Restaurant Rating Prediction

DOMAIN: E COMMERCE

Komal Reddy Koukuntla

Objective :

The main goal of this project is to perform extensive Exploratory Data Analysis(EDA) on the Zomato Dataset and build an appropriate Machine Learning Model that will help various Zomato Restaurants to predict their respective Ratings based on certain features.

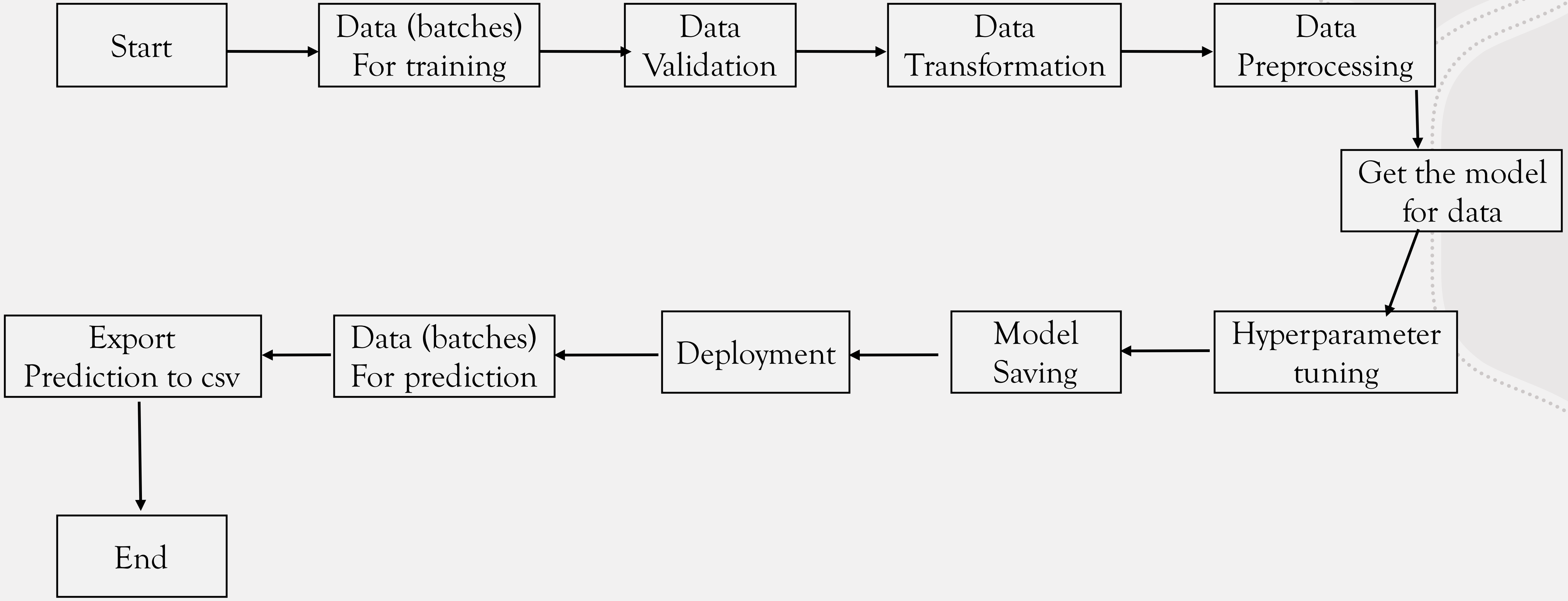
Benefits :

- Gives Better insight of Restaurant
- Predict the rating of restaurant by analyzing certain features

Data Sharing Agreement

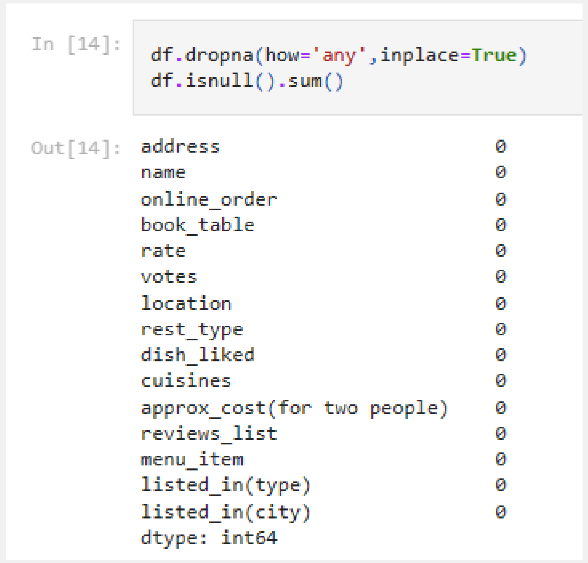
- File name (Zomato.csv)
- Number of columns 17
- Number of rows 56251
- Column Data Type : String (12), Boolean (2), Integer(1), other (2)

Architecture



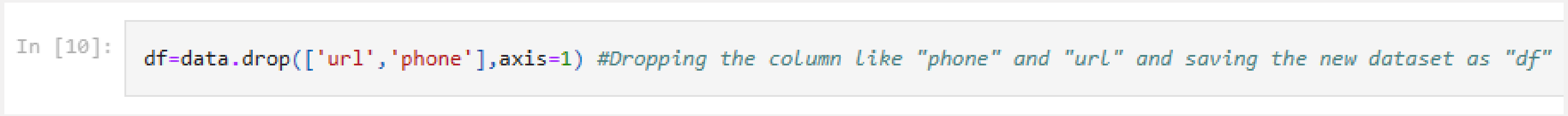
Data Validation and Data Transformation

- Removing Null values



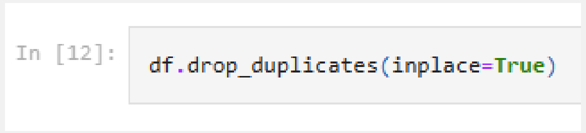
```
In [14]:  df.dropna(how='any',inplace=True)
          df.isnull().sum()

Out[14]:  address                        0
          name                           0
          online_order                   0
          book_table                     0
          rate                           0
          votes                          0
          location                       0
          rest_type                      0
          dish_liked                     0
          cuisines                       0
          approx_cost(for two people)    0
          reviews_list                   0
          menu_item                      0
          listed_in(type)                0
          listed_in(city)                0
          dtype: int64
```

- Remove unnecessary columns (features)



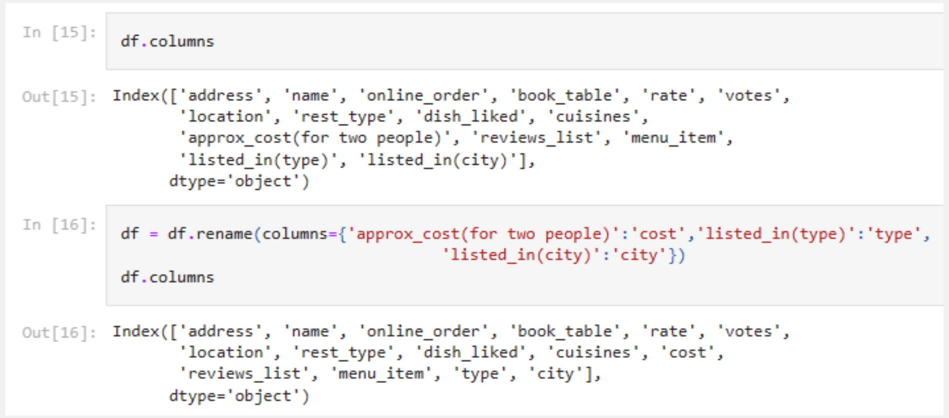
```
In [10]:  df=data.drop(['url','phone'],axis=1) #Dropping the column like "phone" and "url" and saving the new dataset as "df"
```

- Remove duplicates in dataset



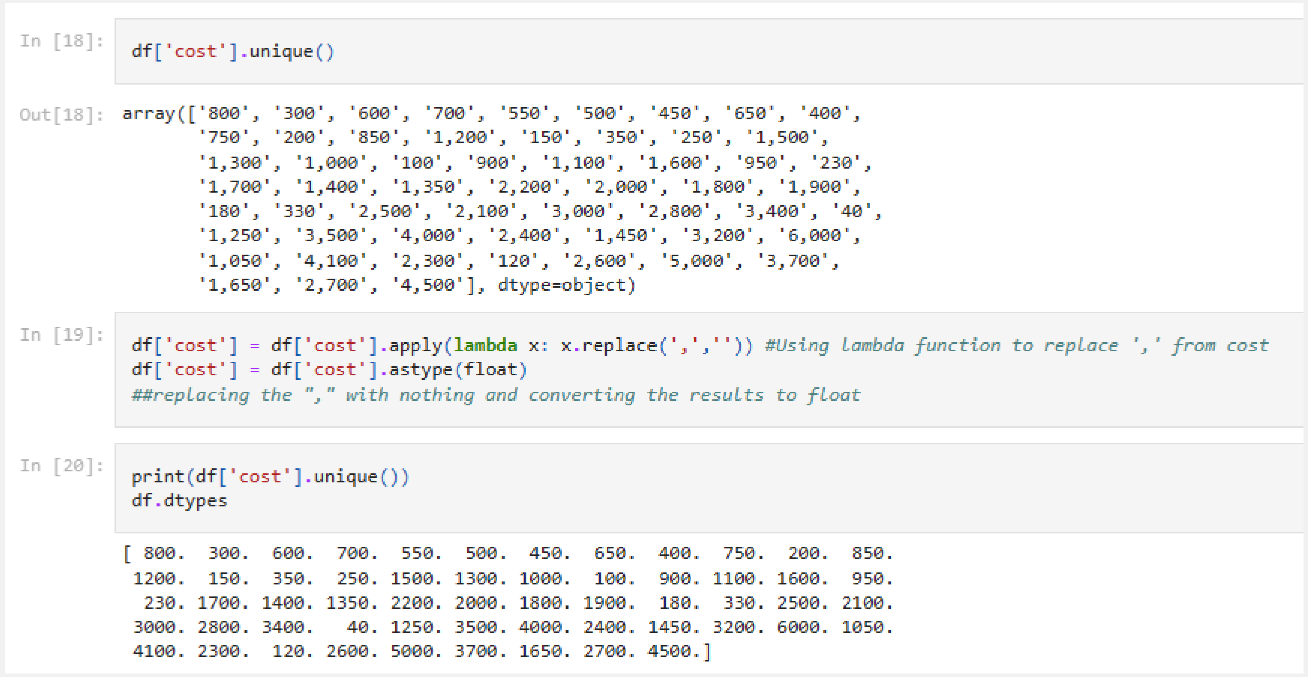
```
In [12]:  df.drop_duplicates(inplace=True)
```

- Renaming columns according to our convenience



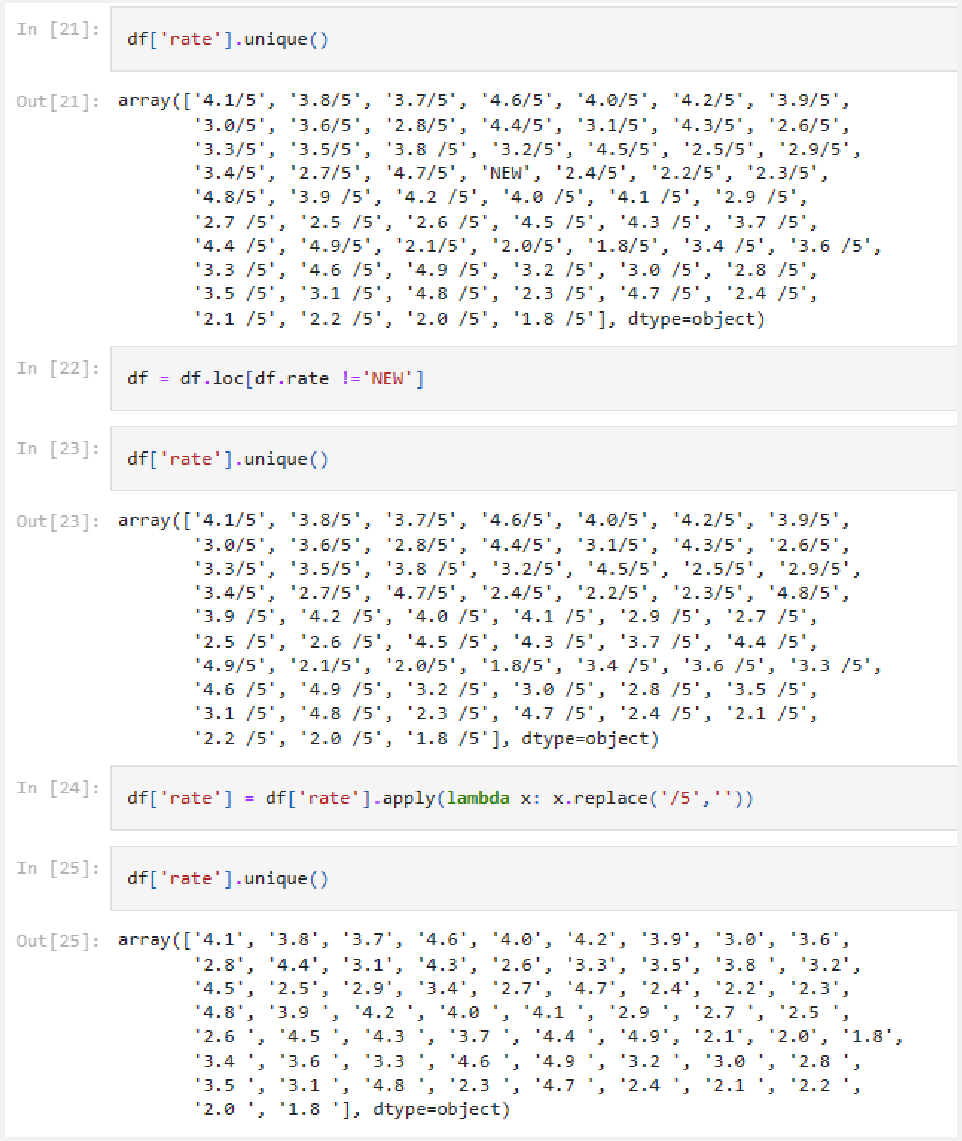
```
In [15]:  df.columns

Out[15]:  Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
                 'location', 'rest_type', 'dish_liked', 'cuisines',
                 'approx_cost(for two people)', 'reviews_list', 'menu_item',
                 'listed_in(type)', 'listed_in(city)'],
                dtype='object')

In [16]:  df = df.rename(columns={'approx_cost(for two people)':'cost','listed_in(type)':'type',
                                          'listed_in(city)':'city'})
          df.columns

Out[16]:  Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
                 'location', 'rest_type', 'dish_liked', 'cuisines', 'cost',
                 'reviews_list', 'menu_item', 'type', 'city'],
                dtype='object')
```

- Changing datatype of cost as float



```
In [18]: df['cost'].unique()

Out[18]: array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
                '750', '200', '850', '1,200', '150', '350', '250', '1,500',
                '1,300', '1,000', '100', '900', '1,100', '1,600', '950', '230',
                '1,700', '1,400', '1,350', '2,200', '2,000', '1,800', '1,900',
                '180', '330', '2,500', '2,100', '3,000', '2,800', '3,400', '40',
                '1,250', '3,500', '4,000', '2,400', '1,450', '3,200', '6,000',
                '1,050', '4,100', '2,300', '120', '2,600', '5,000', '3,700',
                '1,650', '2,700', '4,500'], dtype=object)

In [19]: df['cost'] = df['cost'].apply(lambda x: x.replace(',','')) #Using lambda function to replace ',' from cost
         df['cost'] = df['cost'].astype(float)
         ##replacing the "," with nothing and converting the results to float

In [20]: print(df['cost'].unique())
         df.dtypes

         [ 800.  300.  600.  700.  550.  500.  450.  650.  400.  750.  200.  850.
          1200.  150.  350.  250. 1500. 1300. 1000.  100.  900. 1100. 1600.  950.
           230. 1700. 1400. 1350. 2200. 2000. 1800. 1900.  180.  330. 2500. 2100.
          3000. 2800. 3400.   40. 1250. 3500. 4000. 2400. 1450. 3200. 6000. 1050.
          4100. 2300.  120. 2600. 5000. 3700. 1650. 2700. 4500.]
```

- Removing '/5' from Rates and getting rid of 'NEW' as an entry



```
In [21]:  df['rate'].unique()

Out[21]:  array(['4.1/5', '3.8/5', '3.7/5', '4.6/5', '4.0/5', '4.2/5', '3.9/5',
                 '3.0/5', '3.6/5', '2.8/5', '4.4/5', '3.1/5', '4.3/5', '2.6/5',
                 '3.3/5', '3.5/5', '3.8 /5', '3.2/5', '4.5/5', '2.5/5', '2.9/5',
                 '3.4/5', '2.7/5', '4.7/5', 'NEW', '2.4/5', '2.2/5', '2.3/5',
                 '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5', '4.1 /5', '2.9 /5',
                 '2.7 /5', '2.5 /5', '2.6 /5', '4.5 /5', '4.3 /5', '3.7 /5',
                 '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '3.4 /5', '3.6 /5',
                 '3.3 /5', '4.6 /5', '4.9 /5', '3.2 /5', '3.0 /5', '2.8 /5',
                 '3.5 /5', '3.1 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
                 '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)

In [22]:  df = df.loc[df.rate !='NEW']

In [23]:  df['rate'].unique()

Out[23]:  array(['4.1/5', '3.8/5', '3.7/5', '4.6/5', '4.0/5', '4.2/5', '3.9/5',
                 '3.0/5', '3.6/5', '2.8/5', '4.4/5', '3.1/5', '4.3/5', '2.6/5',
                 '3.3/5', '3.5/5', '3.8 /5', '3.2/5', '4.5/5', '2.5/5', '2.9/5',
                 '3.4/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5', '4.8/5',
                 '3.9 /5', '4.2 /5', '4.0 /5', '4.1 /5', '2.9 /5', '2.7 /5',
                 '2.5 /5', '2.6 /5', '4.5 /5', '4.3 /5', '3.7 /5', '4.4 /5',
                 '4.9/5', '2.1/5', '2.0/5', '1.8/5', '3.4 /5', '3.6 /5', '3.3 /5',
                 '4.6 /5', '4.9 /5', '3.2 /5', '3.0 /5', '2.8 /5', '3.5 /5',
                 '3.1 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5', '2.1 /5',
                 '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)

In [24]:  df['rate'] = df['rate'].apply(lambda x: x.replace('/5',''))

In [25]:  df['rate'].unique()

Out[25]:  array(['4.1', '3.8', '3.7', '4.6', '4.0', '4.2', '3.9', '3.0', '3.6',
                 '2.8', '4.4', '3.1', '4.3', '2.6', '3.3', '3.5', '3.8 ', '3.2',
                 '4.5', '2.5', '2.9', '3.4', '2.7', '4.7', '2.4', '2.2', '2.3',
                 '4.8', '3.9 ', '4.2 ', '4.0 ', '4.1 ', '2.9 ', '2.7 ', '2.5 ',
                 '2.6 ', '4.5 ', '4.3 ', '3.7 ', '4.4 ', '4.9', '2.1', '2.0', '1.8',
                 '3.4 ', '3.6 ', '3.3 ', '4.6 ', '4.9 ', '3.2 ', '3.0 ', '2.8 ',
                 '3.5 ', '3.1 ', '4.8 ', '2.3 ', '4.7 ', '2.4 ', '2.1 ', '2.2 ',
                 '2.0 ', '1.8 '], dtype=object)
```

Model Training

Data preprocessing:

- Performing EDA to get insight of data like identifying distribution , outliers, trend among data (most famous restaurants, top liked foods) etc.
- Check for null values in the columns. If present impute the null values.
- Encode the categorical values with numeric values.

Model Selection:

- Built models using Linear Regression, Random Forest, ExtraTree Regressor
- ExtraTree Regressor is chosen as its r2 score is high (0.9325151755043354)

Prediction

- Model is trained in sets (Train_Test_Split) 30% of data for testing whereas 70% is for training
- The sets are preprocessed and data is predicted using ExtraTree Regressor model
- The predicted data is saved in Zomata_df_komal.csv

Q & A

- What’s the source of data ?

The data for training is provided by the client in multiple batches and each batch contain multiple files.

- What was the type of data ?

The data was the combination of numerical and Categorical values

- What’s the complete flow you followed in this Project ?

Refer Slide no 4 for better Understanding.

- After the File validation what you do with incompatible file or files which didn't pass the validation ?

Files like these are moved to the Achieve Folder and a list of these files has been shared with the client and we removed the bad data folder.

- How logs are managed ?

We are using different logs as per the steps that we follow in validation and modeling like File validation log, Data Insertion, Model Training log, prediction log etc

- What techniques were you using for data pre-processing ?

-> Removing unwanted attributes.

-> Visualizing relation of independent variables with each other and output variables.

-> Checking and changing Distribution of continuous values.

-> Removing outliers

-> Cleaning data and imputing if null values are present.

-> Converting categorical data into numeric values.

- How training was done or what models were used ?

Model is trained in sets (Train_Test_Split) 30% of data for testing whereas 70% is for training

The sets are preprocessed and data is predicted
using ExtraTree Regressor model

The predicted data is saved in Zomata_df_komal.csv

- How Prediction was done ?

The testing files are shared by the client. We Performed the same life cycle on the provided dataset. Then, on the basis of dataset, model is loaded and prediction is performed. In the end we get the accumulated data of predictions.

- What are the different stages of deployment?

First, the scripts are stored on GitHub as a storage interface.

The model is first tested in the local environment, and hosted locally .