MEF UNIVERSITY

A COMPARISON OF ENSEMBLE LEARNING METHODS IN RETAIL SALES FORECASTING

Capstone Project

Serhan Süer

İSTANBUL, 2019

MEF UNIVERSITY

A COMPARISON OF ENSEMBLE LEARNING METHODS IN RETAIL SALES FORECASTING

Capstone Project

Serhan Süer

Advisor: Dr. Evren Güney

İSTANBUL, 2019

MEF UNIVERSITY

Name of the project: A Comparison Of Ensemble Learning Methods In Retail Sales Forecasting

Name/Last Name of the Student: Serhan Süer

Date of Thesis Defense: 04/09/2019

I hereby state that the graduation project prepared by Serhan Süer has been completed under my supervision. I accept this work as a "Graduation Project".

04/09/2019
Dr. Evren Güney

I hereby state that I have examined this graduation project by Serhan Süer which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

04/09/2019
Prof. Dr. Özgür Özlük

Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of Serhan Süer and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Evren Güney	………………………..
2. Prof. Dr. Özgür Özlük	………………………..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Serhan Süer 04.09.2019 Signature

EXECUTIVE SUMMARY

A COMPARISON OF ENSEMBLE LEARNING METHODS
IN RETAIL SALES FORECASTING

Serhan Süer

Advisor: Dr. Evren Güney

SEPTEMBER, 2019, 37 pages

Forecasting has always been an essential skill which companies try to have and implement in various areas. Sales forecasting is one of the major usage areas of forecasting which is used in almost all sectors. This study refers to forecasting sales of Walmart Stores based on several features such as store id, department id, date, and store size. Walmart sales data which was used in this study contains information of stores between 2010 and 2012. At the beginning of the study, the introduction of the dataset and exploratory data analysis were made to identify dependent/independent variables and their characteristics. To apply machine learning algorithms, data preprocessing methods such as missing value treatment, outlier treatment, and feature selection was applied. Ensemble learning methods in machine learning algorithms were applied in the modeling stage. These methods were addressed in three parts such as Bootstrap Aggregation, Boosting, and Stacked Generalization and these parts consist of six different algorithms in total. The models were compared based on four regression metrics as Root Mean Square Error, Mean Absolute Error, R-Squared, and runtime. After selecting the main metric which models were evaluated, cross-validation was applied to achieve unbiased estimates. Finally, parameters of the model which have the highest score in cross-validation were tuned in the hyperparameter optimization stage and a machine learning model which can be used in forecasting sales of Walmart stores and its success score were obtained.

Key Words: retail sales forecasting, regression, exploratory data analysis, ensemble learning methods.

ÖZET

PERAKENDE SATIŞ TAHMİNLEMESİNDE
TOPLULUK ÖĞRENME METODLARININ KARŞILAŞTIRILMASI

Serhan Süer

Tez Danışmanı: Dr. Evren Güney

EYLÜL, 2019, 37 sayfa

Tahminleme her zaman şirketlerin sahip olmaya çalıştığı ve birçok alanda uygulanan önemli bir beceri olmuştur. Satış tahminlemesi ise neredeyse bütün sektörlerde kullanılan tahminlemenin en büyük kullanım alanlarından biridir. Bu çalışma, Walmart mağazalarının mağaza numarası, reyon numarası, tarih, ve mağaza büyüklüğü gibi özellikler üzerinden satış tahminlemesinin yapılması ile ilgilidir. Bu çalışmada kullanılan Walmart satış verisi, 2010 ve 2012 yılları arasındaki mağaza bilgilerini içerir. Çalışmanın başlangıcında bağımlı ve bağımsız değişkenlerinin özelliklerinin belirlenmesi için veri setinin tanıtılması ve keşifçi veri analizi yapılmıştır. Makine Öğrenmesi algoritmalarının uygulanabilmesi için kayıp veri iyileştirmesi, aykırı verilerin işlenmesi ve özellik seçimi gibi veri önişleme yöntemleri kullanıldı. Modelleme aşamasında makine öğrenmesi algoritmaları içinde bulunan Topluluk Öğrenme Yöntemleri uygulandı. Bu yöntemler, Torbalama, Yükseltme, ve İstifli Genelleştirme olarak üç farklı kısımda ele alındı ve toplamda altı farklı algoritma içerecek şekilde hazırlandı. Modeller, Hataların Ortalama Karekökü, Hataların Mutlak Ortalaması, R-Kare ve süre olmak üzere dört metriğe göre karşılaştırıldı. Modellerin değerlendirileceği temel regresyon metriğinin seçiminin ardından tarafsız tahminler elde etmek için çapraz-doğrulama tekniği uygulandı. Son olarak, parametre eniyileme aşamasında en yüksek sonucu veren modelin parametreleri ayarlandı ve Walmart mağaza satışlarını tahminlemekte kullanılabilecek makine öğrenmesi modeli ve modelin başarı oranı elde edilmiş olundu.

Anahtar Kelimeler: perakende satış tahmini, regresyon, keşifçi veri analizi, , topluluk yöntemleri.

TABLE OF CONTENTS

1. INTRODUCTION

Forecasting has always been a significant skill which companies tried to have. The companies which have this skill made their planning more accurate, organized their labor, logistic structures, financial resources, and costs. This provides companies not only productivity and profitability but also a competitive advantage among their competitors.

When the volume of the transactions, size of logistics organization and instability of the market are considered, one of the most important industries is retailing in terms of sales forecasting. Sales forecasting is an important skill which is generally required and used skill, especially in the retailing industry.

The objectives of this study are comparing performances of ensemble learning methods on retail sales forecasting and investigating the importance and effects of independent variables on the target variable.

The study consists of 9 main sections such as introduction, literature review, about the data, project definition, exploratory data analysis, methodology, conclusion, references, and appendix. Besides, exploratory data analysis was made in three parts: Univariate, Bivariate and Multivariate Analysis. In the methodology section, data preprocessing, model building and model evaluation stages were applied.

Six different ensemble learning methods which are Bootstrap Aggregation, Random Forest, Extremely Randomized Trees, Adaptive Boosting, XGBoost, and Voting Regressor were applied in modeling. Ensemble learning methods were compared based on not only Root Mean Square Error (RMSE) but also Mean Absolute Error (MAE), R-Squared and runtime. However, RMSE was selected as the main metric for evaluating the performance of the methods. Furthermore, Python codes related to all steps of the study were presented in the appendix section.

2. LITERATURE REVIEW

Since the scope of this study is ensemble learning methods in terms of machine learning algorithms, articles related to these methods in the scientific literature were scanned and the ones which can contribute to the study were selected in the literature review section.

Bootstrap Aggregation -also known as Bagging- is an ensemble learning method which is used for both classification and regression. Breiman (1996) stated that the main functions which Bootstrap Aggregation offers are improving the stability and accuracy of the model, reducing variance and avoiding overfitting. However, the most important drawback of Bootstrap Aggregation is reducing the simplicity and interpretability of the model. The formulation of Bootstrap Aggregation can be described as below:

$$\hat{y}_{BAG} = \frac{1}{B}\sum_{b=1}^{B}\emptyset(x; T_b)$$

The study which was published by Rapach & Strauss (2010) and a well-known article in the literature shows that Bootstrap Aggregation model performs better than the combination forecasts in terms of accuracy in U.S. employment growth. In addition, the study of D'Haen, Van den Poel & Thorleuchter (2012) shows that Bootstrap Aggregation gave the highest performance regardless of the data source. Extremely Randomized Trees is a type of Bootstrap Aggregation in ensemble learning methods. Geurts, Ernst & Wehenkel (2006) declared that while decision trees have very high standard deviations of the errors, Extremely Randomized Trees can offer reduce bias and variance. Another boosting algorithm which was used in this study is Random Forest. Breiman (2001) explains that avoiding overfitting, being an accurate predictive model and using out-of-bag estimation make Random Forest a powerful ensemble learning method. Another benefit which is offered by Random Forest is reducing bias and variance and this allows improving the model performance.

Boosting algorithms are very powerful machine learning models which have been used for a large number of Kaggle competitions. The result of the study which published by Ruiz-Abellón, Gabaldón & Guillamón (2018) illustrates the impressiveness and importance of both Random Forest and XGBoost methods for load forecasting for a university. In addition, Jain, Menon & Chandra (2015) states that XGBoost gave the best result in compared to Linear Regression and Random Forest Regression for retail sales forecasting. Catal, Ece, Arslan & Akbulut (2019) compared several regression machine learning algorithms which are Bayesian Linear Regression, Linear Regression, Decision Forest Regression, Boosted Decision Tree Regression and Neural Network Regression and time series analysis techniques such as Seasonal ARIMA, Non-Seasonal ARIMA and Seasonal ETS for sales forecasting. The result of the study shows that Boosted Decision Tree Regression algorithm performed better than other models in predicting sales of retail stores. Furthermore, the study which was published by Barboza, Kimura, and Altman (2017) showed that Bagging, Boosting and Random Forest algorithms are effective tools which can enhance the decision-making process in the banking industry. Another study which indicates similar results has published by Son, Hyun, Phan, and Hwang, (2019), and shows that machine learning algorithms such as XGBoost and Random Forest performed better than traditional models for bankruptcy forecasting. Another type of boosting algorithms is Adaptive Boosting also known as AdaBoost. AdaBoost was explained by the article of Freund, & Schapire (1997) and the study shows that it can be a powerful algorithm for both classification and regression problems. The formulation of AdaBoost for regression is as below:

$$h_f(x) = \inf \left\{ y \in Y : \sum_{t:\, h_t(x) \leqslant y} \log(1/\beta_t) \geqslant \tfrac{1}{2} \sum_{t} \log(1/\beta_t) \right\}$$

Krishna and Hegde (2018) showed that AdaBoost performed better than other algorithms based on Root Mean Square Error in their study related to sales forecasting of retail stores.

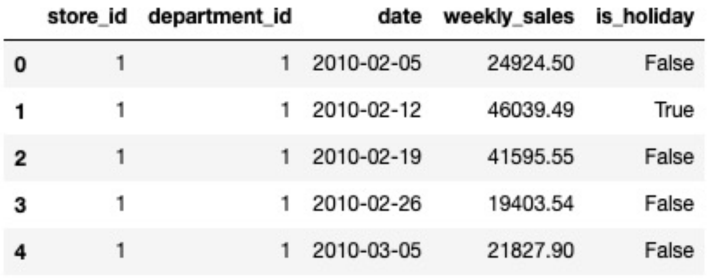
The last type of ensemble learning method which is in the scope of this study is stacked generalization also known as stacking. Wolpert (1992) explained how stacked generalization works and showed that it could be powerful at overcoming errors. Breiman (1996) stated that stacking algorithms performed better when the models which will be used as estimators are not much similar.

3. ABOUT THE DATA

This report contains an analysis of historical sales data for 45 Walmart stores and their departments located in different regions from February 2010 to October 2012. The dataset contains 3 files providing several features related to Walmart retail sales. Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are the Super Bowl, Labor Day, Thanksgiving, and Christmas.

"train.csv" file, which is shown in Table 1, is historical training data, which covers to 2010-02-05 to 2012-10-26. It consists of 421570 rows and 5 columns such as store_id - the store number, department_id - the department number, date - the date of sales, weekly_sales - sales for the given department in the given store and is_holiday - whether the week is a special holiday week.

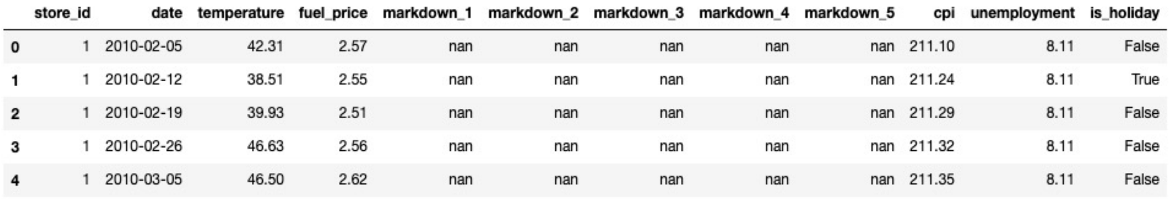
Table 1. First 5 rows of "train.csv" data file



	store_id	department_id	date	weekly_sales	is_holiday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False

"features.csv" file, which can be seen in Table 2, contains additional data related to the store, department, and regional activity. It consists of 8190 rows and 8 columns such as store_id - the store number, date - the date of sales, temperature - average temperature in the region, fuel_price - cost of fuel in the region, markdown_1-5 - anonymized data related to promotional markdowns that Walmart is running, cpi - the consumer price index, unemployment - the unemployment rate and is_holiday - whether the week is a special holiday week.

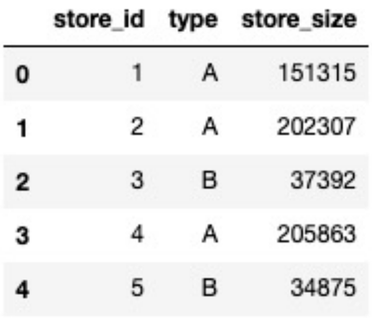
Table 2. First 5 rows of "features.csv" data file



	store_id	date	temperature	fuel_price	markdown_1	markdown_2	markdown_3	markdown_4	markdown_5	cpi	unemployment	is_holiday
0	1	2010-02-05	42.31	2.57	nan	nan	nan	nan	nan	211.10	8.11	False
1	1	2010-02-12	38.51	2.55	nan	nan	nan	nan	nan	211.24	8.11	True
2	1	2010-02-19	39.93	2.51	nan	nan	nan	nan	nan	211.29	8.11	False
3	1	2010-02-26	46.63	2.56	nan	nan	nan	nan	nan	211.32	8.11	False
4	1	2010-03-05	46.50	2.62	nan	nan	nan	nan	nan	211.35	8.11	False

"stores.csv" file, which is shown in Table 3, contains anonymized information about the 45 stores, indicating the type and the size of stores. There are 3 types of stores such as A, B, and C. While store_id column represents the store number, type and size columns represents the type of the store and the size of the store.

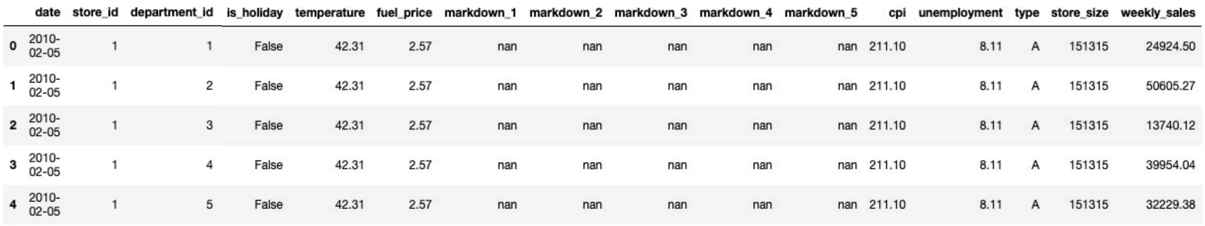
Table 3. First 5 rows of "stores.csv" data file



	store_id	type	store_size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

At the beginning of the project, these three files were merged to make a proper analysis. After that, merged data, which is shown in Table 4, has 16 columns and 421570 observations.

Table 4. First 5 rows of the merged data



	date	store_id	department_id	is_holiday	temperature	fuel_price	markdown_1	markdown_2	markdown_3	markdown_4	markdown_5	cpi	unemployment	type	store_size	weekly_sales
0	2010-02-05	1	1	False	42.31	2.57	nan	nan	nan	nan	nan	211.10	8.11	A	151315	24924.50
1	2010-02-05	1	2	False	42.31	2.57	nan	nan	nan	nan	nan	211.10	8.11	A	151315	50605.27
2	2010-02-05	1	3	False	42.31	2.57	nan	nan	nan	nan	nan	211.10	8.11	A	151315	13740.12
3	2010-02-05	1	4	False	42.31	2.57	nan	nan	nan	nan	nan	211.10	8.11	A	151315	39954.04
4	2010-02-05	1	5	False	42.31	2.57	nan	nan	nan	nan	nan	211.10	8.11	A	151315	32229.38

4. PROJECT DEFINITION

The analysis aims to compare the ensemble learning methods in machine learning by their performance on sales forecasting in the retailing industry.

The main objective of the project is to observe which ensemble learning method provides more accurate results in retail sales forecasting. On the other hand, the other objective of the project is investigating the importance and effects of the independent variables on the target variable using feature engineering techniques.

In the analysis, Bagging, Random Forest, Extremely Randomized Trees, AdaBoost, XGBoost and Voting Regressor have been used as ensemble learning methods. However, traditional machine learning models such as linear regression, decision tree, k-nearest neighbors were left out of scope. Besides, both relationships among independent variables and relationships between dependent and independent variables were examined. Throughout the analysis, the effect of holidays on the sales was analyzed.

5. EXPLORATORY DATA ANALYSIS

5.1. Univariate Analysis

In univariate analysis; unique values, mode, frequency table, histogram and pie chart of the categorical features, mean, median, minimum, maximum, standard deviation, variance, skewness, kurtosis, distributions and box plot of the numerical features have been analyzed.

Date column consists of 143 unique values from 2010-02-05 to 2012-10-26. So the period of the data can be considered as 143 weeks or 2 years and 9 months. In store_id column, there are 45 unique values, from 1 to 45, which means that data contains information related to 45 different Walmart stores. The dataset contains 81 different departments in the stores. Furthermore, it can be understood that while some departments exist in most stores, some exists in only few stores. is_holiday variable has a boolean type with a ratio of 93% False and 7% True which also can be seen in Figure 1. Thus, "is_holiday" variable is highly unbalanced since 93% of the weekly sales did not occur in the holidays. Stores in the dataset have 3 types such as A, B, and C. Figure 2 shows that while roundly half of the weekly sales records are related to Type A stores, almost 40% of them occurred in Type B stores and Type C stores have only 10 percent.

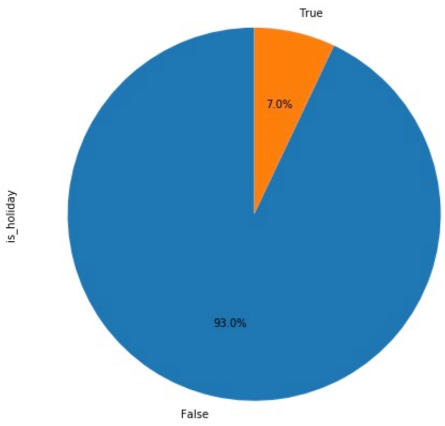


Figure 1. Pie Chart of is_holiday variable

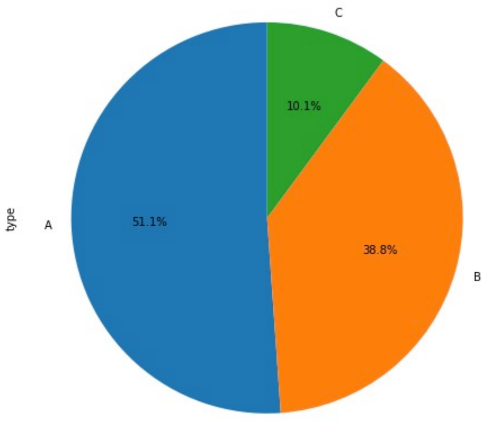


Figure 2. Pie Chart of type variable

After categorical variables were analyzed, numerical variables including the target variable were observed in univariate analysis. In this section, mean, median, minimum, maximum and standard deviation of each numerical variable were investigated. Weekly sales column in the dataset which is shown in Figure 3 has a highly skewed distribution and has negative values which mean that the value of return products exceeds the weekly sales on several dates.

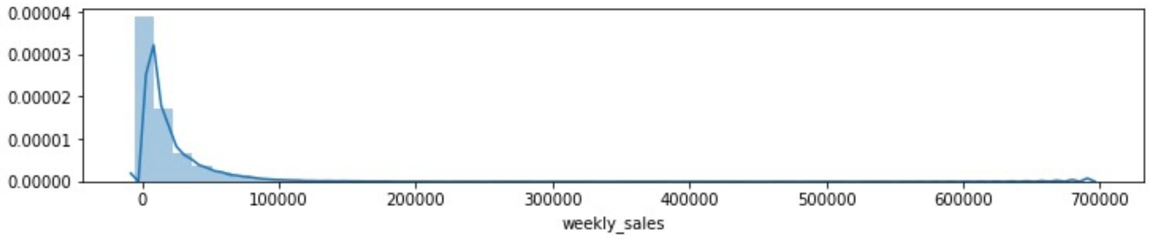


Figure 3. The Distribution of Target Variable

Store size varies from 34875 to 219622 and has a mean of 136727.92. In addition, there are several variables such as temperature, consumer price index, fuel price which may help to predict weekly sales of retail stores in the dataset.

5.2. Bivariate Analysis

Since the distribution of the data is skewed, non-parametric tests were preferred in the bivariate analysis section. While bivariate analysis between categorical and numerical was made, Kruskal Wallis test was applied. According to the Kruskal Wallis test result, it can be understood that sample distributions in the categorical variables are not equal. Correlations between numerical variables were analyzed based on the "Spearman" method since numerical features don't have Gaussian distribution.

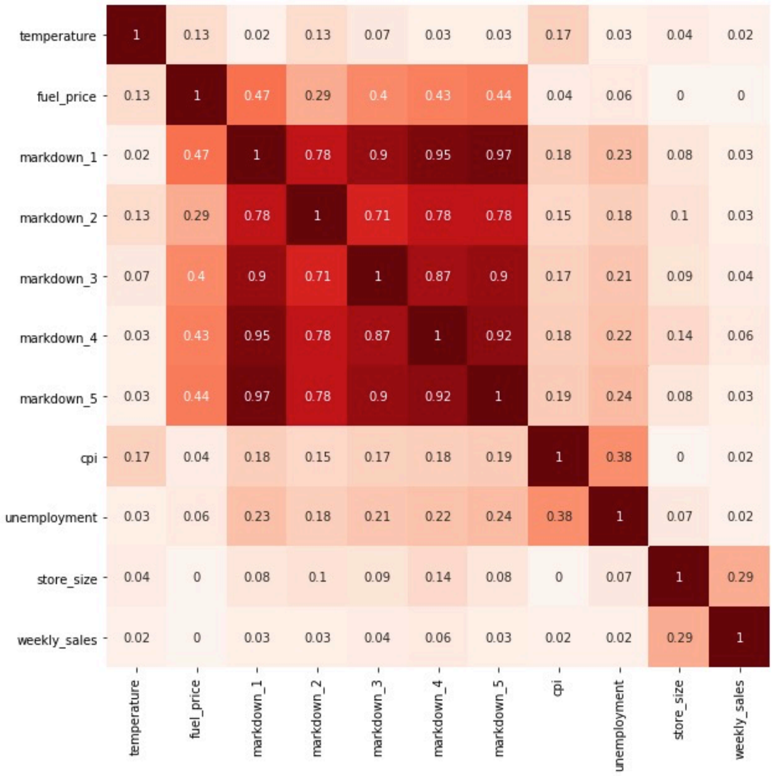
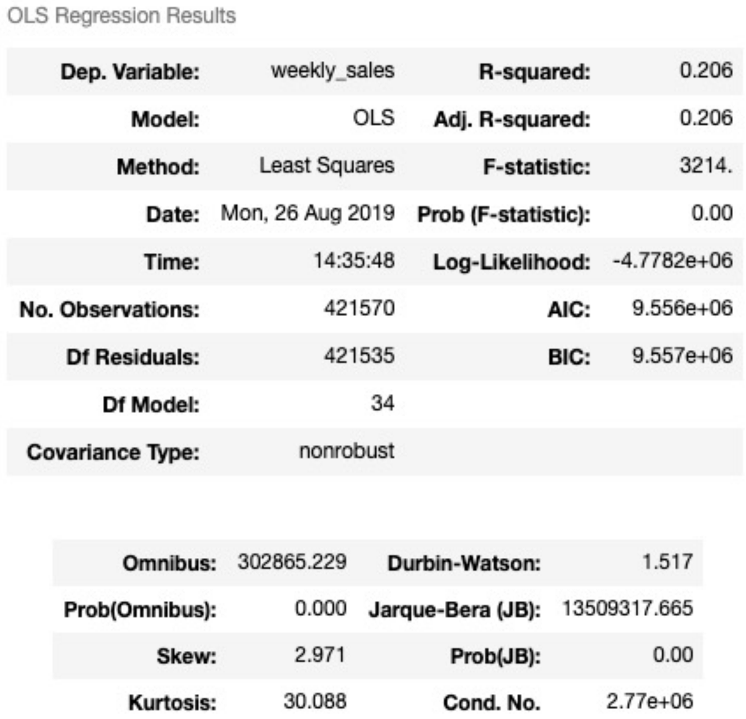


Figure 4. Spearman Correlation Analysis

According to the result of the correlation analysis which can be seen in Figure 4, it can be seen that markdown features are highly correlated with each other.

5.3. Multivariate Analysis

For Multivariate Analysis, Analysis of covariance (ANCOVA), which is shown in Figure 5, will be used since data has both categorical and numerical features.



OLS Regression Results

Dep. Variable:	weekly_sales	R-squared:	0.206
Model:	OLS	Adj. R-squared:	0.206
Method:	Least Squares	F-statistic:	3214.
Date:	Mon, 26 Aug 2019	Prob (F-statistic):	0.00
Time:	14:35:48	Log-Likelihood:	-4.7782e+06
No. Observations:	421570	AIC:	9.556e+06
Df Residuals:	421535	BIC:	9.557e+06
Df Model:	34		
Covariance Type:	nonrobust		

Omnibus:	302865.229	Durbin-Watson:	1.517
Prob(Omnibus):	0.000	Jarque-Bera (JB):	13509317.665
Skew:	2.971	Prob(JB):	0.00
Kurtosis:	30.088	Cond. No.	2.77e+06

Figure 5. Analysis of Covariance (ANCOVA)

The reason why ANCOVA was chosen is that it combines features of both ANOVA and regression. It increases the model of ANOVA with quantitative factors, called covariates, linked to the target variable. The covariates are included to decrease the variance in the error terms and provide more accurate measurement of the treatment effects. ANCOVA is used to test the main and interaction effects of the factors while controlling for the effects of the covariate.

6. METHODOLOGY

6.1. Data Preprocessing

Firstly, variables were split into two sets as features and target to implement preprocessing easily. Missing values in markdown columns were filled with 0 for missing value treatment. Outliers in all columns were identified and it was seen that markdown columns, unemployment, and target variable have outliers. To avoid data loss, doing outlier treatment after feature selection stage was decided. In Feature Engineering, week number as a new feature was created from Date column. Afterward, variables were transformed into proper types. Date column was dropped since the week number variable will be used instead.

For categorical features which have string values, Label Encoding was applied. Since the scope of this study contains only tree-based models, One-Hot Encoding and feature scaling will not be applied. In the feature selection stage, is_holiday column was dropped since week_number represents the information which is_holiday variable offers. Highly correlated variables such as markdown_3, markdown_2, markdown_5, markdown_1, and type were dropped based on their correlation with the target variable. Since fuel_price variable has very low variation (0.21), it was dropped. After the importance of features were identified thanks to the Random Forest algorithm, markdown_4 was dropped. Finally, data was split into three parts such as train, validation, and test set with the ratio of 60%, 20%, and 20% respectively.

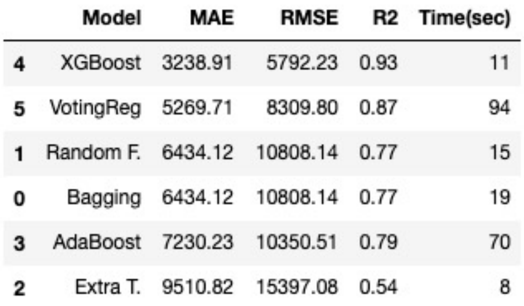
6.2. Model Building

Ensemble Learning Methods such as bootstrap aggregation, boosting and stacked generalization methods were applied after data preprocessing stage. For bootstrap aggregation method Bagging Regressor, Random Forest Regressor and Extremely Randomized Trees; for Boosting method AdaBoost and XGBoost were used and models were compared based on their Root Mean Square Error, Mean Absolute Error, R-Squared scores, and runtimes. In stacked generalization, XGBoost, AdaBoost and Random Forest will be used since they have higher scores than other models. In order to obtain a reasonable comparison, the maximum depth was selected the same in all models as 7 for the beginning. Different values of depth will be tried in the stage of hyperparameter tuning.

6.3. Model Evaluation

In the model evaluation stage of the analysis, mean absolute error was selected as the main performance metric in order to reduce the effect of the outliers to the model performance. The results of the models after cross-validation, which is shown in Table 5 and Figure 6, were compared based on all metrics. XGBoost gave a more successful performance in Root Mean Square Error, Mean Absolute Error, and R-Squared. However, Extremely Randomized Trees performed better in terms of model runtime.

Table 5. The Performances of Models with Cross-Validation



	Model	MAE	RMSE	R2	Time(sec)
4	XGBoost	3238.91	5792.23	0.93	11
5	VotingReg	5269.71	8309.80	0.87	94
1	Random F.	6434.12	10808.14	0.77	15
0	Bagging	6434.12	10808.14	0.77	19
3	AdaBoost	7230.23	10350.51	0.79	70
2	Extra T.	9510.82	15397.08	0.54	8

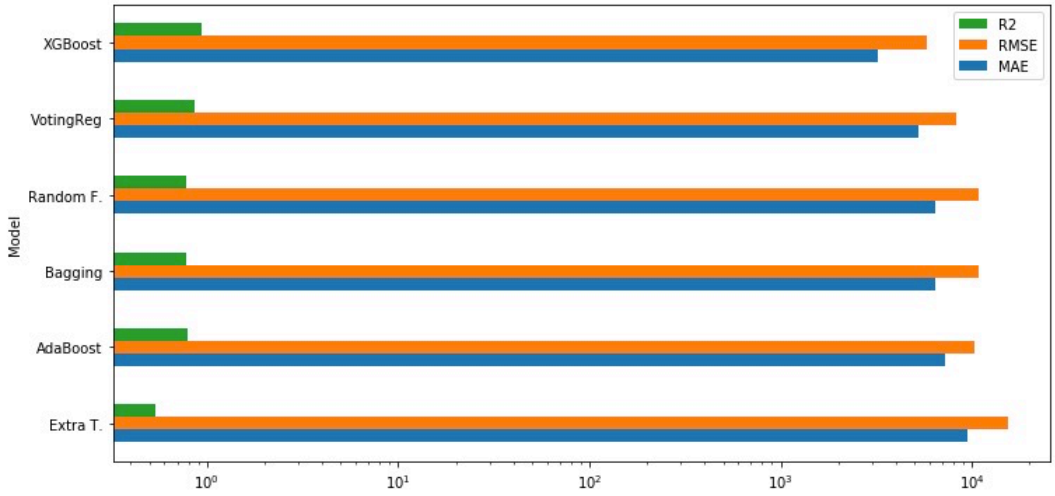


Figure 6. Bar Chart of Comparison of Models

In the final part of the evaluation stage, hyperparameter optimization will be applied to XGBoost method which is more successful based on the results in the cross-validation part. Parameters such as n_estimators, learning_rate, max_depth, gamma, subsample and colsample_bytree were tuned after trying different reasonable values. While trying values in hyperparameter optimization, 5-fold cross-validation was used to obtain consistent results. Best parameters of XGBoost model were 0.05 for learning rate, 4200 for the number of estimators, 10 for maximum depth, 1 for gamma, 0.8 for subsample and 1 for subsample ratio of columns. With these parameters, XGBoost model gave the result of 1298.86 of Mean Absolute Error and 0.98 of R-Squared.

7. CONCLUSION

In this study, the weekly sales of departments in 45 Walmart stores from February 2010 to October 2012 were analyzed and forecasting of weekly sales by using statistical techniques and machine learning algorithms was aimed. After introducing the general structure of the data, characteristics of dependent and independent variables and relationships between them were analyzed by using exploratory data analysis techniques.

Missing value treatment, outlier treatment, the encoding of categorical features, feature engineering, feature selection, and train-validation-test split were applied in data preprocessing stage. As considering the scope of this study, ensemble learning methods in machine learning were performed in three parts as Bootstrap Aggregation, Boosting and Stacked Generalization. Bagging Regressor, Random Forest Regressor, Extra Trees Regressor, AdaBoost, XGBoost and Voting Regressor are the machine learning methods which were compared by Root Mean Square Error, Mean Absolute Error, R-Squared, and runtime with cross-validation in the modeling section. After realizing that XGBoost performed better than other models in all metrics, the best parameters for XGBoost were determined by trying different parameters in the hyperparameter tuning section. Finally, the model with the best parameters was applied to the test data and the result of the model was recorded. In further studies, Artificial Neural Network and Time Series Analysis may be applied to improve the results in terms of performance metrics.

8. REFERENCES

Barboza, F., Kimura, H. & Altman, E. (2017). Machine Learning Models and Bankruptcy Prediction. An International Journal Expert Systems With Applications 83 (2017) 405–417.

Breiman, L. (1996). Bagging Predictors. Kluwer Academic Publishers, Machine Learning Volume 24, Issue 2, pp 123–140.

Breiman, L. (2001). Random Forests. Kluwer Academic Publishers, Machine Learning 45:5.

Catal, C., Ece, K., Arslan, B. & Akbulut, A. (2019). Benchmarking of Regression and Time Series Analysis Techniques for Sales Forecasting. Balkan Journal of Electrical & Computer Engineering, Vol. 7, No. 1.

D'Haen, J., Van den Poel D. & Thorleuchter, D. (2012). Predicting Customer Profitability During Acquisition: Finding The Optimal Combination of Data Source and Data Mining Technique. An International Journal Expert Systems with Applications 40 (2013) 2007–2012.

Freund, Y. & Schapire, R.E. (1997). A Decision-theoretic Generalization of Online Learning and An Application to Boosting". Journal of Computer and System Sciences. 55: 119–139.

Geurts, P., Ernst, D. & Wehenkel, L. (2006). Extremely Randomized Trees. Springer Science + Business Media Mach Learn (2006) 63: 3–42.

Breiman, L. (1996). Stacked Regressions. Kluwer Academic Publishers, Machine Learning 24, 49-64.

Jain, A.K., Menon, M.N., & Chandra, S. (2015). Sales Forecasting for Retail Chains.

Krishna, A. & Hegde, C. (2018). Sales-forecasting of Retail Stores using Machine Learning Techniques. 3rd IEEE International Conference on Computational Systems and Information Technology for Sustainable Solutions.

Rapach, D. & Strauss, J. (2010). Bagging or Combining (or Both)? An Analysis Based on Forecasting U.S. Employment Growth. Econometric Reviews, 29(5–6):511–533.

Ruiz-Abellón, M., Gabaldón, A. & Guillamón, A. (2018). Load Forecasting for A Campus University Using Ensemble Methods Based on Regression Trees. Energies 2018, 11, 2038.

Son, H., Hyun, C., Phan, D. & Hwang, H.J. (2019). Data Analytic Approach for Bankruptcy Prediction. An International Journal Expert Systems With Applications 138 (2019) 112816.

Wolpert, D.H. (1992). Stacked Generalization. Neural Networks Volume 5, Issue 2, 1992, Pages 241-259.

APPENDIX

PYTHON CODES

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')

import time
from math import sqrt
import statistics
from scipy import stats
import category_encoders as ce
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_regression
from sklearn.model_selection import train_test_split, KFold, GridSearchCV,
RandomizedSearchCV
from sklearn.decomposition import PCA

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.svm import libsvm
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
import xgboost as xgb
from sklearn.ensemble import VotingRegressor
from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

import warnings
warnings.filterwarnings('ignore')
pd.options.mode.chained_assignment = None
pd.options.display.max_columns = None
pd.set_option('display.float_format', '{:.2f}'.format)
```

```
print('Versions')
print('Pandas      : ', pd.__version__)
print('Numpy       : ', np.__version__)
print('Seaborn     : ', sns.__version__)
print('Matplotlib  : ', plt.__version__)
print('Statsmodels : ', sm.__version__)
print('Sklearn     : ', sk.__version__)

# Reading the data files:
train = pd.read_csv("data/train.csv", sep=',', header=0,
                    names=['store_id', 'department_id', 'date', 'weekly_sales', 'is_holiday'])
features = pd.read_csv("data/features.csv", sep=',', header=0,
                       names=['store_id', 'date', 'temperature', 'fuel_price', 'markdown_1',
'markdown_2',
                              'markdown_3', 'markdown_4', 'markdown_5', 'cpi', 'unemployment',
'is_holiday'])
stores = pd.read_csv("data/stores.csv", sep=',', header=0,
                     names=['store_id', 'type', 'store_size'])

print(train.shape)
print(features.shape)
print(stores.shape)
train.head()
features.head()
stores.head()

# Merging all data files:
data = pd.merge(train, features, on=['store_id', 'date', 'is_holiday'], how='left')
data = pd.merge(data, stores, on=['store_id'], how='left')
del train, features, stores

# Sorting data by date/store number/department number, moving date column to the
beginning and weekly_sales column to the end:
data = data.sort_values(['date', 'store_id', 'department_id']).reset_index(drop=True)
data = pd.concat([data.date, data.drop('date', axis=1)], axis=1)
data = pd.concat([data.drop('weekly_sales', axis=1), data.weekly_sales], axis=1)
data.shape

# The merged data contains 421570 rows and 16 columns.
# Looking at the first 5 rows of data:
data.head()

data.info()
data.apply(lambda x: [x.nunique()])
data.apply(lambda x: [x.unique()])
data.isna().mean()
```

```
# <br>
# ***
# <br>

# <h1>2. Exploratory Data Analysis</h1><br>

# <h2>2.1. Univariate Analysis</h2>

cats = data[['date', 'store_id', 'department_id', 'is_holiday', 'type']]
nums = data.drop(cats.columns, axis=1).fillna(0)

# <br>
# <h3>- Categorical</h3>

pd.DataFrame({'Categorical Variables':cats.columns})

def analyze_cats(dataframe, column_name):
    print('-' * 100 + '\n' + 'Number of Unique Values:')
    print(str(dataframe[column_name].nunique()) + '\n' + '-' * 100)
    print('Unique Values:')
    print(np.sort(dataframe[column_name].unique()), '\n' + '-' * 100)
    uniques = list(dataframe[column_name].value_counts().index)
    counts = list(dataframe[column_name].value_counts().values)
    percentages = list(dataframe[column_name].value_counts(normalize=True).values)
    freq_table_list =  list(zip(uniques, counts, percentages))
    freq_table = pd.DataFrame(freq_table_list, columns = [column_name.capitalize(),
'Count' , 'Count%'])
    plt.figure(figsize=(18, 8))
    if len(dataframe[column_name].unique()) < 5:
        display(freq_table, dataframe[column_name].value_counts(normalize =
True).plot(kind='pie',
                                                  labels=dataframe[column_name].unique(),
                                                  autopct='%1.1f%%',
                                                  startangle=90))
    else:
        display(freq_table, dataframe[column_name].value_counts(normalize =
True).plot(kind='bar', legend=True))

# <br>
# <b>date</b>
analyze_cats(data, 'date')
# Dataset consists of weekly sales of Wallmart Stores from 2010-02-05 to 2012-10-26.
# So time period of the data can be considered as 2 years and 9 months or 143 weeks.

# <br>
# <b>store_id</b>
analyze_cats(data, 'store_id')
# There are 45 Wallmart Stores in the dataset.
# <br>
# <b>department_id</b>
analyze_cats(data, 'department_id')
# Dataset contains 81 different departments in the stores.
# Furthermore, it can be understood that while some departments exists in most of stores,
some exists in only few stores.

# <br>
# <b>is_holiday</b>
analyze_cats(data, 'is_holiday')
# The outputs show that "is_holiday" variable is highly unbalanced since 93% of the
weekly sales did not occur in the holidays.

# <br>
# <b>type</b>
analyze_cats(data, 'type')
# While roundly half of the weekly sales records are related to Type A stores, almost 40%
of them occurred in Type B stores and Type C stores have only 10 percentage.

# <br>
# <h3>- Numerical</h3>

pd.DataFrame({'Numerical Variables':nums.columns})

nums.describe()

pd.DataFrame({'Features':nums.var().index,
'Variance':nums.var().values}).sort_values('Variance')

pd.DataFrame({'Features':(nums.std() / nums.mean()).index,
            'CV':(nums.std() / nums.mean()).values}).sort_values('CV', ascending=False)

pd.DataFrame({'Features':nums.skew().index,
'Skewness':nums.skew().values}).sort_values('Skewness')

pd.DataFrame({'Features':nums.kurtosis().index,
'Kurtosis':nums.kurtosis().values}).sort_values('Kurtosis')

for i in nums.columns:
    plt.figure(figsize=(12, 2))
    sns.distplot(nums[i])

nums.hist(figsize=(20, 15), bins=20, xlabelsize=9, ylabelsize=9);

for i in nums.columns:
    plt.figure(figsize=(12, 2))
    sns.boxplot(x=nums[i])
```

```
# <br><br>
# <h2>2.2. Bivariate Analysis</h2><br>
# <h3>- Categorical & Numerical</h3>

cats['is_holiday'] = cats['is_holiday'].replace({False:0, True:1})

cats['type'] = cats['type'].replace({'A':3, 'B':2, 'C':1})

cats = cats.astype({'date':'category'})

cats['date'] = cats['date'].cat.codes

for i in cats.columns:
    print(i)
    print(stats.kruskal(cats[i], nums['weekly_sales']))
    print('\n')

# According to Kruskal Wallis test result, sample distributions in the categorical variables
are not equal.

# <br>
# <h3>- Numerical & Numerical</h3>

# Correlation Analysis based on "Spearman" method will be used since numerical features
don't have gaussian distribution.
plt.figure(figsize=(12, 12))
sns.heatmap(round(abs(nums.corr(method ='spearman')), 2), vmin=0, vmax=1,
            center=0.5, annot=True, cmap=plt.cm.Reds, square=True);

round(abs(nums.corr(method ='spearman')), 2)[round(abs(nums.corr(method
='spearman')), 2) > 0.7]     [round(abs(nums.corr(method ='spearman')), 2) <
1.0].dropna(how='all', axis=[0, 1])

for i in nums.drop('weekly_sales', axis=1).columns:
    plt.figure(figsize=(12, 2))
    sns.scatterplot(x=i, y="weekly_sales", data=nums);

# <br><br>
# <h2>2.3. Multivariate Analysis</h2>

# For Multivariate Analysis, ANCOVA will be used since data has both categorical and
numerical features.

encoder = ce.BinaryEncoder(cols=['date', 'store_id', 'department_id', 'type'],
drop_invariant=True)

cats = encoder.fit_transform(cats)
```

```
all_columns = " + ".join(pd.concat([cats, nums], axis=1).columns)[:-15]

formula = "weekly_sales ~ " + " + ".join(pd.concat([cats, nums], axis=1).columns)[:-15]

results = ols(formula, data=pd.concat([cats, nums], axis=1)).fit()

results.summary()

# <br>
# ***
# <br>

# <h1>3. Methodology</h1><br>

# <h2>3.1. Data Preprocessing</h2>

# Firstly, variables will be split into two sets as "features" and "target" in order to
implement preprocessing easily.

features = data.drop('weekly_sales', axis=1)
target = pd.DataFrame(data['weekly_sales'], columns=['weekly_sales'])

# <br>
# <h3>- Missing Value Treatment</h3>

features.isna().mean()

target.isna().mean()

features[['markdown_1','markdown_2','markdown_3','markdown_4', 'markdown_5']] =
features[['markdown_1','markdown_2','markdown_3','markdown_4',
'markdown_5']].fillna(0)

features.isna().sum()

# <br>
# <h3>- Outlier Treatment</h3>

# Identifying the outliers in continuous variables based on IQR Score Method:
def find_outliers(data, column_list):
    for i in column_list:
        Q1 = data[i].quantile(0.25)
        Q3 = data[i].quantile(0.75)
        IQR = Q3 - Q1
        print(i + ' ' * (13 - len(i)) + ': ' +
              str(len(data[i][(data[i] < (Q1 - 3 * IQR)) | (data[i] > (Q3 + 3 * IQR))])))

find_outliers(features, features.select_dtypes(include=['int64', 'float64']).columns)
find_outliers(features[features > 0], ['markdown_1', 'markdown_2', 'markdown_3',
'markdown_4', 'markdown_5'])

find_outliers(target, ['weekly_sales'])

# <br>
# <h3>- Feature Engineering</h3>

features = features.astype({'date':'datetime64'})
features['week_number'] = features['date'].dt.week
features = features.drop('date', axis=1)

# <br>
# <h3>- Label Encoding</h3>
le = LabelEncoder()
features['is_holiday'] = le.fit_transform(features['is_holiday'])
features['type'] = le.fit_transform(features['type'])

# <br>
# <h3>- Feature Selection</h3>
print(sorted(features[features['is_holiday'] == True]['week_number'].unique()))
print(sorted(features[features['is_holiday'] == False]['week_number'].unique()))
# Since "week_number" represents the information which "is_holiday" variable offers,
"is_holiday" will be dropped.
features = features.drop('is_holiday', axis=1)

# <br>
# <b>Based on Pairwise Correlation</b>
corr_data = round(abs(pd.concat([features, target], axis=1).corr(method ='spearman')), 2)
plt.figure(figsize=(12, 12))
sns.heatmap(corr_data, vmin=0, vmax=1, center=0.5, annot=True, cmap=plt.cm.Reds,
square=True);
corr_data[corr_data > 0.7][corr_data < 1.0].dropna(how='all', axis=[0, 1])
pd.DataFrame({'Features':corr_data['weekly_sales'].sort_values(ascending=False).index,
            'Corr. with
Target':corr_data['weekly_sales'].sort_values(ascending=False).values}).drop(0, axis=0)
# Since "markdown_4" and "store_size" features have higher correlation with the target,
other variables having high correlation will be dropped:
features = features.drop(['markdown_3', 'markdown_2', 'markdown_5', 'markdown_1',
'type'], axis=1)

# <br>
# <b>Based on Variance</b>
pd.DataFrame({'Features':features.var().index,
'Variance':features.var().values}).sort_values('Variance')
# Since "fuel_price" has very low variation, it will be dropped.
features = features.drop('fuel_price', axis=1)
```

```
# <br>
# <b>Based on Feature Importance</b>
model = RandomForestRegressor(random_state=1)
model.fit(features, target)
feat_importances = pd.Series(model.feature_importances_, index=features.columns)
plt.figure(figsize=(12, 6))
feat_importances.sort_values().plot(kind='barh', grid=True)
plt.show()
pd.DataFrame({'Features':feat_importances.sort_values(ascending=False).index,
             'Importances':feat_importances.sort_values(ascending=False).values})
# According to feature importance output, 'markdown_4' will be dropped since it doesn't
have a significant feature importance.
features = features.drop('markdown_4', axis=1)

# <br>
# <h3>- One Hot Encoding</h3>
# Since the scope of this study contains only tree-based models, one hot encoding will not
be applied.

# <br>
# <h3>- Feature Scaling</h3>
# Since the scope of this study contains only tree-based models, feature scaling will not be
applied.

# <br>
# <h3>- Train-Test Split</h3>
x_train_val, x_test, y_train_val, y_test = train_test_split(features, target, test_size=0.2,
random_state=0)
x_train, x_val, y_train, y_val = train_test_split(x_train_val, y_train_val, test_size=0.25,
random_state=0)
print('x_train : {0:.{1}f}%'.format(x_train.shape[0] / features.shape[0] * 100, 0))
print('y_train : {0:.{1}f}%'.format(y_train.shape[0] / target.shape[0] * 100, 0))
print('\n')
print('x_val   : {0:.{1}f}%'.format(x_val.shape[0] / features.shape[0] * 100, 0))
print('y_val   : {0:.{1}f}%'.format(y_val.shape[0] / target.shape[0] * 100, 0))
print('\n')
print('x_test  : {0:.{1}f}%'.format(x_test.shape[0] / features.shape[0] * 100, 0))
print('y_test  : {0:.{1}f}%'.format(y_test.shape[0] / target.shape[0] * 100, 0))

# <br><br>
# <h2>3.2. Model Building</h2>

# <b>A. Bootstrap Aggregation</b>
# --- Bagging (BaggingRegressor)
# --- Random Forest (RandomForestRegressor)
# --- Extremely Randomized Trees (ExtraTreesRegressor)

# <b>B. Boosting</b>
# --- Adaptive Boosting (AdaBoostRegressor)
# --- Extreme Gradient Boosting (XGBoost)

# <b>C. Stacked Generalization</b>
# --- Voting Regressor (VotingRegressor)
# <br>

# In order to obtain reasonable comparison, the maximum depths of the model was
selected same in all models as 7 for the beginning.
# Different values of depth will be tried in the stage of hyperparameter tuning.

models = {'Bagging' :
BaggingRegressor(base_estimator=DecisionTreeRegressor(max_depth=7,
random_state=1),
                              n_estimators=100,
                              n_jobs=-1,
                              random_state=1),
        'Random F.' : RandomForestRegressor(n_estimators=100,
                                   max_depth=7,
                                   n_jobs=-1,
                                   random_state=1),
        'Extra T.' : ExtraTreesRegressor(n_estimators=100,
                                 max_depth=7,
                                 n_jobs=-1,
                                 random_state=1),
        'AdaBoost' :
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=7,
random_state=1),
                               n_estimators=100,
                               learning_rate=0.1,
                               random_state=1),
        'XGBoost' : xgb.XGBRegressor(n_estimators=100,
                              learning_rate=0.1,
                              max_depth=7,
                              n_jobs=-1,
                              random_state=1)}

model_name = list(models.keys())
mae_scores = []
rmse_scores = []
r2_scores = []
times = []

for i in models:
    start = time.time()
    model = models[i]
    model.fit(x_train, y_train)
    y_pred = model.predict(x_val)
    mae_scores.append(mean_absolute_error(y_val, y_pred))
    rmse_scores.append(sqrt(mean_squared_error(y_val, y_pred)))
    r2_scores.append(r2_score(y_val, y_pred))
    end = time.time()
    times.append(end - start)

compare_list =  list(zip(model_name, mae_scores, rmse_scores, r2_scores, times))
compare = pd.DataFrame(compare_list, columns = ['Model', 'MAE' , 'RMSE', 'R2',
'Time(sec)'])
compare

# In stacked generalization algorithm, XGBoost, AdaBoost and Random Forest will be
used since they have higher scores than other models.

model = VotingRegressor(estimators=[('xg', xgb.XGBRegressor(n_estimators=100,
                                   learning_rate=0.1,
                                   max_depth=7,
                                   n_jobs=-1,
                                   random_state=1)),
                     ('ad',
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=7,
                                                        random_state=1),
                                   n_estimators=100,
                                   learning_rate=0.1,
                                   random_state=1)),
                     ('rf', RandomForestRegressor(n_estimators=100,
                                     max_depth=7,
                                     n_jobs=-1,
                                     random_state=1))],
              n_jobs=-1)

model_name.append('VotingReg')
models['VotingReg'] = model
start = time.time()
model.fit(x_train, y_train)
y_pred = model.predict(x_val)
mae_scores.append(mean_absolute_error(y_val, y_pred))
rmse_scores.append(sqrt(mean_squared_error(y_val, y_pred)))
r2_scores.append(r2_score(y_val, y_pred))
end = time.time()
times.append(end - start)

compare_list =  list(zip(model_name, mae_scores, rmse_scores, r2_scores, times))
compare = pd.DataFrame(compare_list, columns = ['Model', 'MAE' , 'RMSE', 'R2',
'Time(sec)'])
compare
```

```
compare.plot(kind='barh',
            x='Model',
            y=['MAE' , 'RMSE', 'R2'],
            figsize=(14, 8),
            logx=True,
            grid=True,
            legend='reverse');


# <br><br>
# <h2>3.3. Model Evaluation</h2>

# <h3>3.3.1. Performance Metrics</h3>
def take_second(x):
    return x[1]

# <br>
# <b>Mean Absolute Error</b>
for i, j in sorted(zip(model_name, mae_scores), key=take_second, reverse=False):
    print(i + ' ' * (12 - len(i)) + ': {0:.{1}f}'.format(j, 2))
compare.sort_values('MAE', ascending=False).plot(kind='barh',
                                                 x='Model',
                                                 y='MAE',
                                                 figsize=(12, 6),
                                                 legend=False);

# <br>
# <b>Root Mean Squared Error</b>
for i, j in sorted(zip(model_name, rmse_scores), key=take_second, reverse=False):
    print(i + ' ' * (12 - len(i)) + ': {0:.{1}f}'.format(j, 2))
compare.sort_values('RMSE', ascending=False).plot(kind='barh',
                                                  x='Model',
                                                  y='RMSE',
                                                  figsize=(12, 6),
                                                  legend=False);

# <br>
# <b>R-Squared</b>
for i, j in sorted(zip(model_name, r2_scores), key=take_second, reverse=True):
    print(i + ' ' * (12 - len(i)) + ': {0:.{1}f}'.format(j, 2))
compare.sort_values('R2').plot(kind='barh',
                               x='Model',
                               y='R2',
                               figsize=(12, 6),
                               legend=False);

# <br>
# <b>Runtime</b>
```

```
for i, j in sorted(zip(model_name, times), key=take_second, reverse=False):
    print(i + ' ' * (12 - len(i)) + ': {0:.{1}f}'.format(j, 2))
compare.sort_values('Time(sec)', ascending=False).plot(kind='barh',
                                                        x='Model',
                                                        y='Time(sec)',
                                                        figsize=(12, 6),
                                                        legend=False);

# <br><br>
# <h3>3.3.2. Cross-Validation</h3>
# In order to achieve an unbiased estimate of the model performance, 5-fold cross-
validation will be used.
all_mae_scores = []
all_rmse_scores = []
all_r2_scores = []
all_times = []

for i in models:
    mae_scores = []
    rmse_scores = []
    r2_scores = []
    times = []
    model = models[i]
    cv = KFold(n_splits=5)
    for train_index, test_index in cv.split(x_train_val.values):
        start = time.time()
        x_train, x_val, y_train, y_val = x_train_val.iloc[train_index],
x_train_val.iloc[test_index],                              y_train_val.iloc[train_index],
y_train_val.iloc[test_index]
        model.fit(x_train, y_train)
        y_pred = model.predict(x_val)
        mae_scores.append(mean_absolute_error(y_val, y_pred))
        rmse_scores.append(sqrt(mean_squared_error(y_val, y_pred)))
        r2_scores.append(r2_score(y_val, y_pred))
        end = time.time()
        times.append(end - start)
    all_mae_scores.append(sum(mae_scores) / len(mae_scores))
    all_rmse_scores.append(sum(rmse_scores) / len(rmse_scores))
    all_r2_scores.append(sum(r2_scores) / len(r2_scores))
    all_times.append(round(sum(times) / len(times)))

compare_list_cv =  list(zip(model_name, all_mae_scores, all_rmse_scores, all_r2_scores,
all_times))
compare_cv = pd.DataFrame(compare_list_cv, columns = ['Model', 'MAE' , 'RMSE', 'R2',
'Time(sec)'])
compare_cv.sort_values('MAE')
compare_cv.sort_values('MAE', ascending=False).plot(kind='barh',
                                                     x='Model',
                                y=['MAE' , 'RMSE', 'R2'],
                                logx=True,
                                legend='reverse',
                                figsize=(12, 6));

# <br><br>
# <h3>3.3.3. Hyperparameter Optimization</h3>
grid_param = {'n_estimators'     : range(100, 1000, 100),
              'learning_rate'    : [0.01, 0.05, 0.1],
              'max_depth'        : range(3, 11),
              'gamma'            : [0, 1, 5],
              'subsample'        : [0.8, 0.9, 1.0],
              'colsample_bytree' : [0.8, 0.9, 1.0]}
xgb_grid = xgb.XGBRegressor(n_jobs=-1,
                            random_state=1)
cv = KFold(n_splits=5)
grid = RandomizedSearchCV(estimator=xgb_grid,
                          param_distributions=grid_param,
                          scoring='neg_mean_absolute_error',
                          cv=cv,
                          n_iter=5,
                          n_jobs=-1,
                          random_state=1)

grid.fit(x_train, y_train)
print(grid.best_params_, -grid.best_score_)

y_pred = grid.best_estimator_.predict(x_val)
print('Mean Absolute Error for Validation: {0:.{1}f}'.format(mean_absolute_error(y_val,
y_pred), 2))

y_pred_test = grid.best_estimator_.predict(x_test)
print('Mean Absolute Error for Test: {0:.{1}f}'.format(mean_absolute_error(y_test,
y_pred_test), 2))

model = xgb.XGBRegressor(learning_rate=0.05,
                         n_estimators=200,
                         max_depth=10,
                         gamma=1,
                         subsample=0.8,
                         colsample_bytree=1,
                         n_jobs=-1,
                         random_state=1)
model.fit(x_train, y_train)
y_pred_test = model.predict(x_test)
print('Final Mean Absolute Error: {0:.{1}f}'.format(mean_absolute_error(y_test,
y_pred_test), 2))
```