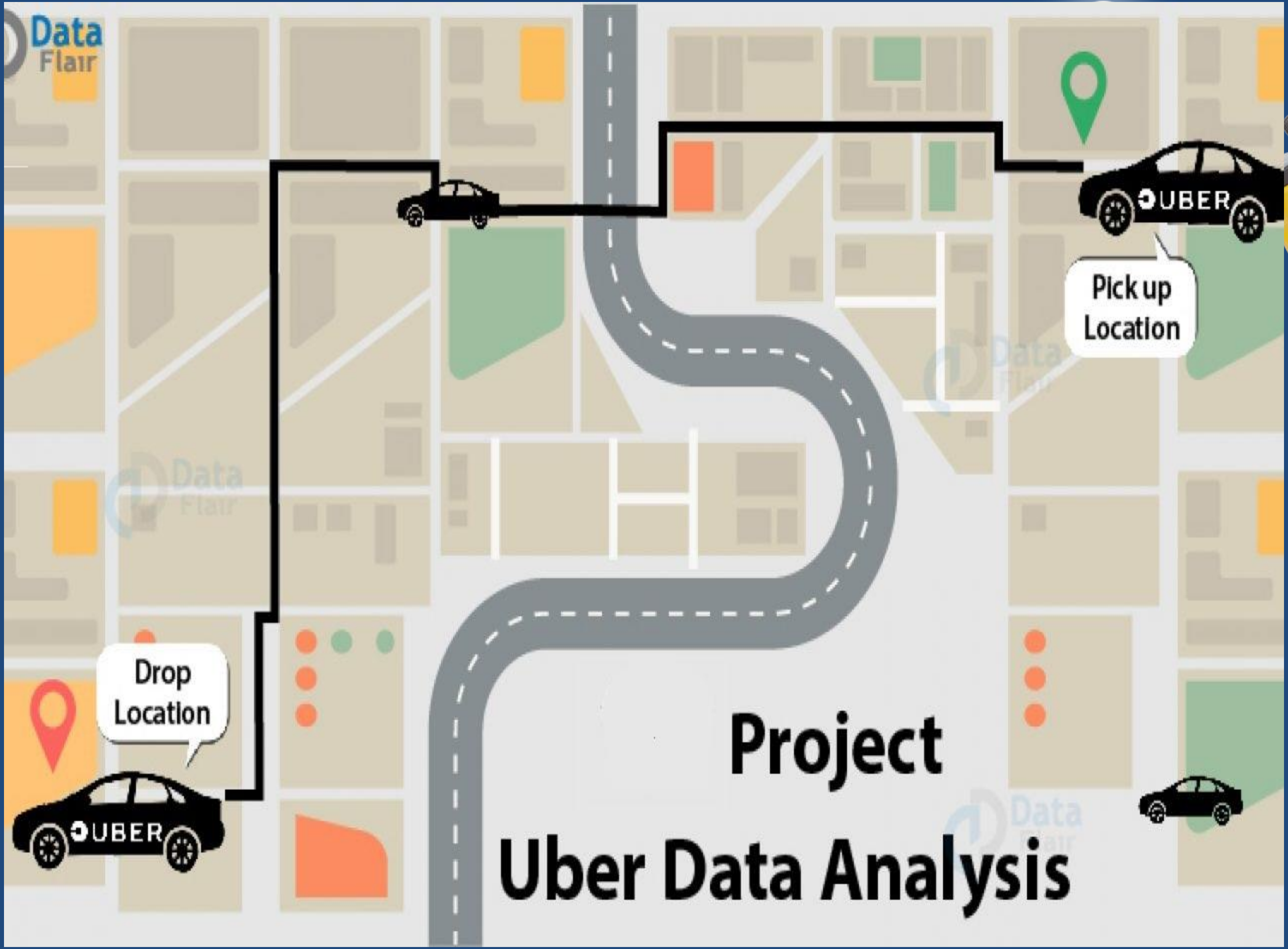


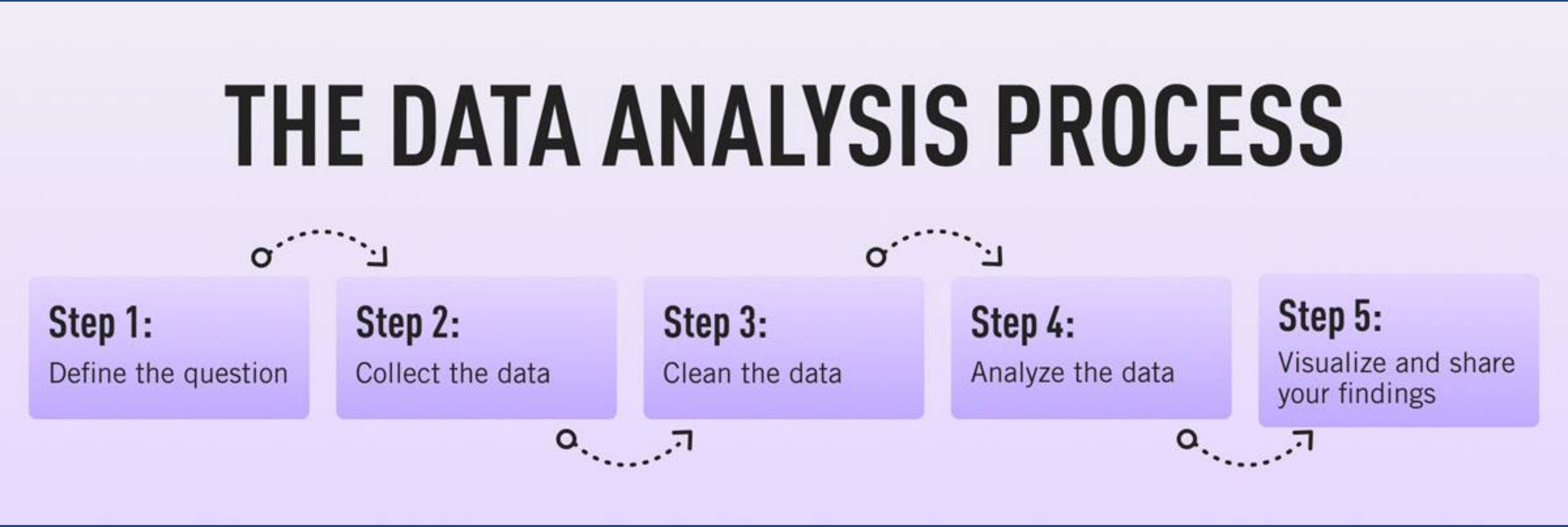
Project Uber Data Analysis

PRESENTED BY :

AMBAR KUMAR
(111903136)

SANKALP MAHESHWARI
(111903154)

VISHAL SHARMA
(111903159)



THE DATA ANALYSIS PROCESS

Step 1: Define the question

Step 2: Collect the data

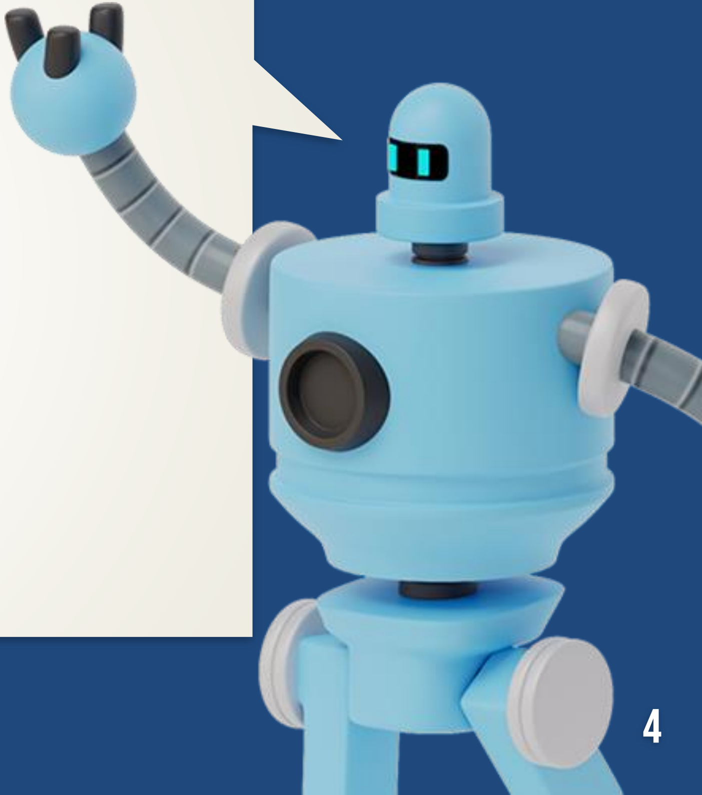
Step 3: Clean the data

Step 4: Analyze the data

Step 5: Visualize and share your findings

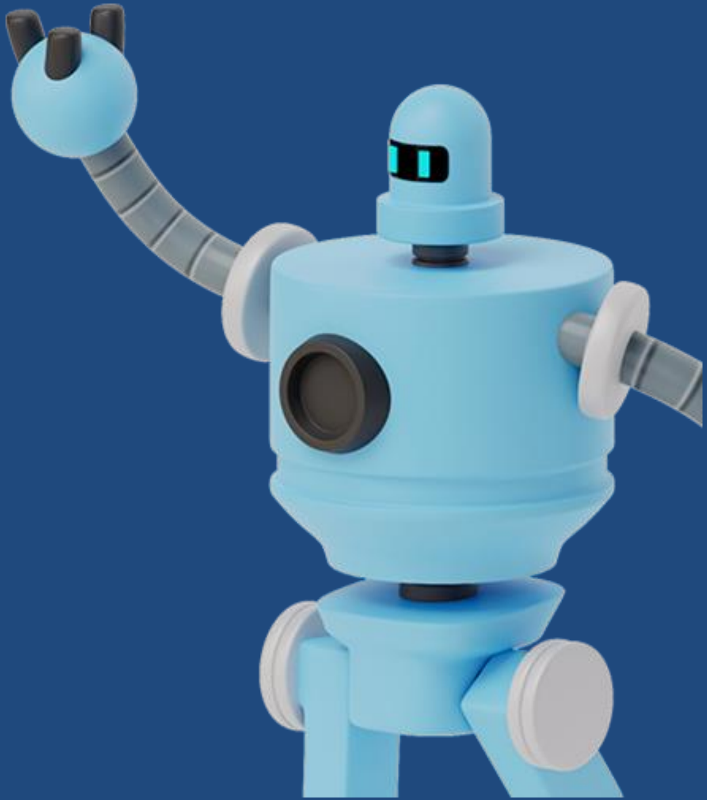
AIM

Complete Data Analysis and Exploration of Uber Dataset



OBJECTIVES :

The objective is to first explore hidden or previously unknown information by applying exploratory data analytics on the dataset and to know the effect of each field on price with every other field of the dataset. Then we apply different machine learning models to complete the analysis. After this, the results of applied machine learning models were compared and analyzed based on accuracy, and then the best performing model was suggested for further predictions of the label ‘Price’.



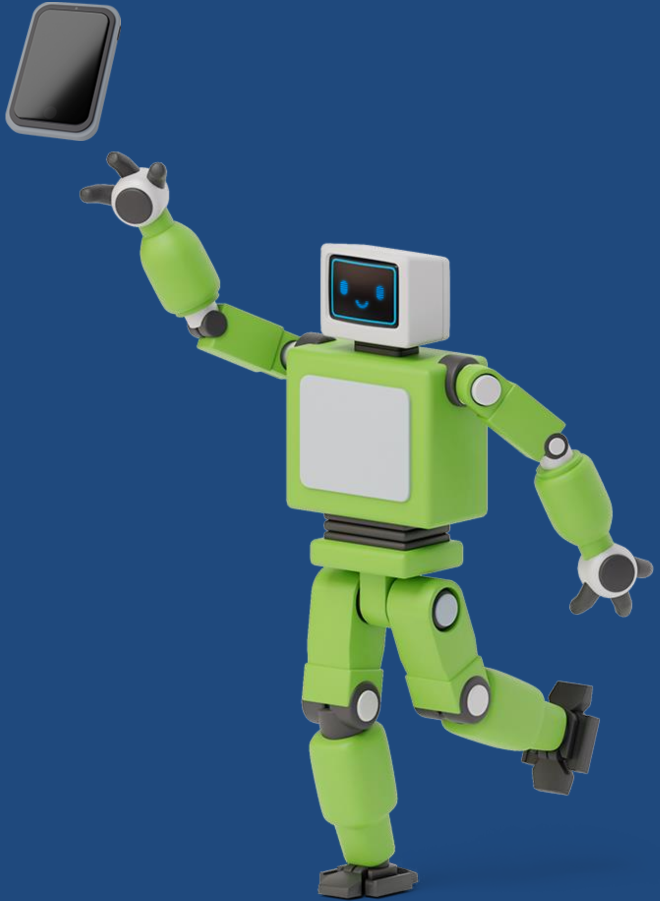
DATA SET :

Kaggle

CSV Format

SHAPE :

322844,56



LIBRARIES

NumPy

matplotlib

Pandas

scikit learn

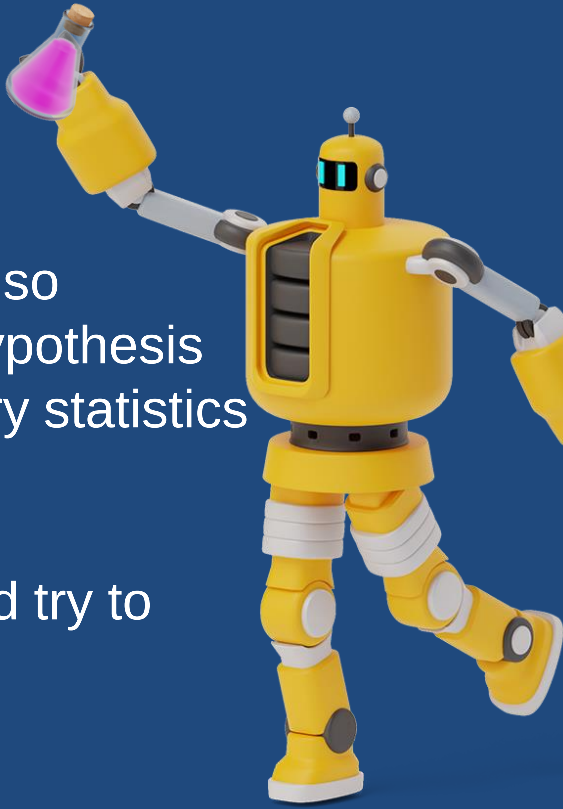
Seaborn

EXPLORATORY DATA ANALYSIS

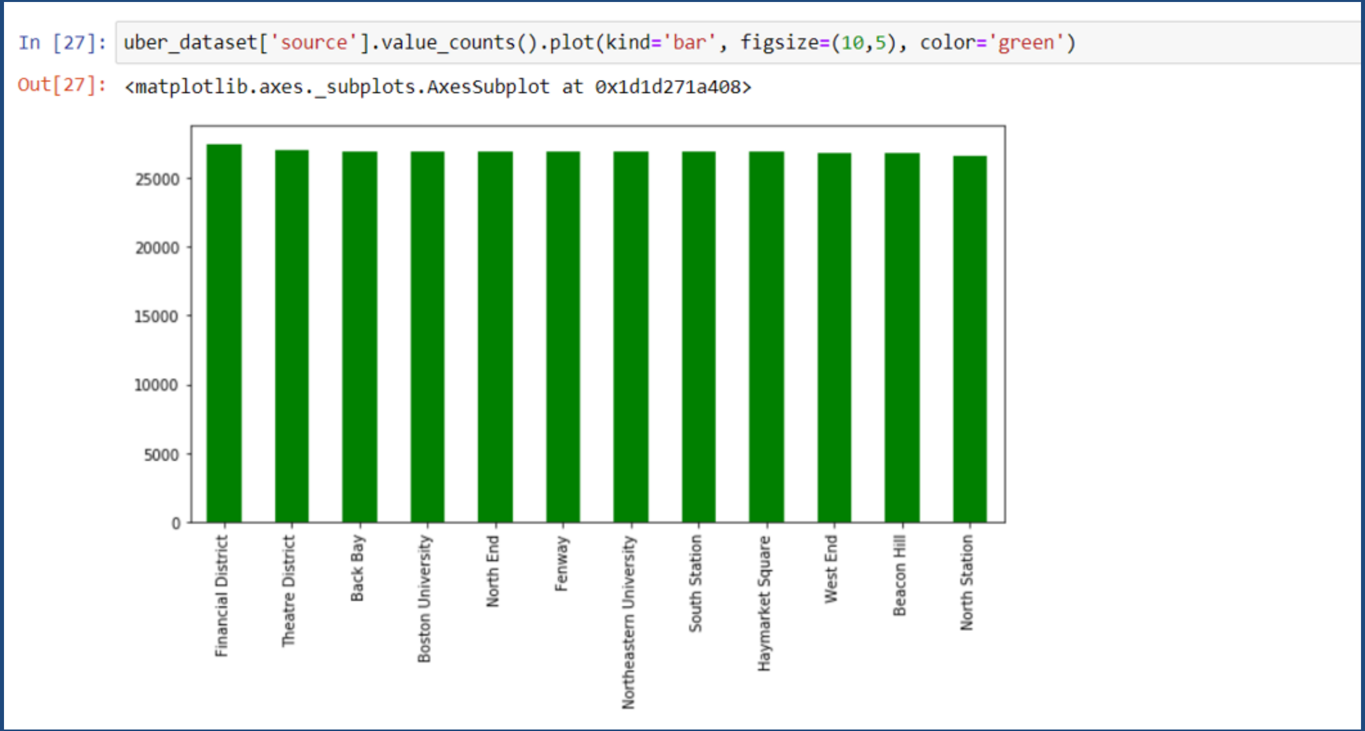
Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns to spot anomalies to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

It is a good practice to understand the data first and try to gather as many insights from it.

EDA is all about making sense of data in hand.

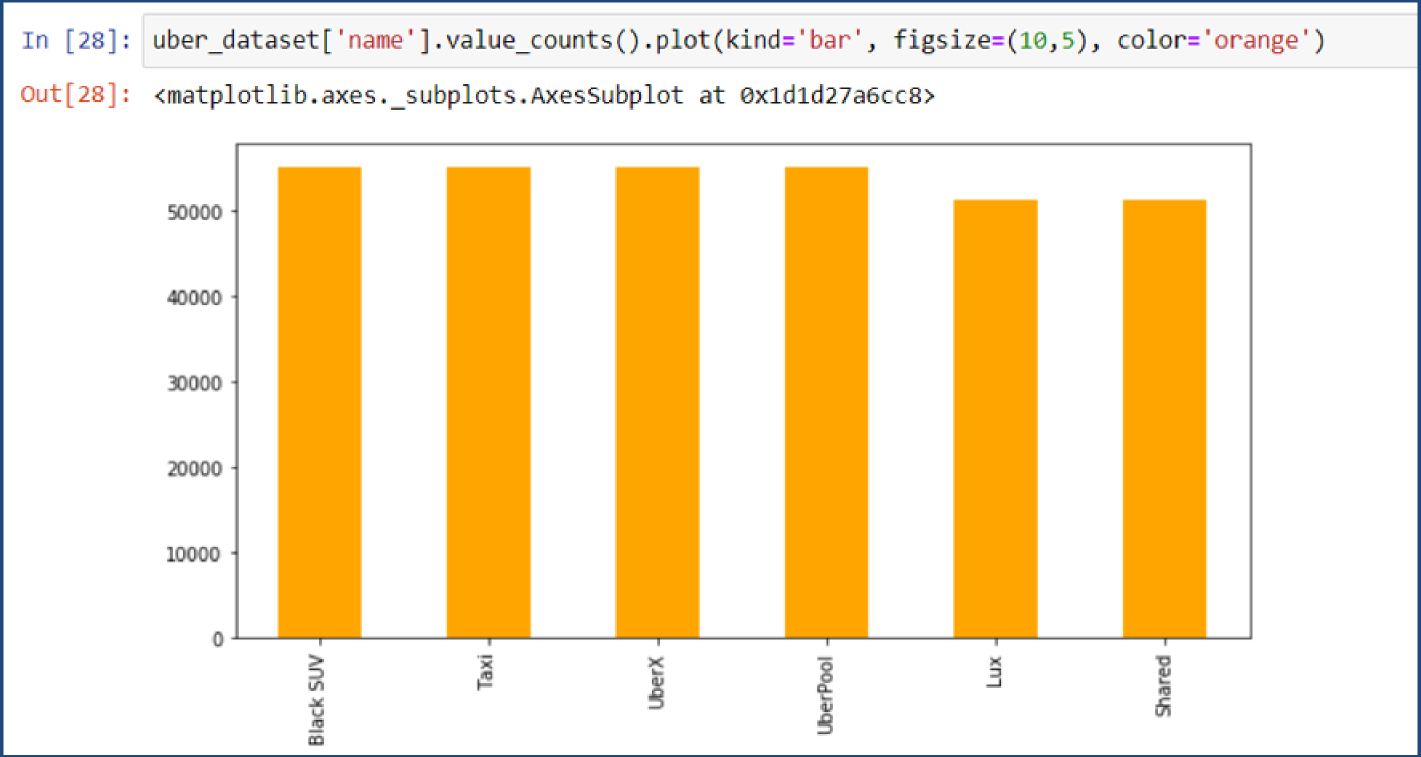


BAR PLOT BETWEEN SOURCE AND CUSTOMERS



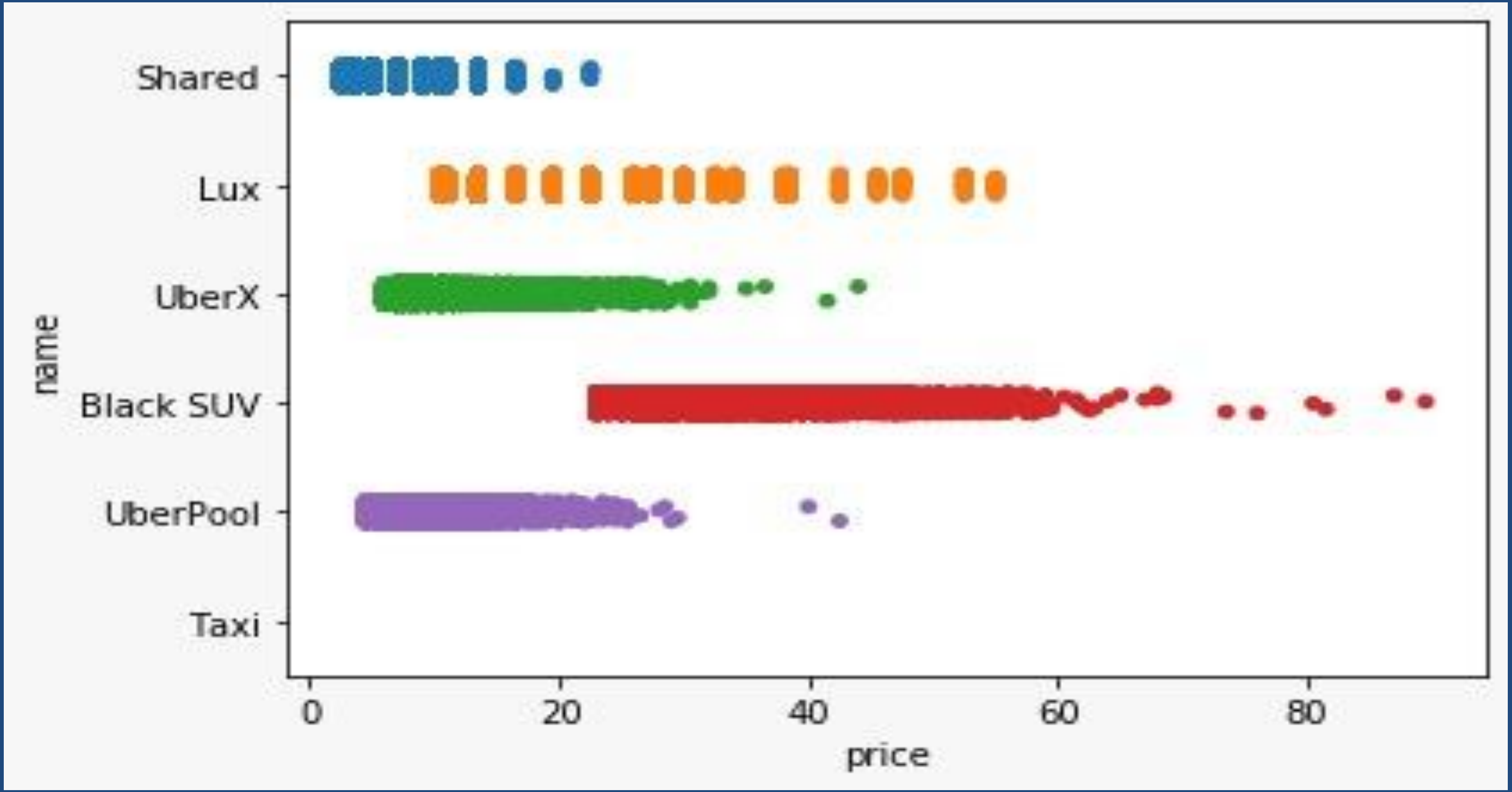
```
In [27]: uber_dataset['source'].value_counts().plot(kind='bar', figsize=(10,5), color='green')
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1d1d271a408>
```

BAR PLOT BETWEEN CAB TYPE AND CUSTOMERS



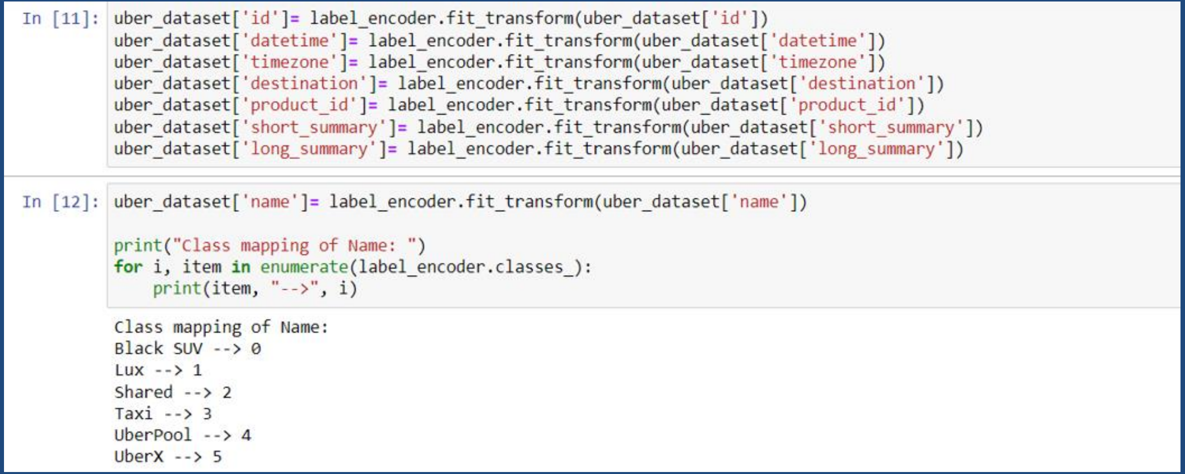
```
In [28]: uber_dataset['name'].value_counts().plot(kind='bar', figsize=(10,5), color='orange')

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1d1d27a6cc8>
```

STRIP-PLOT BETWEEN PRICE AND CAB-TYPE



LABEL ENCODING

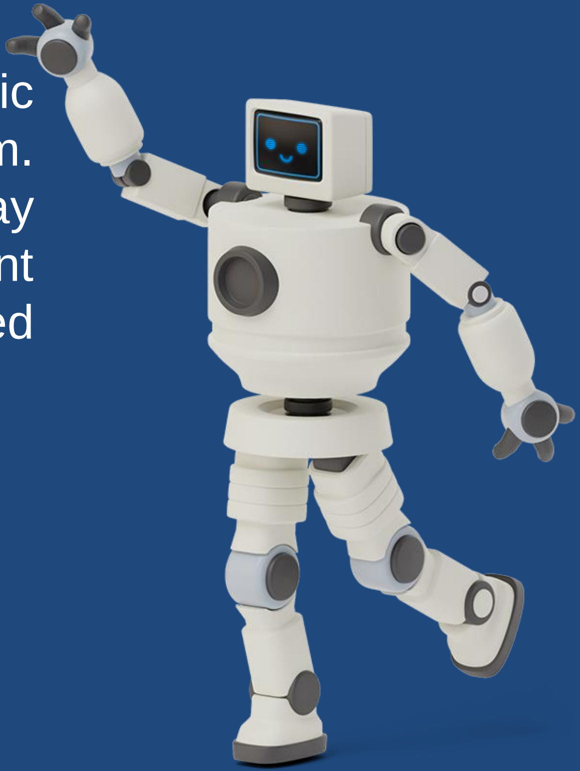
Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.



```
In [11]: uber_dataset['id']= label_encoder.fit_transform(uber_dataset['id'])
         uber_dataset['datetime']= label_encoder.fit_transform(uber_dataset['datetime'])
         uber_dataset['timezone']= label_encoder.fit_transform(uber_dataset['timezone'])
         uber_dataset['destination']= label_encoder.fit_transform(uber_dataset['destination'])
         uber_dataset['product_id']= label_encoder.fit_transform(uber_dataset['product_id'])
         uber_dataset['short_summary']= label_encoder.fit_transform(uber_dataset['short_summary'])
         uber_dataset['long_summary']= label_encoder.fit_transform(uber_dataset['long_summary'])
```

```
In [12]: uber_dataset['name']= label_encoder.fit_transform(uber_dataset['name'])

         print("Class mapping of Name: ")
         for i, item in enumerate(label_encoder.classes_):
             print(item, "-->", i)
```

```
Class mapping of Name:
Black SUV --> 0
Lux --> 1
Shared --> 2
Taxi --> 3
UberPool --> 4
UberX --> 5
```

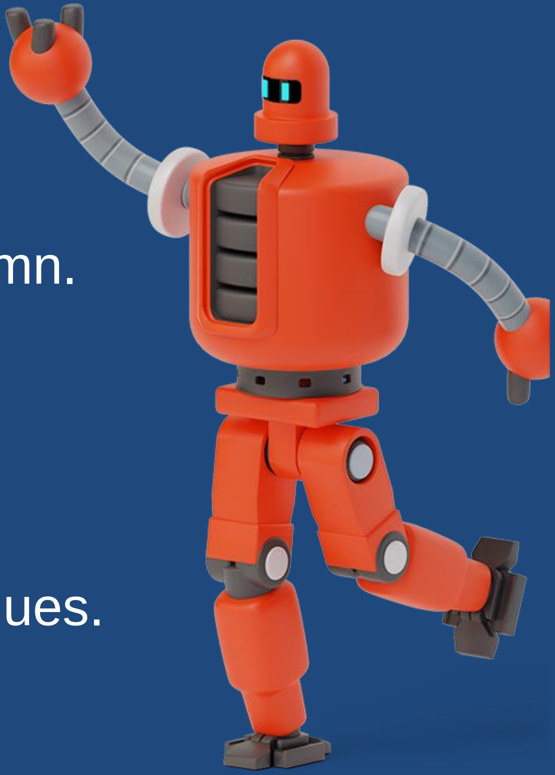


NANS(MISSING VALUES)

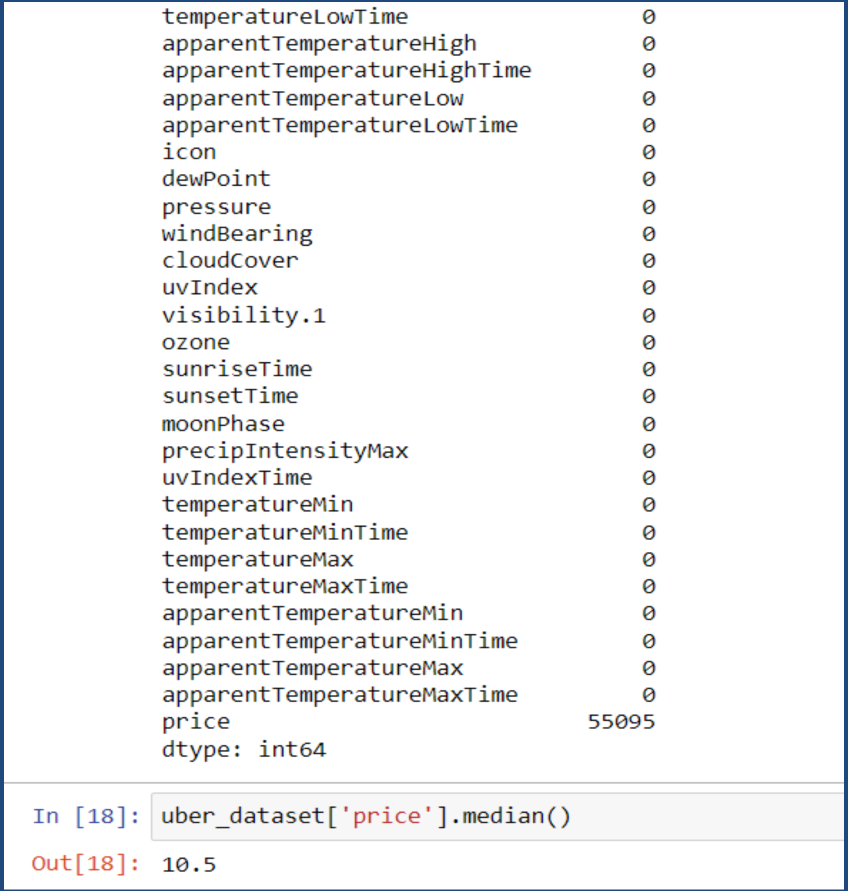
Our data set contain NANs only in Price column.

The count of nans is: 55095

The nans is filled with the median of other values.



NANS (MISSING VALUES)



```
temperatureLowTime                0
apparentTemperatureHigh           0
apparentTemperatureHighTime       0
apparentTemperatureLow            0
apparentTemperatureLowTime        0
icon                              0
dewPoint                          0
pressure                          0
windBearing                       0
cloudCover                        0
uvIndex                           0
visibility.1                      0
ozone                             0
sunriseTime                       0
sunsetTime                        0
moonPhase                         0
precipIntensityMax                0
uvIndexTime                       0
temperatureMin                    0
temperatureMinTime                0
temperatureMax                    0
temperatureMaxTime                0
apparentTemperatureMin            0
apparentTemperatureMinTime        0
apparentTemperatureMax            0
apparentTemperatureMaxTime        0
price                         55095
dtype: int64
```

```
In [18]: uber_dataset['price'].median()

Out[18]: 10.5
```

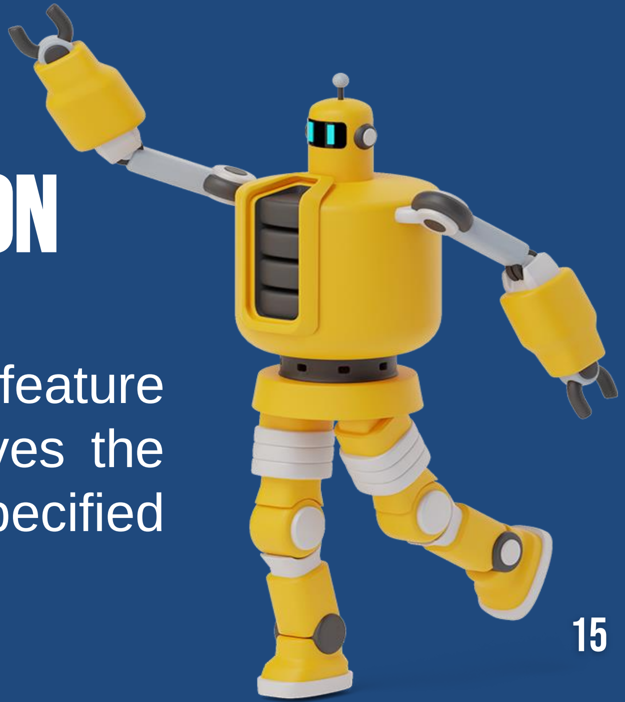
FEATURE SELECTION

Feature Selection is the process of selecting a subset of relevant feature (variables, predictors) for use in model construction.

RECURSIVE FEATURE ELIMINATION

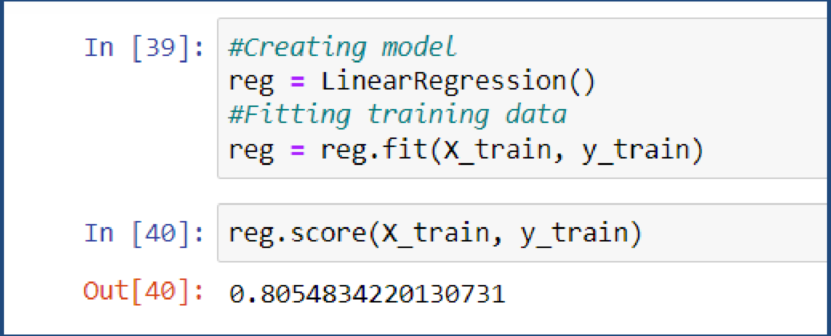
:

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached.



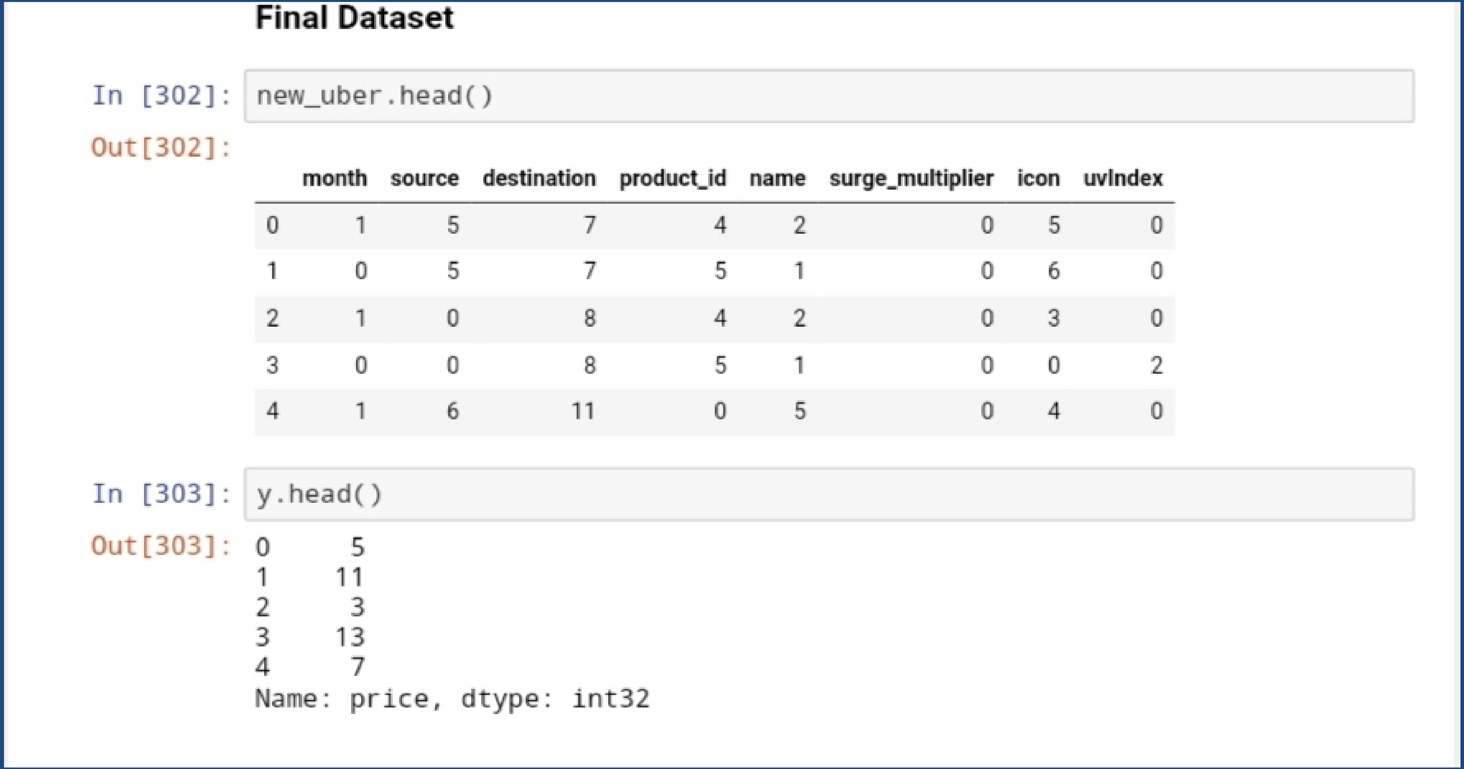
After applying RFE on given data set with Linear Regression, we found accuracy with different no of columns as follows:-

Serial No.	No. of Feature	Accuracy
1	56	0.805483422
2	40	0.8050662132
3	25	0.80553551515
4	15	0.8050457819



```
In [39]: #Creating model
         reg = LinearRegression()
         #Fitting training data
         reg = reg.fit(X_train, y_train)

In [40]: reg.score(X_train, y_train)

Out[40]: 0.8054834220130731
```

FINAL DATASET



Final Dataset

```
In [302]: new_uber.head()
Out[302]:
```

	month	source	destination	product_id	name	surge_multiplier	icon	uvIndex
0	1	5	7	4	2	0	5	0
1	0	5	7	5	1	0	6	0
2	1	0	8	4	2	0	3	0
3	0	0	8	5	1	0	0	2
4	1	6	11	0	5	0	4	0

```
In [303]: y.head()
Out[303]: 0     5
          1    11
          2     3
          3    13
          4     7
          Name: price, dtype: int32
```

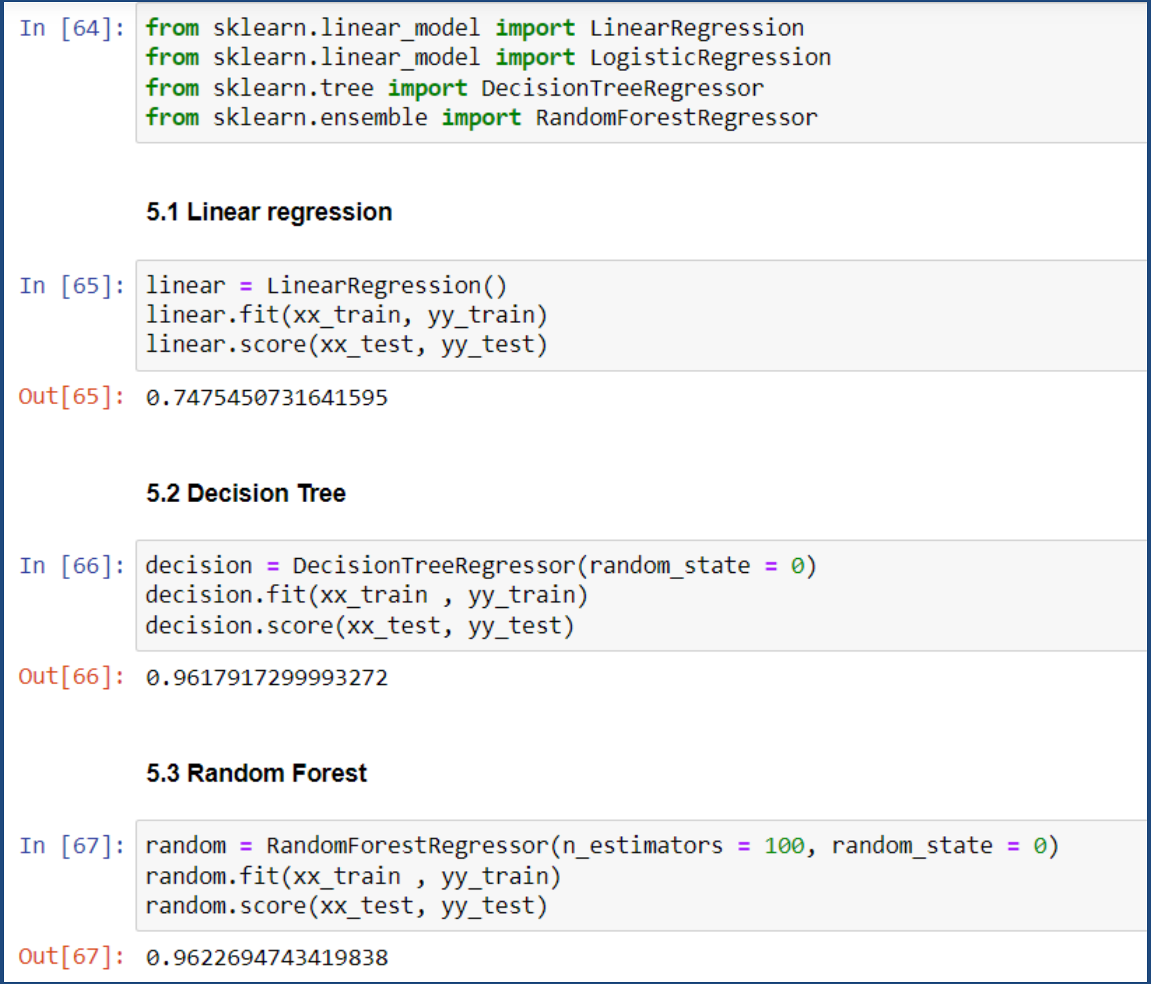
MODELING

After Completion of Recursive Feature Elimination process, we can do Modeling on final dataset.

Linear Regression: Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It’s used to predict values within a continuous range.

Decision Tree: A decision tree is a graphical representation of all the possible solutions to a decision based on certain conditions.

Random Forest: Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.



```
In [64]: from sklearn.linear_model import LinearRegression
         from sklearn.linear_model import LogisticRegression
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.ensemble import RandomForestRegressor
```

5.1 Linear regression

```
In [65]: linear = LinearRegression()
         linear.fit(xx_train, yy_train)
         linear.score(xx_test, yy_test)

Out[65]: 0.7475450731641595
```

5.2 Decision Tree

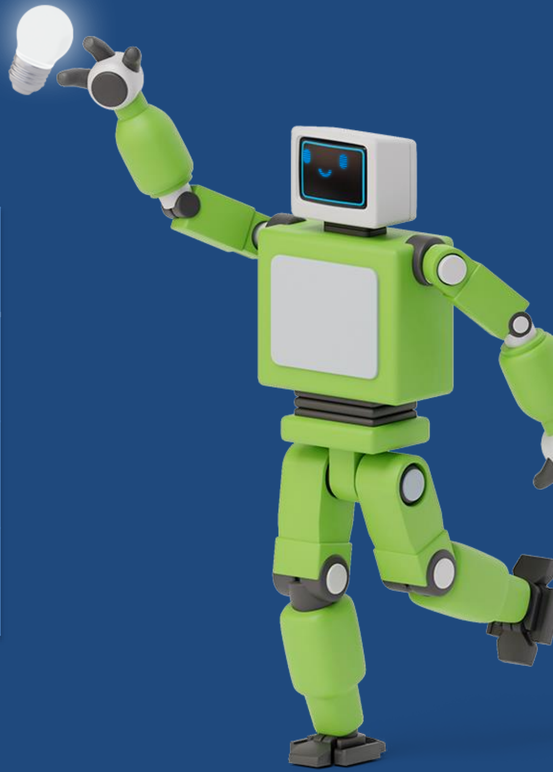
```
In [66]: decision = DecisionTreeRegressor(random_state = 0)
         decision.fit(xx_train , yy_train)
         decision.score(xx_test, yy_test)

Out[66]: 0.9617917299993272
```

5.3 Random Forest

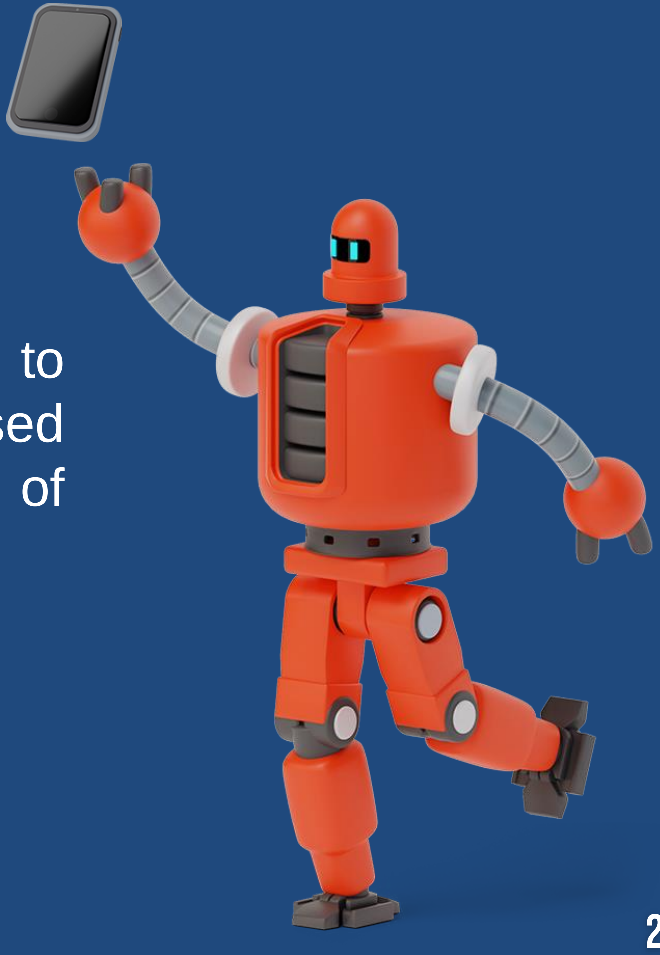
```
In [67]: random = RandomForestRegressor(n_estimators = 100, random_state = 0)
         random.fit(xx_train , yy_train)
         random.score(xx_test, yy_test)

Out[67]: 0.9622694743419838
```

After applying different models on final dataset, we found different accuracy as given below :-

Serial No.	Models	Accuracy
1	Linear Regression	0.74754507316
2	Decision Tree	0.961791729999
3	Random Forest	0.96226947434198



TESTING

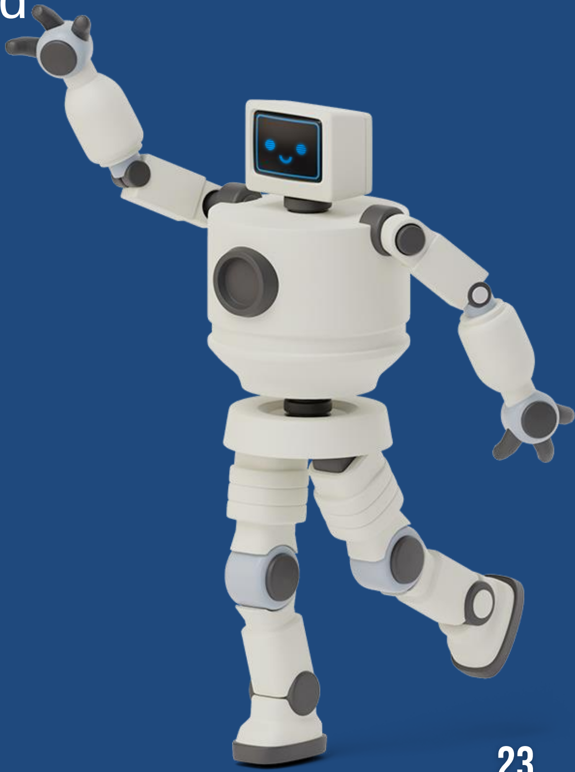


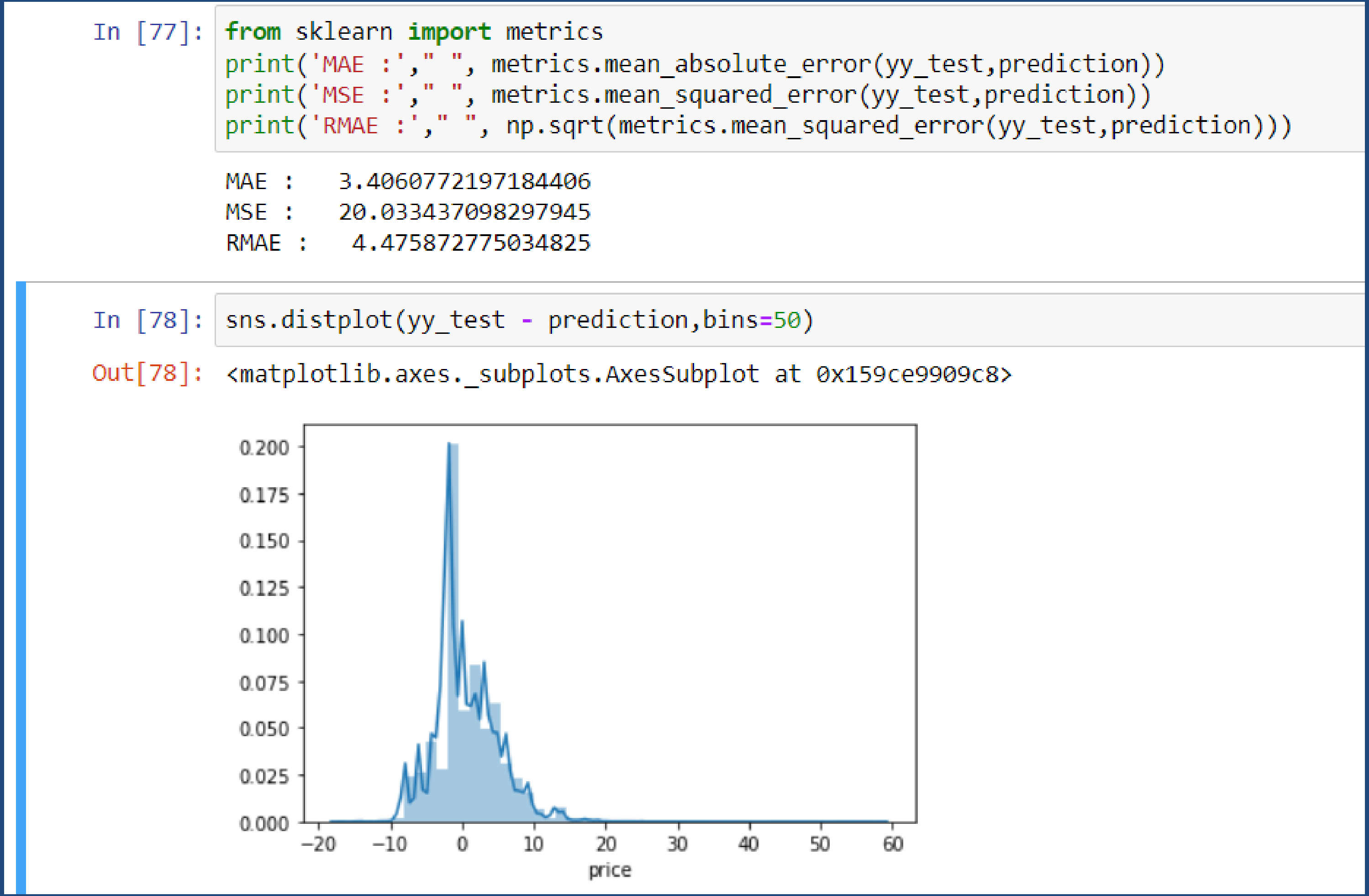
The usage of the word "testing" in relation to machine learning models is primarily used for testing the model performance in terms of accuracy/precision of the model.

With the help of linear regression, we predicted the price, plot a graph between actual and predicted values and find the following errors:-

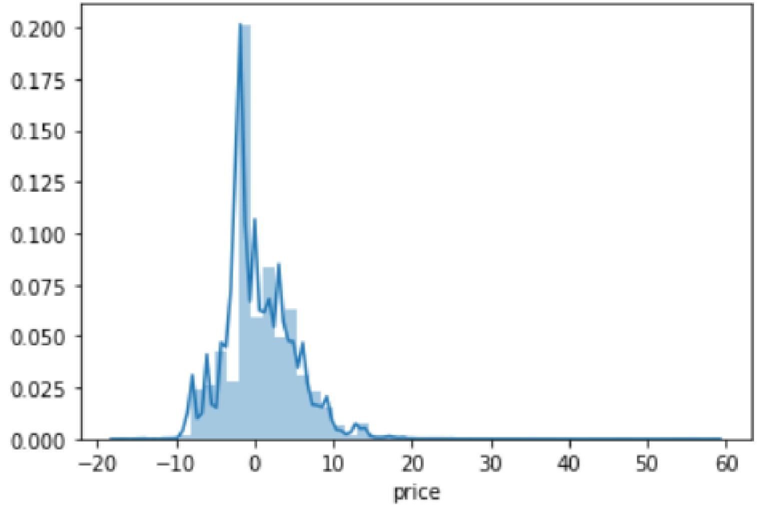
For linear regression:-

- ❖MAE : 3.406077219
- ❖MSE : 20.03343709
- ❖RMAE : 4.47587277

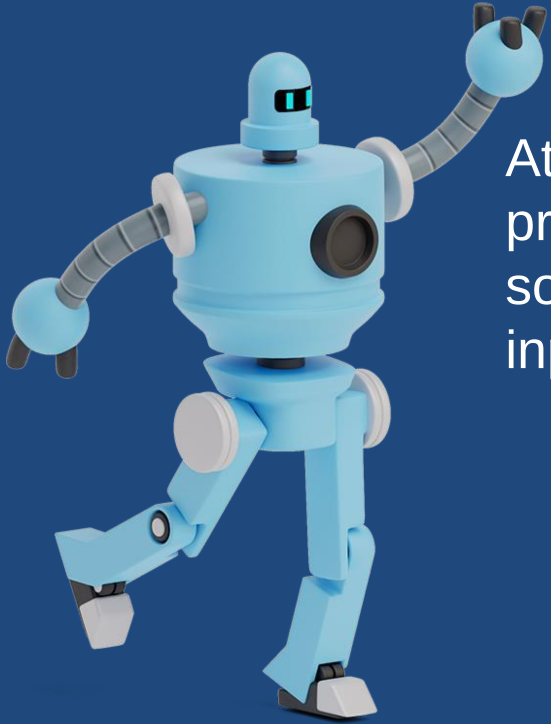




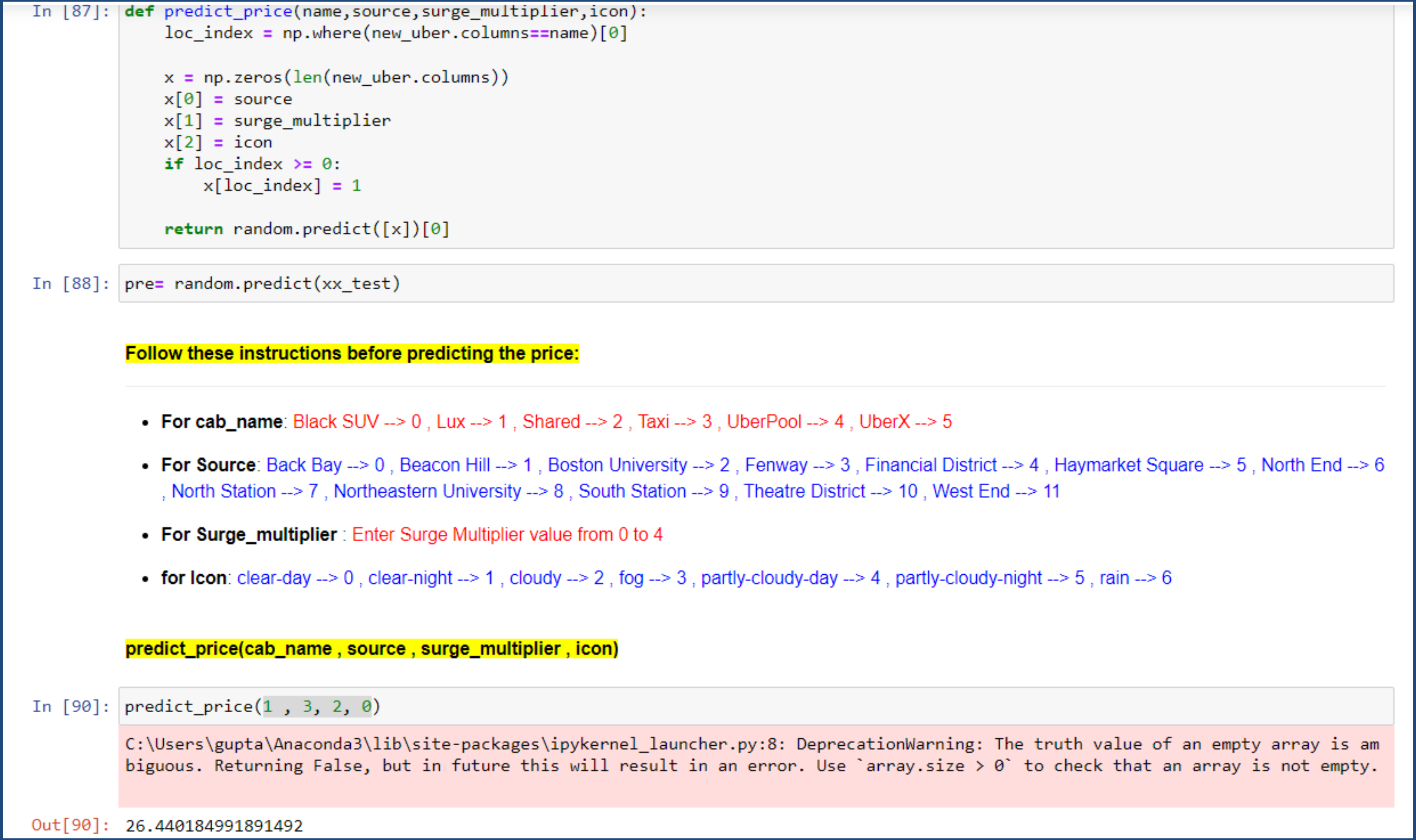
```
In [77]: from sklearn import metrics
         print('MAE :'," ", metrics.mean_absolute_error(yy_test,prediction))
         print('MSE :'," ", metrics.mean_squared_error(yy_test,prediction))
         print('RMAE :'," ", np.sqrt(metrics.mean_squared_error(yy_test,prediction)))

         MAE :   3.4060772197184406
         MSE :   20.033437098297945
         RMAE :   4.475872775034825

In [78]: sns.distplot(yy_test - prediction,bins=50)

Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x159ce9909c8>
```



PRICE PREDICTION FUNCTION



At last, we create a function for price prediction which take cab name, source, surge multiplier and icon as input and predict the price.



```
In [87]: def predict_price(name,source,surge_multiplier,icon):
             loc_index = np.where(new_uber.columns==name)[0]

             x = np.zeros(len(new_uber.columns))
             x[0] = source
             x[1] = surge_multiplier
             x[2] = icon
             if loc_index >= 0:
                 x[loc_index] = 1

             return random.predict([x])[0]
```

```
In [88]: pre= random.predict(xx_test)
```

Follow these instructions before predicting the price:

- **For cab_name**: Black SUV --> 0 , Lux --> 1 , Shared --> 2 , Taxi --> 3 , UberPool --> 4 , UberX --> 5
- **For Source**: Back Bay --> 0 , Beacon Hill --> 1 , Boston University --> 2 , Fenway --> 3 , Financial District --> 4 , Haymarket Square --> 5 , North End --> 6 , North Station --> 7 , Northeastern University --> 8 , South Station --> 9 , Theatre District --> 10 , West End --> 11
- **For Surge_multiplier** : Enter Surge Multiplier value from 0 to 4
- **for Icon**: clear-day --> 0 , clear-night --> 1 , cloudy --> 2 , fog --> 3 , partly-cloudy-day --> 4 , partly-cloudy-night --> 5 , rain --> 6

predict_price(cab_name , source , surge_multiplier , icon)

```
In [90]: predict_price(1 , 3, 2, 0)

C:\Users\gupta\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: DeprecationWarning: The truth value of an empty array is am
biguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check that an array is not empty.

Out[90]: 26.440184991891492
```

CONCLUSION :

Before working on features first we need to know about the data insights which we get to know by EDA. Apart from that, we visualize the data by drawing various plots, due to which we understand that we don’t have any data for taxi’s price, also the price variations of other cabs and different types of weather. Other value count plots show the type and amount of data the dataset has. After this, we convert all categorical values into continuous data type and fill price Nan by the median of other values. Then the most important part of feature selection came which was done with the help of recursive feature elimination. With the help of RFE, the top 25 features were selected. Among those 25 features still, there are some features which we think are not that important to predict the price, so we drop them and left with 8 important columns. We apply three different models on our remaining dataset among which Decision Tree and Random Forest prove best with 96%+ accuracy on training for our model. This means the predictive power of all these three algorithms in this dataset with the chosen features is very high but, in the end, we go with random forest because it does not prone to overfitting and design a function with the help of the same model to predict the price.

Thank you