ANNUAL RESTAURANT SALES PREDICTION

PROJECT TEAM MEMBERS

A N V SREEVISHNU – RA1811003010333

DIBYA DEBAYAN DASH - RA1811004010210

RAHUL DIT – RA1811003010288

REHAN SINGH – RA1811003010305

CONTENTS

ACKNOWLEDGEMENTS

We would like to express immense gratitude to our Project Trainer Mr. Naveen Kumar C for training us in Machine Learning and helping us in every way possible towards our learning, development and completion of our project within stipulated time.

We express our sincere thanks to our Project Mentor Mr. R Rakesh for leading us seamlessly through the project and being our guiding light throughout. Thank you, sir, for providing us ways to keep our internship project efficient and accurate.

We are grateful to Mr. Durga Naveen Kandregula, CEO and Co-Founder Goalstreet, Mr. Srikanth sir, Mr. Seshu sir and all other members at Goalstreet without whom this whole internship programme would not have been possible.

ABSTRACT

With over 1,200 quick service restaurants across the globe, TFI (Tab Food Investments) is the company behind some of the world's most well-known brands: Burger King, Sbarro, Popeyes, Usta Donerci, and Arby's. They employ over 20,000 people in Europe and Asia and make significant daily investments in developing new restaurant sites.

Right now, deciding when and where to open new restaurants is largely a subjective process based on the personal judgement and experience of development teams. This subjective data is difficult to accurately extrapolate across geographies and cultures.

New restaurant sites take large investments of time and capital to get up and running. When the wrong location for a restaurant brand is chosen, the site closes within 18 months and operating losses are incurred.

Finding a mathematical model to increase the effectiveness of investments in new restaurant sites would allow TFI (Tab Food Investments) to invest more in other important business areas, like sustainability, innovation, and training for new employees. Using demographic, real estate, and commercial data, the goal of this project is to predict the annual restaurant sales of 100,000 regional locations.

TFI has provided a dataset with 137 restaurants in the training set, and a test set of 100000 restaurants. The data columns include the open date, location, city type, and three categories of obfuscated data: Demographic data, Real estate data, and Commercial data. The revenue column indicates a (transformed) revenue of the restaurant each year and is the target of predictive analysis.

The aim of this project is to build various regression-based models and evaluate the performance of the models using RMSE, R^2 etc.

INTRODUCTION

This project was focused around helping Tab Food Industries to study annual restaurant sales and constructing a viable model that predicts accurate values of revenue based on the features provided. In this project we implemented various regression models like Logistic Regression, Multiple Regression and Random Forest. We had also implemented some advanced models like Ridge and Lasso Regression and selected that which gave the best accuracy.

The existing system of Linear Regression did not have regularization parameter and hence overfit the data. The system also does not provide enough pre-processing and visualization or Exploratory Data Analysis (EDA). Our new models and implementation include all of these.

DATA REPRESENTATION

Project 1 database consists of 3 datasets. Training set, test set and sample submission dataset. The Training dataset consists of Id, Open Date, City, City Group, Type and P1 to P37 features. The training set has 137 entries and its target being revenue. Test set has the same features with 1,00,000 entries. Our aim is to create revenue predictions for all these entries in test dataset.

The data was given input and studied. All the necessary processes like standardisation elimination and visualization were done. Visualizations were done in the form of bar charts and histograms to get an idea on which features are to be eliminated and which ones are paramount. Since there were no missing data, there was not much handing done there.

After data pre-processing we implemented various advanced models like Lasso, Ridge and Random Forest Regressors. These models were then tested for their accuracy and root mean square error. All these models were implemented and their accuracy specific to each model was coded right below for ease of access and representation.

VISUALIZING THE DATASET

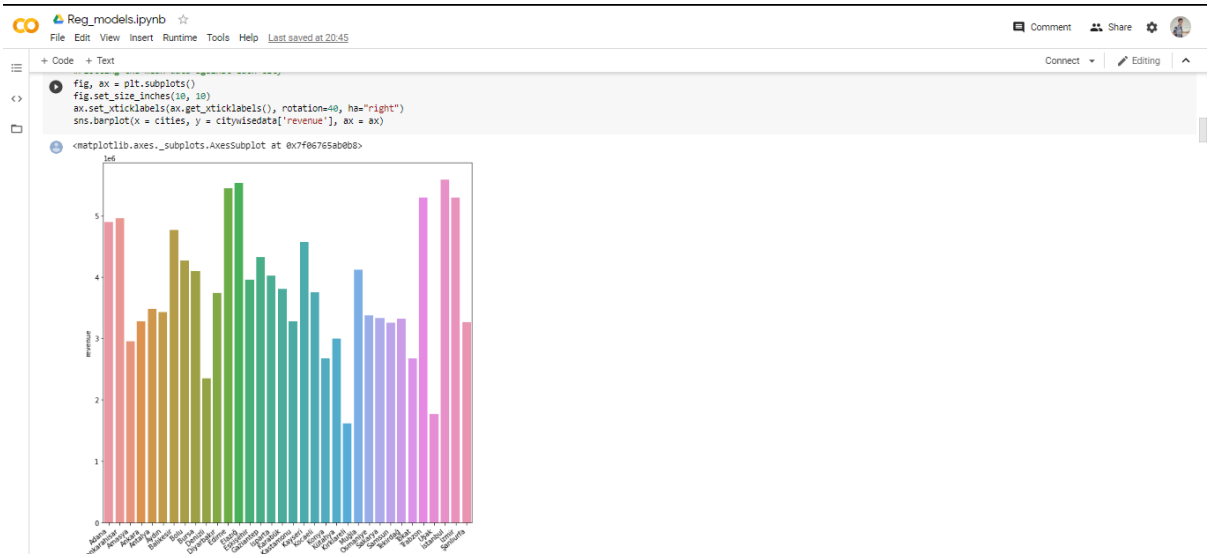
The original dataset – the training dataset looks like this.

```
#importing datasets
data_train = pd.read_csv('train-Project1.csv')
data_test = pd.read_csv('test-Project1.csv')
data_train.head()
```

	Id	Open Date	City	City Group	Type	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26
0	0	07/17/1999	İstanbul	Big Cities	IL	4	5.0	4.0	4.0	2	2	5	4	5	5	3	5	5.0	1	2	2	2	4	5	4	1	3	3	1	1	1.0
1	1	02/14/2008	Ankara	Big Cities	FC	4	5.0	4.0	4.0	1	2	5	5	5	5	1	5	5.0	0	0	0	0	0	3	2	1	3	2	0	0	0.0
2	2	03/09/2013	Diyarbakır	Other	IL	2	4.0	2.0	5.0	2	3	5	5	5	5	2	5	5.0	0	0	0	0	0	1	1	1	1	1	0	0	0.0
3	3	02/02/2012	Tokat	Other	IL	6	4.5	6.0	6.0	4	4	10	8	10	10	8	10	7.5	6	4	9	3	12	20	12	6	1	10	2	2	2.5
4	4	05/09/2009	Gaziantep	Other	IL	3	4.0	3.0	4.0	2	2	5	5	5	5	2	5	5.0	2	1	2	1	4	2	2	1	2	1	2	3	3.0

As shown in this, the dataset consists of the following features; Id which gives a serial number to each revenue. Open Date tells us the date when the restaurant was opened. City is the location which is classified into Big Cities or Other Cities depending on the city. There are 3 Types, IL, FC and DT. Then there are 37 numerical features namely P1 – P37. The target has the label revenue.

We then plot city wise effect on the target – revenue.

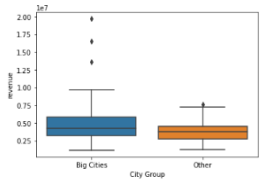


We can see that each city has a different effect on the revenue that is being generated there annually. This kind of bar graphs are our deciding factor on which features to consider and which ones to eliminate. The ones with minimal effect like Open Date are eliminated.

For the categorical data we used box plot.

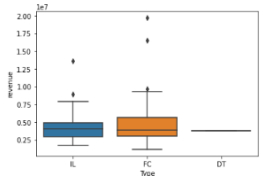
```
#Plotting the categorical data against the target values

sns.boxplot(x = 'City Group', y = 'revenue', data = data_train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f067c484c88>
```




```
[ ] sns.boxplot(x = 'Type', y = 'revenue', data = data_train)
```

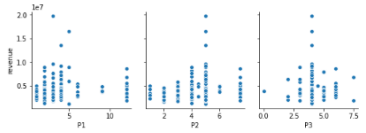
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06796c41d0>
```



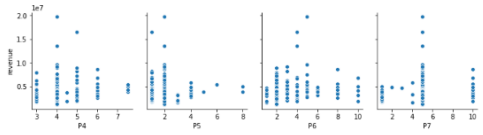
As we can see this plot shows us the relation between Big City or Other Cities to revenue. Since this makes an impact on the target, this feature is important.

Our plot for the 37 numerical features looks like this

```
[ ] #Plotting the Numerical data against the Target Variable
pp = sns.pairplot(data=data_train,
                  y_vars=['revenue'],
                  X_vars=['P1', 'P2', 'P3'])
```




```
pp = sns.pairplot(data=data_train,
                  y_vars=['revenue'],
                  X_vars=['P4', 'P5', 'P6', 'P7'])
```




```
[ ] pp = sns.pairplot(data=data_train,
                  y_vars=['revenue'],
                  X_vars=['P8', 'P9', 'P10', 'P11'])
```

Each feature has a different impact on the revenue. Hence these were prime.

DATA PREPROCESSING

```
[ ] #Creating a flag for each type of restaurant
    data_train['Type_IL'] = np.where(data_train['Type'] == 'IL', 1, 0)
    data_train['Type_FC'] = np.where(data_train['Type'] == 'FC', 1, 0)
    data_train['Type_DT'] = np.where(data_train['Type'] == 'DT', 1, 0)

    #Creating a flag for 'Big Cities'
    data_train['Big_Cities'] = np.where(data_train['City Group'] == 'Big Cities', 1, 0)

    #Converting Open_Date into day count
    #Considering the same date the dataset was made available
    data_train['Days_Open'] = (pd.to_datetime('2015-03-23') - pd.to_datetime(data_train['Open Date'])).dt.days

    #Removing unused columns
    data_train = data_train.drop('Type', axis=1)
    data_train = data_train.drop('City Group', axis=1)
    data_train = data_train.drop('City', axis=1)
    data_train = data_train.drop('Open Date', axis=1)

    #Adjusting test data as well
    data_test['Type_IL'] = np.where(data_test['Type'] == 'IL', 1, 0)
    data_test['Type_FC'] = np.where(data_test['Type'] == 'FC', 1, 0)
    data_test['Type_DT'] = np.where(data_test['Type'] == 'DT', 1, 0)
    data_test['Big_Cities'] = np.where(data_test['City Group'] == 'Big Cities', 1, 0)
    data_test['Days_Open'] = (pd.to_datetime('2015-03-23') - pd.to_datetime(data_test['Open Date'])).dt.days
    data_test = data_test.drop('Type', axis=1)
    data_test = data_test.drop('City Group', axis=1)
    data_test = data_test.drop('City', axis=1)
    data_test = data_test.drop('Open Date', axis=1)
    data_train.head()
```

	Id	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30	P31	P32	P33	P34	P35	P36	P37	revenue	Type_IL	Type_FC	Type_DT	Bi
0	0	4	5.0	4.0	4.0	2	2	5	4	5	5	3	5	5.0	1	2	2	2	4	5	4	1	3	3	1	1	1.0	4.0	2.0	3.0	5	3	4	5	5	4	3	4	5653753.0	1	0	0	
1	1	4	5.0	4.0	4.0	1	2	5	5	5	5	1	5	5.0	0	0	0	0	0	3	2	1	3	2	0	0	0.0	0.0	3.0	3.0	0	0	0	0	0	0	0	0	6923131.0	0	1	0	
2	2	2	4.0	2.0	5.0	2	3	5	5	5	5	2	5	5.0	0	0	0	0	0	1	1	1	1	1	0	0	0.0	0.0	1.0	3.0	0	0	0	0	0	0	0	0	2055379.0	1	0	0	
3	3	6	4.5	6.0	6.0	4	4	10	8	10	10	8	10	7.5	6	4	9	3	12	20	12	6	1	10	2	2	2.5	2.5	2.5	7.5	25	12	10	6	18	12	12	6	2675511.0	1	0	0	

As one can see there is some processing we have done here. Firstly, we encode the Types column. We encode them with the code

```
data_train['Type_IL'] = np.where(data_train['Type'] == 'IL', 1, 0)
```

Once it is encoded, we now encode City Group. This is done by

```
data_train['Big_Cities'] = np.where(data_train['City Group'] == 'Big Cities', 1, 0)
```

We then drop all the unwanted columns which include Type, City Group, City and Open Date. Then we apply the same pre-processing methods to the test dataset as well. If both the datasets are not processed in the same way, the trained model would give an error as it would not recognise any pattern.

The final product of the set after pre-processing is shown in the image above. data_train.head() shows us the modified set. Here Id is not modified. P1 – P37 are lined. Then comes the revenue, Type and City Group. Reducing number of columns results in a faster model that would run seamlessly.

We then take the features from dataset into variables X and Y by the code

```
X = data_train.drop(['Id', 'revenue'], axis=1)

Y = data_train.revenue
```

Now we move forward to the most important part of the project, implementation of models.

MODEL IMPLEMENTATIONS

When we started out with the ideation and discussion of which models to implement, we thought of many models like Lasso, Ridge and Random Forest regression. We implemented the existing system of linear regression to asses the performance. We found that it has no regularization parameter and hence was overfitting. Some proposed methods were lasso and ridge. Then we test out with every regression algorithm to verify a model that would best fit the data.

Here are some images of the models that we implemented.

```
[ ] #Implementing Multiple Regression
    from sklearn.linear_model import LinearRegression
    from sklearn import metrics
    regressor = LinearRegression()
    regressor.fit(X, Y)

    test_predicted_mreg = pd.DataFrame()
    test_predicted_mreg['Id'] = data_test.Id
    test_predicted_mreg['Prediction'] = regressor.predict(data_test.drop('Id', axis=1))
    test_predicted_mreg.head()
```

	Id	Prediction
0	0	4.669238e+06
1	1	2.741557e+06
2	2	1.815584e+06
3	3	5.312600e+06
4	4	5.493380e+06

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

    regressor_accuracy = LinearRegression()
    regressor_accuracy.fit(X_train, y_train)

    y_pred = regressor_accuracy.predict(X_test)

    print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
    print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
    print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 3004513.8682692987
Mean Squared Error: 19196941411388.652
Root Mean Squared Error: 4381431.434062235
```

As seen in the image above, this is the implementation of the multiple regression model. For this we import some methods from scikit-learn. We then use the LinearRegression() method to train the model. This is done by using regressor.fit(X, Y). Then we implement this on the test data by the code

```
test_predicted_mreg = pd.DataFrame()

test_predicted_mreg['Id'] = data_test.Id
test_predicted_mreg['Prediction'] = regressor.predict(data_test.drop('Id', axis=1))
```

Our aim here is to measure the mean squared error and root mean square error. As shown in the image the mse and rmse value show to be 19196941411388.652 and 4381431.434062235.

Similarly we implemented other models like Decision Tree regression, Support Vector Machines, Random Forest regression, Ridge and Lasso Regression.

```
#Implementing Ridge and Lasso Model
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn import metrics

#Lasso Regression
model = Lasso(alpha=5.5)
model.fit(X, Y)

test_predicted = pd.DataFrame()
test_predicted['Id'] = data_test.Id
test_predicted['Prediction'] = model.predict(data_test.drop('Id', axis=1))
test_predicted.head()
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_coordinate_descent.py:476: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 268250168671059.75,
  positive)
```

	Id	Prediction
0	0	4.669774e+06
1	1	2.740680e+06
2	2	1.815398e+06
3	3	5.311360e+06
4	4	5.492584e+06

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

regressor_accuracy = Lasso(alpha=5.5)
regressor_accuracy.fit(X_train, y_train)

y_pred = regressor_accuracy.predict(X_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
```

The image above shows pat of the implementation of the Lasso Regression model. Ridge and Lasso models provided us with the best predictions because Ridge and Lasso models use regularisation which sets a regularisation term which in tun reduces the effect of non-important features, highlighting the more important ones.

CONCLUSION

Restaurants are a high investment and high profit business ventures when done at the right location with the right resources. One such organization that helps in studying and deciding the location and resources required for a start of a restaurant is Tab Food Industries. One of the projects here was predicting the annual sales revenue based on objective measurements. With the help of Machine Learning models, we could make the necessary predictions. These models predict the revenue of a restaurant by training a model based on the dataset given where the features are studied, and a model is trained.

In this specific project, Ridge and Lasso models provided us with the best predictions. This is because these models use regularisation which sets a regularisation term which in turn reduces the effect of non-important features and highlights the more important ones. This also solved the problem of overfitting which was prominent in the existing model of linear regression.

REFERENCES

Code link:

https://colab.research.google.com/drive/1ejlonRPISFJj8Stp5G0Z9MqEukYLxMd_#scrollTo=6-R5rjoTwuzY

https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114

https://www.studytonight.com/post/what-is-mean-squared-error-mean-absolute-error-root-mean-squared-error-and-r-squared

https://machinelearningmastery.com/data-visualization-methods-in-python/