**ZOMATO RESTAURANT PREDICTION**

*Project Description-*

Zomato Data Analysis is one of the most useful analysis for foodies who want to taste the best cuisines of every part of the world which lies in their budget. This analysis is also for those who want to find the value for money restaurants in various parts of the country for the cuisines.

Additionally, this analysis caters the needs of people who are striving to get the best cuisine of the country and which locality of that country serves that cuisines with maximum number of restaurants.

The collected data has been stored in the Comma Separated Value file **Zomato.csv**. Each

restaurant in the dataset is uniquely identified by its Restaurant Id. Every Restaurant contains the following variables:

• Restaurant Id: Unique id of every restaurant across various cities of the world

• Restaurant Name: Name of the restaurant

• Country Code: Country in which restaurant is located

• City: City in which restaurant is located

• Address: Address of the restaurant

• Locality: Location in the city

• Locality Verbose: Detailed description of the locality

• Longitude: Longitude coordinate of the restaurant&#39;s location

• Latitude: Latitude coordinate of the restaurant&#39;s location

• Cuisines: Cuisines offered by the restaurant

• Average Cost for two: Cost for two people in different currencies

• Currency: Currency of the country

• Has Table booking: yes/no

• Has Online delivery: yes/ no

• Is delivering: yes/ no

• Switch to order menu: yes/no

• Price range: range of price of food

• Aggregate Rating: Average rating out of 5

• Rating color: depending upon the average rating color

• Rating text: text on the basis of rating of rating

• Votes: Number of ratings casted by people

***Problem statement : In this dataset we are making two predictions –***

1. Average Cost for two- Build a predictive regression model to predict the average cost of 2 customers dining/ordering.
2. Price range- Build a predictive classification model to predict the range

*Source of data-*

* <https://github.com/dsrscientist/dataset4/blob/main/Country-Code.xlsx>
* <https://github.com/dsrscientist/dataset4/blob/main/zomato.csv>

From all the Data available, we can bring out some neat insights or conclusions such as-

* What is the Average cost for 2 persons?
* How many types of Restaurant types are there?
* Does location affect the price/cost?
* How many have a book table facility?
* How many have online delivery service?
* How rating affects the price?
* What is the most liked Restaurant type?

**Tools used:-**

* Numpy
* Pandas
* Matplotlib
* Seaborn
* Machine learning techniques
* Sklearn

*DATA ANALYSIS-*

The data has been imported and converted into data frame for analysis purpose.

The dataset contains 9551 entries/records of Data and 22 columns/features such as

‘Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',

'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',

'Average Cost for two', 'Currency', 'Has Table booking', 'Has Online delivery', 'Is delivering now', 'Switch to order menu', 'Price range',

'Aggregate rating', 'Rating colour', 'Rating text', 'Votes', 'Country’.

We have used dataframe.info() method to check the nulls and datatype for each variable.

Most of the columns in the dataset have object type data.

Columns like Restaurant Name, City, Address, Locality, Locality Verbose, Cuisines, Currency, Has Online delivery, Is delivering now, Switch to order menu, Rating colour and Country have object type data and remaining columns have integer and float data type values.

There are no null values in any of the columns.

*EXPLORATORY DATA ANALYSIS-*

Exploratory Data Analysis (EDA) refers to the method of studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables. EDA is normally carried out as a preliminary step before undertaking extra formal statistical analyses or modelling.

Let’s analyse the data visually to draw insights.

***Data Visualization:***

EDA employs visual techniques to represent the statistics graphically. Visualizations consisting of histograms, box plots, scatter plots, line plots, heatmaps, and bar charts assist in identifying styles, trends, and relationships within the facts.

We are using data visualization to find patterns and gain further insights about the features/variables.

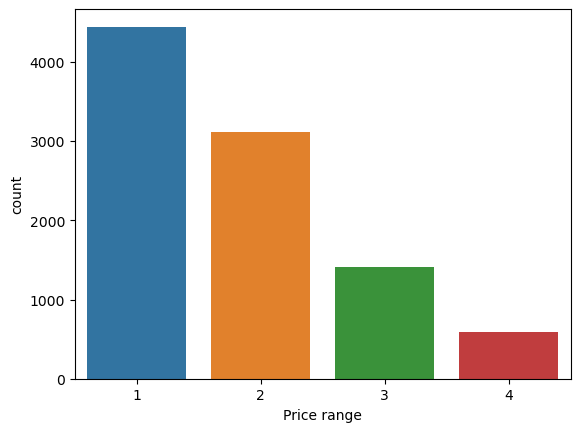
***Price range-***

This variable has 4 unique values.

*#Checking distribution price range-*

sns.countplot(x='Price range', data=data)

plt.show()

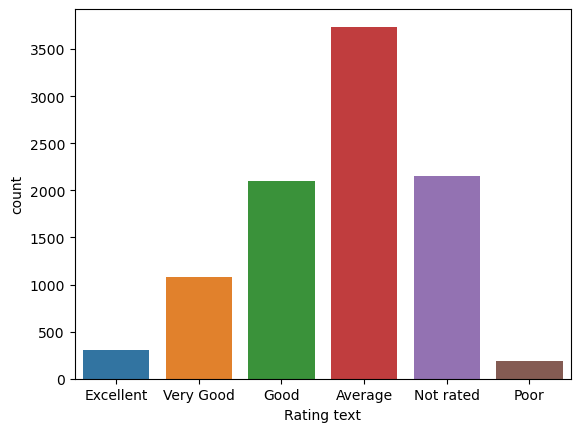


Price Range 1 has the highest count, followed by Price range 2, Price rang 3 and the lowest count is for price range 4. The distribution seems a bit imbalanced which may have to be treated at a later stage.

***Rating text –***

When customers like the food/cuisine in a restaurant, they leave text rating as a feedback. Rating text has 6 unique values- 'Excellent', 'Very Good', 'Good', 'Average', 'Not rated', 'Poor'.

The distribution of the customers leaving specific rating text-

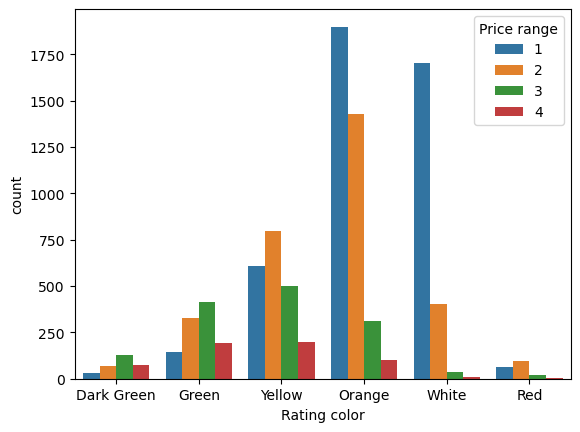


Most customers have rated the restaurants/food in the restaurant as 'Average'. A high number of customers have not rated at all. 'Poor' is the least in number in terms of rating.

***Rating colour-***

Often specific colours are used for rating to describe satisfaction or dissatisfaction of customers. There are 6 unique values in Rating colour.

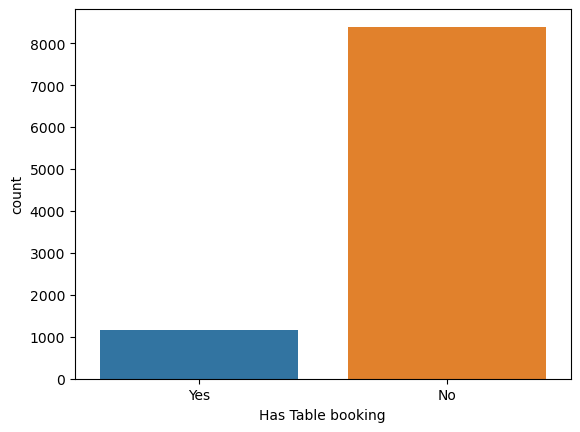
The distribution of the customers rating specific in specific colours-



It is evident that most customers have used Orange, yellow and white colour to rate.

***Has Table booking-***

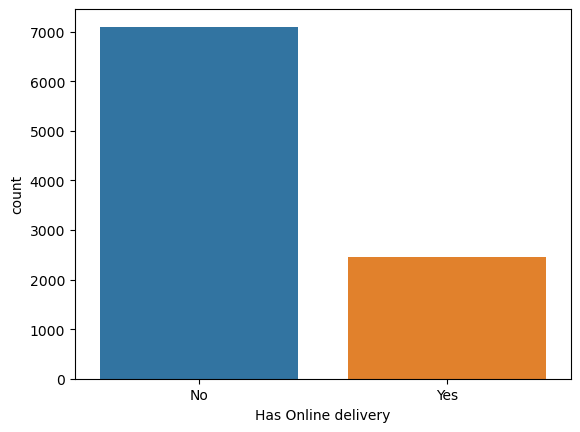
This variable explains if restaurants provide option to book tables in advance. It has 2 unique values.



We can see that most restaurants offer dining on arrival and does not give table pre-booking option.

***Has Online delivery-***

This variable explains if restaurants provide option to order food online. It has 2 unique values.

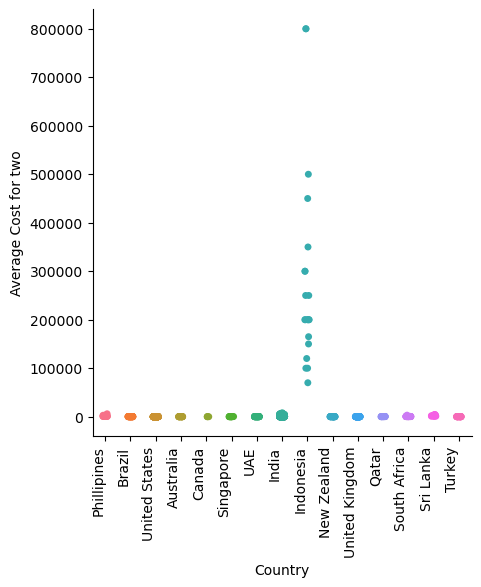


We can conclude that most restaurants provide food service on dining only and are not open to online delivery option.

***Average Cost for two-***

This is a continuous variable. For customers to decide on the price range for two that they can choose among restaurants, most restaurants share this detail on their page. In this dataset, there are 140 unique values in Average Cost for two.

Let’s check how average cost for two varies country wise-



It is evident that average cost of two is highest in Indonesia.

***Feature Engineering:***

 EDA allows for the exploration of various variables and their adjustments to create new functions or derive meaningful insights. Feature engineering can contain scaling, normalization, binning, encoding express variables, and creating interplay or derived variables.

We have seen in the initial phase of the analysis that majority of the columns have object type data that restricts us for further analysis as python techniques require data to be numerical to analyse further and build Machine Learning model.

We are using Encoding technique to convert the data, specifically Label Encoding.

Encoding is the process of converting the data or a given sequence of characters, symbols, alphabets etc., into a specified format, for the secured transmission of data.

**Label Encoding** is a technique that is used to convert categorical columns into numerical ones so that they can be fitted by machine learning models which only take numerical data. It is an important pre-processing step in a machine-learning project.

Label encoder can be imported from sklearn.preprocessing.

We have used the below code(we have compiled the object type variables into a list)-

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()  
for i in list:

data[i]=le.fit\_transform(data[i])

We have converted all the data into numerical values to further analyze the data.

***Descriptive Statistics:***

EDA utilizes precise records to recognize the important tendency, variability, and distribution of variables. Measures like suggest, median, mode, preferred deviation, range, and percentiles are usually used.

We have used dataframe.describe() method to gain further insights about the dataset.

**Observations-**

1. There are no null values for any of the variables.

2. Range of standard deviation shows that the scale of distribution varies a lot.

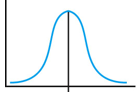
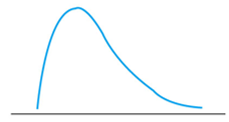
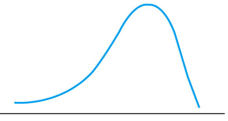
3. Difference in 75% and max is high for many variables like Longitude, Cuisines etc which indicate presence of outliers.

***Distribution of data-***

Data distribution refers to the way data values are spread or distributed in a dataset. It aims at providing valuable insights, informs decision-making, and ensures that appropriate methods are used for statistical analysis and modelling.

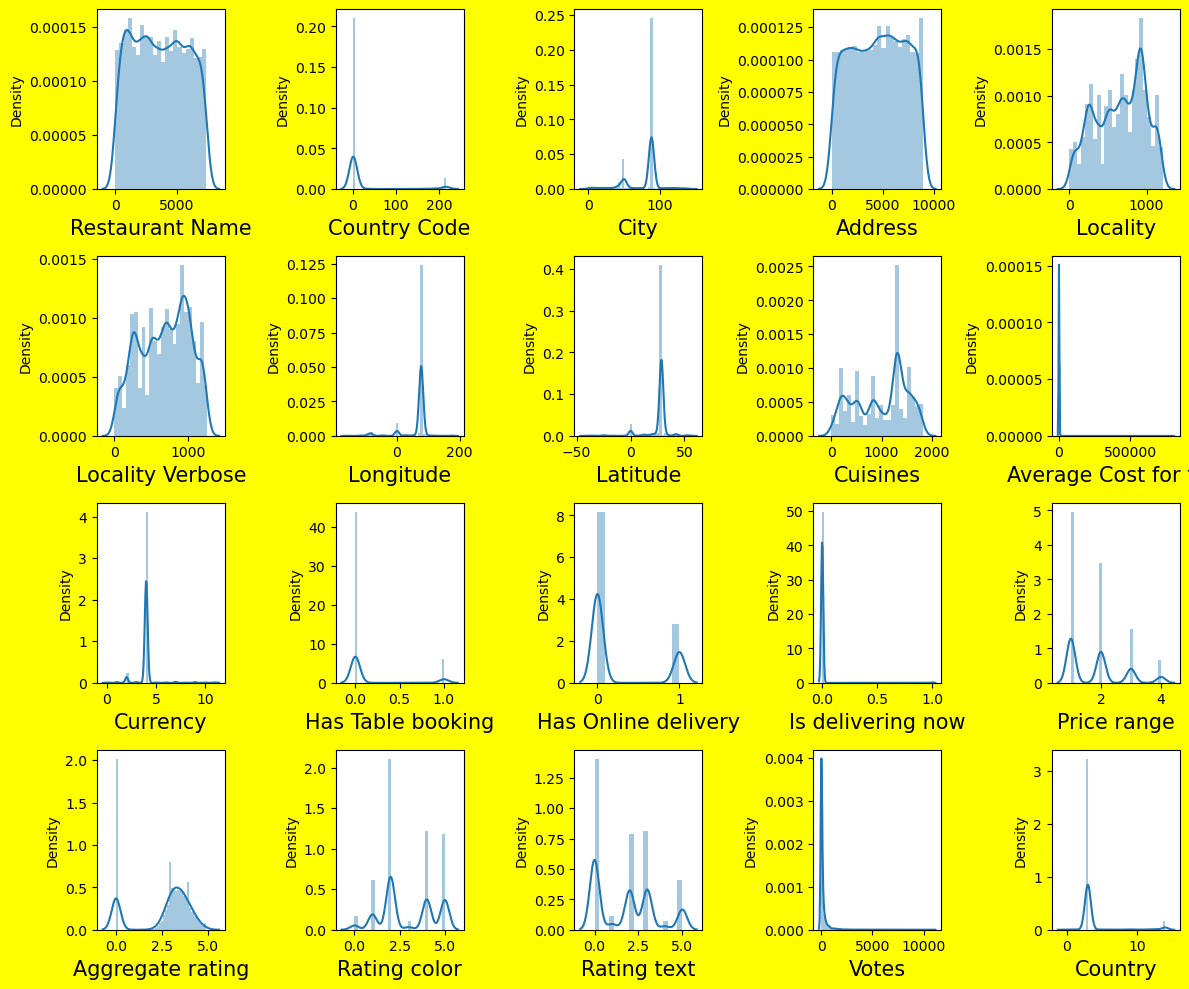
Data distribution refers to the representation of data points scattered or clustered around with specific values and quantitative ranges. When joining the data points together, they typically create a shape that expresses a pattern following the underlying factor. Moreover, it provides insights into the patterns, central tendencies, and variability of the data.

**T**he shapes of data distribution can be symmetrical or asymmetrical. A typical example of symmetry is a normal distribution, while an asymmetric distribution is considered skewed.

Normal Distribution Positively skewed Negatively skewed

Let’s check what the distribution of data looks like-



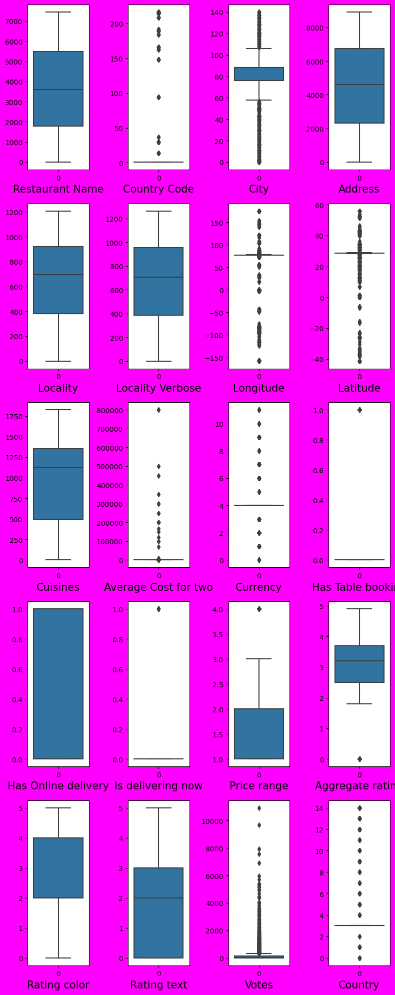
Most of the variables like Votes, Is delivering now, Average cost for two etc are skewed to right. A lot of other variables are skewed. This clearly means there can be outliers in the data set. Outliers are extreme values that stand out greatly from the overall pattern of values in a dataset or graph.

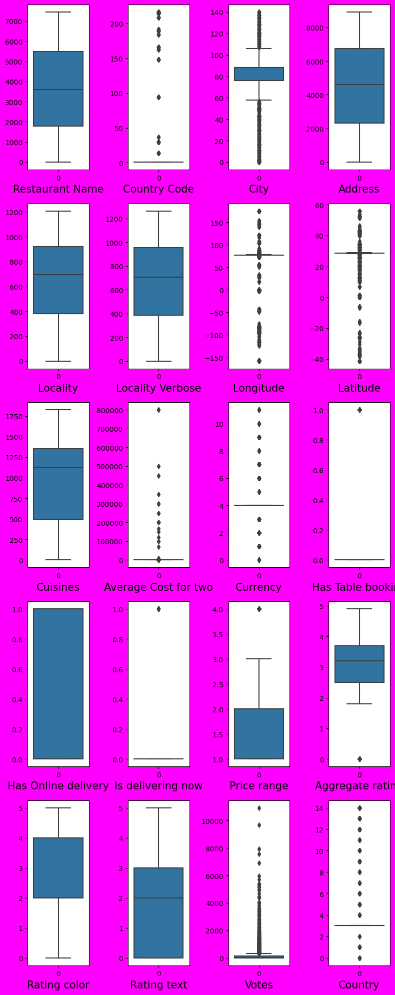
Outliers can give helpful insights into the data you're studying, and they can have an effect on statistical results. This can potentially help you discover inconsistencies and detect any errors in your statistical processes.

There are **four ways** to identify outliers:

1. [Sorting method](https://www.scribbr.com/statistics/outliers/#sorting)
2. [Data visualization method](https://www.scribbr.com/statistics/outliers/#visualizations)
3. [Statistical tests (*z* scores)](https://www.scribbr.com/statistics/outliers/#statistical)
4. [Interquartile range method](https://www.scribbr.com/statistics/outliers/#interquartile)

We are using Boxplot- Box Plot is a graphical method to visualize data distribution for gaining insights and making informed decisions. Box plot is a type of chart that depicts a group of numerical data through their quartiles.



Variables like Country code, City, Longitute, Latitude, Average Cost for two, Currency, Has Table booking, Aggregate rating, Votes and Country have outliers.

Rest of the columns don't have outliers at all.

***Correlation and Relationships:***

EDA allows discover relationships and dependencies between variables. Techniques such as correlation analysis, scatter plots, and pass-tabulations offer insights into the power and direction of relationships between variables.

We are using Heatmap to find the correlation between features.

A heatmap (or heat map) is a graphical representation of data where values are depicted by colour.

Codes used-

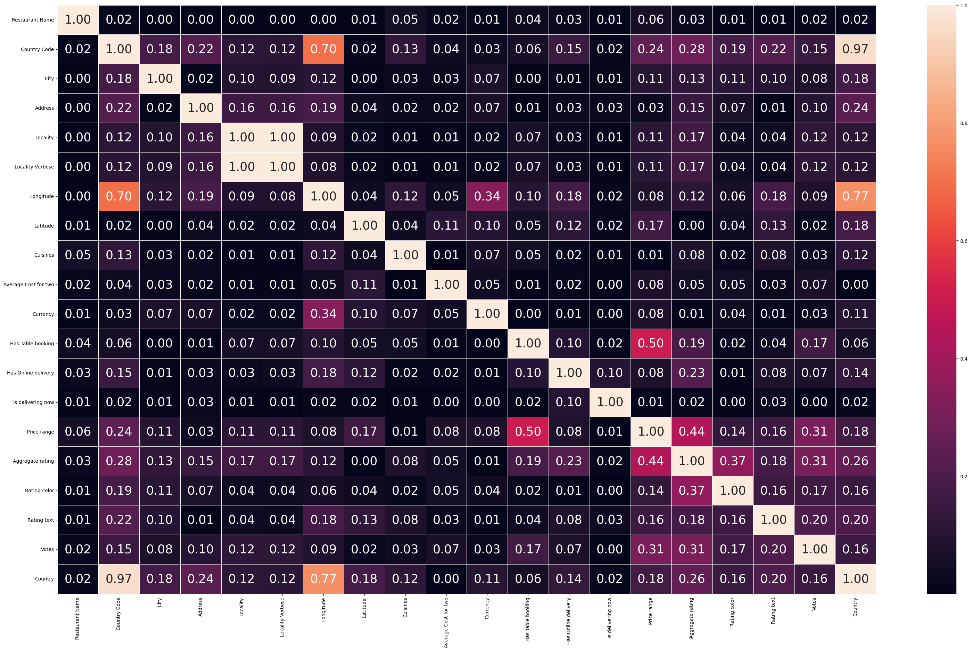
data\_corr=data.corr().abs()

plt.figure(figsize=(38,22))

sns.heatmap(data\_corr, annot=True, fmt='0.2f', annot\_kws={'size':25}, linewidth=.5)

plt.show()

***Heatmap***



**Observations-**

1. Features like Country code and Country are highly correlated. The score for correlated features are highlighted in Orange/Red/Pink volour.

2. Locality and Locality Verbrose have the perfect correlation score of 1 which means they have the highest correlation among features. We see multicollinearity issue here.

Multicollinearity arises when two or more features are highly correlated. This can lead to issues in estimating the individual coefficients of the correlated features because their effects on the target variable become difficult to distinguish.

Multicollinearity can result in unstable and inaccurate coefficient estimates. The model becomes sensitive to small changes in the data, making it less reliable.

3. Features like Currency and Is Delivering Now seem to be least correlated with the label Price Range.

4. Features like Locality, Locality Verbose, Has Table booking are least correlated to the label Average cost for two and features like Is delivering now and Country are not correlated at all.

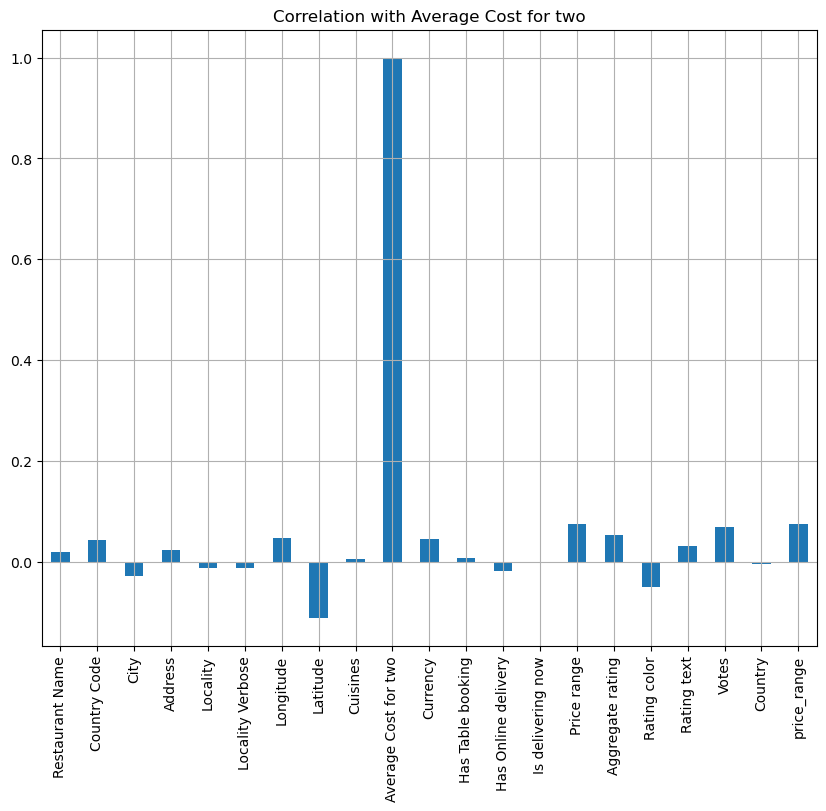
It is imperative to pick the right number of features for building a model and for that, we need to remove features that are correlated with each other or the features that least impact(minimum or no impact) the label.

Correlated features in general don't improve models (although it depends on the specifics of the problem like the number of variables and the degree of correlation), but they affect specific models in different ways and to varying extents.

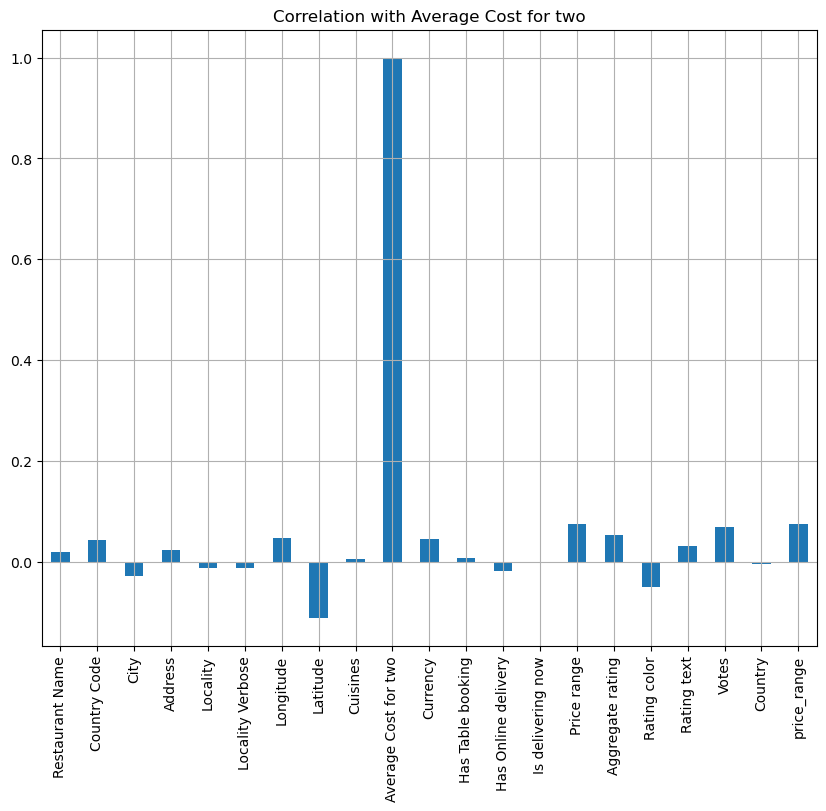
However, if the correlated features are also highly correlated with the label, it is wise to keep them.

To determine which are the right features to proceed with, we need to check their correlation with the label.

Since we are making 2 predictions, there are two labels and we need to check the correlation of featured with both the labels.



Correlation of features with the label-“Average cot for two”



Correlation of features with the label=”Price range”

We can conclude that Country and Is delivering now are least contributors for determining the label Average Cost for two.

Now, as we have identified which are the best features to proceed with, we will be removing the features that are irrelevant using dataframe.drop() method.

After removing the irrelevant variables, we have 9551 entries and 17 columns.

***Handling Outliers-***

Outliers, deviating significantly from the norm, can distort measures of central tendency and affect statistical analyses.

We have analysed and detected outliers in the dataset using Boxplot method.

Now, we need to treat/remove the outliers for proceeding to building a model.

We are using Z-score to remove the outliers here.

Z-score is also called a standard score. This value/score helps to understand that how far is the data point from the mean. And after setting up a threshold value one can utilize z score values of data points to define the outliers.

Z-score can be imported from scipy.stats.

Now, to define an outlier threshold value is chosen which is generally 3.0. As 99.7% of the data points lie between +/- 3 standard deviation (using Gaussian Distribution approach). However, if there are too many outliers detected and we feel that taking outliers threshold as 3 is resulting to a loss of too much data, we can also take the threshold as 3.1 or 3.2.

***Let’s remove rows where Z value is greater than 3.2.***

In this example, we sets a threshold value of 3.2 and then uses NumPy’s np.where() to identify the positions (indices) in the Z-score array z where the absolute Z score is greater than the specified threshold (3.2). It prints the positions of the outliers in the ‘age’ column based on the Z-score criterion.

Codes used-

z=np.abs(zscore(data))

abs\_z\_score=np.abs(z)

filt\_ent=(abs\_z\_score<3.2).all(axis=1)

data=data[filt\_ent]

After removing the outliers, we are left with 8482 records and 17 columns.

In the initial stage of analysing data and presenting it visually, we had also seen a lot of skewness in the variables.   
  
Addressing skewness through data transformation is not merely a statistical adjustment; it is a crucial step toward uncovering precise, actionable insights.

We will be removing skewness from data using different methods.

There are different methods to deal with skewness depending on the mildness or high skewness.

We have observed that features like 'City','Longitude','Latitude’ and ‘Currency’ are negatively skewed and 'Votes' and 'Country Code' are positively correlated.

We have used np.sqrt(square root method) to treat positive skewness and square method to treat negative skewness.   
  
For highly skewed variable 'Country Code', we have used natural log method.

Th skeweness in the data has been treated.

We have completed EDA and we can derive the following conclusion-

1. We have used feature engineering techniques like label encoding to encode object type data in numerical values.
2. Visualizing data has helped us determine the most important features or features that impact label the most.
3. There are 17 variables we are going to build the model with.

*PRE-PROCESSING PIPELINE-*

***Separating data-***

To proceed further, we need to separate data into features and labels so that we can scale the data, split into train test and build the model for prediction.

Since we are building two types of models (one regression and one classification), we are separating data for each of them.

For Classification model using label as ‘Price range’, we have stored features in X and label in y.

Similarly for Regression model using label as ‘Average cost for two’, we have stored features in U and label in v.

***Scaling data-***

When a dataset has values of different columns at drastically different scales, it gets tough to analyze the trends and patterns and comparison of the features or columns. So, in cases where all the columns have a significant difference in their scales, are needed to be modified in such a way that all those values fall into the same scale. This process is called **Scaling**.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

Here, we are using StandardScaler to scale the data.

Since it is one of the pre-processing technique, StandardScaler can be imported from sklearn.preprocessing .

We will be using this standardizing technique on features only which are X and U to get X\_scaled and U\_scaled(standardized features).

***Splitting data into train and test-***

One of the key aspects of supervised [machine learning](https://realpython.com/learning-paths/machine-learning-python/) is model evaluation and validation. When we evaluate the predictive performance of the model, it’s essential that the process be unbiased

Testing the model on the same data as it was trained on will lead to an overfit and poor performance in real-life scenarios.

In order to avoid that, split your data into 2 pieces: train set and test set. The most common practice is to do a 80-20/70-30/75-25 split.

Splitting a dataset might also be important for detecting if your model suffers from one of two very common problems, called underfitting and overfitting:

1. **Underfitting** is usually the consequence of a model being unable to encapsulate the relations among data. For example, this can happen when trying to represent nonlinear relations with a linear model. Underfitted models will likely have poor performance with both training and test sets.
2. **Overfitting** usually takes place when a model has an excessively complex structure and learns both the existing relations among data and noise. Such models often have bad generalization capabilities. Although they work well with training data, they usually yield poor performance with unseen (test) data.

For this, we are using **train\_test\_split()** which can be imported from sklearn.model\_selection.

For classification model, the data has been split into X\_train, X\_test, y\_train, y\_test and for regression model, data has been split into U\_train, U\_test, v\_train, v\_test.

We have completed all the pre-processing steps and now we will be proceeding with building model for prediction.

*MODEL BUILDING-*

From now we will talk about machine learning and its models that work to predict the PRICE RANGE and AVERAGE COST FOR TWO.

So far we have,

* visually represented data to understand the patterns and relationship
* encoded object type data to numerical values using Label encoder
* detected and removed outliers
* treated skewness
* detected and handles multicollinearity issue( by dropping highly correlated columns that have least impact on the label)
* separated, scaled and splitted data

***FOR CLASSIFICATION MODEL-***

For classification model, we are using a function code(writing one function to call it for multiple models).

We are using **accuracy\_score**  as evaluation metric. It can be imported from sklearn.metrics.

We have built 4 models using different classification models so that we can choose the one that works the best on this dataset.

***Cross validation-***

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model’s performance. Cross validation is an important step in the machine learning process and helps to ensure that the model selected for deployment is robust and generalizes well to new data.

The main purpose of cross validation is to prevent overfitting, which occurs when a model is trained too well on the training data and performs poorly on new, unseen data. By evaluating the model on multiple validation sets, cross validation provides a more realistic estimate of the model’s generalization performance, i.e., its ability to perform well on new, unseen data.

We will be using this technique as well for the model built. We will be using CV=3,4 and 5 for each model.

1. *Random Forest Classifier-*

Random Forest Classifier can be imported from sklearn.ensemble.

The Random Forest or Random Decision Forest is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees. Random Forests are particularly well-suited for handling large and complex datasets, dealing with high-dimensional feature spaces, and providing insights into feature importance.

The accuracy score for train set is 100% and Accuracy Score for test set it 98.59%

The Cross validation score(CV score) at CV=3 is 97.4% which proves that the model is overfitting.

1. *KNeighbors Classifier*

KNeighbors Classifier can be imported from sklearn.neighbors.

KNN is a simple, supervised machine learning (ML) algorithm that can be used for classification or regression tasks - and is also frequently used in missing value imputation. It is based on the idea that the observations closest to a given data point are the most "similar" observations in a data set, and we can therefore classify unforeseen points based on the values of the closest existing points. By choosing K, the user can select the number of nearby observations to use in the algorithm.

The accuracy score for train set is 82.38% and Accuracy Score for test set it 74.42

%.

The Cross validation score(CV score) at CV=3 is 70.9% which is the highest among all other cross folds(3,4,5).

1. *Gradient Boosting Classifier-*

Gradient Boosting Classifier can also be imported from sklearn.ensemble.

This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n\_classes\_ regression trees are fit on the negative gradient of the loss function, e.g. binary or multiclass log loss. Binary classification is a special case where only a single regression tree is induced.

The accuracy score for train set is 99.97% and Accuracy Score for test set it 98.66

%.

The Cross validation score(CV score) at CV=5 is 90.11% which is the highest among all other cross folds(3,4,5).

1. *SVC-*

SVC can be imported from sklearn.svm.

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible.

The accuracy score for train set is 95.22% and Accuracy Score for test set it 95.36

%.

The Cross validation score(CV score) at CV=5 is 79% which is the highest among all other cross folds(3,4,5).

Clearly, we are getting one of the best accuracy score 98.47% from Random Forest Classifier model. The cross validation score at cv=3 is 97.43% which is a great score and proves that the model is generalized(isn't overfitted or underfitted).

Gradient Boosting classifier model has the highest accuracy score of 98.66% however, the CV score at CV=5 is 90.11%.

We can conclude that Random Forest Classifier model is the best fitted model as it works great on test data.

***Hyper-parameter tuning-***

 Machine Learning model is defined as a mathematical model with several parameters that need to be learned from the data. By training a model with existing data, we can fit the model parameters.   
However, there is another kind of parameter, known as ***Hyperparameters***, that cannot be directly learned from the regular training process.

Hyperparameter tuning is the process of selecting the optimal values for a machine learning model’s hyperparameters. Hyperparameters are settings that control the learning process of the model, such as the learning rate, the number of neurons in a neural network, or the kernel size in a support vector machine. The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task.

We are using GridSearchCV method for Hyper-parameter tuning.

Grid search can be considered as a “brute force” approach to hyperparameter optimization. We fit the model using all possible combinations after creating a grid of potential discrete hyperparameter values. We log each set’s model performance and then choose the combination that produces the best results. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

Since we have finalized RandomForest Classfier as our final model, we will be fine tuning the parameters of this model.   
Below parameters are the best combination to train the model and predict-

***criterion='entropy', max\_depth= 30,max\_leaf\_nodes=45,n\_estimators=50***

Using these parameters, we have built a model and got 97.76% accuracy score.

***CONCLUSION-***

After tuning the parameters, we are still getting 97.76% as accuracy score which implies that this is a generalized model that will work well on unseen data.

Prediction made by tuned model-

| **Actual** | **RF predicted** |
| --- | --- |
| **6529** | 4 | 4 |
| **2988** | 2 | 2 |
| **5170** | 2 | 2 |
| **1027** | 2 | 2 |
| **9152** | 3 | 2 |
| **...** | ... | ... |
| **6906** | 2 | 2 |
| **4283** | 2 | 2 |
| **4321** | 4 | 4 |
| **4901** | 1 | 1 |
| **1636** | 2 | 2 |

***FOR REGRESSION MODEL-***

For regression model, we are using mean\_absolute\_error and r2\_score

As evaluation metrics. They can be imported from sklearn.metrics.

Like we did for classification model, we will be building multiple models to figure out which model works the best on this data.

1. *Linear Regression-*

Linear Regression can be imported from sklearn.linear\_model.

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.

The interpretability of linear regression is a notable strength. The model’s equation provides clear coefficients that elucidate the impact of each independent variable on the dependent variable, facilitating a deeper understanding of the underlying dynamics. Its simplicity is a virtue, as linear regression is transparent, easy to implement, and serves as a foundational concept for more complex algorithms.

The accuracy score for train set is 75.2% and Accuracy Score for test set it 70.9

%. The means absolute score is 169.

The Cross validation score(CV score) at CV=5 is 70% which is the highest among all other cross folds(3,4,5).

1. *Gradient Boosting Regressor-*

Gradient Boosting regressor can be imported from sklearn.ensemble.

This estimator builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

The accuracy score for train set is 91.5% and Accuracy Score for test set it 86

%. The means absolute score is 115.

The Cross validation score(CV score) at CV=5 is 82.9% which is the highest among all other cross folds(3,4,5).

1. *RandomForestRegressor-*

Random Forest Regressor can be imported from sklearn.ensemble.

The accuracy score for train set is 98.2% and Accuracy Score for test set it 85.7

%. The means absolute score is 108.

The Cross validation score(CV score) at CV=5 is 83.7% which is the highest among all other cross folds(3,4,5).

1. *SVR-*

SVR can be imported from sklearn.svm.

Support vector regression (SVR) is a type of support vector machine (SVM) that is used for regression tasks. It tries to find a function that best predicts the continuous output value for a given input value.

SVR can use both linear and non-linear kernels. A linear kernel is a simple dot product between two input vectors, while a non-linear kernel is a more complex function that can capture more intricate patterns in the data. The choice of kernel depends on the data’s characteristics and the task’s complexity.

The accuracy score for train set is 27% and Accuracy Score for test set it 25

%. The means absolute score is 233.

The Cross validation score(CV score) at CV=5 is 83% which is the highest among all other cross folds(3,4,5).

It is evident that Random Forest Regressor model works best on test data and Mean absolute score is the lowest. We are getting r2 score of 85.4% with this model and Mean Absolute Error score of 109. At CV=5, we are getting 83% accuracy when actual test score is 85.4% and MAE is 109.

Gradient Boosting Regressor is giving the best r2-score of 86% however, MAE is 115 which is higher than Random Forest Regressor model and CV score at CV=5 is 82% which is again lower than Random Forest Regressor model.

Hence, Random Forest Regressor model seems to be the best model and we will move ahead with this.

Same as classification model, we will be fine-tuning the final model Random forest regressor. The parameters selected are- criterion='squared\_error',max\_depth= 20,max\_leaf\_nodes=50,n\_estimators=50.

***CONCLUSION-***

We have used these parameters to build a final model which gives the accuracy score of 84.9%

Prediction made by final model-

|  | **Actual** | **RFR predicted** |
| --- | --- | --- |
| **3536** | 350 | 322.329281 |
| **4549** | 250 | 222.706974 |
| **7602** | 200 | 225.826488 |
| **6266** | 550 | 604.210110 |
| **8987** | 700 | 602.140503 |
| **...** | ... | ... |
| **3347** | 700 | 604.210110 |
| **1870** | 650 | 604.210110 |
| **6157** | 200 | 261.671301 |
| **7573** | 150 | 218.878849 |
| **7257** | 400 | 296.173134 |

FiNAL CONCLUSION-

We have chosen random forest regressor and random forest classifier with tuned parameters as our final models for prediction.

We have got good r2\_score and accuracy for both the predictions and we can also conclude that both the models are not overfitted/underfitted as they give good CV score and the score after tuning parameters doesn’t vary much.