Prediction of Sales using Ensemble of Regressors *

Karthik Prasad Phani Shekhar Rishabh Shah
Sushruth Gopal †

University of California Irvine
{prasadkr,pmantrip,rishabas,smallipa}@uci.edu

Abstract

Linear regression is a statistical approach for modeling relationships between a scalar dependent variable `y` and explanatory variables `X`. In this report, we present our experiments and approaches with different regression techniques in order to predict the sales of Rossmann Stores in Europe. By analyzing the data and features, we were able to draw helpful insights to train our models better. We studied and tried multiple linear regression modules, ranging from a Linear regression using Stochastic Gradient Descent to Support Vector Machines. We observed that an ensemble of bagging and boosting techniques on the transformed data patterns could better accommodate the richness in the data and the features yielding higher accuracy.

1 Introduction

The work presented in this report is based on one of the first Kaggle competition hosted by Rossmann Store. Kaggle, Inc. provides a platform for data mining competitions via the website `https://kaggle.com`. Companies and researchers use this platform to host predictive modeling and analytics competitions, where data miners from all over the world compete to produce best models.

Rossmann[2] is a retail company founded in 1972 by Dirk Rossmann. Currently, they operate over 3,000 drug stores in 7 European countries and are the second largest drug store chain in Germany[3]. Apart from Germany, they also have establishments in Poland, Hungary, Czech Republic, Albania and Turkey. This competition challenges people to predict the daily sales for a time period of 6 weeks for 1115 stores located across Germany.

*The competition was worked on as a project for CS 273A: Machine Learning, Fall 2015, taught by Prof. Eric Mjolsness.

†Karthik Rajendra Prasad: Student# 42686317, Phani Shekhar Mantripragada: Student# 85686586, Rishabh Amit Shah: Student# 79403075, Sushruth Mallipatna Gopal: Student# 57803787

Rossmann has provided the historical sales of those 1115 stores[4]. The data provided by them has been organized into the following CSV files:

- train.csv - Historical sales data of stores
- store.csv - Additional information about the stores

The data captures details such as number of customers visiting the stores, amount of sales, details of promotional offers, type of store, assortment of products sold by the store, details of the nearest competition and other related factors such as school holidays, state and public holidays. Given the above data we are supposed to predict the sales for the next 6 weeks at all stores.

2 Previous Work

The forum on Kaggle for Rossmann Store Competition was active and posted external data [5] that can be used by everyone.
We leveraged the data mapping stores to states[6] to understand the state dynamics and its effect on the sales of the store.
We also came across a post[7] (Fig. 1) on the forum which proved that using log transformation of the sales (instead of sales value) while calculating root-mean-squared-percentage-error(RMSPE) improves the accuracy.

$$RMSPE(p,t) = \sqrt{\frac{1}{n}\sum\left(\frac{p-t}{t}\right)^2}$$

$$P \equiv f(p) = f(t+\delta) \approx f(t) + f'(t)\delta + \ldots$$

$$T \equiv f(t)$$

$$\text{where } \delta \equiv p - t$$

Here, P and T are the transformed p and t. Our goal is to choose f in order to make $RMS(P,T)$ approximately equal to $RMSPE(p,t)$:

$$RMS(P,T) = \sqrt{\frac{1}{n}\sum(P-T)^2}$$

All we need to get these two metrics to agree is choose f such that:

$$P - T \approx \frac{p-t}{t}$$

If we assume that δ is small, and drop all the higher order terms of the Taylor series, then we can make $RMS(P,T)$ approximately equal to $RMSPE(p,t)$:

$$\Rightarrow f(t) + f'(t)\delta - f(t) \approx \frac{p-t}{t}$$

$$\Rightarrow f'(t)\delta \approx \frac{\delta}{t}$$

$$\Rightarrow f'(t) \approx \frac{1}{t}$$

$$\Rightarrow f(x) \approx \ln(x) + C$$

Figure 1: Relationship between between RMSPE and RMS(`log(y)`)

3 Overview and Decomposition of the Project

We tackled the problem by trying out various techniques. We started of with simple techniques and then let the project evolve organically by incorporating the learnings at every stage.

Section 2 provides the information that we gathered by exploring the Kaggle forum to understand the work done by other competitors. Insightful external data and tips to start on a Kaggle project were gathered as well. This work was done by ***Rishabh***.
Section 4.1 details our efforts with data analysis and building the exploratory graphs. This work was done by ***Karthik*** and ***Phani*** using the Pandas library[8] in Python.
Section 4.2 expands on important data and feature engineering techniques such as One-Hot encoding, normalization, feature synthesis and mergers, outlier detection and removal, and imputing missing data. This work was done by ***all four***.
Section 5.1 talks about Stochastic Gradient Descent on Linear Regression and the implementation details of this algorithm. This work was done by ***Karthik*** and ***Sushruth*** using the SGD Regressor library[11] in Python.
Section 5.2 discusses Support Vector Machine and its implementation to build the prediction model for this problem. This work was done by ***Rishabh*** using the Linear SVM library[10] in Python.
Section 5.3 highlights one of the successful techniques employed by us – Bagging: Random Forest algorithm. The section presents our learnings from this algorithm. The feature was implemented using Random Forest Regressor Scikit Library[12] in Python. It also presents our observations about the feature importance weight matrix and the RMSPE achieved by varying the various parameters of Random Forest Algorithm. This work was done by ***Rishabh*** and ***Karthik***.
Section 5.4 presents yet another successful technique – Gradient Boosting Algorithm and describes the implementation of this algorithm using XGBoost library[14] in Python. It lists the important features and RMSPE result achieved by changing the parameters of Gradient Boosting Algorithm. This work was done by ***Sushruth*** and ***Phani***.
After considerable analysis and experimentation, we decided to proceed with the ensemble of the above two ensemble techniques. Section 5.5 discusses the reasons and motivations behind this approach. This part of the project was implemented by ***Rishabh***.

4 Data Engineering

4.1 Data Insights

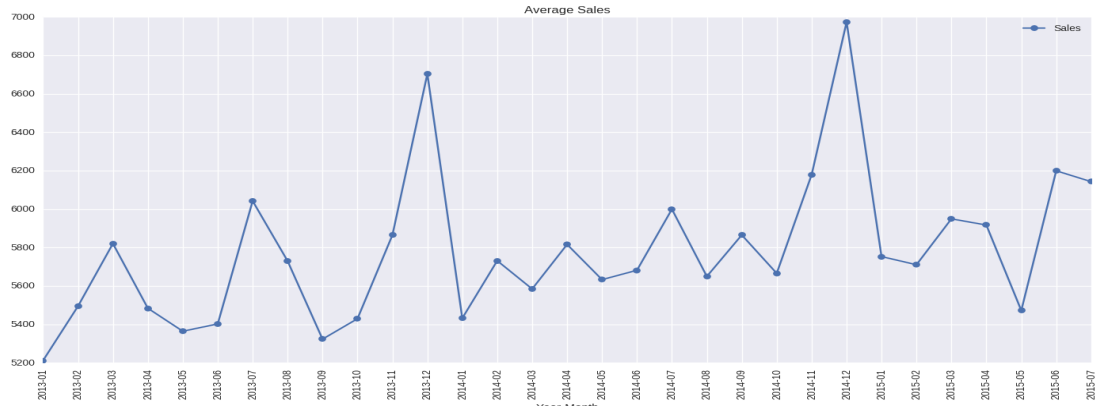


Figure 2: Month-wise Average Sales

- Plotting the average sales per month (Fig. 2) revealed that the sales were high during the months of December, March and July; perhaps due to Christmas, Easter festivals and public holidays.
- Plotting the average sales per day (Fig. 3) revealed that the average sales of the stores that were open on Sundays were higher than the ones that were not.

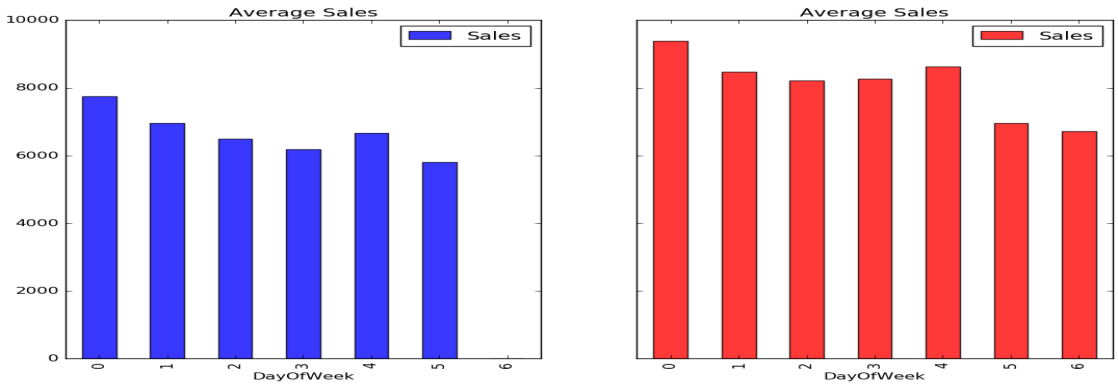
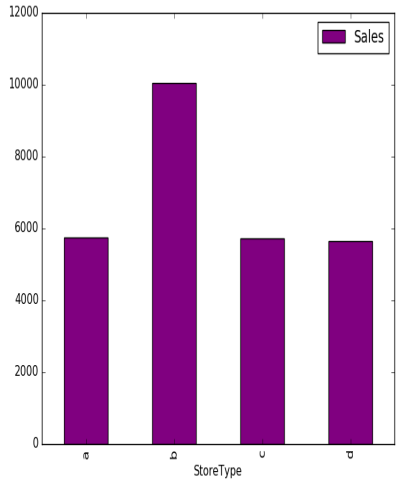


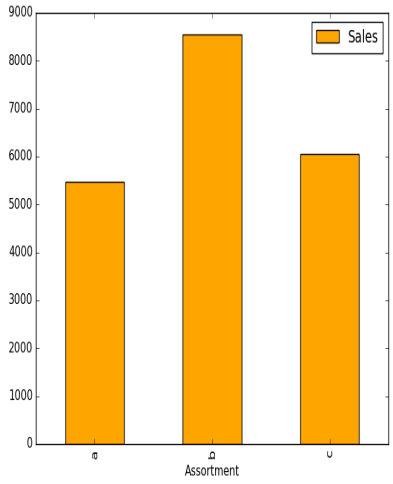
Figure 3: Average Sales by Day of Week for stores closed and open on Sundays

- While the stores A, C, D reported the same average sales, the store type B reported much higher sales. Added to this the stores with extended assortment supplies reported larger sales. (Fig. 4(a) and 4(b))

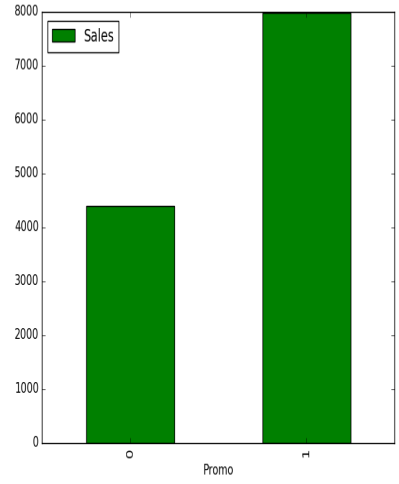
- Promo played an important role in the sales of the store. The average sales of the stores almost doubled in the promotion weeks. Similarly, sales were higher on school holidays. (Fig. 4(c) and 4(d))



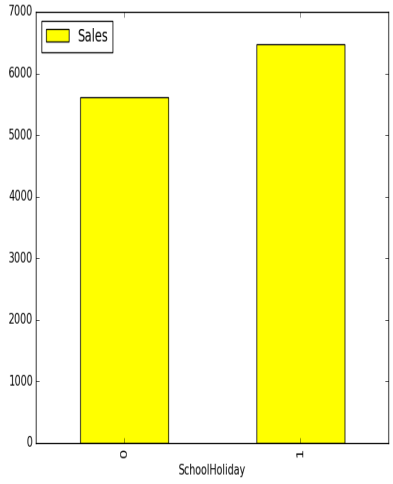
(a) Average Sales by Store Type



(b) Average Sales by Assortment



(c) Average Sales by Promotion



(d) Average Sales by School Holiday

Figure 4: Sales by Features

- Plotting the sales of stores with time exposed anomalies in the data. Sales data for some months were missing for certain stores, whereas some stores reported spike in sales just before they were closed as shown in Fig. 5. These sets were one time events and considered anomalies as they had a negative impact on the average sales being predicted.

- Similarly, some stores which were open on sundays never reported zero

sales. Out of the 1115 stores some were open on sundays, where as the others were not. But plotting the sales time graph(Fig. 6) we realized that some stores were open on sundays for certain weeks and closed for other.

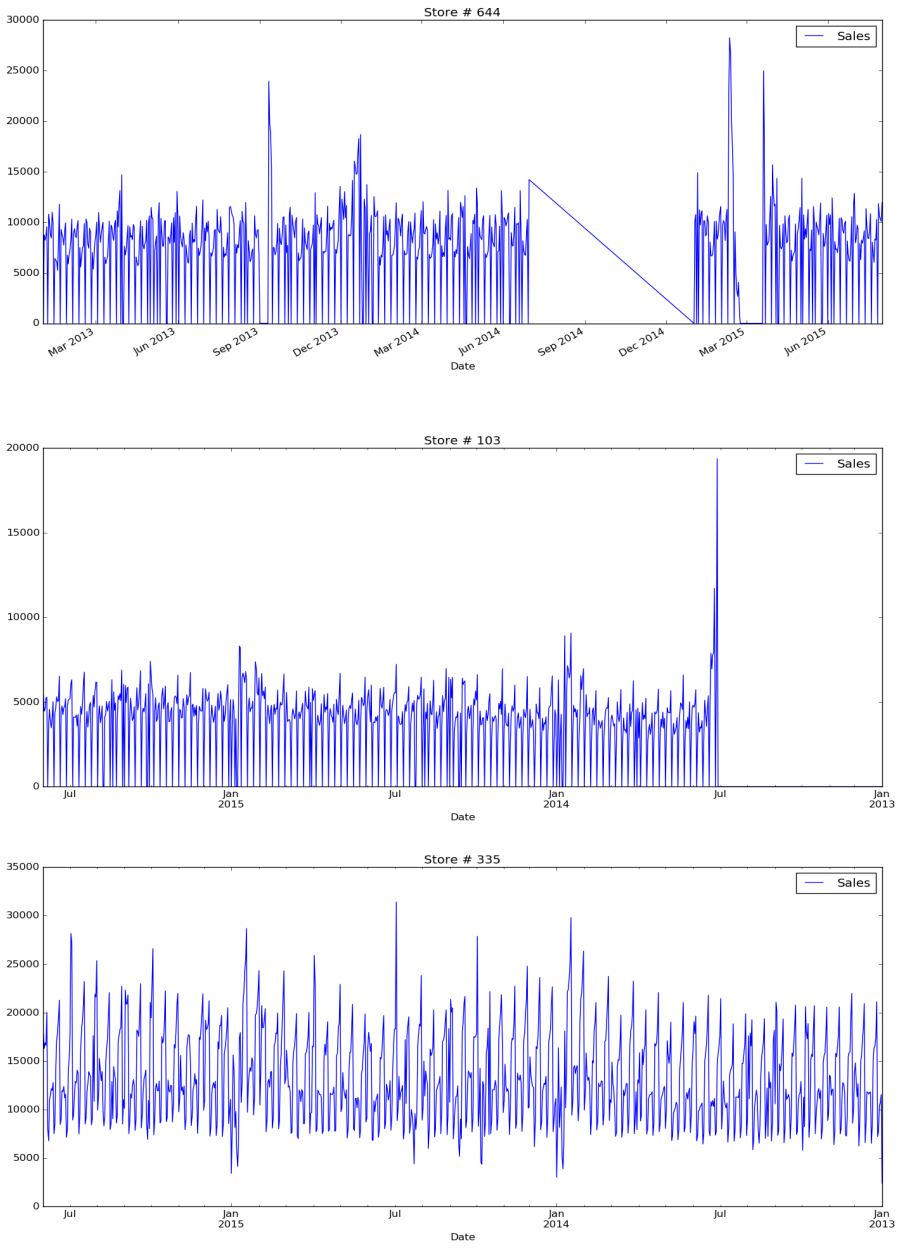


Figure 5: Interesting Data Patterns (Sales per store v/s Time)

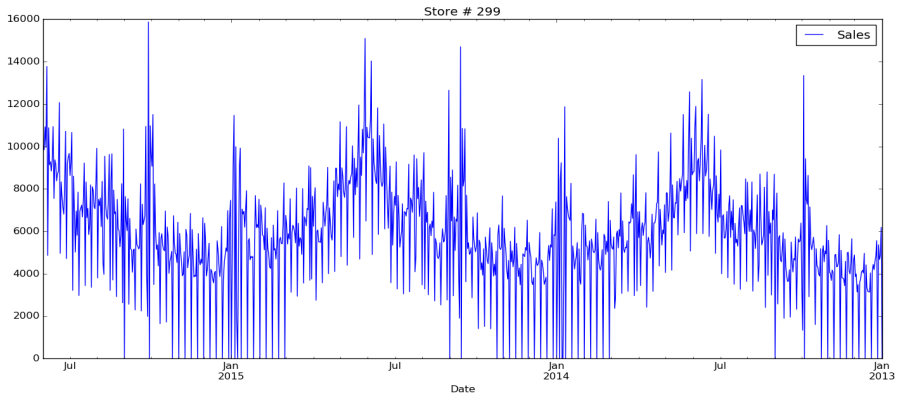


Figure 6: Interesting Data Patterns (Sales per store v/s Time)

4.2 Data Transformation

- **One Hot Encoding**: Many categorical variables were provided in the raw feature list such as - DayOfWeek, StoreType, Assortment, MonthOfYear. By one-hot encoding, we bloated the feature space as DayOfWeek got converted to 6 features (Sunday represented as all zeroes). This resulted in RMSPE score improvement by 0.15.
- **Removing data outliers**: Unexpectedly high sales (four times the mean value) were reported on few days for limited stores. These outliers and anomalies were clipped and limited to mean ± 3*standard deviation
- **Addition of New features**: Stores open on sundays throughout the year reported higher average sales than the other stores which were closed on sundays. A new feature was added to report such stores. Also, promo interval was introduced by merging few features from the training set.
- **Imputation of Missing Data**: Few stores were closed for more than six months due to refurbishment. This data was affecting the final model and had to be excluded from the final training set.
- **Normalizing and Data Binning**: CompetitionDistance and CompetitionOpen had a high cardinality in their data and ranged from 0 to 35000. This had to be normalized and binned to reduce its effect on sales.

5 Implementation

5.1 Linear Regression Model with Stochastic Gradient Descent

Our approach to predict the sales from the given set of features was to first check the efficiency achieved using a simple linear regression model. As we had

more than a million entries to train on, we used Stochastic Gradient Descent[11] with the simple regression model on this large data set. We experimented with different error functions, learning rate and number of iterations. The best result was attained with the Huber loss function[16], 1000 iterations and a learning rate of 0.01. The RMSPE was around *0.4271*. From this experiment it was evident that the simple linear regressor would not be able to capture all the patterns in the data, and we needed a better regression model.

5.2 Support Vector Machines

A linear regression model was not able to fit the data points accurately. We decided to use Support Vector Regression to project the data to a higher dimensional space using a kernel where we could perform linear curve fitting. We implemented this algorithm using the SVR Library[10] in Python with Linear, polynomial of degree 4, Radial Basis Function kernels[13] and epsilon insensitive loss function[1]. Training the data with the entire sample set was computationally expensive, O(Number of features * Training samples^2)[15]. Training the model using smaller chunks of 20000 samples (out of one million data points) did not yield good results(RMSPE > 10).

5.3 Bagging - Random Forest Algorithm

Bagging involves training multiple models using bootstrap data sets and aggregating the result of the individual models[1]. We used a particular type of Bagging technique called Random Forest. Random Forest[17] constructs a collection of decision trees[19] during training phase and outputs the mean prediction of the individual trees. A random set of features are used to train the decision tree over random training data samples. This is done to make sure the individual models are not correlated.

Random Forest is trained on the data and the performance of the algorithm is recorded by varying the number of trees used. Cross validation error is used to record the performance.

As evident from the results (Fig. 7), the performance of the algorithm saturates after the number of trees is around 100 which is used to calculate the error on the test data on the Kaggle Competition. Random Forests are biased in favor of categorical variables with different number of levels while calculating the feature importance[17]. One-hot encoding described in Section Data Transformation helped us overcome this problem. The feature importance as ranked by the Random Forest is as shown in Fig. 8

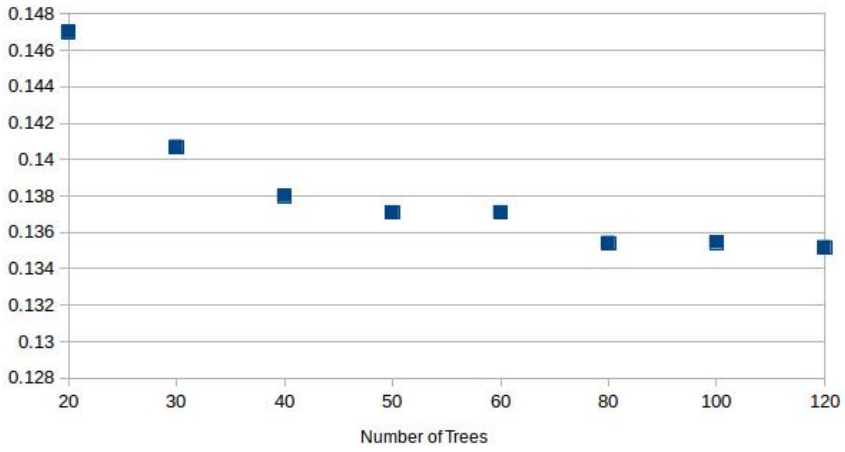


Figure 7: RMSPE v/s No. of Trees

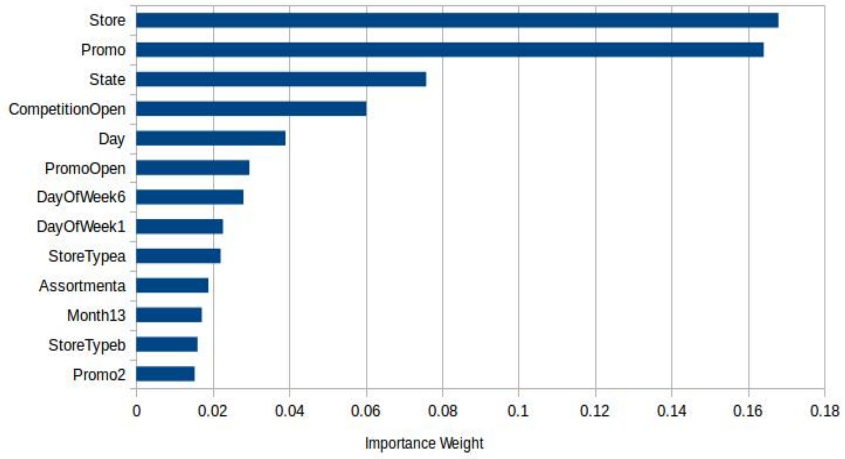


Figure 8: Feature importance as determined by Random Forest

5.4 Gradient Boosting Algorithm

Boosting involves training multiple models in sequence in which the error function used to train a particular model depends on the performance of previous models.[1] Gradient Boosting[18] is a technique which constructs a collection of decision trees during training phase and outputs the weighted average of prediction of the individual trees. Gradient Boosting is known to perform better than Random Forest in the case of correlated features[20]. The performance of the algorithm is recorded by varying the number of boosting iterations.

As indicated by the results (Fig. 9), 800 iterations are used to calculate the error on the test data. The feature importance as ranked by the Boosting algorithm is as shown in Fig. 10

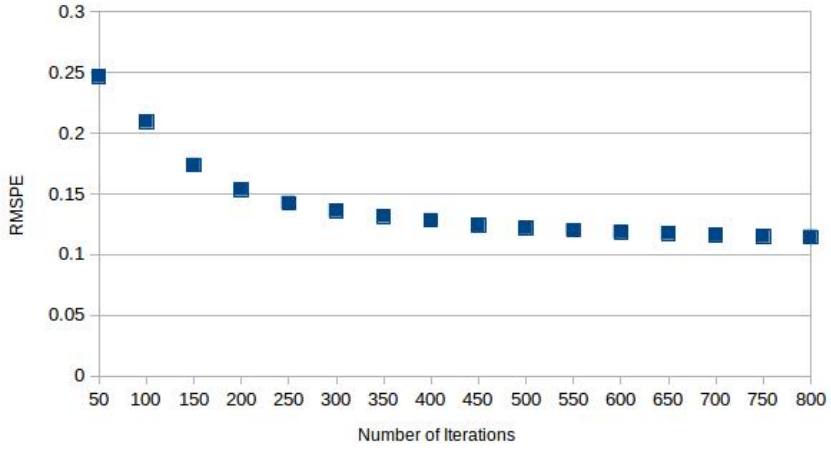


Figure 9: RMSPE v/s No. of iterations

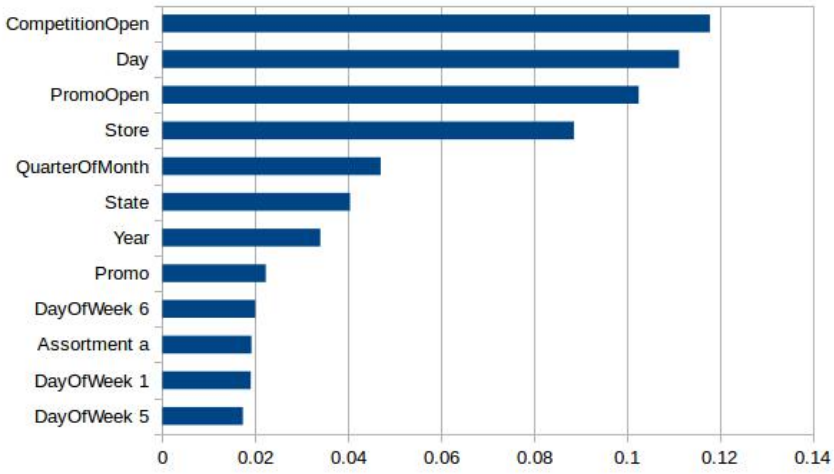


Figure 10: Feature importance as determined by Gradient Boosting

5.5 Ensemble of Ensembles

The Gradient boosting technique yielded slightly better accuracy than the Random Forest algorithm. However, boosting has a tendency to over-fit the training data[17][18]. Therefore, we tried to use weighted average of the predictions made by the two models by giving higher weight to the better model (boosting) and obtained improved results (Table 1).

	SGD on Linear Regression	Gradient Boosting	Random Forest	Ensemble of Ensembles
Raw Data	0.5483	0.1348	0.1536	
Engineered Data	0.4271	0.1190	0.1328	**0.1179**

Table 1: RMSPE rates for different algorithms and data combinations

6 Future Work

The current set of features provided a strong model to play with and we were successful to an extent to get the best out of them. Nevertheless, addition of more features that could potentially influence the predictions are to be considered. The information about weather, for instance, can play a vital role in determining the customers' willingness to visit the store which is directly related to sales. Similarly, clustering the stores based on the geographical regions and modeling each region separately might help in improving the accuracy. Since the stores are located in majorly Germany, the local events such as festivals, football matches can heavily influence the sales figures. Most importantly, given time-series nature of data, we strongly feel that running time-series modeling techniques such as ARIMA[21] can significantly boost our accuracy.

7 Conclusion

The project helped in solidifying our understanding of the different machine learning techniques and algorithms. We started by directly applying machine learning algorithms on raw data and then organically evolved by polishing the data and using robust algorithms. The exercise emphasized the importance of data analysis and data engineering before running the algorithms on them. We learnt that each technique and model comes with its own set of advantages and disadvantages and no one model can accommodate the data. Accordingly, after the initial attempts of running various algorithms such as Support Vector Machines and SGD on Linear Regression, we obtained better results with ensemble regression techniques: boosting(Gradient Boosting) and bagging(Random Forest). Further, to prevent over-fitting and be robust to correlated features, we used a weighted average of the above techniques and obtained even better results.

References

[1] Bishop, Christopher M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. Print.

[2] "*Wikipedia*. (Rossmann Company), n.d. Web. 09 Dec. 2015." *Web*. `https://en.wikipedia.org/wiki/Rossmann_(company)`

[3] "Rossmann Store Sales" *Description*. N.p., n.d. Web. 09 Dec. 2015. *Web*. `https://www.kaggle.com/c/rossmann-store-sales`

[4] "Rossmann Store Sales" Data. *Web*. `https://www.kaggle.com/c/rossmann-store-sales/data`

[5] "Rossmann Store Sales" External Data and Other Information. *Web.* https://www.kaggle.com/c/rossmann-store-sales/forums/t/17229/external-data-and-other-information.

[6] "Rossmann Store Sales". Putting Stores on the Map. *Web.* https://www.kaggle.com/c/rossmann-store-sales/forums/t/17048/putting-stores-on-the-map.

[7] "Rossmann Store Sales". Connection between RMSPE and the log transform. *Web.* https://www.kaggle.com/c/rossmann-store-sales/forums/t/17026/a-connection-between-rmspe-and-the-log-transform.

[8] "Python Data Analysis Library". N.p., n.d. Web. http://pandas.pydata.org/

[9] "Welcome to Python.org." Python. N.p., n.d. Web. 9 Dec. 2015. https://www.python.org/

[10] "Sklearn svm LinearSVR" *Linear SVR.* N.p., n.d. Web. 9 Dec. 2015. http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html#sklearn.svm.LinearSVR

[11] "Sklearn linear_model SGDRegressor" *SGD Regressor.* N.p., n.d. Web. 9 Dec. 2015. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html

[12] "Sklearn.ensemble.RandomForestRegressor" *Random Forest Regressor.* N.p., n.d. Web. 9 Dec. 2015. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[13] "Kernel Approximation" *Radial Basis Function Kernel.* N.p., n.d. Web. 9 Dec. 2015. http://scikit-learn.org/stable/modules/kernel_approximation.html#radial-basis-function-kernel

[14] "Dmlc/xgboost" *GitHub - XGBoost* N.p., n.d. Web. 9 Dec. 2015. https://github.com/dmlc/xgboost

[15] Chapelle, Olivier. "Training a Support Vector Machine in the Primal." *Neural Computation* 19.5 (2007): 1155-178. Web. http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/neco_%5b0%5d.pdf

[16] "Wikipedia. Wikimedia Foundation", Huber Loss n.d. Web. 09 Dec. 2015. https://en.wikipedia.org/wiki/Huber_loss

[17] "Wikipedia. Wikimedia Foundation", Random Forest n.d. Web. 09 Dec. 2015. https://en.wikipedia.org/wiki/Random_forest

[18] "Wikipedia. Wikimedia Foundation", Gradient Boosting n.d. Web. 09 Dec. 2015. https://en.wikipedia.org/wiki/Gradient_boosting

[19] "Wikipedia. Wikimedia Foundation", Decision Tree Learning n.d. Web. 09 Dec. 2015. https://en.wikipedia.org/wiki/Decision_tree_learning

[20] "Dmlc/xgboost" *GitHub* N.p., n.d. Web. 09 Dec. 2015. https://github.com/dmlc/xgboost/blob/master/R-package/vignettes/discoverYourData.Rmd

[21] "Autoregressive Integrated Moving Average" *Wikiwand*. N.p., n.d. Web. 10 Dec. 2015. https://www.wikiwand.com/en/Autoregressive_integrated_moving_average