ZOMATO BANGALORE RESTAURANTS INSIGHTS AND PREDICTIVE MODELING FOR RATING

25th July 2020

OVERVIEW:

We live in an amazing age where we have numerous delivery outlets all around us to choose from a wide range of lip smacking cuisines from all around the world. If you are in the mood to gobble famous Mexican Nachos, just place an order from an app and in a regular time span, your food is rolled-up at your doorstep. Zomato is one such food delivery startup which helps us do this. Zomato has a tie up with most of the restaurants around the world and has rich data of these restaurants which could give us great insights.We will keep our data restricted to Bangalore restaurants.

BUSINESS PROBLEM TO SOLVE/GOALS:

- Ratings and reviews play a very important role in attracting new and retaining customers.
- Our target would be improving ratings based on the insights and based on these factors predict ratings using predictive modelling for a prospect restaurant.

- Understand what people like the most in a highly rated restaurant, in a particular locality, which are related to ratings for a prospect restaurant.
- Have an insight of approx_cost (cost for two), which is based on many factors like neighborhood, restaurant type that can be related to ratings.
- Given a locality, a prospect restaurant can have an insight of the factors to get the best rating.
- Marketing strategies like personalized notifications, discounts etc. can be set up to attract an audience.

WHO CARES ABOUT THE PROBLEM WE TRYING TO FIX ?

- Potential clients would be existing restaurant owners and prospects restaurants.
- Having insights on such factors could help the decision makers take actions which would eventually increase the ratings and audience.

DATA COLLECTION AND WRANGLING:

The data is scraped from Zomato(https://www.zomato.com/bangalore) using the Python package 'Beautiful soup' as of Jan 2020.

Ref script: Web scraping script (Zomato)

- Most of the data is cleaned/formatted while scraping.
- Some columns are manipulated to tuples.
- Opening and closing timings are transformed to datetime formats.
- Missing values are transformed to np.NAN
- Duplicates rows, if any, are removed based on the restaurant_id.
- No outliers.

Data csv: [data csv]

EXPLORATORY DATA ANALYSIS SUMMARY:

Ref script :[Data cleaning and wrangling script]

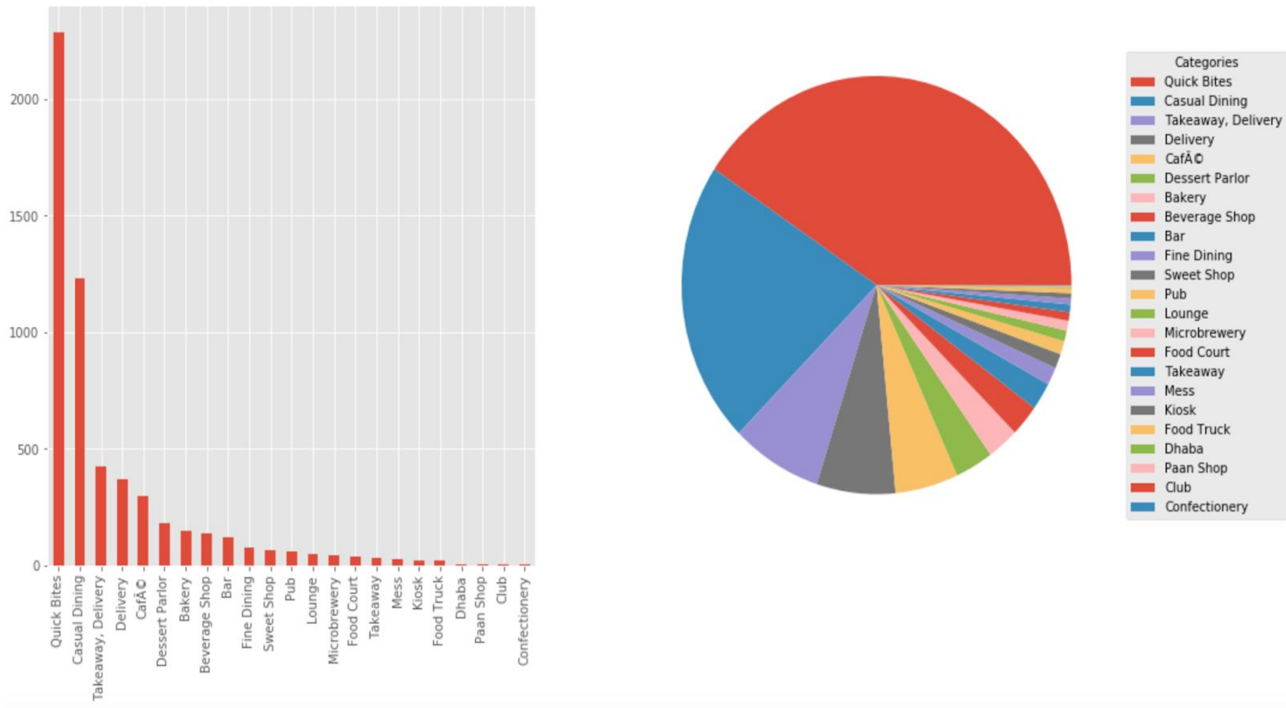
- This dataset is relatively small, having information of around 6k restaurants.
- Let's have a quick glance of the data first and start exploring,

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6147 entries, 0 to 6146
Data columns (total 23 columns):
restaurant_link          6147 non-null object
restaurant_ID            6147 non-null int64
restaurant_name          6147 non-null object
locality                 6147 non-null object
restaurant_category      6142 non-null object
zomato_gold              347 non-null object
discounts                738 non-null object
photos_taken             6147 non-null int64
rating                   5328 non-null float64
votes                    5310 non-null float64
cuisines                 6147 non-null object
approx_cost_for_2        6147 non-null int64
opening_timings          6143 non-null object
address                  6041 non-null object
latitude                 5682 non-null float64
longitude                5682 non-null float64
more_info                6147 non-null object
featured_in              761 non-null object
known_for                290 non-null object
most_liked_Food          2607 non-null object
most_liked_Service       1382 non-null object
most_liked_Look & Feel   1071 non-null object
reviews                  6147 non-null object
dtypes: float64(4), int64(3), object(16)
memory usage: 1.1+ MB
```

	restaurant_link	restaurant_ID	restaurant_name	locality	restaurant_category	zomato_gold	discounts	photos_taken	rating	vote
0	https://www.zomato.com/bangalore/abs-absolute-...	56618	AB's - Absolute Barbecues	Marathahalli	Casual Dining	zomato gold	NaN	4665	4.8	14700.
1	https://www.zomato.com/bangalore/uru-brewpark-...	19122613	URU Brewpark	JP Nagar	Microbrewery	NaN	NaN	776	4.3	1421.
2	https://www.zomato.com/bangalore/the-big-barbe...	19203051	The Big Barbeque	Marathahalli	Casual Dining	NaN	NaN	609	4.7	1744.

- ***Different Restaurant categories***



Quick bites and Casual dining are the most common of all restaurant categories.

- ***Cuisines***

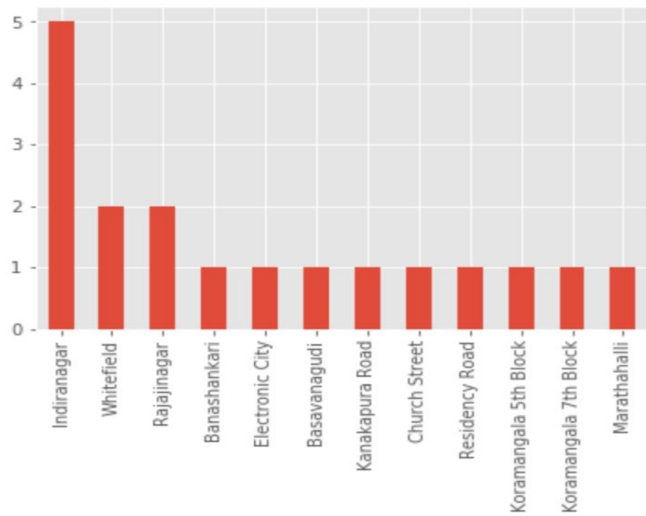
North Indian and Chinese are the most popular whereas Belgium, Portuguese are some of the rare cuisines.

- ***Average Restaurant rating in a locality:***

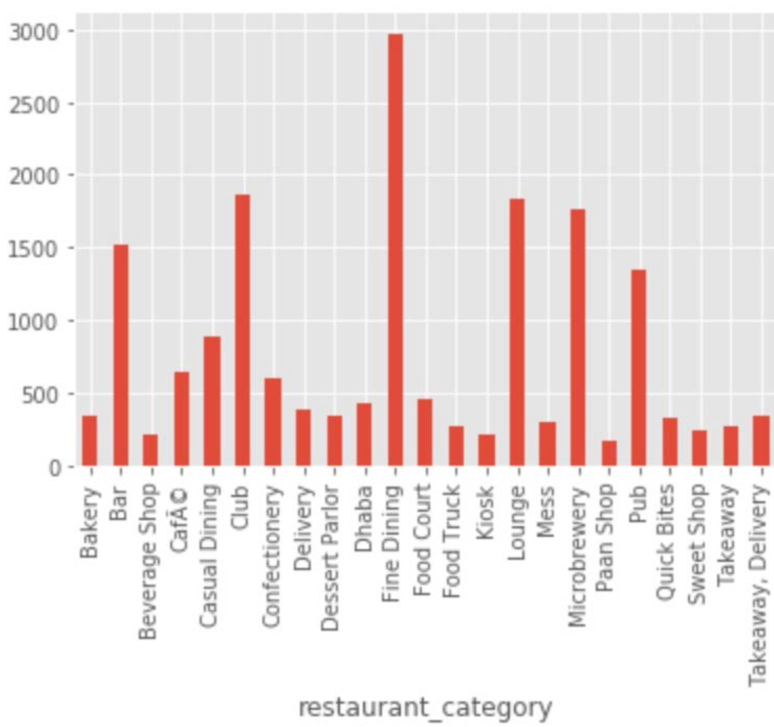
Sankey road, Lavelle road and Church street have highest average ratings.There is definitely locality playing a part in Restaurant rating.

- ***Highest number of newly opened restaurants in a locality***

Indiranagar has the highest number of ‘Newly opened” restaurants.

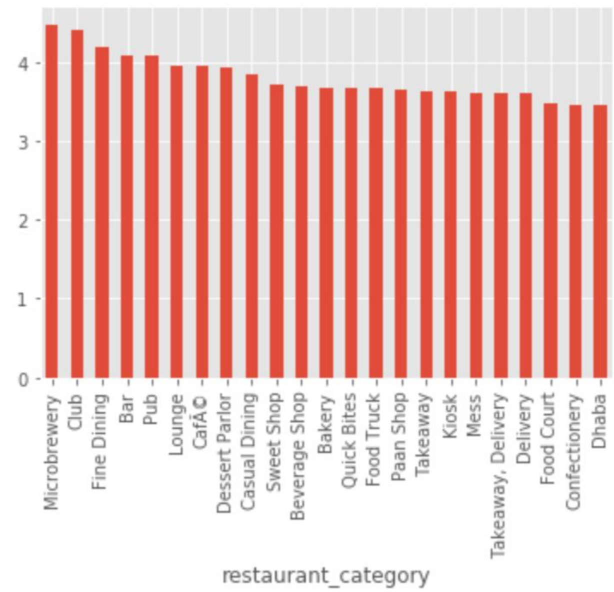


- ***Restaurant category having highest 'Cost for 2'***



Fine Dining being the highest, followed by Club and Lounge have the highest 'Cost for 2'.

- ***Highest and lowest rated restaurant categories :***

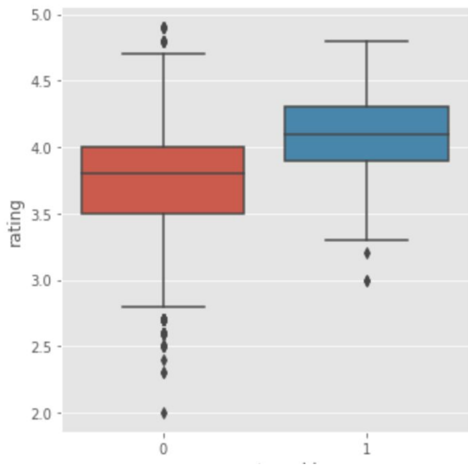


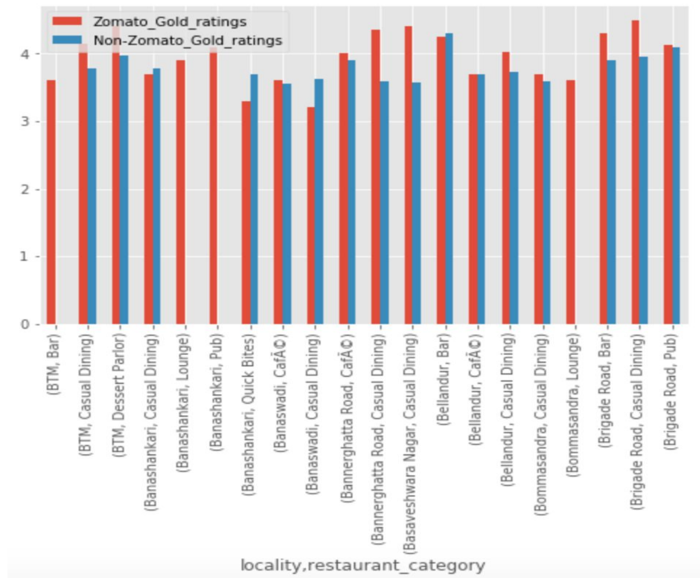
Microbrewery and Club are highest average ratings whereas Confectionery and Dhaba have the lowest average rating.

- ***The following figure summarises the Most liked features in the top rated restaurant categories(microbrewery and club respectively) :***

Staff, Chicken, Friendly, Courteous, Beer are some important features for these categories.

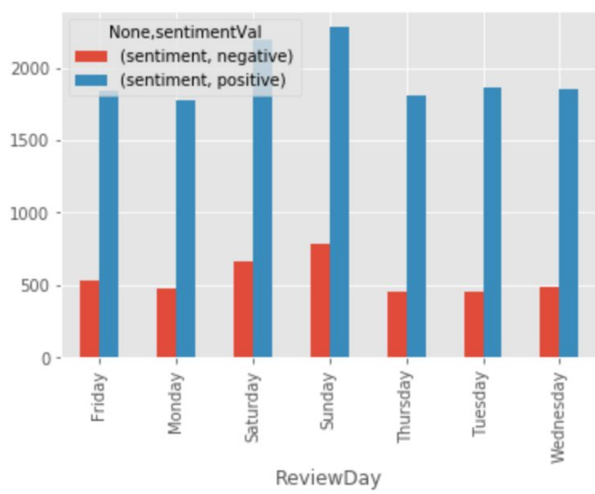
- ***Average rating of a Zomato Gold and a Non Zomato Gold restaurant in a locality:***





Majority of the times Zomato gold restaurants have higher ratings compared to a Non Zomato restaurant.

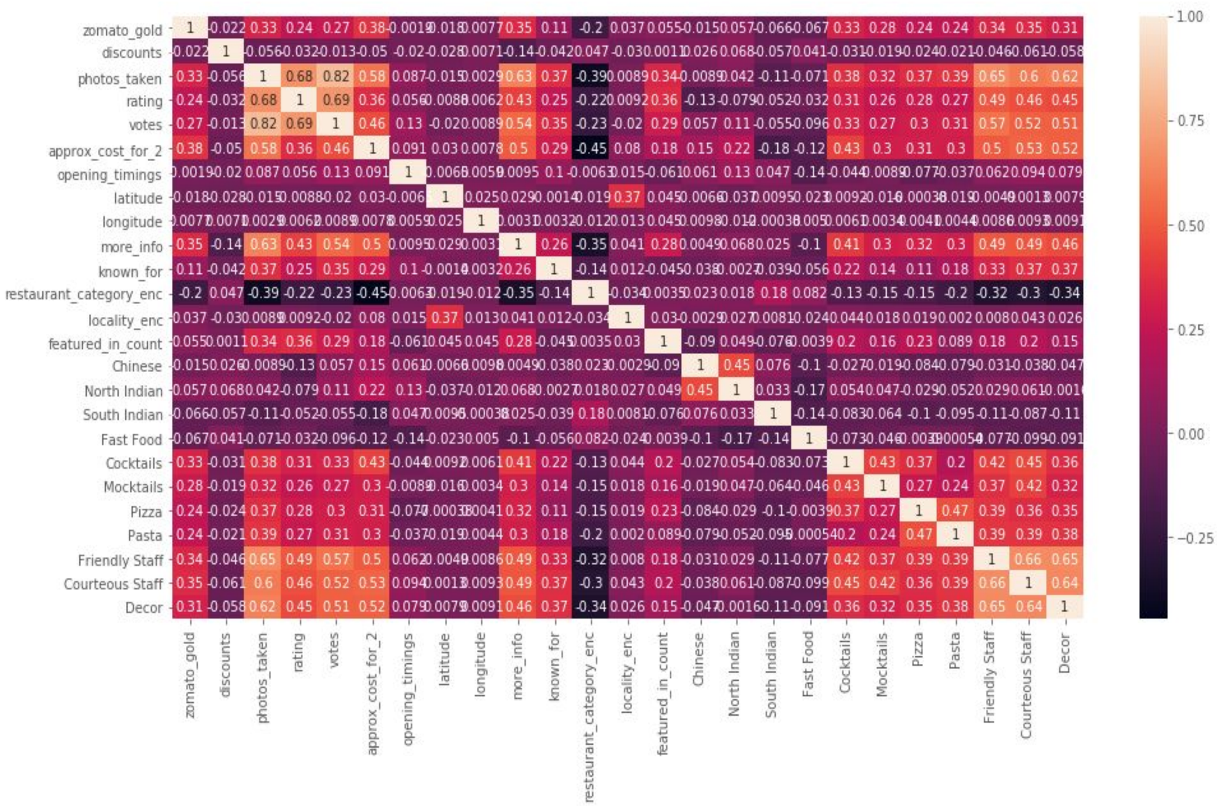
- ***Sentiments over a week:***



Looking at sentiments from latest user reviews, we see relatively high positive sentiments, also on weekends have high sentiment responses.

EXPLORING CORRELATIONS AND TRENDS:

To explore correlation and trends between variables we use the pandas correlation and seaborn pairplot visual representations.



- *Photos_taken* are positively correlated with *ratings* and *votes*, and negatively correlated with *approx_cost_for_2*.
- Features like *photos_taken*, *votes*, *approx_cost_for_2*, *featured_in_count*, *zomato_gold* tend to have positive correlation with restaurant *rating*.
- *Discounts* have a negative correlation with *ratings*.
- Few localities have high *approx_cost_for_2*.
- From the above restaurant_category distribution,we see that a very few categories are very popular.
- Few certain restaurant categories have high approx_cost_for_2.
- From the distribution of *zomato_gold*, we see that relatively few restaurants have zomato_gold.
- From the distribution of discounts, we see that relatively few restaurants have discounts.

- Localities seem not correlated with *rating*.
- *Featured_in_count* looks positively correlated with *rating*.

FEATURE ENGINEERING:

Keeping in mind getting the best from model learnings, some feature engineering is required on the data.These are some of the techniques performed.

Imputation:

- Missing values replaced with NaN.
- *Restaurant_id* considered as a unique identifier, duplicates removed.
- In some cases, Numeric imputation i.e replacing NaN with a number say '0' made more sense like in case Zomato_gold, etc.
- Some *Restaurant_categories* were very low in number, Categorical imputation i.e replacing such categories with 'Others' made more sense.

Log Transform:

- Log transformation helps to handle highly skewed data.On transforming the distribution is approximated to a normal.
- Log transformation helps to decrease the effect of outliers.
- Referring to the pairplots, we can see the *photos_taken*, *votes*, *approx_cost_for_2* are highly right skewed which should be log transformed.

Encoding:

Encoding converts non-numeric categorical data to numeric data, so as to convert it into a machine-readable form .

- *Label Encoding:* Encodes categories to uniques numbers.Locality and Restaurant_categories are Label Encoded.

- *MultilabelBinarizer:* Encodes in a binary matrix indicating presence of class.We used this to encode *cuisines,* then expanded this to new features having only most prominent cuisines, most liked features.

mato_gold	discounts	photos_taken	rating	votes	cuisines	...	North Indian	South Indian	Fast Food	Cocktails	Mocktails	Pizza	Pasta	Friendly Staff	Courteous Staff	Decor
0	0	0.693147	3.2	2.079442	[North Indian]	...	1	0	0	0	0	0	0	0	0	0
0	0	4.521789	4.2	4.779123	[Cafe, Italian, Continental]	...	0	0	0	0	0	0	0	1	1	1
0	0	1.386294	3.5	2.197225	[South Indian]	...	0	1	0	0	0	0	0	0	0	0

Here, we have expanded to new features like ‘North indian’, ‘Cocktails’ ,etc which are most prominent.

- *Target Encoding:* In this encoding technique, we map or encode the variable, to mean of the target variable. Here, we generally split the data first, encode the variable in the train data then map respectively on the test data.Any unseen category in the test data can be filled with mean target variable. In this dataset, we have encoded *Restaurant_category* using target encoding.

Statistical inferences:

Ref script: Statistical inference script

We have performed some statistical tests using Bootstrap techniques to verify or have confidence of certain observations seen in Exploratory data analysis.

1. Null hypothesis test resulted in positive correlation between approx_cost_for_2 and rating.
2. Null hypothesis test resulted in Zomato Gold have high ratings compared to Non Zomato Gold restaurants.

Now that we have a cleaner data set we can move to predictive modelling of the rating ,

Predictive Modelling for Restaurant rating:

Ref script: Predictive modeling script

Referring to the pair plots and correlation matrix, we have selected features which show certain relation with the target variable and we split the data into train and test.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)
```

```
X_test = X_test[['restaurant_category_enc_mean_target','photos_taken','votes','approx_cost_for_2',
                 'featured_in_count','zomato_gold','discounts',
                 'known_for',
                 'more_info',
                 'Cocktails','Mocktails','Pizza','Pasta',
                 'Friendly Staff', 'Courteous Staff','Decor']]

X_train = X_train[['restaurant_category_enc_mean_target','photos_taken','votes','approx_cost_for_2',
                   'featured_in_count','zomato_gold','discounts',
                   'known_for',
                   'more_info',
                   'Cocktails','Mocktails','Pizza','Pasta',
                   'Friendly Staff', 'Courteous Staff','Decor']]
```

Linear Regression:

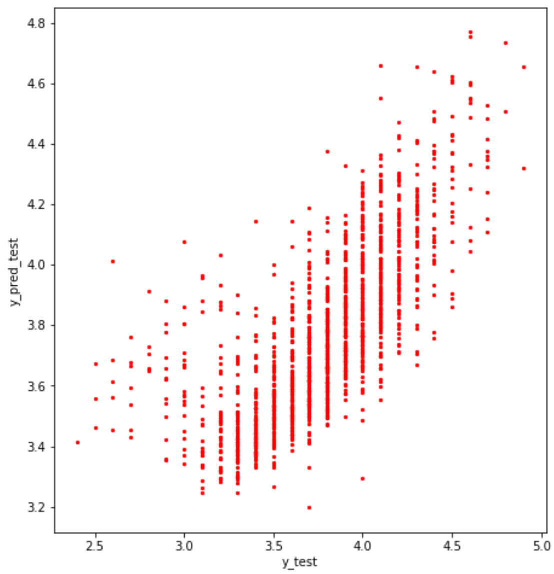
Starting with linear regression to check for any linear relationships and create baseline model,

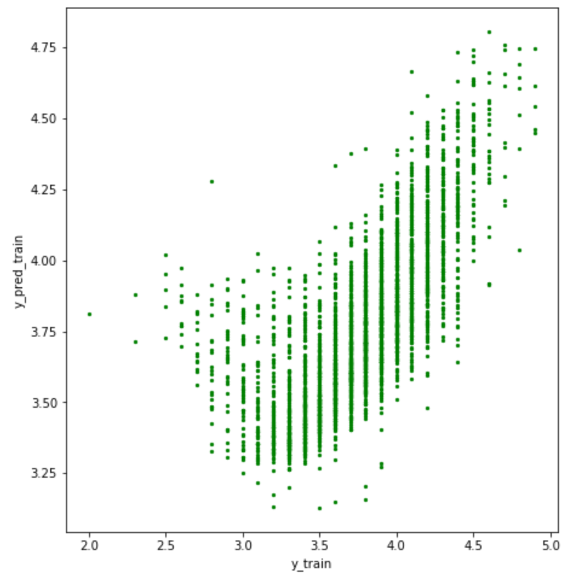
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Create the regressor: reg_all
reg_all = LinearRegression()

# # Fit the regressor to the training data
reg_all.fit(X_train, y_train)

# # Predict on the test data: y_pred
y_pred_test = reg_all.predict(X_test)

from sklearn.metrics import r2_score
print('Test accuracy',r2_score(y_test,y_pred_test))

y_pred_train = reg_all.predict(X_train)

from sklearn.metrics import r2_score
print('Train accuracy',r2_score(y_train,y_pred_train))
```

```
Test accuracy 0.5650701106273819
Train accuracv 0.5168134628097882
```

Linear regression a test accuracy of **0.565** and train accuracy of **0.516**

Further plotting prediction values against actual values,





Further, lets try some decision trees models,

Random Forest Regressor:

```
from sklearn.ensemble import RandomForestRegressor

RForest = RandomForestRegressor(n_estimators=100,random_state=329,min_samples_leaf=.0001)

RForest.fit(X_train,y_train)

y_pred_test = RForest.predict(X_test)

from sklearn.metrics import r2_score
print('Test accuracy',r2_score(y_test,y_pred_test))

y_pred_train = RForest.predict(X_train)

from sklearn.metrics import r2_score
print('Train accuracy',r2_score(y_train,y_pred_train))
```

```
Test accuracy 0.5476368081176365
Train accuracy 0.9319015725052022
```

Random forest gives a test accuracy of **0.547** and train accuracy of **0.931** which indicates an overfitting.

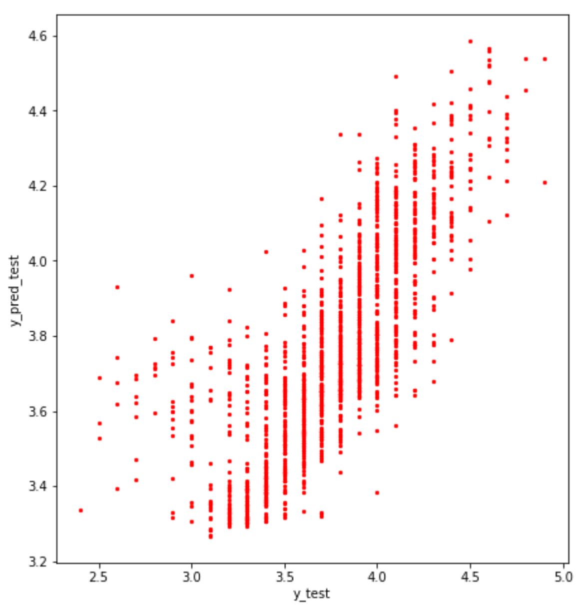
Further we regularise and choose the best hyperparameters using *RandomizedSearchCV.*

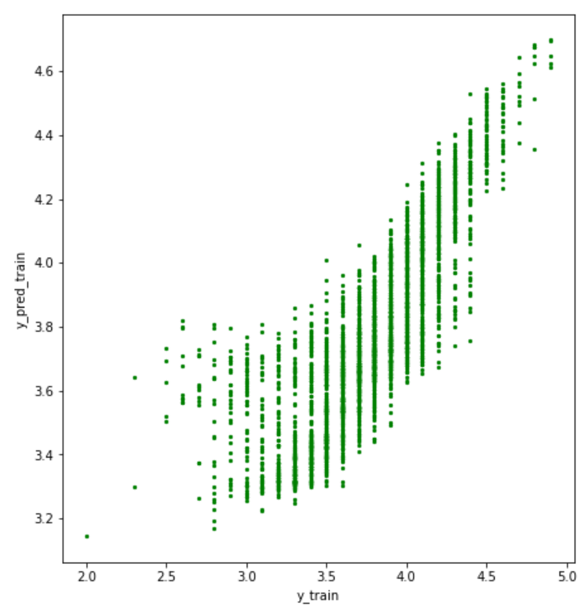
From the best estimator with the best parameters from *RandomizedSearchCV*, we predict,

```
best_random.fit(X_train,y_train)

y_pred_test = best_random.predict(X_test)

from sklearn.metrics import r2_score
print('Test accuracy',r2_score(y_test,y_pred_test))

y_pred_train = best_random.predict(X_train)

from sklearn.metrics import r2_score
print('Train accuracy',r2_score(y_train,y_pred_train))
```

```
Test accuracy 0.6023152073595284
Train accuracy 0.6858187175578061
```

So, we have slightly better test accuracy which is **0.602** and train accuracy of **0.685.**





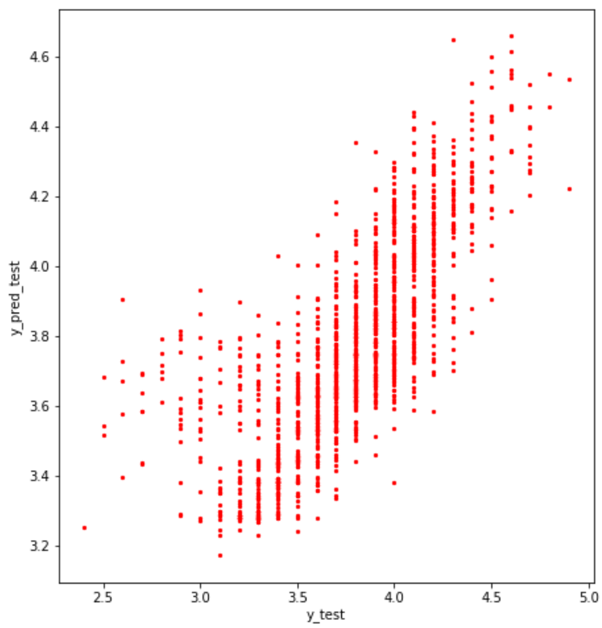
XG Boost:

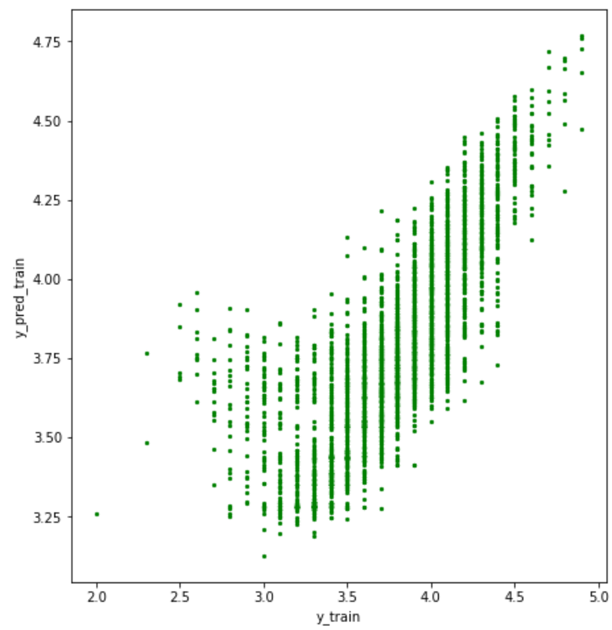
```python
import xgboost

xgb = xgboost.XGBRegressor()

xgb.fit(X_train,y_train)

y_pred_test = xgb.predict(X_test)

from sklearn.metrics import r2_score
print('Test accuracy',r2_score(y_test,y_pred_test))

y_pred_train = xgb.predict(X_train)

from sklearn.metrics import r2_score
print('Train accuracy',r2_score(y_train,y_pred_train))
```

```
/Users/Anand/anaconda3/lib/python3.7/site-packages/xgboost/core.py:587: FutureWarning:
will be removed in a future version
  if getattr(data, 'base', None) is not None and \
```

```
[12:21:37] WARNING: src/objective/regression_obj.cu:152: reg:linear is now deprecated
Test accuracy 0.6013679785025707
Train accuracy 0.6214343741282311
```

XG Boost gives a test accuracy of **0.601** and train accuracy of **0.621.**





Hyperparameter tuning with *RandomizedSearchCV* gives following results,

```
param_xg = {   'learning_rate':[0.05,0.1,0.15,0.2,0.25,0.3],
                'max_depth':[3,4,5,6,8,10,12,15],
                'min_child_weight':[1,3,5,7],
                'reg_lambda':[0,0.1,0.2,0.3,0.4]
           }
```

```
import xgboost
from sklearn.model_selection import RandomizedSearchCV

xg_random = RandomizedSearchCV(estimator = xgb, param_distributions = param_xg, n_iter = 100, cv = 5, verbose=2, rando

xg_random.fit(X_train,y_train)
```

```
xgb.fit(X_train,y_train)

y_pred_test = xgb.predict(X_test)

from sklearn.metrics import r2_score
print('Test accuracy',r2_score(y_test,y_pred_test))

y_pred_train = xgb.predict(X_train)

from sklearn.metrics import r2_score
print('Train accuracy',r2_score(y_train,y_pred_train))
```

```
[12:24:02] WARNING: src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor c
Test accuracy 0.6024138407402954
Train accuracy 0.6184483009683994
```

XG boost gives a test accuracy of **0.602** after hyperparameter tuning.

Test Accuracy from Predictive modeling:

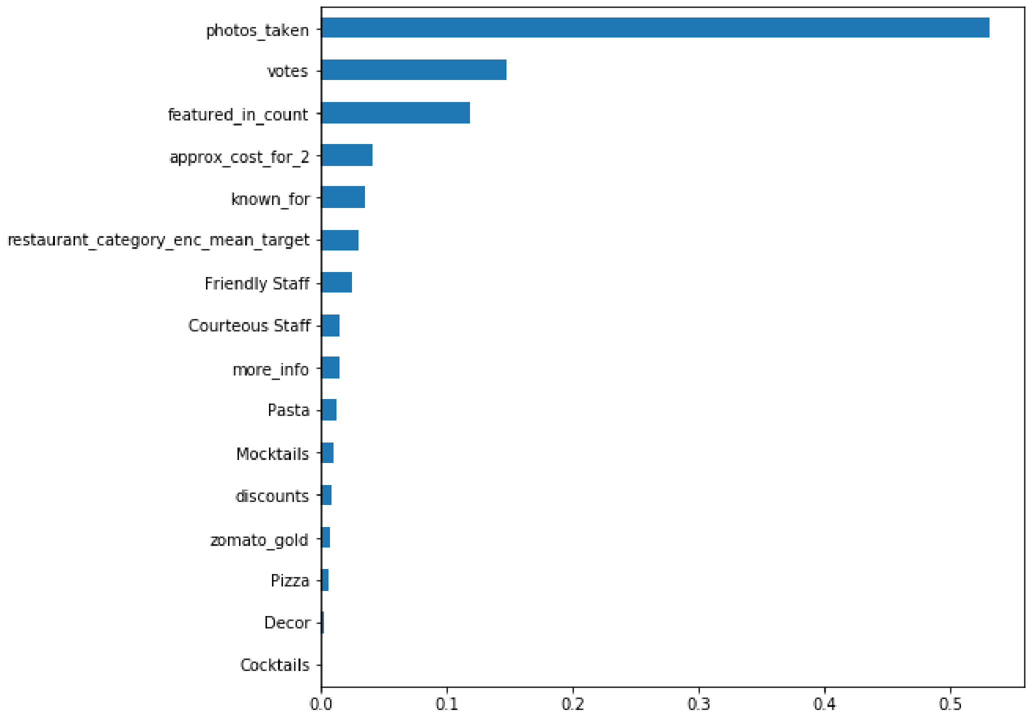
So far ***RandomForestRegressor*** and ***XGBoost*** have a relatively higher test accuracy of ***0.60*** post hyperparameter tuning.

Feature importance:

Let's see the important features of the XG boost model,

```
xgb.feature_importances_
```

```
array([0.02962964, 0.5313309 , 0.1476723 , 0.04040176, 0.11814345,
       0.00669769, 0.00831953, 0.03495465, 0.01436714, 0.        ,
       0.01008315, 0.0054883 , 0.01200172, 0.02461362, 0.01450068,
       0.00179543], dtype=float32)
```



From the above plot of feature importance, we see that features like ***photos_taken, votes, featured_in_count*** and ***approx_cost_for_2*** are significant features for predicting "***Rating***" of a restaurant.

Insights from low model accuracy:

From the above analysis, we summarise that there is a good amount of noise in the data, also we need to include more features/factors and incorporate more data for better learning of the model.

Future Development: As a part of future development and refinements,

- Update the web scraping to incorporate more data.
- Explore and incorporate more features like commercial rates etc.
- Explore packages like HYPEROPT for hyperparameter tuning.