Restaurant Revenue Prediction Using Regression Techniques

Kavya Mistry, MS in Data Science, Rowan University

1. Introduction

Anything that comes under food industry plays a vital role for the development of a country. Especially restaurants have large amount of share in the societal growth and economic boost. According to Forbes, with approximately 11 million jobs and over 4% of the GDP attributable to the restaurant industry, one could make a very well-founded argument that “as restaurants go, so goes the economy.”

Restaurants being such crucial into business sector comes with its own challenges, like choosing location, type of restaurant, tough competitions in market, varying cultures, subjective personal judgement, inventory decisions, and other demographic, commercial and real estate factors. Fortunately, most of these challenges could be tackled using developing technologies in Machine learning and Artificial Intelligence. The focus of this project/paper is to explore with various methods to create an efficient Revenue prediction model for a restaurant using Data Mining applications.

2. Business Problem and Tasks

The main aspect of this challenge is to determine revenue for a restaurant and obtain significant insights on whether that restaurant will be profitable or will it incur loss of money in the near future.

The steps involved to achieve the above goal are Data-preprocessing, Exploratory Data Analysis (EDA), Data Visualization, Feature Engineering, Training and Testing of machine learning model. As this is supervised regression problem, the paper also talks about evaluation of various Regression machine learning models to check their performance for the dataset.

3. Dataset

For this project, I took the dataset from one of the Kaggle competition. With over 1,200 quick service restaurants across the globe, TFI is the company behind some of the world's most well-known brands: Burger King, Sbarro, Popeyes, Usta Donerci, and Arby’s. TFI has provided a dataset with 137 restaurants in the training set, and a test set of 100000 restaurants. The description of the data fields is stated below:

- Id: Restaurant id.
- Open Date: opening date for a restaurant
- City: City that the restaurant is in.
- City Group: Type of the city. Big cities, or Other.
- Type: Type of the restaurant. FC: Food Court, IL: Inline, DT: Drive Thru
- P1, P2 - P37: There are three categories of these obfuscated data. Demographic data are gathered from third party providers with GIS systems. These include population in any given area, age and gender distribution, development scales. Real estate data mainly relate to the m2 of the location, front facade of the location, car park availability. Commercial data mainly include the existence of points of interest

including schools, banks, other QSR operators.

- Revenue (Target): The revenue column indicates a (transformed) revenue of the restaurant in a given year and is the target of predictive analysis.

4. Data-Preprocessing

Using Python libraries like pandas and numpy, I imported my dataset. Luckily, there were no null values in the dataset for all attributes. The Open Date column for both training and testing data was converted to proper format using to_datetime function. For better understanding and EDA process I differentiated the data into two separate variables, namely categorical values and numerical values.

```
1 data['Open Date'] = pd.to_datetime(data['Open Date'])
2 data1['Open Date'] = pd.to_datetime(data1['Open Date'])
```

```
1 data.head()
```

	Id	Open Date	City	City Group	Type	P1	P2	P3	P4	P5	...	P29	P30	P31	P32	P33	P34	P35	P36	P37	revenue
0	0	1999-07-17	Istanbul	Big Cities	IL	4	5.0	4.0	4.0	2	...	3.0	5	3	4	5	5	4	3	4	5653753
1	1	2008-02-14	Ankara	Big Cities	FC	4	5.0	4.0	4.0	1	...	3.0	0	0	0	0	0	0	0	0	6923131
2	2	2013-03-09	Diyarbakır	Other	IL	2	4.0	2.0	5.0	2	...	3.0	0	0	0	0	0	0	0	0	2055379
3	3	2012-02-02	Tokat	Other	IL	6	4.5	6.0	6.0	4	...	7.5	25	12	10	6	18	12	12	6	2675511
4	4	2009-05-09	Gaziantep	Other	IL	3	4.0	3.0	4.0	2	...	3.0	5	1	3	2	3	4	3	3	4316715

5 rows × 43 columns

```
1 data1.head()
```

	Id	Open Date	City	City Group	Type	P1	P2	P3	P4	P5	...	P28	P29	P30	P31	P32	P33	P34	P35	P36	P37
0	0	2011-01-22	Niğde	Other	FC	1	4.0	4.0	4.0	1	...	2.0	3.0	0	0	0	0	0	0	0	0
1	1	2011-03-18	Konya	Other	IL	3	4.0	4.0	4.0	2	...	1.0	3.0	0	0	0	0	0	0	0	0
2	2	2013-10-30	Ankara	Big Cities	FC	3	4.0	4.0	4.0	2	...	2.0	3.0	0	0	0	0	0	0	0	0
3	3	2013-05-06	Kocaeli	Other	IL	2	4.0	4.0	4.0	2	...	2.0	3.0	0	4	0	0	0	0	0	0
4	4	2013-07-31	Afyonkarahisar	Other	FC	2	4.0	4.0	4.0	1	...	5.0	3.0	0	0	0	0	0	0	0	0

5 rows × 42 columns

```
<class 'pandas.core.frame.DataFrame
RangeIndex: 137 entries, 0 to 136
Data columns (total 43 columns):
Id              137 non-null int64
Open Date       137 non-null object
City            137 non-null object
City Group      137 non-null object
Type            137 non-null object
P1              137 non-null int64
P2              137 non-null float64
P3              137 non-null float64
P4              137 non-null float64
P5              137 non-null int64
P6              137 non-null int64
P7              137 non-null int64
P8              137 non-null int64
P9              137 non-null int64
P10             137 non-null int64
P11             137 non-null int64
P12             137 non-null int64
P13             137 non-null float64
```

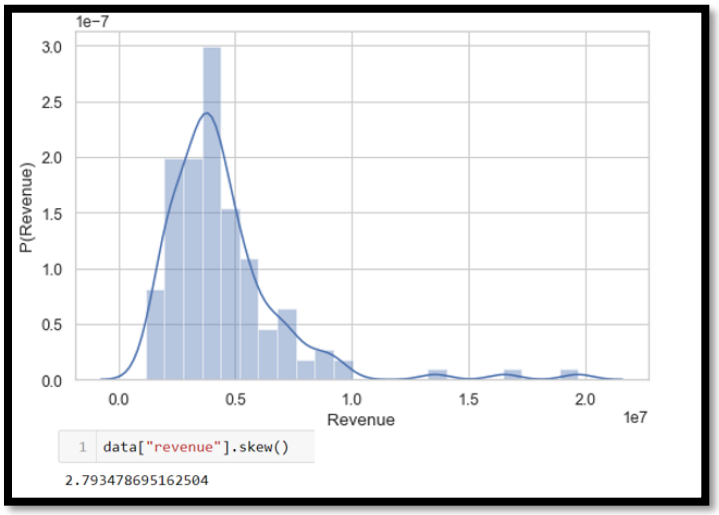
```
1 numerical_features = data.select_dtypes([np.number]).columns.tolist()
2 categorical_features = data.select_dtypes(exclude = [np.number,np.datetime64]).columns.tolist()

1 categorical_features

['City', 'City Group', 'Type']

1 numerical_features

['Id',
 'P1',
 'P2',
 'P3',
 'P4',
 'P5',
```

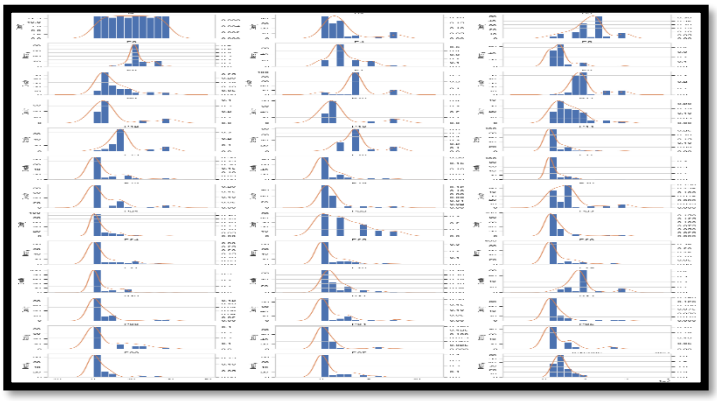
5. EDA & Data Visualization

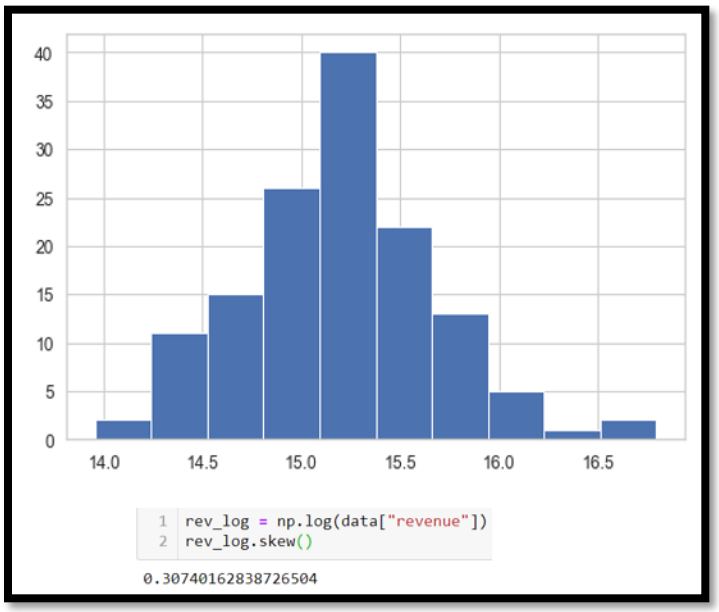
5.1 Analyzing Numerical Features

To get understanding of the distribution of Revenue in the given dataset, I plotted a histogram chart and from the below distribution we can see that the data is mostly right skewed (because of outliers?). It is essential that the target variable is normally distributed for increasing accuracy of the model. There are two ways we can make the target variable with normal distribution, either removing the outliers or perform Log transformation on revenue data values. I would go for the latter as it seems to be more efficient approach.

After performing it can be observed that the value of the skewness decreased from almost 2.80 to 0.30, which is pretty decent.



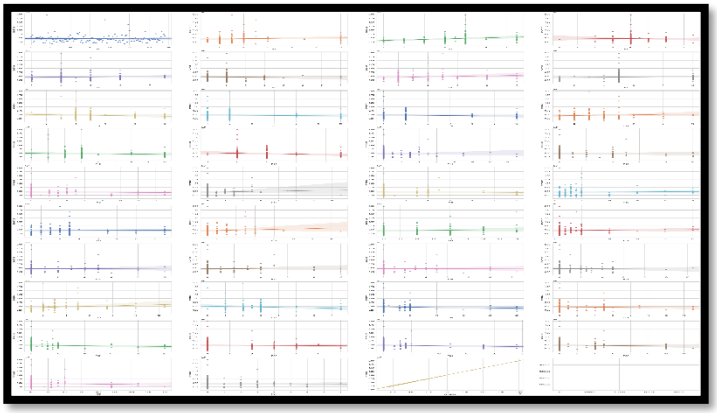


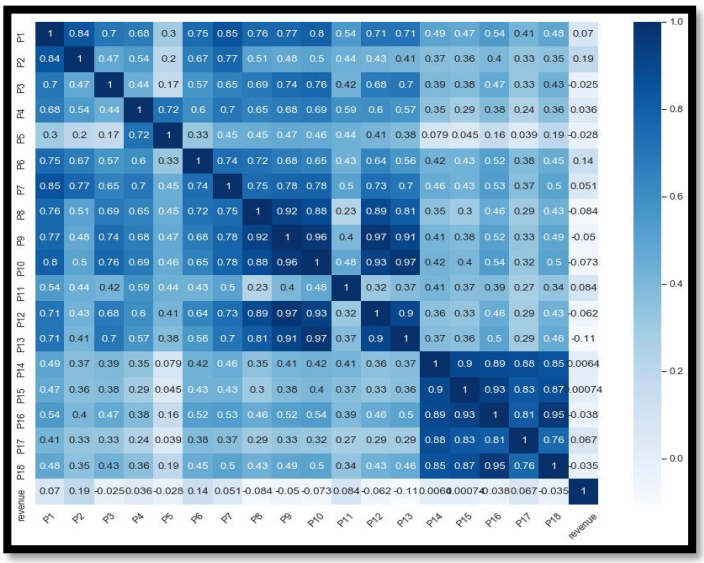


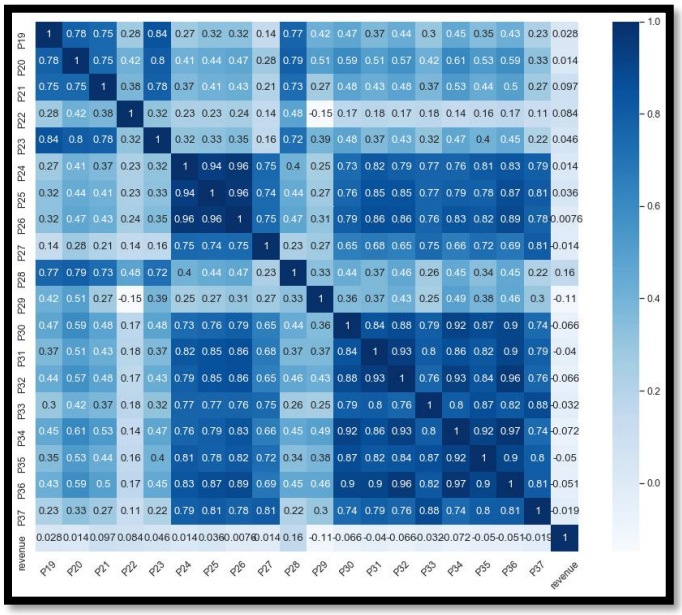
Moreover, I also checked for the distribution of P1 to P37 via Histogram and from below distributions it can be observed that there are clear anomalies in some of the features in the numerical aspect of the data, the suitable method is to study these anomalies with an expert in the field, also the distributions describe certain biases that made that data distribution either not fair or skewed for some reason.

For instance, if we look at P37 value zero is more likely to show among the feature values, possibly this is a data entry error or is there something important that number zero represents?

Next, to analyze the relationship between Revenue and these numerical attributes, I plotted a graph using Regression plot and Correlation plot.

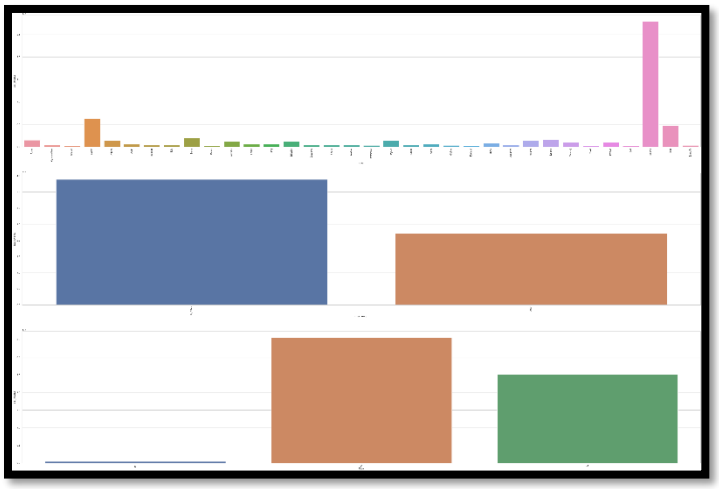






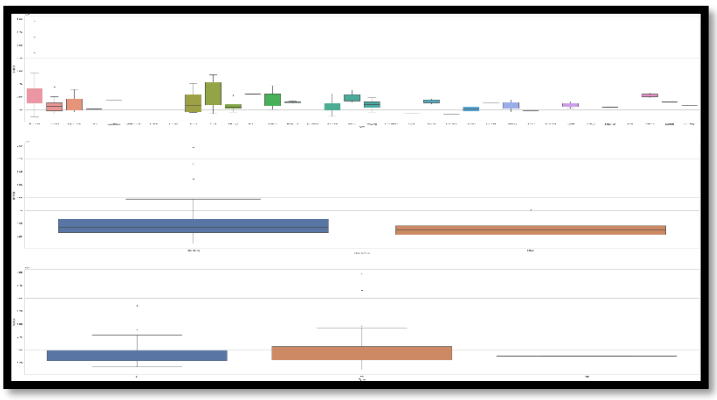
As the heatmap and the scatter distribution clearly shows that there is no obvious relation between numerical variables independently with the target.

5.2 Analyzing Categorical Features Vs Revenue

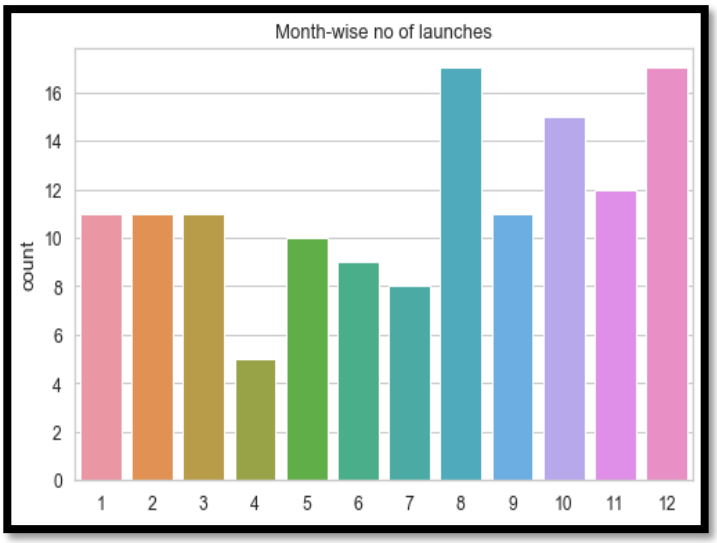


Using Matplotlib library, insights obtained from categorical variables charts shows that Istanbul is making the most revenues compared to other cities, hence opening a restaurant in Istanbul could be a nice idea. It is also visible that big cities are making more revenues than smaller cities. It can also be noted that the gap between them is not an obvious one that is much bigger, may be because even small cities have good tourist attraction. Moreover, if you are about to open a restaurant it better be either FC (Food court) or IL (inline).

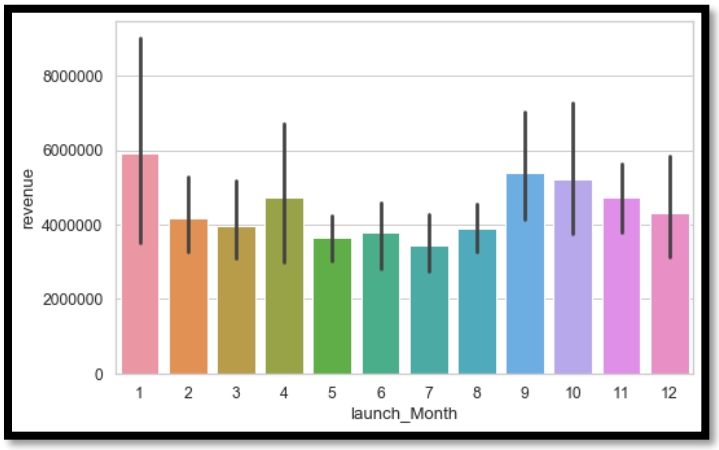
I also made a boxplot to visualize the anomalies in the categorical attributes.



5.3 Analyzing Open Date Column



Using Seaborn library, From the count plot in the above figure, seems like the first three months (January to March) and ending months (August to December) have more launches.



From the bar plot in the right figure, it can be seen that the months that are having more revenue are: January, September, October, November and December. Hence, it's safe to say that one can launch their restaurants in these months to have a better chance of generating profits!

6. One Hot Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data[['City', 'City Group', 'Type']] = data[['City', 'City Group', 'Type']].apply(le.fit_transform)
```

```
data.head()
```

	Id	Open Date	City	City Group	Type	P1	P2	P3	P4	P5	...	P29	P30	P31	P32	P33	P34	P35	P36	P37	revenue
0	0	1999-07-17	31	0	2	4	5.0	4.0	4.0	2	...	3.0	5	3	4	5	5	4	3	4	5653753
1	1	2008-02-14	3	0	1	4	5.0	4.0	4.0	1	...	3.0	0	0	0	0	0	0	0	0	6923131
2	2	2013-03-09	10	1	2	2	4.0	2.0	5.0	2	...	3.0	0	0	0	0	0	0	0	0	2055379
3	3	2012-02-02	28	1	2	6	4.5	6.0	6.0	4	...	7.5	25	12	10	6	18	12	12	6	2675511
4	4	2009-05-09	14	1	2	3	4.0	3.0	4.0	2	...	3.0	5	1	3	2	3	4	3	3	4316715

5 rows × 43 columns

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data1[['City', 'City Group', 'Type']] = data1[['City', 'City Group', 'Type']].apply(le.fit_transform)
```

```
data1.head()
```

	Id	Open Date	City	City Group	Type	P1	P2	P3	P4	P5	...	P28	P29	P30	P31	P32	P33	P34	P35	P36	P37
0	0	2011-01-22	38	1	1	1	4.0	4.0	4.0	1	...	2.0	3.0	0	0	0	0	0	0	0	0
1	1	2011-03-18	27	1	2	3	4.0	4.0	4.0	2	...	1.0	3.0	0	0	0	0	0	0	0	0
2	2	2013-10-30	3	0	1	3	4.0	4.0	4.0	2	...	2.0	3.0	0	0	0	0	0	0	0	0

Before feeding the data to the model, the categorical data should be converted to numbers, so that we can feed these variables in machine learning algorithms using sklearn. For this I imported Label Encoder and did one-hot encoding on City, City type and Type for both train and test data.

7. Model Building &Evaluation

For this purpose, we have training data and testing data separate files. The model is trained on train dataset and to predict the revenue we are using testing dataset. There is another sample file which already has results according to the independent variables. The output obtained from testing data is stored in data frame so that one can compare those predicted results and sample revenue values. Through this we can know about the accuracy of the model, and what fits best to the model.

To train the dataset, numerous regression models were used. For evaluation purpose, Root Mean Square Error (RMSE) was used. Firstly, logistic regression was trained with the below results:

```
logisticRegr.score(x_train,y_train)
```

```
1.0
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
label_list=ypred_label['Prediction'].tolist()
#print('Root Mean squared error {}'.format(sqrt(mean_squared_error(label_list, predictions))))
rmse1=round(sqrt(mean_squared_error(label_list, predictions)),2)
print(rmse1)
```

```
2366286.68
```

Secondly, Gradient Boosting Regression was applied and below were the results:

```
gbRegr.score(x_train, y_train)
```

```
0.9403860943246145
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
label_list=ypred_label['Prediction'].tolist()
#print('Root Mean squared error {}'.format(sqrt(mean_squared_error(label_list, predictions))))
rmse2=round(sqrt(mean_squared_error(label_list, prediction_rr)),2)
print(rmse2)
```

```
655991.78
```

Afterwards, with Linear Regression below results were seen:

```
reg.score(x_train, y_train)
```

```
0.3260403831258316
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
label_list=ypred_label['Prediction'].tolist()
#print('Root Mean squared error {}'.format(sqrt(mean_squared_error(label_list, predictions))))
rmse3=round(sqrt(mean_squared_error(label_list, prediction_lr)),2)
print(rmse3)
```

```
218217353.19
```

With Random Forest Regressor below was the output:

```
regrr.score(x_train, y_train)
```

```
0.30770677147819403
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
label_list=ypred_label['Prediction'].tolist()
#print('Root Mean squared error {}'.format(sqrt(mean_squared_error(label_list, predictions))))
rmse4=round(sqrt(mean_squared_error(label_list, prediction_rf)),2)
print(rmse4)
```

```
1538357.74
```

Lastly, using Neighbors Regressor, below output was observed:

```
neigh.score(x_train, y_train)
```

```
0.49045279578697765
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
label_list=ypred_label['Prediction'].tolist()
#print('Root Mean squared error {}'.format(sqrt(mean_squared_error(label_list, predictions))))
rmse5=round(sqrt(mean_squared_error(label_list, prediction_knn)),2)
print(rmse5)
```

```
2346324.81
```

7.1 Stacking for Regression

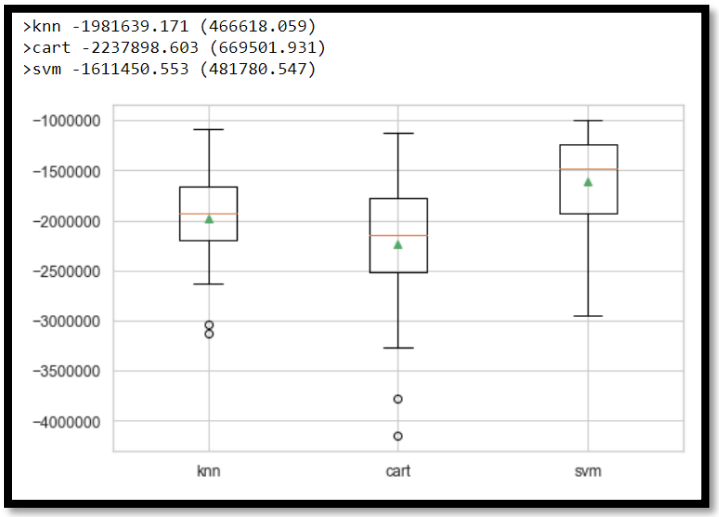
Stacked Generalization or “Stacking” for short is an ensemble machine learning algorithm. It involves combining the predictions from multiple machine learning models on the same dataset, like bagging and boosting. The architecture of a stacking model involves two or more base models, often referred to as level-0 models, and a meta-model that combines the predictions of the base models, referred to as a level-1 model.

Level-0 Models (Base-Models): Models fit on the training data and whose predictions are compiled.

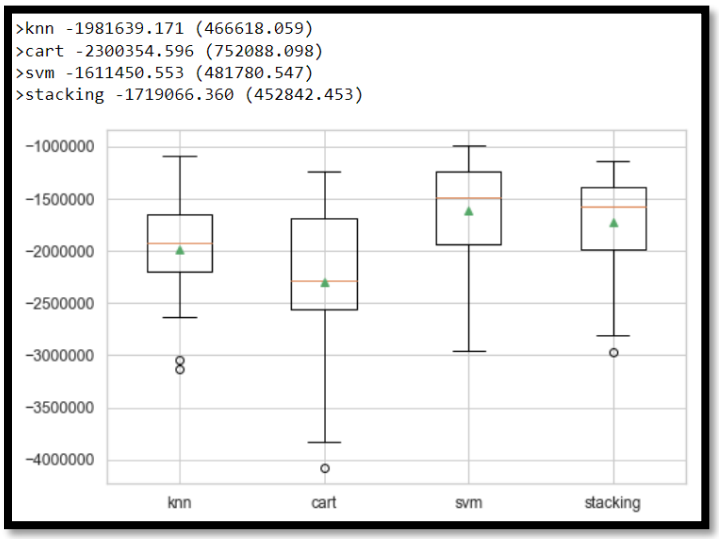
Level-1 Model (Meta-Model): Model that learns how to best combine the predictions of the base models.

A box-and-whisker plot is then created comparing the distribution negative MAE scores for each model.

KNRegressor, Decision Tree Regressor and Support Vector Regressor were used for Level-0 that is as base models for stacking. After implementing Cross validation and model fit, the below figure represents the output:



For level-1, get_stacking function defines the StackingRegressor model by first defining a list of tuples for the three base models, then defining the Linear regression meta-model to combine the predictions from the base models using 5-fold cross-validation. Implementing this, we get the below outputs:



8. Conclusion

From the tasks I did in this project, I got the knowledge about best city & city type, best months and the relationship between the numerical attributes and target variable. Gradient Boosting seems to work best for the dataset as it has good score and pretty reasonable root mean square error value. When done stacking, the negative MSE of stacking and the MSE for support vector regressor doesn't show much difference, hence both can fit well for the dataset.