




FEBRUARY 7, 2019

CSCI5901 - Special Topics in Appl. Com Sc
ASSIGNMENT-1

SHARMILA THIRUMALAINATHAN (B00823663)
AISHWARYA NARAYANAN (B00820313)



Contents

1. INTRODUCTION.....	1
2. DATA PREPROCESSING AND UNDERSTANDING.....	3
DATASET LOADING.....	3
FREQUENCY DISTRIBUTION	4
VISUALIZATION	4
ATTRIBUTES SIGNIFICANCE.....	4
DROPPING DUPLICATES	5
NEIGHBOURHOOD WITH HIGHEST AVERAGE RATING	5
3. MODELS.....	5
DATA PREPROCESSING.....	5
FEATURE ENGINEERING	6
STANDARDIZATION	6
MODEL TYPE.....	6
MODELS CHOSEN	6
EVALUATION METRICS.....	7
OVERFITTING AVOIDANCE	7
MODEL PERFORMANCE AND TESTING	8
MODEL TUNING	9
RELIEF FEATURE SELECTION.....	10
4. REFERENCES.....	10

1. INTRODUCTION

The Zomato dataset was retrieved from Kaggle to forecast the approximate cost of two people in for a restaurant in Bangalore, India based on various input variables with a statistical learning algorithm. It contains 51717 records and 17 columns. Below is the brief description of each attribute in the dataset.

Target Variable: - Approx Cost of two people (Continuous variable)

Explanatory Variables: -

Discrete variables: -

- URL - The zomato link for the restaurant (String)
- Address - The address of the restaurant (String)
- Name - Defining name for each restaurant (String)
- online_order - Values of this attribute will be either "Yes" or "no".
- Specifies whether online ordering is available in that particular hotel. (Boolean)
- book_table - Values of this attribute will be either "Yes" or "no".
- Specifies whether the particular hotel is having the feature of booking a table. (Boolean)
- phone - Phone number for each hotel (Long)
- location - Defines in which area the hotel is located (String)
- rest_type - Uniquely identifies each restaurant type for a hotel. Each restaurant can have more than one restaurant type (String Array)
- dish_liked - Identifies the dishes which are most liked by people for a hotel. Each restaurant can have more than one dish liked (String Array)
- Cuisines - Describes the cuisine type for each hotel. Multiple cuisines can exist in each hotel. (comma separated values)
- reviews_list - Provides reviews given for that hotel and the rating. Number of reviews can be more than one for each hotel (Array of tuples)
- menu_item - Specifies the list of menus for each hotel. (Array of string)
- listed_in(type) - Specifies in which restaurant type the hotel is listed under in zomato. It could have been listed in more than one restaurant type (comma separated values)
- listed_in(city) - Specifies in which area of Bangalore the hotel is located. It could have been listed in more than one neighbourhood based on the distance. (comma separated values)

Continuous variables: -

- approx_cost(for two people) - Provides approximate cost of two people for each hotel.
- rate - Identifies the average rating of each hotel.
- votes - Number of votes obtained for each hotel.

Description of the dataset: -

votes	
count	51717.000000
mean	283.697527
std	803.838853
min	0.000000
25%	7.000000
50%	41.000000
75%	198.000000
max	16832.000000

```
zomato_data = pd.read_csv('drive/My Drive/Dataset/zomato.csv')  
zomato_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 51717 entries, 0 to 51716  
Data columns (total 17 columns):  
 url                51717 non-null object  
 address            51717 non-null object  
 name               51717 non-null object  
 online_order       51717 non-null object  
 book_table         51717 non-null object  
 rate              43942 non-null object  
 votes             51717 non-null int64  
 phone             50509 non-null object  
 location          51696 non-null object  
 rest_type         51490 non-null object  
 dish_liked        23639 non-null object  
 cuisines          51672 non-null object  
 approx_cost(for two people) 51371 non-null object  
 reviews_list      51717 non-null object  
 menu_item         51717 non-null object  
 listed_in(type)   51717 non-null object  
 listed_in(city)   51717 non-null object  
 dtypes: int64(1), object(16)  
 memory usage: 6.7+ MB
```

2. DATA PREPROCESSING AND UNDERSTANDING

DATASET LOADING: -

The data was loaded into the google collaboratory notebook by mounting the drive with Zomato data. The CSV file was then loaded with pandas through "pd.read_csv" command.

DATASET LOADING											
<pre>zomato_data = pd.read_csv('drive/My Drive/Dataset/zomato.csv') zomato_data.head()</pre>											
	url	address	name	online_order	book_table	rate	votes	phone	location	rest_type	dish_liked
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	42297555/vn+91 9743772233	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...

Zomato Dataset

FREQUENCY DISTRIBUTION: -

The dataset was visualized using pyplot of matplotlib library. All the attributes were plotted against their frequency to infer the significance of each. All the attributes except "approx_cost(for two people)" was not plotted, since it is the target variable which is to be predicted.

VISUALIZATION: -

Histogram was chosen for plotting frequency distribution of each attribute. It is because plotting the categorical values as bar chart would ultimately forecast the comparison among each attribute. Whereas with histogram, the underlying distribution of data points and how widely they are distributed can be finely projected. Hence histogram was chosen for visualization of the frequency distribution.

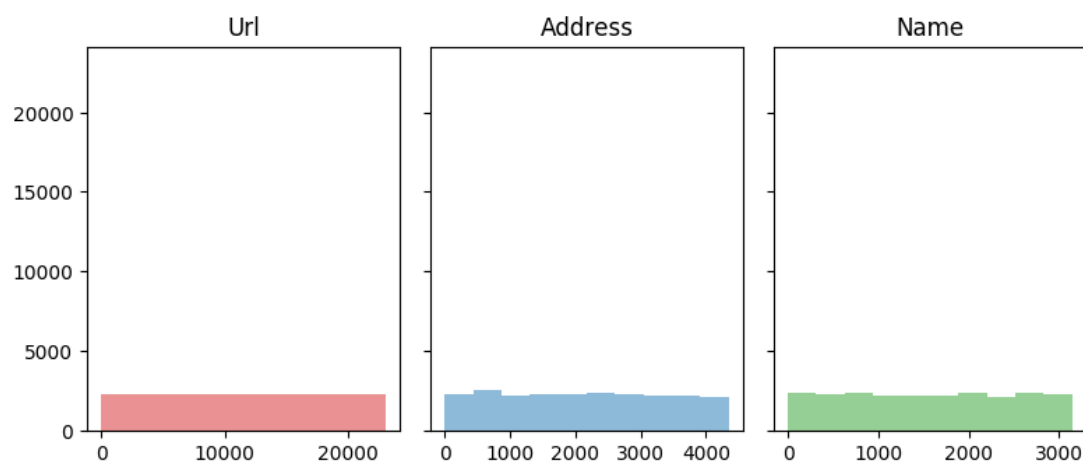
Since histograms are known for plotting continuous values, all the categorical values in the data are transformed to continuous values through label encoder.

The attributes which has comma separated values are processed separately to plot multiple values in an instance.

ATTRIBUTES SIGNIFICANCE: -

1. By plotting frequency distribution for all the features, the attributes name, URL, address and phone were **uniformly distributed**, and the x-limit was equal to the number of instances in the dataset. Hence, they were discarded as they had no significance in predicting the class variable.

Frequency Distribution for Explanatory Variables



Frequency distribution of sample attributes

hot encoding was used in the cases where the instances had one only value and multi hot encoding was used when instances had multiple values.

FEATURE ENGINEERING: -

1. On observing the values of the attributes "rest_type" and "listed_in(type)" the values were identical but with different inflectional endings. Hence in order to transform to their root word while preserving the data significance, lemmatization was performed. By doing this, the number of columns was reduced after encoding.

2. From the inference in the number of duplicates in the dataset, the records which had the same name and address values but different listed_in(type) and listed_in(city) values were merged into a single row. It was achieved by joining different listed_in(type) and listed_in(city) values with identical names and address records as comma-separated values.

STANDARDIZATION: -

In the given dataset, the features "approximate cost of two people" and "rate" are measured at different scales. In order to normalize them, "StandardScaler" was used from sklearn.

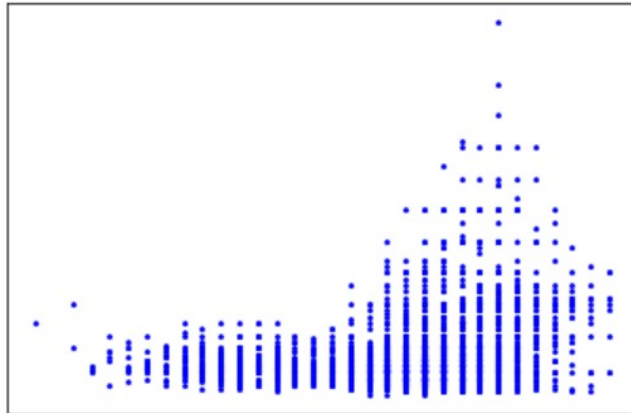
MODEL TYPE: -

It is a supervised learning approach as the data is labeled. As we need to predict continuous variable (approx_cost(for two people)), it is a regression problem.

MODELS CHOSEN: -

1. Linear Regression - The data was plotted in scatter plot to identify if there is linearity in the data. On visualizing it was observed that the data was scattered throughout and the model could not fit the data well. In order to gain further insight, the model was evaluated by finding the accuracy and the result was very low as expected.

```
[289] import matplotlib.pyplot as plt
      y_pred = lr.predict(test)
      plt.figure()
      plt.scatter(xTrain['rate'], yTrain, color='b',s=5,marker='*')
      plt.xticks(())
      plt.yticks(())
      plt.show()
```



Scatter plot of data shape

2. Decision tree regressor (Cart Trees) - As there was non-linearity in data, decision tree regressor would be a simple and ideal model to predict class variables for the given dataset.

EVALUATION METRICS: -

The evaluation metrics that are used here are

1. Mean Absolute Error - It provides absolute error between the predicted values and actual values.

2. Accuracy (in percentage) - A customized function was made to produce MAE values in percentage to have good inference.

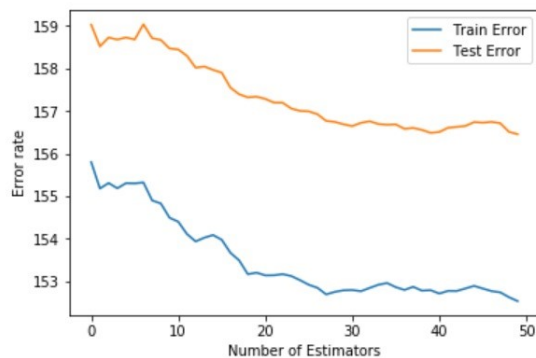
3. R2 Score - The R2 score for regression determines how close the model fits the actual data.

OVERFITTING AVOIDANCE: -

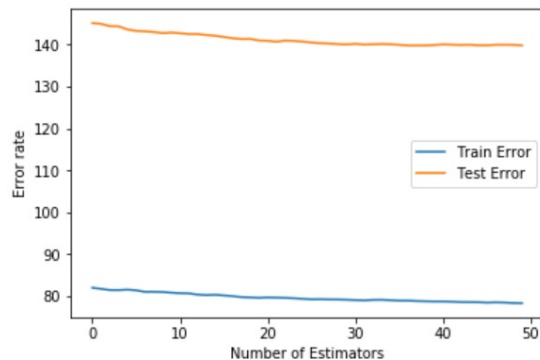
Pruning: - Trees have been pruned by specifying values for minimum number of samples and maximum depth to avoid overfitting.

Findings: - To ensure that number of estimators doesn't overfit the data, overfitting graph was plotted with number of estimators against the mean squared error.

Overfitting graph: -



Before Pruning (Min no of samples = 20)



After Pruning (Min no of samples = 2)

MODEL PERFORMANCE AND TESTING: -

Initially, decision tree technique was chosen to utilize ensembling and boosting with random forest and Gradient Boosting.

The models that have been implemented are

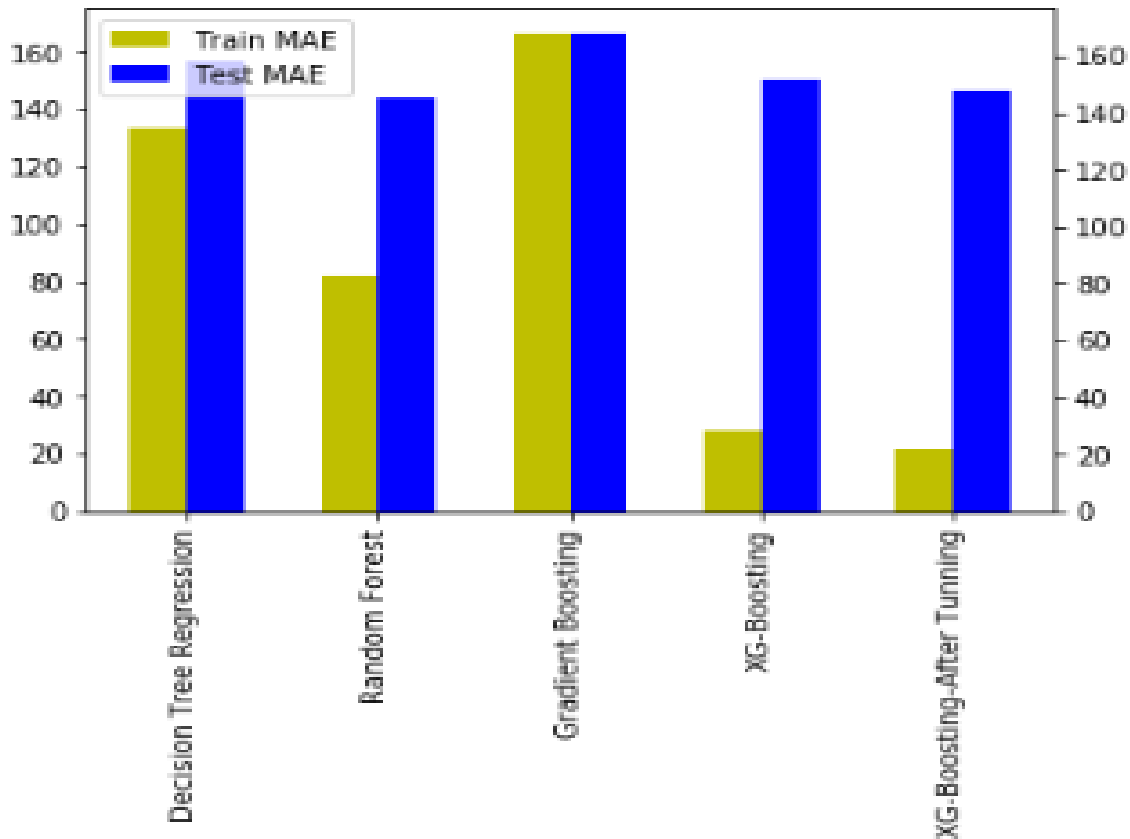
- Decision tree regressor
- Random forest
- Gradient Boosting
- XG Boost

Evaluation Approach: -

Cross-validation approach was used to evaluate all the models in order to identify how well the model will generalize on unseen data. All the models were plotted with train errors and test errors in a bar graph.

Observations: -

Although XG Boosting had less error rate in training data, it was observed that model did not generalize well for unseen data. **Whereas with “Random Forest” the error rate was less when comparted with all the other models.** Hence “Random Forest” was considered good model for this dataset.

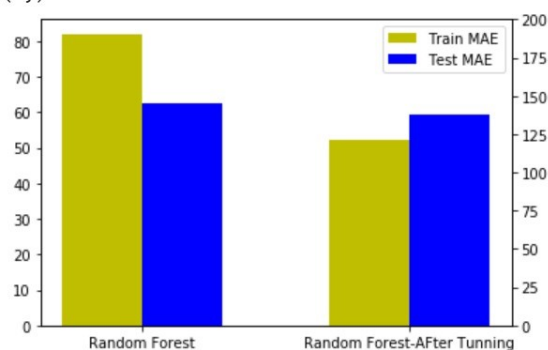


Bar graph with models plotted against the “Mean Absolute Error”

MODEL TUNING: -

Initially, hyperparameters were specified manually while training the “RandomForestRegressor” model. In order to further tune it, “**RandomizedSearchCV**” algorithm was run on random forest regressor to identify best hyperparameters.

	model	train_error	test_error
0	Random Forest	81.99630	145.15400
1	Random Forest-After Tuning	52.13073	137.97062
(2,)			



Error rate of random forest after tuning

RELIEF FEATURE SELECTION: -

Random forest regressor will automatically select the best features on its own based on the feature's significance. But in order to specify the maximum number of features to be fed for the model, "Relief feature Algorithm" can be used.

4. REFERENCES

- [1] /@rakshithvasudev. (2019, April 03). What is One Hot Encoding? Why And When do you have to use it? Retrieved from <https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f>
- [2] 136762297237701. (2018, July 21). A Guide to Pandas and Matplotlib for Data Exploration. Retrieved from <https://towardsdatascience.com/a-guide-to-pandas-and-matplotlib-for-data-exploration-56fad95f951c>
- [3] Albon, C. (2017, December 20). Hyperparameter Tuning Using Random Search. Retrieved from https://chrisalbon.com/machine_learning/model_selection/hyperparameter_tuning_using_random_search/
- [4] Drakos, G., & Drakos, G. (2018, August 26). How to select the Right Evaluation Metric for Machine Learning Models: Part 1 Regression Metrics. Retrieved from <https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regression-metrics-3606e25beae0>
- [5] Random Forest Overfitting. (n.d.). Retrieved from <https://mljar.com/blog/random-forest-overfitting/>