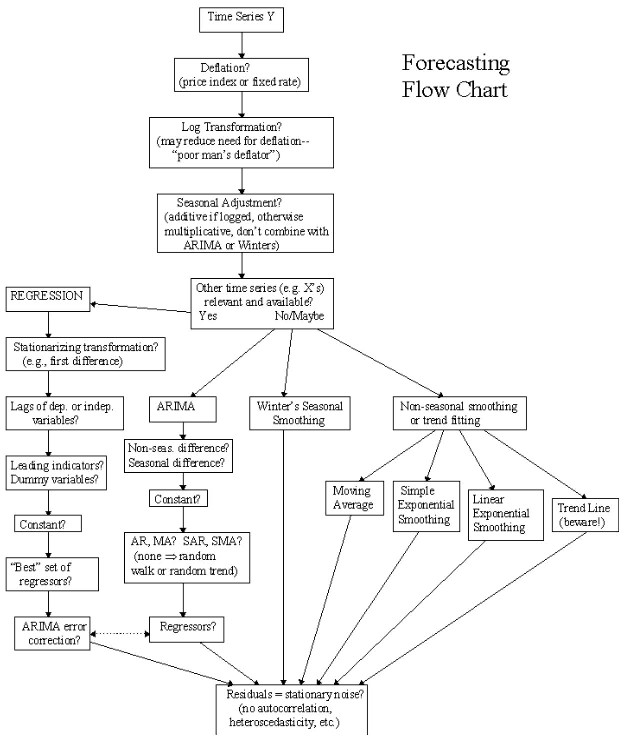
Table 2: FMTS Technique Selection Guidelines

Time Series Component Characteristics	*FMTS Technique*
Stable Level, with No Trend or Seasonality	Exponential Smoothing
Changing Level, with No Trend or Seasonality	Adaptive Smoothing
Level and Trend	Exponential Smoothing with Trend
Level, Trend, and Seasonality	Exponential Smoothing with Trend and Seasonality
Changing Level, Trend, and Seasonal Patterns	AEES

B. **OMTS:** OMTS techniques first try to analyze the components) nature as well as existence of the components) in the time series and then decide on what on which forecasting formulae unique to that time series to use. Thus, the formulae are "open" until the analysis of the components are made. Various OMTS methods exist, such as decomposition analysis, spectral analysis, fourier analysis, and auto-regressive moving average (ARMA) or Box-Jenkins analysis. Once the formulae for modeling is decided then the forecasts are made.[12]

The diagram below shows a flowchart of how forecasting for time series is done.[13]

Figure 1: Time Series Forecasting Flow Chart


Forecasting Flow Chart
Time Series Y
Deflation? (price index or fixed rate)
Log Transformation? (may reduce need for deflation-- "poor man's deflator")
Seasonal Adjustment? (additive if logged, otherwise multiplicative, don't combine with ARIMA or Winters)
Other time series (e.g. X's) relevant and available? Yes No/Maybe
REGRESSION
Stationarizing transformation? (e.g., first difference)
Lags of dep. or indep. variables?
Leading indicators? Dummy variables?
Constant?
"Best" set of regressors?
ARIMA error correction?
ARIMA
Non-seas. difference? Seasonal difference?
Constant?
AR, MA? SAR, SMA? (none ⇒ random walk or random trend)
Regressors?
Winter's Seasonal Smoothing
Non-seasonal smoothing or trend fitting
Moving Average
Simple Exponential Smoothing
Linear Exponential Smoothing
Trend Line (beware!)
Residuals = stationary noise? (no autocorrelation, heteroscedasticity, etc.)

6.3. Ensemble

The ensemble methods have been mostly studied in supervised and unsupervised learning communities separately, but they are powerful for improving the robustness and accuracy of both supervised and unsupervised solutions. And they share the same basic principles. For example, they tend to yield better results when there is a significant diversity among the models as the combination of diversified base models strengthen weak models. In other words, an ensemble method is a technique for combining many weak learners in an attempt to produce a strong learner. The shortcoming of ensemble method is that evaluating the prediction of an ensemble typically requires more computation than evaluating the prediction of a single model, so ensembles may be thought of as a way to compensate for poor learning algorithms by performing a lot of extra computation. Fast algorithms such as decision trees are commonly used with ensembles like random forest, although slower algorithms can benefit from ensemble techniques as well.

As the goal of our data analytic task is to figure out a good prediction of Walmart's future weekly sales, a supervised like learning process, we think ensemble methods like random forest or random committee (provided by Weka) may help us to find a good model for this purpose.

7. METHODS

7.1. Linear Regression

lm is the package in R which helps us perform linear regression. It provides us the option of performing both single and multiple values regressions.

In our case we performed the following steps to decide on the model and perform predictions.

1. Imported the denormalized data and cleaned certain values such as dates, booleans as per requirements.
2. Next we created a test and a train set from the data itself. We ran the following function
   (1:nrow(mydata), nrow(mydata)*0.66)
3. Following step was to create the model. This was done running the lm function on single or multiple values as needed.
   a. Single value
   lm(Weekly_Sales ~ ., data = train)
   b. Multi value
   lm(Weekly_Sales ~ Date + CPI + …., data = train)
4. Prediction models were run to basically deduce the RMSE (Root mean square error).

The **root-mean-square deviation (RMSD)** or **root-mean-square error (RMSE)** is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. Basically, the RMSD represents the sample standard deviation of the differences between predicted values and observed values. These individual differences are called residuals when the calculations are performed over the data sample that was used for estimation, and are called *prediction errors* when computed out-of-sample. The RMSD serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. RMSD is a good measure of accuracy, but only to compare forecasting errors of different models for a particular variable and not between variables, as it is scale-dependent.[14]

The function used to predict the model is **predict(fit, newdata=test)**

The error formula which was used **mean((test $Weekly_Sales - yHat)^2)**

7.2. SMOReg

SMOReg refers to the Sequential Minimal Optimization (SMO) algorithm for Support Vector Machine (SVM) regression. SMOReg is a supervised learning algorithm which can be used for classification or regression. SMOreg's parameters can be learned using various algorithms, the most popular of them being RegSMOImproved, which is the default RegOptimizer in Weka.[15] [16]

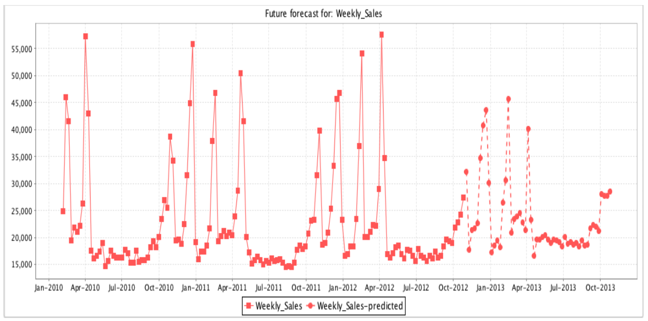
Weka also provides us with the ability to change various parameters in SMOReg to optimize the output. The following are the parameters that we tried tweaking:

c - The complexity parameter
filterType - Normalize or standardize the training data
kernel - The kernel to use for SMOReg

Of the above options, the best results were produced by changing the kernel from a PolyKernel to a NormalizedPolyKernel.

Using the Weka Java API, we split up the Walmart dataset into all 4,455 store-department combinations and ran SMOReg on each time series. We then forecasted this data 39 weeks into the future. Below is a sample forecast of the store1-department1 combination using SMOReg.

Figure 2: SMOReg Forecast - Store 1 Dept 1



SMOReg was one of our highest performing algorithms on the Kaggle leaderboard. The vanilla SMOReg algorithm was ranked 312 with a RMSE of 3527 on the Kaggle leaderboard. The SMOReg algorithm with a NormalizedPolyKernel was ranked 244 with a RMSE of 3274 on the Kaggle leaderboard. The Java code to integrate with the WEKA API is attached in the appendix.

7.3. IBk

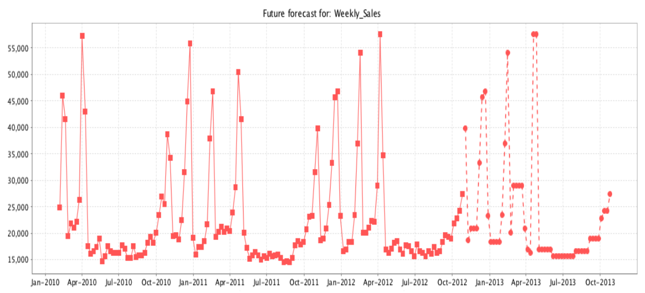
IBk refers to Instance Based Kth Nearest Neighbor algorithm. The Euclidean distance is used by default. The sum of the squared differences between normalized attribute values is computed; this is then normalized by the number of attributes in the data; finally the square root is taken. Following this, weighting can be applied to the distances. It may return k neighbors. If there are ties in the distance, neighbors are voted to form the final classification.[17]

Weka also provides us with the ability to change various parameters in IBk to optimize the output. The following are the parameters that we tried tweaking:

KNN - The number of neighbors to use.
nearestNeighborSearchAlgorithm - The algorithm to use to find the nearest neighbor. LinearNNSearch is used by default.

Using the Weka Java API, we split up the Walmart dataset into all 4,455 store-department combinations and ran IBk on each time series. We then forecasted this data 39 weeks into the future. Below is a sample forecast of the store1-department1 combination using IBk.

Figure 3: IBk Forecast - Store 1 Dept 1



IBk was one of our highest performing algorithms on the Kaggle leaderboard. The vanilla IBk algorithm was ranked 383 with a RMSE of 3444 on the Kaggle leaderboard. The Java code to integrate with the WEKA API is attached in the appendix.

7.4. Generalized Boosted Regression Model (gbm)

gbm is an R package that implements the generalized boosted modeling framework. Boosting is the process of iteratively adding basis functions in a greedy fashion so that each additional basis function further reduces the selected loss function. This implementation closely follows Friedman's Gradient Boosting Machine (Friedman, 2001).

In addition to many of the features documented in the Gradient Boosting Machine, gbm offers additional features including the out-of-bag estimator for the optimal number of iterations, the ability to store and manipulate the resulting gbm object, and a variety of other loss functions that had not previously had associated boosting algorithms, including the Cox partial likelihood for censored data, the poisson likelihood for count outcomes, and a gradient boosting implementation to minimize the AdaBoost exponential loss function.[18]

In most of the other models that we employed for forecasting store sales, we used per store per department combination, but we tried a different approach with training the data on per department basis.

Steps followed in R for forecasting Walmart store sales using gbm package:

1. Importing the stores, features, train, test and sampleSubmission dataset into R.
2. Merging the data for creating denormalized train and test data.
3. Converting the date into the required format and splitting it into Year, Month and Day variables
4. Creating additional variables such as day_index, holiday_binary, Holiday, last_holiday, next_holiday, days_from_next_holiday, days_to_next_holiday, Last_Year_Sales52 (weekly sales of last year).
5. Now we train the model for each department using **gbm.fit** and number of trees as 5000. Then inside the loop we check the performance of the model using **gbm.perf** for each department.
6. Summary of the model provides the influence of each of the variable on the model. Variables with least influence are weighed less.
7. Now we use the output of optimum number of iterations from the model performance and predict the output on the test data.
8. After all the departments are trained and predicted, we write the output in the Kaggle submission format to a .csv file.

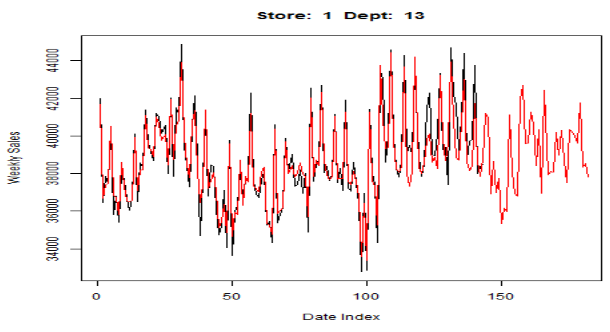
We played around with various parameters available in gbm.fit model till optimum prediction is achieved which had achieved lowest rank and Weighted Mean Absolute Error on Kaggle submission. They are explained below:

- **n.trees:** The total number of trees to fit. This is equivalent to the number of iterations and the number of basis functions in the additive expansion. We selected 5000 for the model to adjust the loss function
- **distribution:** We used Laplace and Gaussian distribution as the seasonality of the data is appropriate for these distributions.
- **shrinkage:** a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction.
- **interaction.depth:** The maximum depth of variable interactions. 1 implies an additive model, 2 implies a model with up to 2-way interactions, etc.
- **n.minobsinnode:** minimum number of observations in the trees terminal nodes.
- **bag.fraction:** The fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomness into the model fit. This value has to be selected carefully to reduce overfitting.

Overall, high interaction depth and low minobservations can help train the model properly for optimum results.

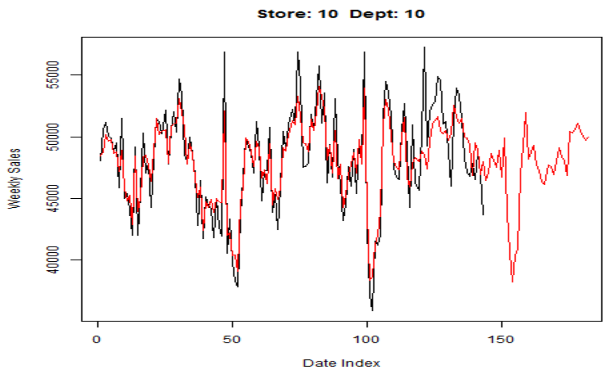
Sample forecast for train and test data for Store 1 and Dept 13:

Figure 4: gbm Forecast - Store 1 Dept 13



Sample forecast for train and test data for Store 10 and Dept 10:

Figure 5: gbm Forecast - Store 10 Dept 10



Insights:

We played around with parameters such as interaction depth, n.minobsinnode and bag.fraction. High interaction depth with a combination of minimum value of n.minobsinnode can achieve optimum results. Of course bag.fraction is to be adjusted to avoid overfitting of the model. On the whole, the model which has the least rank and weighted mean average error has the following parameters:

- Shrinkage: 0.1
- interaction.depth: 10
- n.minobsinnode: 2
- bag.fraction: 0.7

This gbm algorithm was ranked 472 with a WMAE of 5871 on the Kaggle leaderboard.

The above model can still be improved with adjusting of the above parameters, but a customized gbm model with own function can also achieve better results which is our next step.

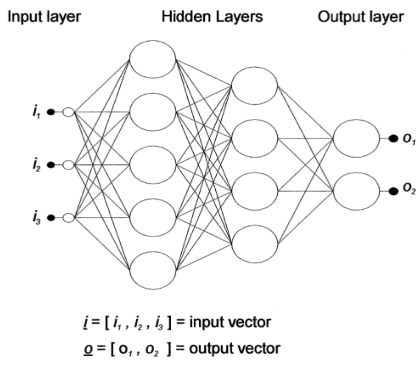
The script used for pre-processing of the dataset and prediction models is attached in the Appendix.

7.5. Perceptron

The multilayer perceptron consists of a system of simple interconnected neurons, or nodes, as illustrated in Fig [insert reference], which is a model representing a nonlinear mapping between an input vector and an output vector. The nodes are connected by weights and output signals which are a function of the sum of the inputs to the node modified by a simple nonlinear transfer, or activation, function. It is the superposition of many simple nonlinear transfer functions that enables the multilayer perceptron to approximate extremely non-linear functions. If the transfer function were linear then the multilayer perceptron would only be able to model linear functions. Due to its easily computed derivative a commonly used transfer function is the logistic function. The output of a node is scaled by the connecting weight and fed forward to be an input to the nodes in the next layer of the network. This implies a direction of information processing, hence the multilayer perceptron is known as a feed-forward neural network.

The terms input and output vectors refer to the inputs and outputs of the multilayer perceptron and can be represented as single vectors, as shown in Fig [insert reference]. A multilayer perceptron may have one or more hidden layers and finally an output layer. Multilayer perceptrons are described as being fully connected, with each node connected to every node in the next and previous layer. By selecting a suitable set of connecting weights and transfer functions, it has been shown that a multilayer perceptron can approximate any smooth, measurable function between the input and output vectors (Hornik et al., 1989).

Figure 6: A Multilayer Perceptron Model


Input layer
Hidden Layers
Output layer
i_1
i_2
i_3
o_1
o_2
$\underline{i} = [i_1, i_2, i_3]$ = input vector
$\underline{o} = [o_1, o_2]$ = output vector

Multilayer perceptrons have the ability to learn through training. Training requires a set of training data, which consists of a series of input and associated output vectors. During training the multilayer perceptron is repeatedly presented with the training data and the weights in the network are adjusted until the desired input—output mapping occurs. Multilayer perceptrons learn in a supervised manner. During training the output from the multilayer perceptron, for a given input vector, may not equal the desired output. An error signal is defined as the difference between the desired and actual output. Training uses the magnitude of this error signal to determine to what degree the weights in the network should be adjusted so that the overall error of the multilayer perceptron is reduced. There are many algorithms that can be used to train a multilayer perceptron, for our experiments we used the backpropagation algorithm.[19]

The backpropagation algorithm is summarized below: [20]

1. Initialize network weights
2. Present first input vector, from training data, to the network
3. Propagate the input vector through the network to obtain an output
4. Calculate an error signal by comparing actual output to the desired (target) output
5. Propagate error signal back through the network
6. Adjust weights to minimize overall error
7. Repeat steps 2—7 with next input vector, until overall error is satisfactorily small

It has been shown that the multilayer perceptron can be trained to approximate virtually any smooth, measurable function[21.] Unlike other statistical techniques the multilayer perceptron makes no prior assumptions concerning the data distribution. It can model highly non-linear functions and can be trained to accurately generalize when presented with new, unseen data.

Specifically, we had observed correlation of weekly sales with a subset of mutually dependent features and that the data had cyclical and irregular components. The univariate time series analysis and other regression methods we considered:

1. Needed to make assumptions about underlying trends.
2. Work well for single time series with some hidden regularity.

Univariate time series analysis showed us a strong irregular component and to address the same we considered the neural network method in multilayer perceptron via backpropagation training (MLP), thus we need to handle multidimensional time series with mutual non-linear dependencies.

Approach: [22]

We set up two processed streams of features by converting the dates to cycling numbers denoting weeks, with the middle of the year as 1. Another column for the holiday was added giving a weight of 10 to the holiday week, 5 to before and after holiday weeks, and a weight of 1 to other weeks in the year. For the 5 markdown fields, we filled in the missing markdown values by extrapolating from previous year or near time data.

To model the MLP with backpropagation, we used 10% of the provided training data set for validation, while 66% of the remaining was used for training, and rest for test.

Using the setup mysql-java-weka-java experiment workbench, experiments on the following data partitions were run to determine the fit:

1. store type
2. department
3. store, department
4. store, department, included new date features (week + holiday)
5. store, department, included new date features (week + holiday), removed markdown features

The experiments used WEKA's MLP implementation to apply backpropagation learning on different combination of input feature sets MultilayerPerceptron[23] and MLPRegressor[24]. Given the test data format and the results, the feature set: store, department, week + holiday features, while removing the markdown features gave the best results. As per our initial attempts, we were able to train the MultilayerPerceptron based architecture better than the MLPRegressor and hence, the later experiments were dedicated to improving that architecture. WEKA's MultilayerPerceptron autobuild network feature was used to build the architecture initially. The best results were observed on a 7 hidden layer network. This may not be ideal given smaller feature set and proneness of the algorithm to overfit. Steps to improvement included preparing processed feature sets for input and adjusting the learning parameters in the following:

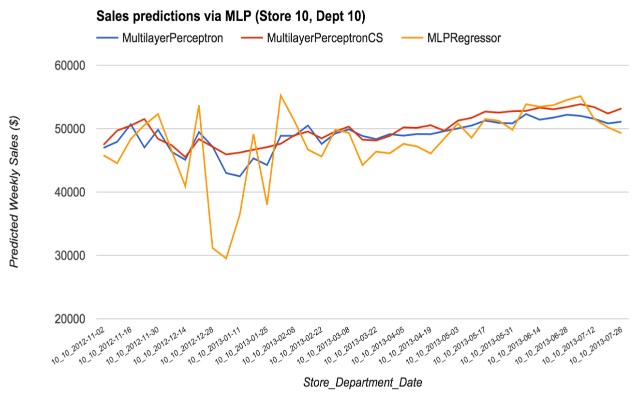
1. Backpropagation learning rate
2. Backpropagation learning momentum
3. Backpropagation training epochs
4. Threshold for number of consecutive errors
5. Learning rate decay

The final iteration of the results submitted to Kaggle with MultilayerPerceptron based 7 layer (autogenerated) architecture with the following learning parameters resulted in a Weighted Mean Absolute Error (WMAE) of 3482.79, as defined by the Kaggle competition

1. Learning rate = 0.25
2. Momentum = 0.4
3. Training time = 500
4. Validation Threshold = 20

In comparison, we could obtain a best WMAE of 5038 for the MLPRegressor implementation with Number of Functions = 2; Ridge = 0.05; Tolerance = 1.0E-4, with conjugate gradient descent enabled. MLP Regressor trains a multilayer perceptron with one hidden layer using WEKA's Optimization class by minimizing the squared error plus a quadratic penalty with the BFGS method. The ridge parameter is used to determine the penalty on the size of the weights.

Figure 7: Perceptron forecast - Store 10 Dept 10



In summary, we attempted to model for the irregular behavior in the time series with a perceptron-backpropagation implementation with average results. It would need more work to look into creating appropriate input features and ensuring that the build model is not overtrained. We have let WEKA autogenerate the MLP architecture, which leads to a big hidden layer, which might not be very effective. It will help to explore a simpler architecture (lesser hidden layers) with appropriate cost functions and regularization parameters to ensure that the architecture itself is not limiting the learning and recall performance.

7.6. Random Forest

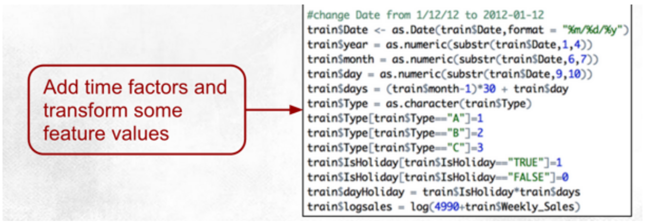
Random Forest is an ensemble learning method for classification and regression that constructs a number of decision trees on train data and outputs the class that is the mode of the classes output by individual trees. More specifically, Random Forest is a combination of tree predictors where each tree depends on the values of a random vector sampled independently with the same distribution for all trees in the forest. The basic principle is that a group of "weak learners" can come together to form a "strong learner". Random Forest is a wonderful tool for making predictions.

Considering that Random Forest has few parameters to tune and can be easy to use with default parameter settings, it could be a simple tool without having a model or to produce a reasonable model fast and efficiently. And we can safely make more accurate predictions without most basic mistakes common to other methods. All of these characters of Random Forest are very suitable for our case, and what's more, as we also want to create more time factors based on current date we have and abandon factors like fuel price, CPI, temperature which are not that relevant, we think Random Forest may be a useful tool for our more randomized new data.

Here are what we do with R programming to experiment this method (R code will be in appendix):

1. Denormalize historical data and remove the unnecessary fields like fuel price, CPI, temperature.
2. Create fields like year, month, day, days and weight on holiday by duplication.

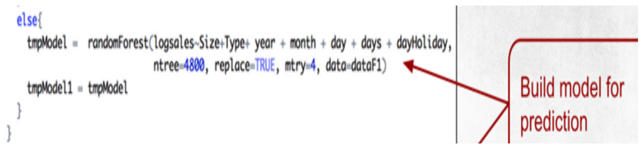
Figure 8: Adding time factor fields based on date



```
#change Date from 1/12/12 to 2012-01-12
train$Date <- as.Date(train$Date,format = "%m/%d/%y")
train$year = as.numeric(substr(train$Date,1,4))
train$month = as.numeric(substr(train$Date,6,7))
train$day = as.numeric(substr(train$Date,9,10))
train$days = (train$month-1)*30 + train$day
train$Type = as.character(train$Type)
train$Type[train$Type=="A"]=1
train$Type[train$Type=="B"]=2
train$Type[train$Type=="C"]=3
train$IsHoliday[train$IsHoliday=="TRUE"]=1
train$IsHoliday[train$IsHoliday=="FALSE"]=0
train$dayHoliday = train$IsHoliday*train$days
train$logsales = log(4990+train$Weekly_Sales)
```

3. Try different combinations of ntree(number of trees) and mtry(number of randomly selected attributes) to find one model with smallest Weighted Mean Absolute Error.

Figure 9: Building models with tree number and selected attribute number



```
  else{
    tmpModel =  randomForest(logsales~Size+Type+ year + month + day + days + dayHoliday,
                             ntree=4800, replace=TRUE, mtry=4, data=dataF1)
    tmpModel1 = tmpModel
  }
}
```

4. Predict for all departments of all stores and fill the submission file.

We got a rank of 475 on Kaggle leaderboard with the WMAE 6077 which seems to be a so method for our case. Better result may be gotten by keeping some original attributes and increase the number of trees when generating the model, and that would require a huge amount of time.

7.7. Holt Winters

Holt Winters (HW) is a fixed model time series technique that was proposed by Peter Winters in 1960.This method built over Holt's linear model (suitable only for non-seasonal data) and it extended to suit data that had both seasonality and trend. Thus, this method is also known as Triple Exponential Smoothing, which is an extension of the Basic Smoothing Exponential Model.[25]

Seasonality can be classified as additive or multiplicative. Additive seasonality refers when the seasonal change can be quantified as a fixed increment or decrement over the current average values of the time series. Multiplicative seasonality on the other hand, depicts increasing amplitudes of seasonal change and the variation can be expressed as a multiple of the current average value of the time series.

HW method caters to both the two seasonality types and thus there an Additive model version as well as Multiplicative model. HW estimates the level, trend and seasonal components at a given point in time and utilize the respective smoothing parameters alpha, beta, and gamma to do so. These parameters vary in magnitude (between 0 and 1) and the value of these parameters signify the weightage given to recent data points to forecast the future values.

Additive Model: As discussed above the forecast is a sum of level, trend and seasonal component. The formula is as follows: [26]

Level $L_t = \alpha(y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1})$
Trend $b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}$
Seasonal $S_t = \gamma(y_t - L_t) + (1-\gamma)S_{t-s}$
Forecast $F_{t+k} = L_t + kb_t + S_{t+k-s}$

Where s is the length of the seasonal cycle, for $0 \le \alpha \le 1$, $0 \le \beta \le 1$ and $0 \le \gamma \le 1$.

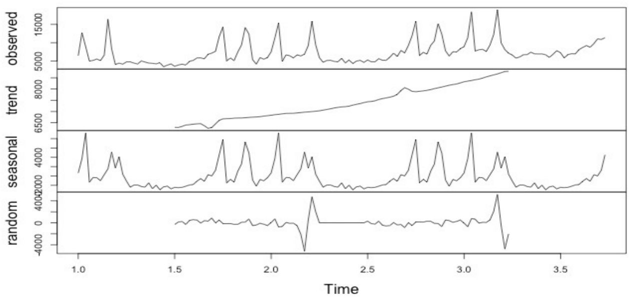
Multiplicative Model: This model forecasts future values by multiplying the level and trend components with the seasonal component. The formula is as follows: [27]

Level $L_t = \alpha\, y_t\, S_{t-s} + (1-\alpha)(L_{t-1} + b_{t-1})$
Trend $b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}$
Seasonal $S_t = \gamma\, y_t\, L_t + (1-\gamma)S_{t-s}$
Forecast $F_{t+k} = (L_t + kb_t)S_{t+k-s}$

Where s is the length of the seasonal cycle, for $0 \le \alpha \le 1$, $0 \le \beta \le 1$ and $0 \le \gamma \le 1$.

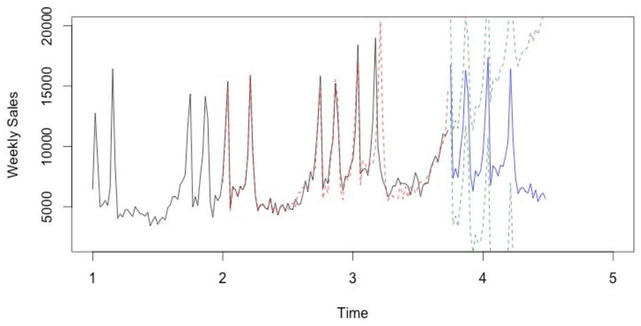
Upon decomposing the Walmart weekly sales time series for each Department and Store combination, it was determined that they each had an upward trend, a strong additive seasonal component as well as a random component. This decomposition confirmed the applicability of Holt Winter's Additive method to both model and forecast future weekly sales values. [28]

Figure 10: Decomposition of weekly sales for Store 1 Dept 3



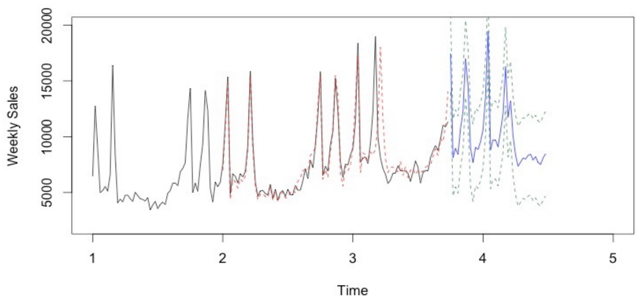
Using the additive model in R, the two years weekly sales data was fed into as input.

Initially several attempt was made by manually entering the smoothing coefficients for fitting the model. In one of the attempts the following values of smoothing coefficients were used: alpha = 0.2, beta = 0.1, gamma = 0.1. The Root Mean Squared Error was recorded as RMSE = 6171.276. The forecasting using this model was as follows:



The blue line represents the point forecast and the two green lines represent the 95% prediction interval. As it can be seen, the two lines are diverging and hence it depicts the level of prediction error that forecasts will have.

The next attempt utilized the auto selection of smoothing coefficients by R and hence no values of alpha, beta and gamma were entered explicitly into the mode. The Generated values of coefficients were alpha = 0.0314, beta = 1, gamma = 0.6048. The Root Mean Squared Error for this model was RMSE = 6160.105. The forecasting using this model was as follows:



The green lines (now are almost parallel) representing 95% prediction interval for this forecasting model, have considerable improvement over the previous model. This change can also be seen in the decrease of RMSE value.
This forecasted values for 39 weeks in future using the latter model was uploaded to Kaggle. The ranking of the result set was 323. The Weighted Mean Average Error for the uploaded result was 3612.74.

The learnings and from exploring this model were as follows:

1. Holt Winters is suitable for this data set as it has a strong seasonality component.
2. The seasonality is additive in nature

A future step for the further exploration would be to analyze the random component of the data.

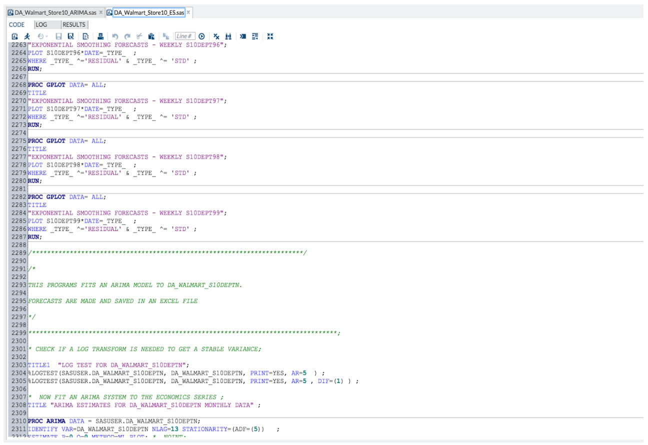
7.8. Exponential Smoothing & ARIMA Using SAS

Exponential Smoothing is a method for extrapolative forecasting from series data seeking to isolate trends or seasonality from irregular variation. When used for forecasting, exponential smoothing uses weighted average of the past data. The effect of recent observations is expected to decline exponentially over time. The further back along the historical time path one travels, the less influence each observation has on the forecasts.

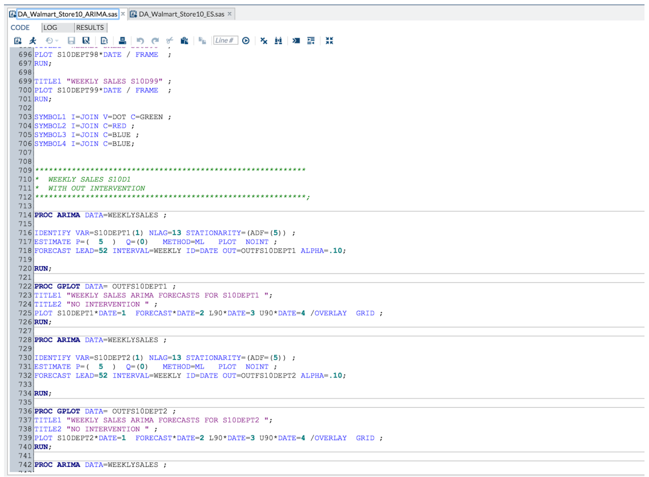
Referred to as an [ARIMA(p,0,q)] model, ARIMA stands for autoregressive integrated moving average, and is also an algorithm used for forecasting time-series data. In this notation, the p is the order of the autoregressive process and the q is the order of the moving average process. With this stationary model, a zero in the middle position signifies the order of differencing required. If there are autoregressive and moving average components to the differenced series, such a series may be modeled as an ARIMA (p,d,q) model, where d is the order of differencing that is required to render the series stationary. In sum, autoregression is the extent to which current output observation is a function of past outputs of the system. The order of autoregression signifies the number of previous observations of which the series is a significant function. The moving average process is only a function of a finite number of past shocks to the system.

With the SAS software suite as one of the tools used for evaluating the dataset, the first step employed was coding exponential smoothing and ARIMA algorithms for each store and its respective departments. The steps taken for this process were to:

1. Import the Walmart dataset
2. Run the dataset against the exponential smoothing algorithms for all 45 stores and each of their 99 departments
3. Run the dataset against the ARIMA (with and without Intervention) algorithms for all 45 stores and each of their 99 departments
    a. Check if the series needs to be transformed by taking logs.
    b. Check if the series needs to be first differenced. This is done in the IDENTIFY step of PROC ARIMA with the augmented Dickey-Fuller test. In PROC ARIMA, if the series needs to be first-differenced you enter a “(1)” when you specify the variable that is being studied. SAS then does the first difference transformation, estimates the model and will perform the inverse transformation before reporting the forecasts.
    c. Use the ACF and PACF to determine the order of p and q. The functions are estimated in the IDENTIFY step of PROC ARIMA.
    d. Estimate the model. Check for significant parameters. Check for the size of the SBC. Check for the significance of the residuals’ ACF and PACF. These are all calculated and reported in the ESTIMATE step of PROC ARIMA.
    e. Search different p and q values until the best model is determined. The best model will have: all its parameters significant, tests will imply that its residuals are NOT significantly different from white noise and its SBC will be the smallest of the models estimated.
    f. Use the FORECAST step to generate forecasts and confidence intervals. This step will print the forecasts and the confidence intervals.
    g. Apply any needed inverse transformations.
4. Plot the data, the forecasts and the confidence intervals.
5. Save the data, forecasts and confidence intervals in an Excel file.
6. Consolidate forecasts for benchmarking



```
"EXPONENTIAL SMOOTHING FORECASTS - WEEKLY S10DEPT96";
PLOT S10DEPT96*DATE=_TYPE_  ;
WHERE _TYPE_ ^='RESIDUAL' & _TYPE_ ^= 'STD' ;
RUN;

PROC GPLOT DATA= ALL;
TITLE
"EXPONENTIAL SMOOTHING FORECASTS - WEEKLY S10DEPT97";
PLOT S10DEPT97*DATE=_TYPE_  ;
WHERE _TYPE_ ^='RESIDUAL' & _TYPE_ ^= 'STD' ;
RUN;

PROC GPLOT DATA= ALL;
TITLE
"EXPONENTIAL SMOOTHING FORECASTS - WEEKLY S10DEPT98";
PLOT S10DEPT98*DATE=_TYPE_  ;
WHERE _TYPE_ ^='RESIDUAL' & _TYPE_ ^= 'STD' ;
RUN;

PROC GPLOT DATA= ALL;
TITLE
"EXPONENTIAL SMOOTHING FORECASTS - WEEKLY S10DEPT99";
PLOT S10DEPT99*DATE=_TYPE_  ;
WHERE _TYPE_ ^='RESIDUAL' & _TYPE_ ^= 'STD' ;
RUN;

/*********************************************************************/

/*

THIS PROGRAMS FITS AN ARIMA MODEL TO DA_WALMART_S10DEPTN.

FORECASTS ARE MADE AND SAVED IN AN EXCEL FILE

*/

***********************************************************************************;

* CHECK IF A LOG TRANSFORM IS NEEDED TO GET A STABLE VARIANCE;

TITLE1  "LOG TEST FOR DA_WALMART_S10DEPTN";
%LOGTEST(SASUSER.DA_WALMART_S10DEPTN, DA_WALMART_S10DEPTN, PRINT=YES, AR=5  ) ;
%LOGTEST(SASUSER.DA_WALMART_S10DEPTN, DA_WALMART_S10DEPTN, PRINT=YES, AR=5 , DIF=(1) ) ;

*  NOW FIT AN ARIMA SYSTEM TO THE ECONOMICS SERIES ;
TITLE "ARIMA ESTIMATES FOR DA_WALMART_S10DEPTN MONTHLY DATA" ;

PROC ARIMA DATA = SASUSER.DA_WALMART_S10DEPTN;
IDENTIFY VAR=DA_WALMART_S10DEPTN NLAG=13 STATIONARITY=(ADF=(5))   ;
```



```
PLOT S10DEPT98*DATE / FRAME  ;
RUN;

TITLE1 "WEEKLY SALES S10D99" ;
PLOT S10DEPT99*DATE / FRAME  ;
RUN;

SYMBOL1 I=JOIN V=DOT C=GREEN ;
SYMBOL2 I=JOIN C=RED ;
SYMBOL3 I=JOIN C=BLUE ;
SYMBOL4 I=JOIN C=BLUE;


*************************************************************
*  WEEKLY SALES S10D1
*  WITH OUT INTERVENTION
*************************************************************;

PROC ARIMA DATA=WEEKLYSALES ;

IDENTIFY VAR=S10DEPT1(1) NLAG=13 STATIONARITY=(ADF=(5)) ;
ESTIMATE P=(  5  )  Q=(0)   METHOD=ML   PLOT  NOINT ;
FORECAST LEAD=52 INTERVAL=WEEKLY ID=DATE OUT=OUTFS10DEPT1 ALPHA=.10;

RUN;

PROC GPLOT DATA= OUTFS10DEPT1 ;
TITLE1 "WEEKLY SALES ARIMA FORECASTS FOR S10DEPT1 ";
TITLE2 "NO INTERVENTION " ;
PLOT S10DEPT1*DATE=1  FORECAST*DATE=2 L90*DATE=3 U90*DATE=4 /OVERLAY  GRID ;
RUN;

PROC ARIMA DATA=WEEKLYSALES ;

IDENTIFY VAR=S10DEPT2(1) NLAG=13 STATIONARITY=(ADF=(5)) ;
ESTIMATE P=(  5  )  Q=(0)   METHOD=ML   PLOT  NOINT ;
FORECAST LEAD=52 INTERVAL=WEEKLY ID=DATE OUT=OUTFS10DEPT2 ALPHA=.10;

RUN;

PROC GPLOT DATA= OUTFS10DEPT2 ;
TITLE1 "WEEKLY SALES ARIMA FORECASTS FOR S10DEPT2 ";
TITLE2 "NO INTERVENTION " ;
PLOT S10DEPT2*DATE=1  FORECAST*DATE=2 L90*DATE=3 U90*DATE=4 /OVERLAY  GRID ;
RUN;

PROC ARIMA DATA=WEEKLYSALES ;
```

Figure 11: ARIMA Forecast - Store 10 Dept 10

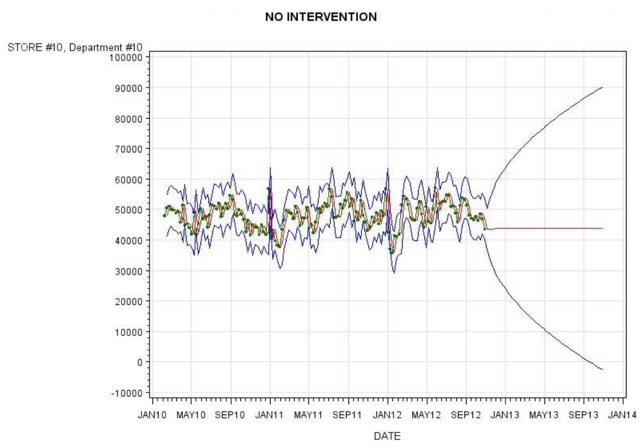
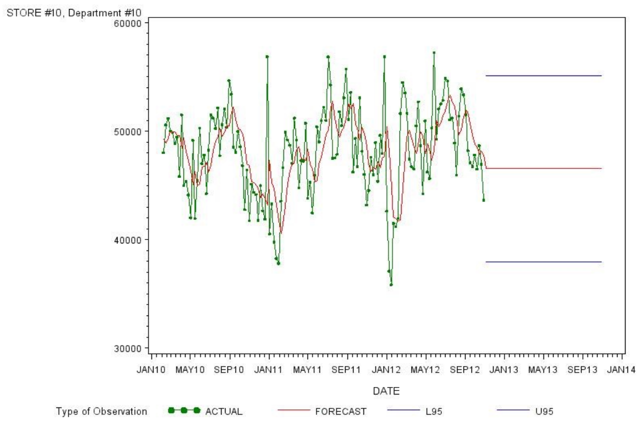


Figure 12: Exponential Smoothing Forecast - Store 10 Dept 10



7.9. ARIMA Using R

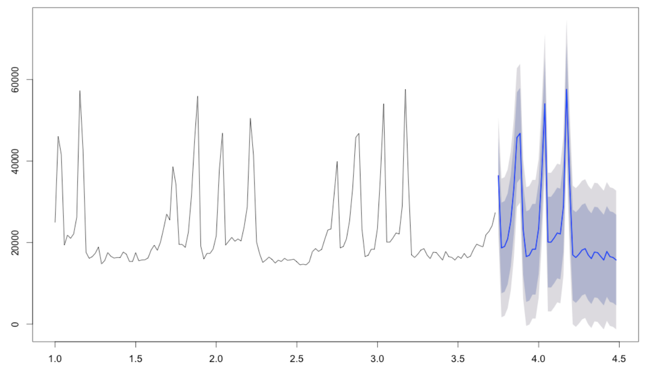
In statistics and econometrics, and in particular in time series analysis, an **autoregressive integrated moving average (ARIMA)** model is a generalization of an autoregressive moving average (ARMA) model. These models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). They are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied to reduce the non-stationarity. [29]

We used two approaches to predict our data set namely auto and manual ARIMA. The steps were.

Auto ARIMA

1. First step was to plot the data to understand the unusual observations and try to stabilize the variance if needed.
2. For auto ARIMA use auto-ARIMA to find the best ARIMA model and predict the sales
    a. ts(ar11, frequency = 52)
    b. auto.ARIMA()
    c. forecast.ARIMA(<on the model>, h=39)

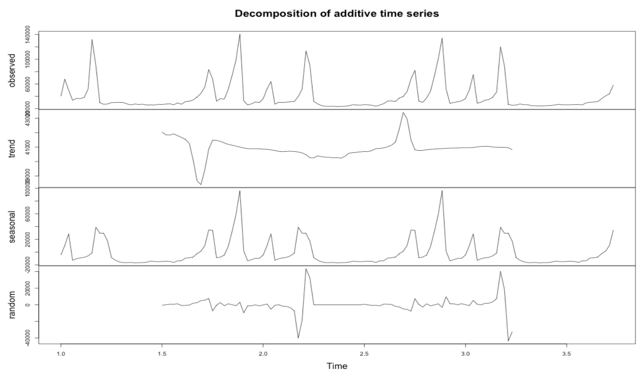
Figure 13: Forecasts from ARIMA(0,0,1)(0,1,0)[52]



Manual ARIMA

When fitting an ARIMA model to a set of time series data, the following procedure provides a useful general approach.

- Plot the data. Identify any unusual observations.



- If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
- If the data are non-stationary: take first differences of the data until the data are stationary. In our case we had to do the difference just once.
- In time series analysis, the partial autocorrelation function (PACF) plays an important role in data analyses aimed at identifying the extent of the lag in an autoregressive model. The use of this function was introduced as part of the Box–Jenkins approach to time series modeling, where by plotting the partial auto-correlative functions one could determine the appropriate lags p in an AR (p) model or in an extended ARIMA (p,d,q).
- In our case we used a difference of 1 and p and q values of 2 and 2 respectively.

7.10. SAS Auto-Model Fit

The second option for using the SAS software suite was to evaluate our dataset using the auto-model fit built into the software. The series of steps involved using the auto-model fit option entail the following:

1. Starting the Project
2. Viewing the Series
3. Generating Models
4. Automatic Model Selection Options
5. Model Selection List
6. Generate Models Results
7. Produce Forecasts
8. Series Selection
9. Plots
10. Series Diagnostics
11. p-Value for Unit Root Test
12. Autocorrelation Plots
13. Plot of First Differences
14. Autocorrelations of First Differences
15. Develop Models
16. ARIMA Model Specification
17. View Models
18. Residual Tests
19. Forecast Combination Model Specification
    a. Combination of forecast models can be done with equal weighting or regression weighting to improve overall forecast based on selected criterion.
    b. *This step needs to be performed*
20. Adding explanatory variables to the model - custom model specification
    a. *This step needs to be performed*
21. Residual ACF and PACF
22. Error Model Options

23. Time Range Specification

The following selection criteria fields are presented within the auto-model fit option to select from:

Table 3: Auto-Model-Fit Selection Criteria

Auto-Model-Fit Selection Criteria		
Sum of Square Error	Mean Square Error	Root Mean Square Error
Mean Absolute Error	Mean Absolute Percent Error	Akaike Information Criterion
Schwarz Bayesian Information Criterion (SBC)	R-Square	Adjusted R-Square
Random Walk R-Square	Amemiya's Adjusted R-Square	Amemiya's Prediction Criterion

In deciding upon which criteria would be best for the Walmart dataset, we selected:

1) **Schwarz Bayesian Information Criterion** - the SBC option is considered to be one of the better measurement criterions when conducting forecasts.

1. The Akaike information criteria (AIC)

$$AIC = \exp\left(\frac{2k}{T}\right)\frac{\sum_{t=1}^{T}\hat{e}_t^2}{T} = \exp\left(\frac{2k}{T}\right)MSE.$$

2. The Schwarz information criteria (SIC) or Bayesian information criteria (BIC). Note that SAS reports this as the SBC.

$$SIC = T^{k/T}\frac{\sum_{t=1}^{T}\hat{e}_t^2}{T} = \left(T^{k/T}\right)MSE.$$

2) **Mean Absolute Error** - due to the fact that the Kaggle result uses this criteria for benchmarking forecast output results and to have a common selection criteria across toolsets.

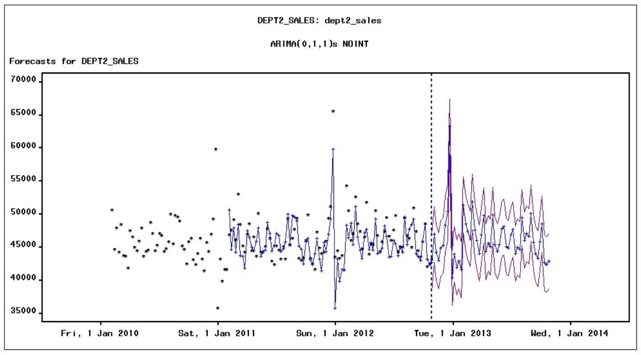
The following models were available for us to choose from for evaluating each series throughout the dataset using the SAS auto-model fit:

Table 4: Models available for SAS Auto model fit

MODELS				Seasonal Dummy	Simple Exponential Smoothing
Double (Brown) Exponential Smoothing	Linear (Holt) Exponential Smoothing	Damped Trend Exponential Smoothing	Seasonal Exponential Smoothing	Winters Method -- Additive	Winters Method -- Multiplicative
Random Walk with Drift	Airline Model	ARIMA(0,1,1)s NOINT	ARIMA(0,1,1)(1,0,0)s NOINT	ARIMA(2,0,0)(1,0,0)s	ARIMA(0,1,2)(0,1,1)s NOINT
ARIMA(2,1,0)(0,1,1)s NOINT	ARIMA(0,2,2) (0,1,1)s NOINT	ARIMA(2,1,2)(0,1,1)s NOINT	Log Mean	Log Linear Trend	Log Linear Trend with Autoregressive Errors
Log Linear Trend with Seasonal Terms	Log Seasonal Dummy	Log Simple Exponential Smoothing	Log Double (Brown) Exponential Smoothing	Log Linear (Holt) Exponential Smoothing	Log Damped Trend Exponential Smoothing
Log Seasonal Exponential Smoothing	Log Damped Trend Exponential Smoothing	Log Seasonal Exponential Smoothing	Log Winters Method -- Additive	Log Winters Method -- Multiplicative	Log Random Walk with Drift
Log Airline Model	Log ARIMA(0,1,1)s NOINT	Log ARIMA(0,1,1) (1,0,0)s NOINT	Log ARIMA(2,0,0) (1,0,0)s	Log ARIMA(0,1,2) (0,1,1)s NOINT	Log ARIMA(2,1,0) (0,1,1)s NOINT
Log ARIMA(0,2,2) (0,1,1)s NOINT	Log ARIMA(2,1,2) (0,1,1)s NOINT	Mean	Linear Trend	Linear Trend with Autoregressive Errors	Linear Trend with Seasonal Terms

In the example below, the auto-model fit for Store 1, Department 2 selected the ARIMA(0,1,1)s NOINT series as the best model based on the selection criterion of SBC.

Figure 14: Forecast - Dept 12



By importing the entire Walmart dataset into the SAS software suite, on a store-by-store basis, the software provides the option to iteratively cycle through each department optimally choosing the best model that fits the time series data. Once a model is selected for each series throughout each department, the store's department forecasts are generated. Once all forecasts for all stores were generated, by consolidating the forecast output for all 45 stores and their respective 99 departments into the format of the submission file, we were able to measure our performance relative to the Kaggle benchmark.

7.11. Random Committee

Random Tree based Committee Learning, Random Committee in short, is a committee learning to use "Random Trees" as the base classifiers. It's been mentioned to have an accurate prediction of the future profit of enterprises from the financial data in their past terms. [30]

As for the similar reason we mentioned in Random Forest section, we also want to experiment on Random Committee method provided by Weka.

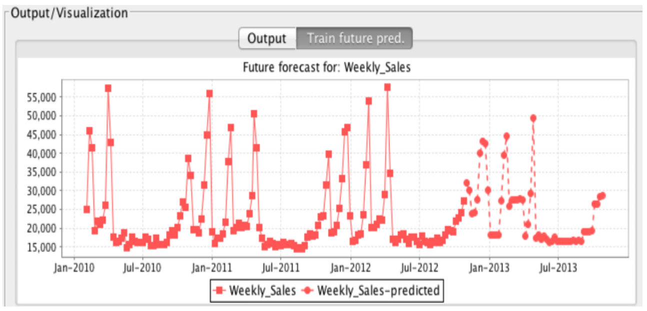
How

First of all, we tried Random Committee on Weka based on one segment of our training data (for department one of store one) to check if it could provide a fit prediction model. Here are what we do:

1. Install Forecast addon to Weka as it significantly helps us on time series analysis.
2. Load the subset of training data into Weka, format Date field and set Weekly_Sales as the class.
3. Set parameters:
   Number of time units to forecast - 52
   Time stamp - Date
   Periodicity - automatically
   Base learner - RandomCommittee and the number of iterations to be performed is set to 10, the random number seed is set to 1 as default.
   Lag length - week 51 – 53
   Mean Absolute Error and Root Mean Squared Error - checked in the evaluation metrics.

The future prediction on the training data seems to be a good result as shown in the following snapshot.

Figure 15: Sample output for department 1 of store 1 with random committee method



Next we want to come up with models for all the departments of all stores so that we can give the sales prediction to fill the submission file which spans from Nov 2012 to July 2013. We use Weka java API to perform this task.

Here are what we do in our Java code (Java code will be in appendix):

1. Read both training and test data.
2. Format the Date field and split into per department of store.
3. Call API of random committee class with the same configuration mentioned in previous method trial.
4. Use the model we get to predict weekly sales data in submission file which includes the given future date.

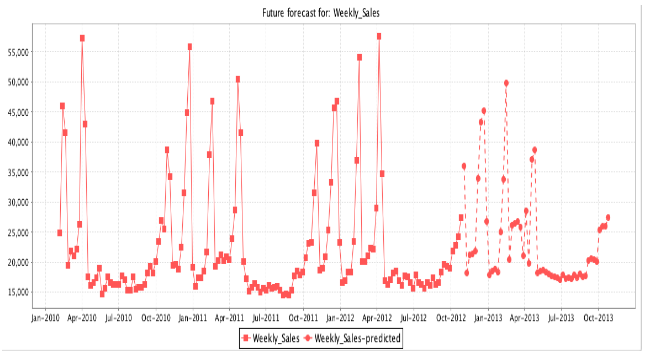
After submission to Kaggle, we got a rank of 265 on leaderboard with the WMAE 3360. This is a pleasant result and if we want to make it better, we may also try adding offset to special holidays and weekends or alter the lag length into certain pieces.

7.12. Vote

In Weka, Vote is a meta classifier that is used to combine the output of multiple underlying classifiers. It runs the underlying classifiers on the data and then combines them using the specified Combination Rule. The common combination rules are average, min, max, product, medium and majority voting. We combined various underlying classifiers like SMOReg, IBk and RandomTree. In our experimentation the average of SMOReg and IBk performed the best on Kaggle. [31]

Using the Weka Java API, we split up the Walmart dataset into all 4,455 store-department combinations and ran Vote(Avg(SMOReg, IBk)) on each time series. We then forecasted this data 39 weeks into the future. Below is a sample forecast of the store1-department1 combination using Vote(Avg(SMOReg, IBk)).

Figure 16: Vote(Avg(SMOReg, IBk)) Forecast - Store 1 Dept 1



Vote(Avg(SMOReg, IBk)) was one of our highest performing algorithms on the Kaggle leaderboard. The Vote(Avg(SMOReg, IBk)) algorithm was ranked 198 with a RMSE of 3079 on the

Kaggle leaderboard. The Java code to integrate with the WEKA API is attached in the appendix.

8. KAGGLE SUBMISSION SUMMARY

The team used Java (with Weka), R and SAS to generate the formatted Kaggle results for Walmart store sales. These results were then submitted to Kaggle for processing using the described evaluation criteria and were then ranked along with all the other submissions and placed on a leaderboard. The team submitted >50 submissions to Kaggle. Below is a summary of our top performing results per category of algorithms:

Table 5: Kaggle WMAE and Leaderboard Rank Summary

Model	Weighted Mean Absolute Error (WMAE)	Kaggle Leaderboard Rank
Weka - VOTE AVG(SMOReg + IBk)	3079	198
Weka - SMOReg (+NormalizedPolyKernel)	3274	244
Weka - Random Committee	3360	265
Weka - IBk	3444	283
Weka - MultilayerPerceptron	3482	302
Weka - SMOReg	3527	312
R - Holt Winters	3612	323
R - ARIMA	3612	324
R - gbm	5871	472
SAS - AutoModel Fit	14034	572
SAS - Exponential Smoothing/ARIMA	TBD	TBD

For comparison, here is the first ranked algorithm on Kaggle and an all zero submission.

Table 6: Kaggle Benchmarks

Model	Weighted Mean Absolute Error (WMAE)	Kaggle Leaderboard Rank
Kaggle - Rank #1	2237	1
Kaggle - All 0s Benchmark	21924	646

9. FUTURE STEPS

Given the fact that our research comprised just 7.5 weeks of effort, with the variety of forecasting tools and methods that were used to determine the forecasted output for Walmart Sales Revenue, improving the accuracy of the sales forecasts would be the first task. Several of the forecasting algorithms work well independently, but combining forecast models with equal weighting or regression weighting could technically improve overall forecasts. Additionally, combining already existing forecasting methods with machine learning algorithms would allow us to leverage automation with the variability that each time series provides.

Additionally, the team has also began light-weight implementation of a real-time forecasting platform which would take as input the actual weekly sales revenue for all stores and departments and automatically generate the forecasts as the information is provided. This forecasting platform leverages full stack software tools from the backend databases (MySQL), to the middleware computing layer, to the front-end using jQuery and D3.js. The process by which this would work is as follows:

1. Create Server-Client Stack to host and interact with client data
2. Create Database/Cloud Architecture to retrieve & store metric data (MySQL)
3. Incorporate Computing Layer to the Stack (R, SAS, Weka)
4. Run Forecasts
5. Run Comparative Analytics from Computing Layer
6. Publish Forecasts & Comparative Analytics to UI (D3.js, jQuery)

10. CONCLUSION

In conclusion, across all forecasting methods, the models explored work well independently. Given the variability and the seasonality of the time series, combining algorithms in specific scenarios using regression weighting would provide the optimal forecasts, while minimizing errors.

11. REFERENCES

[1] *What is R*
http://www.r-project.org/about.html

[2] *R for Mac OS X*
http://cran.r-project.org/bin/macosx

[3] Download Rstudio from www.rstudio.com

[4] W. Andrew Barr, February 25, 2013. *Top 6 reasons you need to be using Rstudio*
http://www.r-bloggers.com/top-6-reasons-you-need-to-be-using-rstudio/

[5] R-project.org. forecast: *Forecasting Functions for Time Series and Linear Models*
http://cran.r-project.org/web/packages/forecast/index.html

[6] Campbell, D. and Campbell, S. 1993. Statlab Workshop. *Introduction to Regression and Data Analysis,* October 28, 2008
http://statlab.stat.yale.edu/workshops/IntroRegression/StatLab-IntroRegressionFa08.pdf

[7] minitab.com. Regression and correlation. Types of regression analysis, *what is multiple linear regression*
http://support.minitab.com/en-us/minitab/17/topic-library/modeling-statistics/regression-and-correlation/basics/types-of-regression-analyses/#what-is-multiple-linear-regression

[8] Cornell University, Learning Strategies Center, Statistics Lab, *Regression*
https://lsc.cornell.edu/Sidebars/Stats%20Lab%20PDFs/Topic9.pdf

[9] Jagdish Hiray, February 21, 2008. *Time-series methods of forecasting*
https://businessmanagement.wordpress.com/2008/02/21/time-series-methods-of-forecasting/

[10] http://www.sagepub.com/upm-data/4913_Mentzer_Chapter_3_Time_Series_Forcasting_Techniques.pdf

[11] http://www.sagepub.com/upm-data/4913_Mentzer_Chapter_3_Time_Series_Forcasting_Techniques.pdf

[12] http://www.sagepub.com/upm-data/4913_Mentzer_Chapter_3_Time_Series_Forcasting_Techniques.pdf

[13] http://people.duke.edu/~rnau/411flow.gif

[14] "Root-mean-square-deviation", en.wikipedia.org. http://en.wikipedia.org/wiki/Ringelmann_effect

[15] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy: Improvements to the SMO Algorithm for SVM Regression. In: IEEE Transactions on Neural Networks, 1999, http://www.keerthis.com/smoreg_ieee_shevade_00.pdf

[16] A.J. Smola, B. Schoelkopf (1998). A tutorial on support vector regression, http://www.svms.org/regression/SmSc98.pdf

[17] D. Aha, D. Kibler (1991). Instance-based learning algorithms. Machine Learning. 6:37-66. http://link.springer.com/article/10.1007%2FBF00153759#page-1

[18] Ridgeway, G and Southworth, H. *Generalized Boosted Regression Models,* version 2.1.1, March 11, 2015 http://cran.r-project.org/web/packages/gbm/gbm.pdf

[19] Gardner, MW, and SR Dorling. "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences." *Atmospheric environment* 32.14 (1998): 2627-2636.

[20] Bishop, Christopher M. "Neural networks for pattern recognition." 4 (1995): 5.

[21] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989): 359-366.

[22] Kaastra, Iebeling, and Milton Boyd. "Designing a neural network for forecasting financial and economic time series." *Neurocomputing* 10.3 (1996): 215-236.

[23] "MultilayerPerceptron - Weka." 2010. 10 May. 2015 http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html

[24] "MLPRegressor - Weka." 2013. 10 May. 2015 http://weka.sourceforge.net/doc.packages/multiLayerPerceptrons/weka/classifiers/functions/MLPRegressor.html

[25] Winters, P. R. (April 1960). "Forecasting Sales by Exponentially Weighted Moving Averages". *Management Science* **6** (3): 324–342.DOI:10.1287/mnsc.6.3.324.

[26]http://s3.amazonaws.com/zanran_storage/www.cec.uchile.cl/ContentPages/107548415.pdf

[27]http://s3.amazonaws.com/zanran_storage/www.cec.uchile.cl/ContentPages/107548415.pdf

[28] Avril Coghlan, Parasite Genomics Group, Wellcome Trust Sanger Institute, Cambridge, U.K. A Little Book of R for Time Series https://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#time-series-analysis

[29] en.wikipedia.org. "Autoregressive integrated moving average", last modified at March 31 2015 http://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

[30] New Frontiers in Artificial Intelligence: JSAI 2006 Conference And Workshops By Takashi Washio, Ken Satoh, Hideaki Takeda

[31] Ludmila I. Kuncheva (2004). Combining Pattern Classifiers: Methods and Algorithms. John Wiley and Sons, Inc. http://www.ccas.ru/voron/download/books/machlearn/kuncheva04combining.pdf

12. APPENDIX

Scripts and code for various models described in this document is attached as a zip file. Below explains the content of the zip:

- **MultilayerPerceptron:** Script for this model is available as Walmart_MLP_Weka_Experiments.zip
- **gbm:** Script for this model is available as Walmart_gbm_R_Model.zip
- **Random Forest:** Script for this model is available as Walmart_RandomForest_Model.zip
- **Holt Winters:** Script for this model is available as Time Series Code.R
- **Random Committee:** Script for this model is available as WalmartForecastRandomCommittee.java
- **ARIMA using R:** Script for this model is available as ArimCode.R
- **SMOReg, IBk, and Vote:** Script for this model is available as WekaJavaIntegration.zip
- **Exponential Smoothing and ARIMA using SAS:** Script for this model is available as SAS.zip
- Script for consolidating multiple outputs from SAS models is Rcode_Consolidation_SAS_Output_Files.R

13. DOCUMENT REVISION HISTORY

Date	Version	Description	Author
May 9, 2015	0.1	Initial Document Created	Team 6
May 9, 2015	0.2	Abstract	Felix Amoruwa, Neha Arora
May 9, 2015	0.2	Methods – Exponential Smoothing & ARIMA using SAS	Felix Amoruwa
May 9, 2015	0.2	Methods – Auto-model Fit	Felix Amoruwa
May 9, 2015	0.3	Introduction	Neha Arora
May 9, 2015	0.3	Infrastructure Setup – MySQL-Java/Weka	Neha Arora
May 9, 2015	0.3	Methods – Perceptron	Neha Arora
May 9, 2015	0.4	Methods – SMOReg	Lloyd D'Silva
May 9, 2015	0.4	Methods – IBk	Lloyd D'Silva
May 9, 2015	0.4	Methods – Vote	Lloyd D'Silva
May 9, 2015	0.4	Kaggle Submission Summary	Lloyd D'Silva
May 9, 2015	0.5	Infrastructure Setup – R	Aparna Kumar
May 9, 2015	0.5	Method Categories – Time Series	Aparna Kumar
May 9, 2015	0.5	Methods – Holt Winters	Aparna Kumar
May 9, 2015	0.6	Data Description	Vignan Uppugandla
May 9, 2015	0.6	Pre-Processing Steps	Vignan Uppugandla
May 9, 2015	0.6	Method Categories – Regression	Vignan Uppugandla
May 9, 2015	0.6	Methods – Generalized Boosted Regression Model (gbm)	Vignan Uppugandla
May 9, 2015	0.7	Evaluation Criteria	Mrikondu Barkakati
May 9, 2015	0.7	Methods – Linear Regression	Mrikondu Barkakati
May 9, 2015	0.7	Methods – ARIMA using R	Mrikondu Barkakati
May 9, 2015	0.8	Method Categories – Ensemble	Feng Liang
May 9, 2015	0.8	Methods – Random Forest	Feng Liang
May 9, 2015	0.8	Methods – Random Committee	Feng Liang
May 9, 2015	0.9	Future Steps	Felix Amoruwa
May 9, 2015	0.9	Conclusion	Felix Amoruwa
May 10, 2015	1.0	Appendix	Vignan Uppugandla
May 10, 2015	1.1	Content Consolidation and Formatting	Vignan Uppugandla