FOOD DELIVERY TIME PREDICTION.

A Project Report

On

FOOD DELIVERY TIME PREDICTION.

By,

RUDRAKSHA PAWAR (21091033)

RAJ SEKHAR CHAKRABORTY (21091036)

Project Guide: Prof. Amruta Shinde

Master Of Science – Department of Mathematics

Savitribai Phule Pune University, Pune.

411007

ACKNOWLEDGEMENT

We express our deepest sense of gratitude, reverence, and indebtedness to our project guide **Prof. Amruta Shinde**, Professor, Department of Mathematics, SPPU, Pune. For her keen interest, ability to persist, inspire and guide with immense zeal for hard work, insistence of perfection positive approach towards the statistics.
We express our deepest sense of gratitude to **Prof. Vinayak Joshi** Head of Department of Mathematics, SPPU, Pune. For his encouragement for this project.
We would also like to acknowledge all other faculties of Department of Mathematics, SPPU, Pune.
Our friends also deserve a delightful thank for their continuous encouragement throughout the project period till completion.
Finally, we thank our parents for their moral support and blessings.

ABSTRACT

An Online Food Ordering System is proposed here which simplifies the food ordering process. The proposed system shows a user interface and update the menu with all available options so that it eases the customer work. Customer can choose more than one item to make an order and can view order details before logging off. The order confirmation is sent to the customer.

While considering the system one more thing to be focused on is delivery time. Many apps have been collecting data regarding time of delivery, personnel, vehicle, and overall conditions while a delivery is being made and providing customers with estimated time of delivery based on it.

In recent times the concern over reducing time of delivery is rising over all the delivery apps. They have been employing different methods and coming up with different strategies. They have been achieving a little success of their vision.

The study has imitated delivery time prediction or estimation with knowledge saturated through learning of concerned subjects. The collected data has been analyzed using Python and results are achieved using different machine learning models used.

INTRODUCTION

Retail **food delivery** is a service in which a restaurant, store, or independent food-delivery company delivers food to a customer. An order is typically made either through a restaurant or grocer's app, or through a food ordering company. The delivered items can include entrees, sides, drinks, desserts, or grocery items and are typically delivered in boxes or bags. The delivery person will normally drive a car, but in bigger cities where homes and restaurants are closer together, they may use bikes or motorized scooters.

The companies involved in delivery services have been keen to understand the time require to complete the delivery and achieving it in profitable margin. The data is being collected extensively for it and is being used for to achieve the considered vision of profit.

They have employing strategies or are coming up new wherever they fell short. In coming times, many apps have promised delivery in short amount of time to attract customers as well as to increase their profits.

Objectives

The primary objective of our project is to check appropriate model to data and predict the food delivery time. Our aim is to explore and understand factors responsible for prediction of delivery time.

Key Objectives.

1. To study and analyze the factors and performing exploratory data analysis.
2. To identify the factors involved in estimation of delivery time.
3. To check a machine learning model for prediction of delivery time.

Data Collection

Secondary data:

The data that is used in this project is in the form of secondary nature. The data is collected from secondary sources such as various websites.

The data has been cleaned for any missing values by removing the row observations as such.

METHODOLOGY

The first step is to study the data in detail. Exploratory data analysis was performed on data used. It is done using line plots, bar graphs, correlation heat map, pie chart, etc.

The second step is to fit an appropriate machine learning model to the data. That is to train a model and then use it to predict delivery time over test data.

Linear prediction and random forest algorithm have been trained and used for prediction.

Bar Graph:

Bar charts should be used **when you are showing segments of information**. Vertical bar charts are useful to compare different categorical or discrete variables.

Line Plots:

A line plot is a graph that shows frequency of data along a number line. It is a quick, simple way to organize data.

Correlation Heatmaps:

Correlation heatmaps can be used to find potential relationships between variables and to understand the strength of these relationships.

Pie Charts:

A pie chart helps organize and show data as a percentage of a whole. True to the name, this kind of visualization uses a circle to represent the whole, and slices of that circle, or "pie", to represent the specific categories that compose the whole.

Linear Regression Model:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.

Each feature variable must model the linear relationship with the dependent variable.

MLR tries to fit a regression line through a multidimensional space of data-points.

Random Forest Algorithm:

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.

A random forest regressor. A random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Random forests can handle categorical and continuous variables equally well. It can automatically handle missing values. Random forest can handle missing values automatically

It takes less training time as compared to other algorithms.

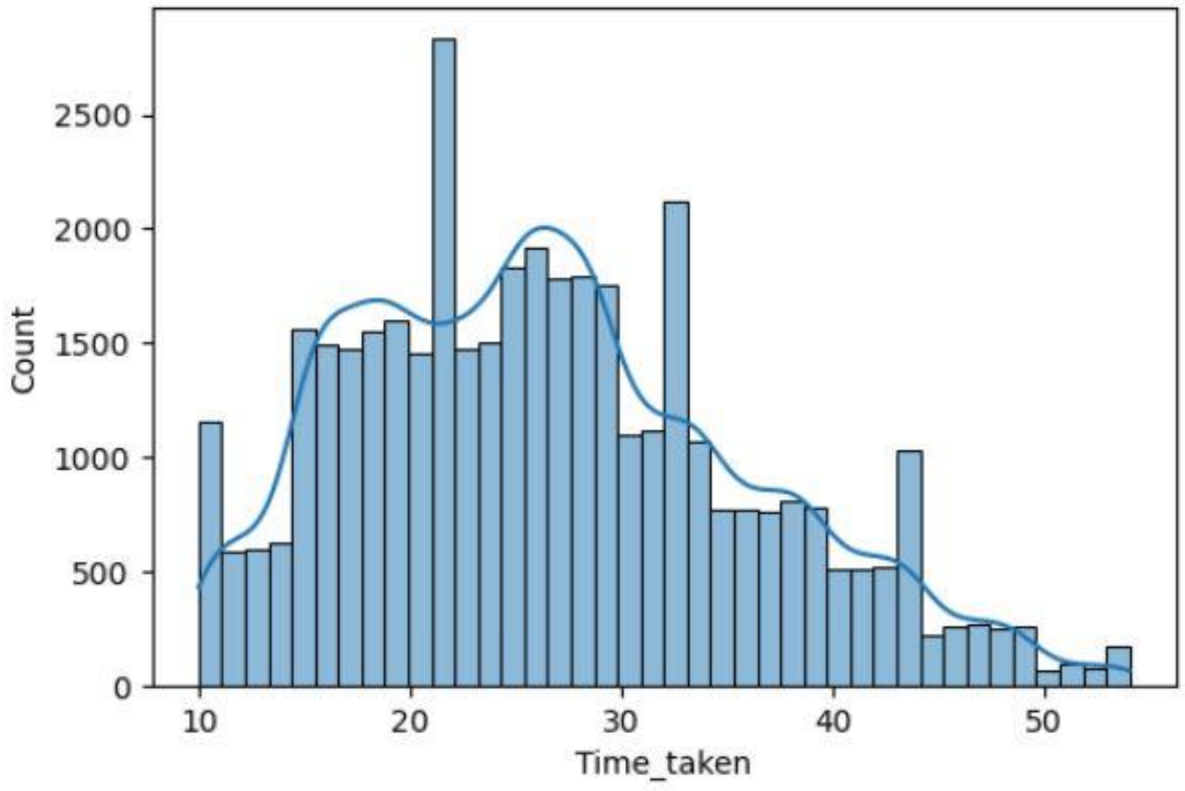
It predicts output with high accuracy, even for the large dataset it runs efficiently.

It can also maintain accuracy when a large proportion of data is missing.

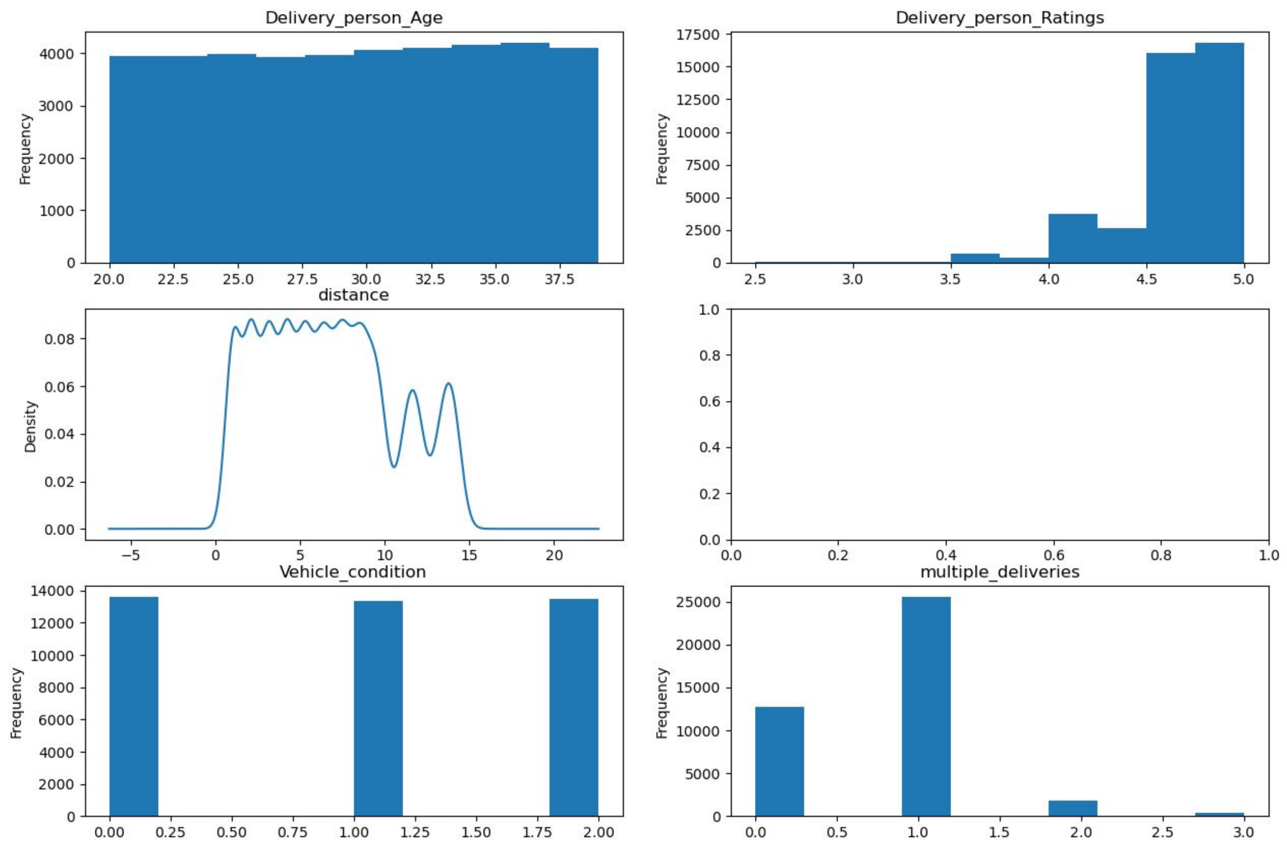
Data Visualization

Exploratory data analysis :--

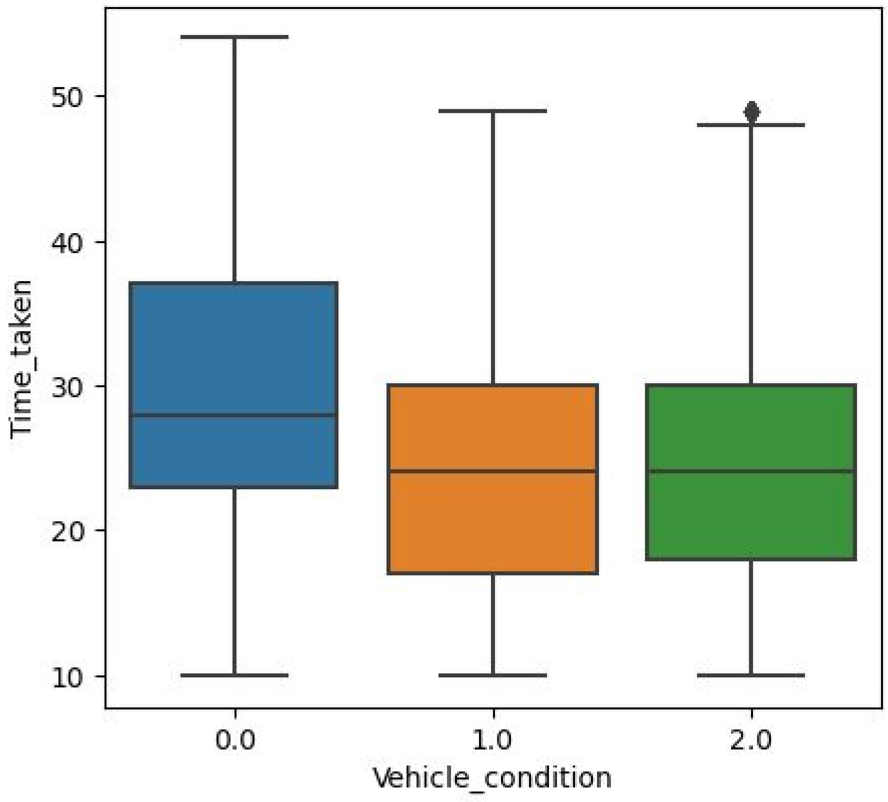
Histogram:

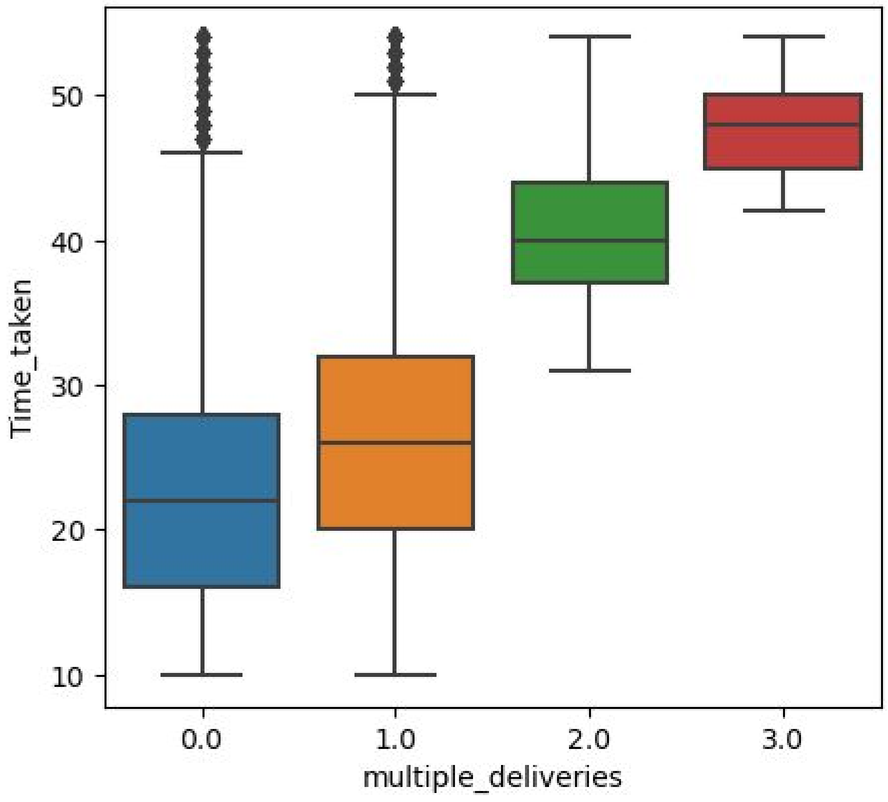


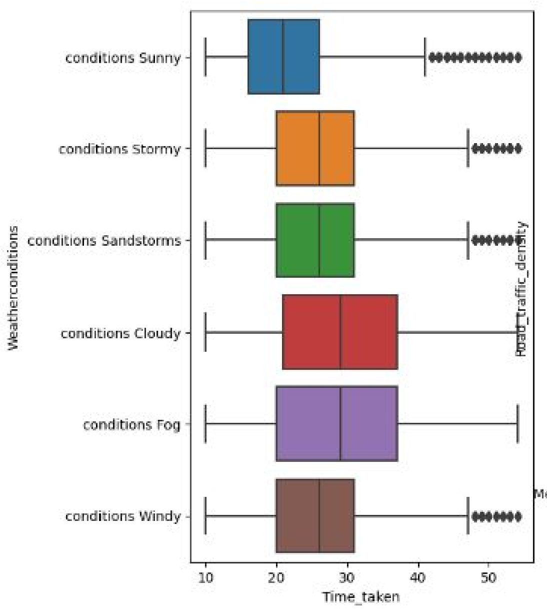
Interpretation : From this histogram we see that delivery time mostly from 15 min to 30 min

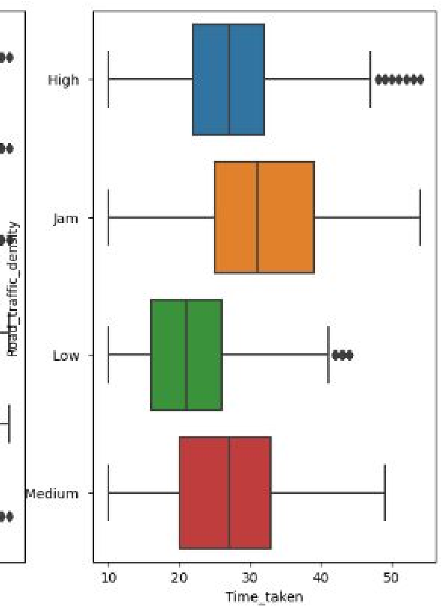

Delivery_person_Age
Frequency
Delivery_person_Ratings
Frequency
distance
Density
Vehicle_condition
Frequency
multiple_deliveries
Frequency

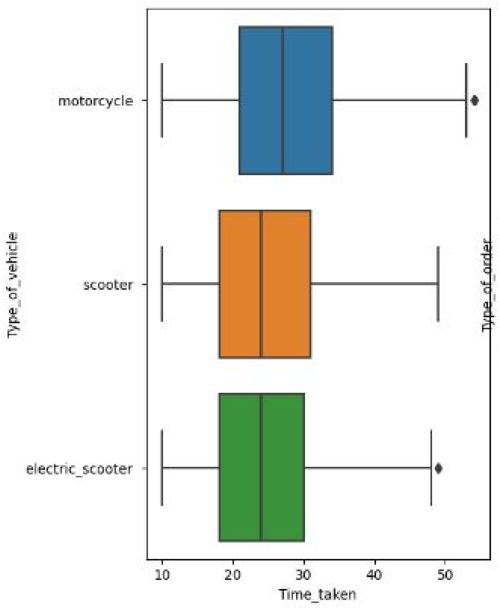
Box Plot

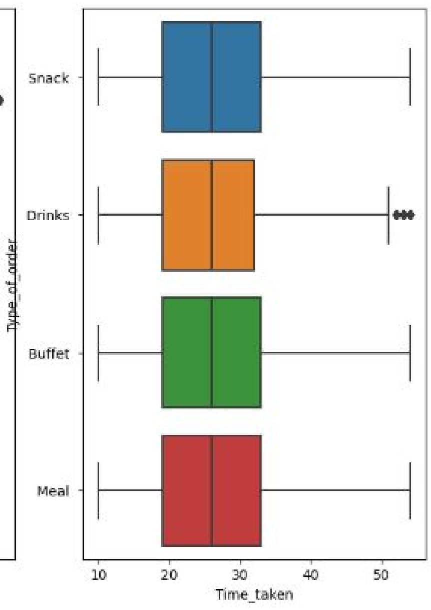

Time_taken
50
40
30
20
10
0.0
1.0
2.0
Vehicle_condition

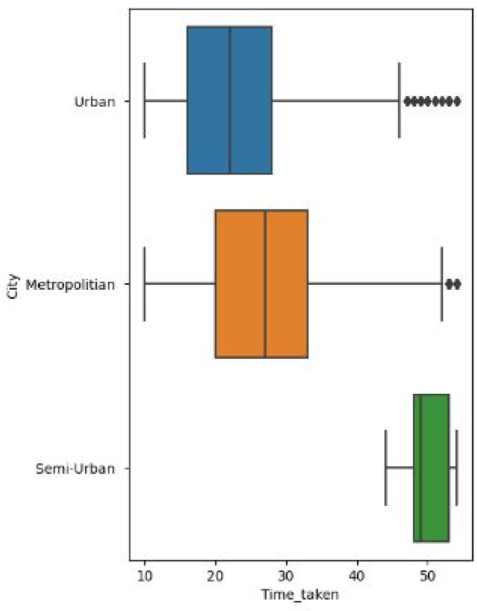

Time_taken
50
40
30
20
10
0.0
1.0
2.0
3.0
multiple_deliveries

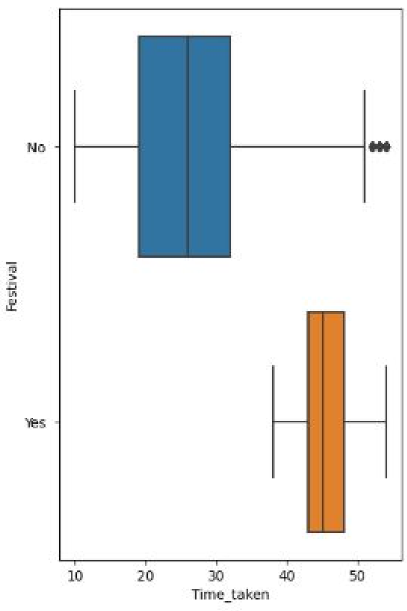

Weatherconditions
conditions Sunny
conditions Stormy
conditions Sandstorms
conditions Cloudy
conditions Fog
conditions Windy
10
20
30
40
50
Time_taken


Road_traffic_density
High
Jam
Low
Medium
10
20
30
40
50
Time_taken

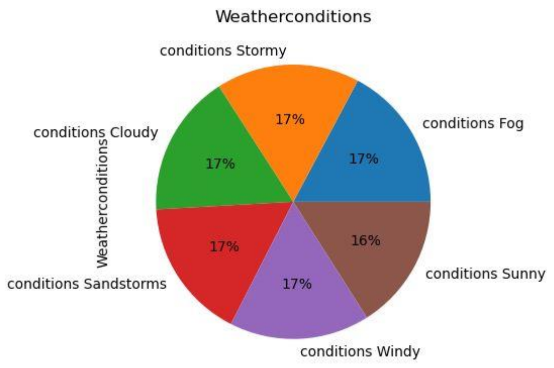

Type_of_vehicle
motorcycle
scooter
electric_scooter
10
20
30
40
50
Time_taken

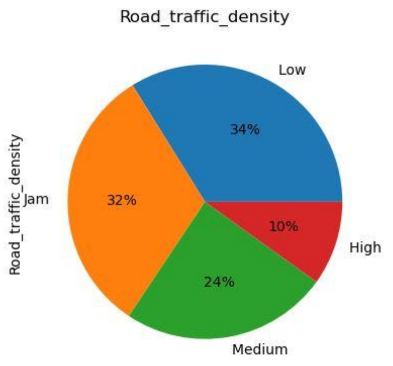

Type_of_order
Snack
Drinks
Buffet
Meal
10
20
30
40
50
Time_taken

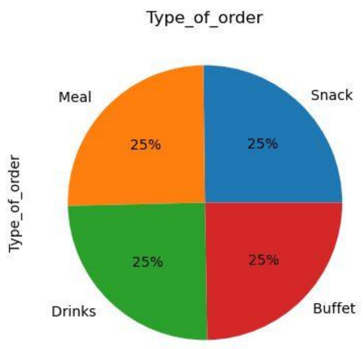

City
Urban
Metropolitian
Semi-Urban
10
20
30
40
50
Time_taken

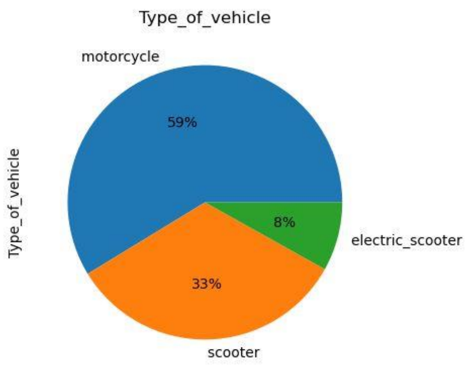

Festival
No
Yes
10
20
30
40
50
Time_taken

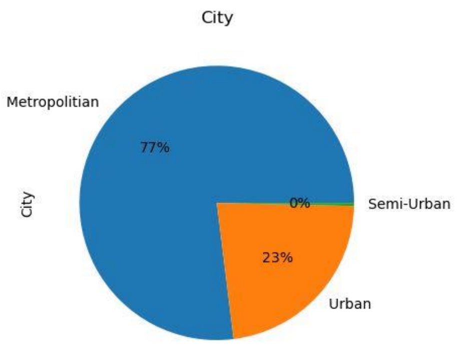
Pie Chart

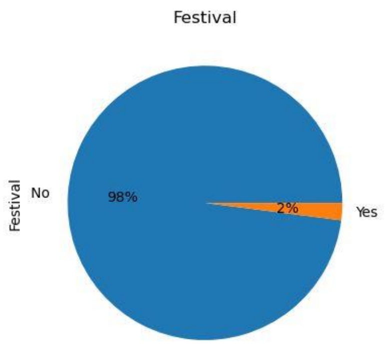

Weatherconditions
Weatherconditions
conditions Stormy
17%
conditions Fog
17%
conditions Cloudy
17%
conditions Sandstorms
17%
conditions Windy
17%
conditions Sunny
16%


Road_traffic_density
Road_traffic_density
Low
34%
Jam
32%
Medium
24%
High
10%

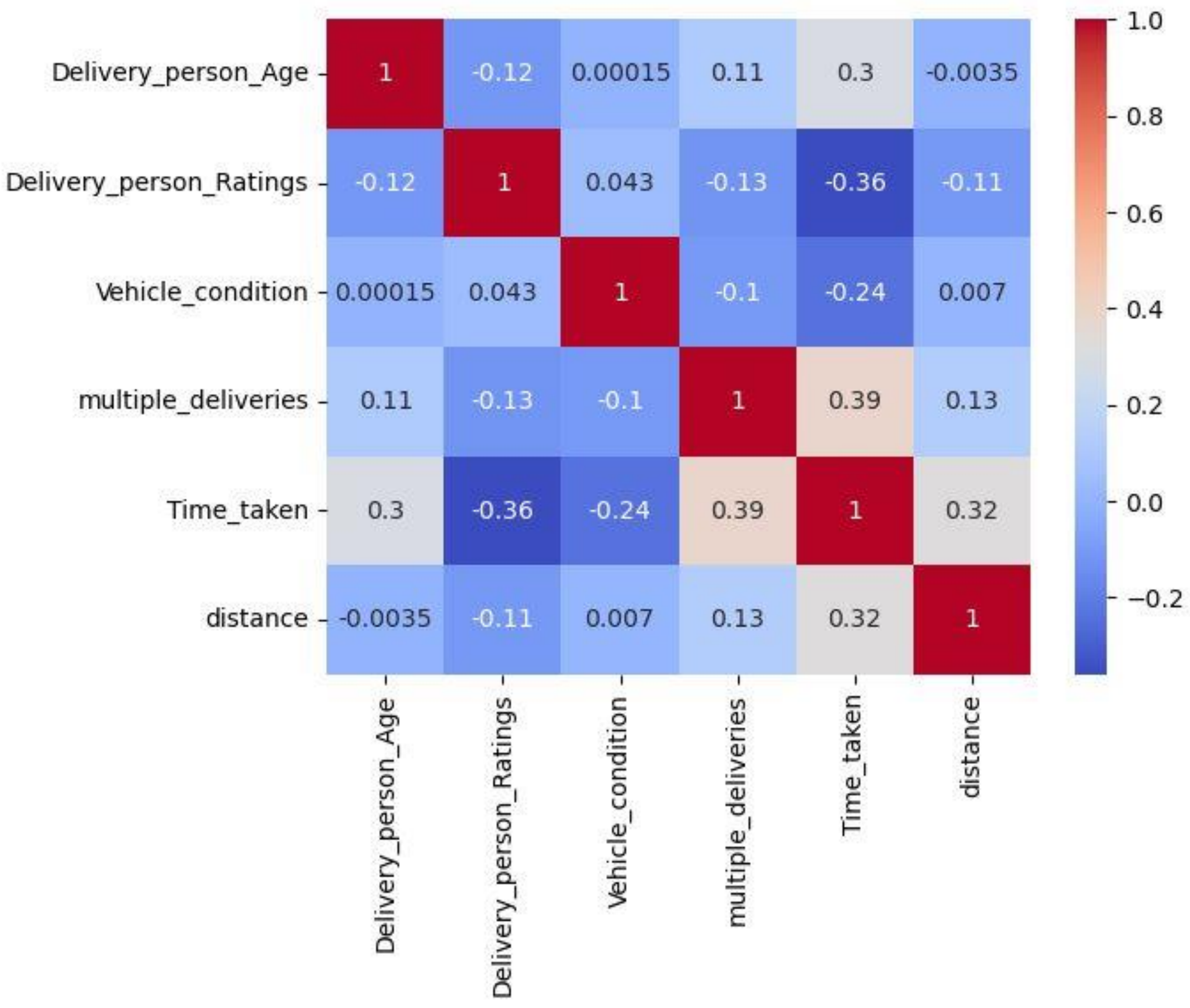

Type_of_order
Type_of_order
Snack
25%
Meal
25%
Drinks
25%
Buffet
25%


Type_of_vehicle
Type_of_vehicle
motorcycle
59%
scooter
33%
electric_scooter
8%

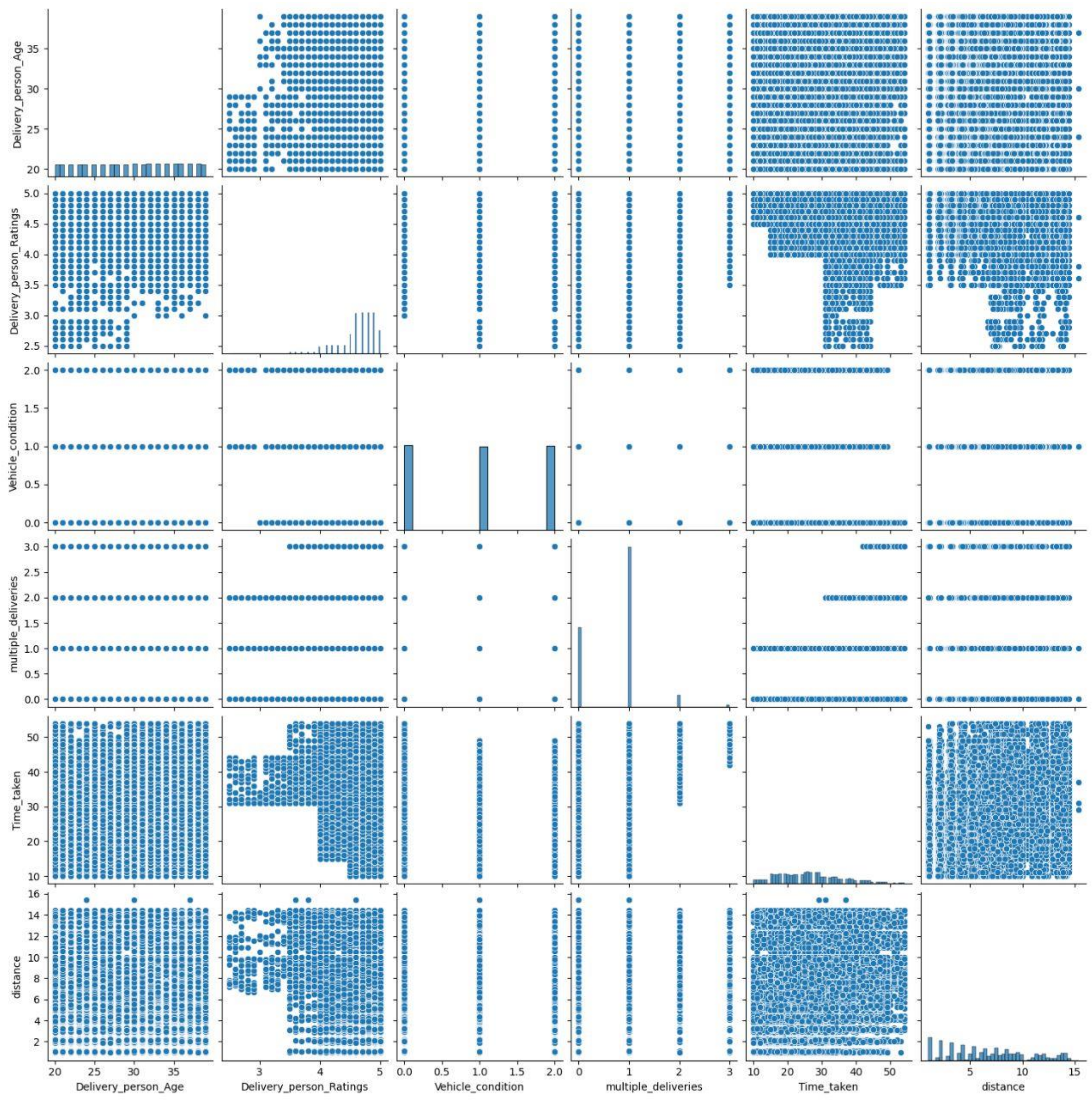

City
City
Metropolitian
77%
Semi-Urban
0%
Urban
23%


Festival
Festival
No
98%
Yes
2%

Correlation Heatmap:

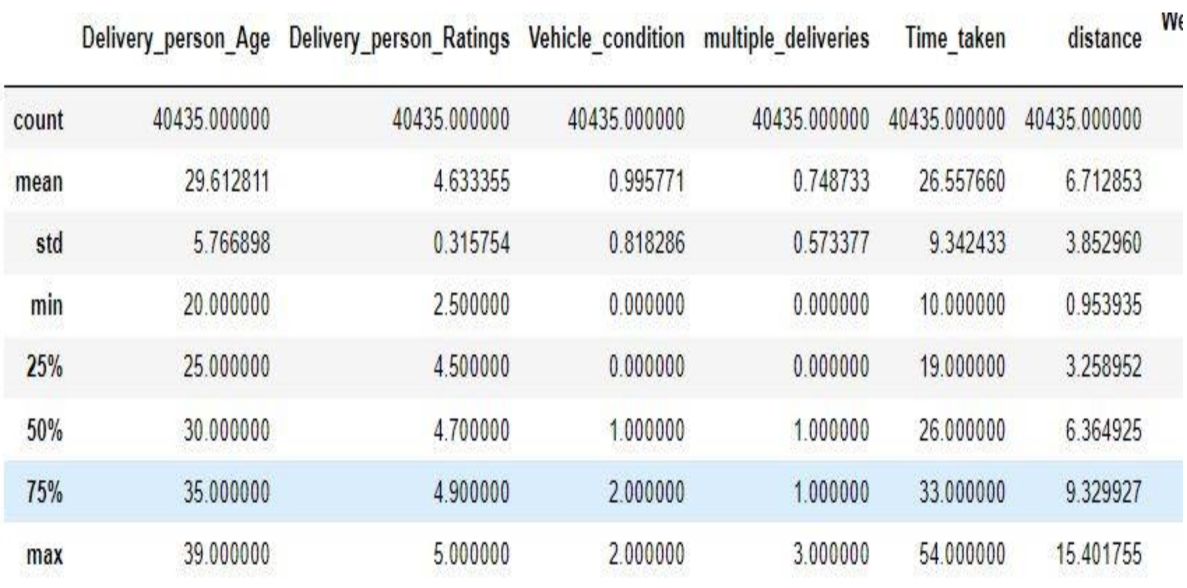


Pair plot


Delivery_person_Age
Delivery_person_Ratings
Vehicle_condition
multiple_deliveries
Time_taken
distance

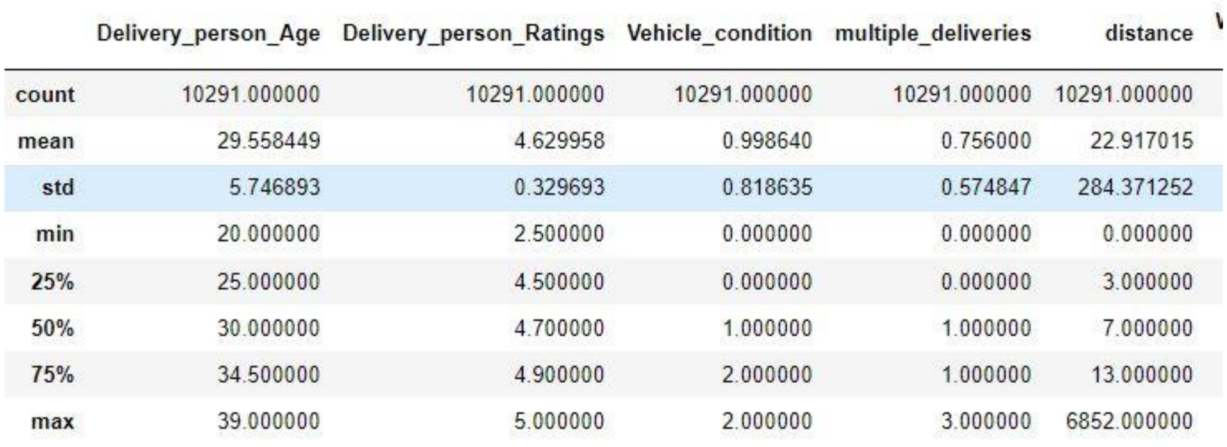
Data Description:

A)Training Data



	Delivery_person_Age	Delivery_person_Ratings	Vehicle_condition	multiple_deliveries	Time_taken	distance	We
count	40435.000000	40435.000000	40435.000000	40435.000000	40435.000000	40435.000000	
mean	29.612811	4.633355	0.995771	0.748733	26.557660	6.712853	
std	5.766898	0.315754	0.818286	0.573377	9.342433	3.852960	
min	20.000000	2.500000	0.000000	0.000000	10.000000	0.953935	
25%	25.000000	4.500000	0.000000	0.000000	19.000000	3.258952	
50%	30.000000	4.700000	1.000000	1.000000	26.000000	6.364925	
75%	35.000000	4.900000	2.000000	1.000000	33.000000	9.329927	
max	39.000000	5.000000	2.000000	3.000000	54.000000	15.401755	

B) Testing Data



	Delivery_person_Age	Delivery_person_Ratings	Vehicle_condition	multiple_deliveries	distance	V
count	10291.000000	10291.000000	10291.000000	10291.000000	10291.000000	
mean	29.558449	4.629958	0.998640	0.756000	22.917015	
std	5.746893	0.329693	0.818635	0.574847	284.371252	
min	20.000000	2.500000	0.000000	0.000000	0.000000	
25%	25.000000	4.500000	0.000000	0.000000	3.000000	
50%	30.000000	4.700000	1.000000	1.000000	7.000000	
75%	34.500000	4.900000	2.000000	1.000000	13.000000	
max	39.000000	5.000000	2.000000	3.000000	6852.000000	

Describing Train Data:

DATA PRE-PROCESSING

The libraries we took help are as follows:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import metrics

from sklearn.model_selection import train_test_split, learning_curve

from sklearn.ensemble import RandomForestRegressor

from xgboost.sklearn import XGBRegressor

from sklearn.ensemble import ExtraTreesRegressor

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error,

mean_absolute_error,r2_score
```

Pandas – is mainly used for manipulating data like cleaning for missing values, analysing, and exploring.

NumPy - it allows you to work with high-performance multidimensional array objects and it also provides tools for working with these arrays.

Matplotlib – is used for dynamic visualization of data.

Scikit-learn / sklearn - contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction.

Seaborn - is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

a)Data Cleaning:

i)For finding the Null or NaN values:

```
for i in train.columns:

    train[i].loc[train[i] == 'NaN '] = np.nan

    train[i].loc[train[i] == 'NaN'] = np.nan


for j in test.columns:

    test[j].loc[test[j] == 'NaN '] = np.nan

    test[j].loc[test[j] == 'NaN'] = np.nan
```

ii)For removing null values or NaN values:

```
train.dropna(subset=['Time_Orderd'], axis=0, inplace=True)

test.dropna(subset=['Time_Orderd'], axis=0, inplace=True)
```

dropna() function from Pandas was used to drop columns containing NaN values.

```
train.dropna(axis=0, inplace=True)


test.dropna(axis=0, inplace=True)
```

Now we get no null values for train and test as follows:

Train:

```
ID                             0
Delivery_person_ID             0
Delivery_person_Age            0
Delivery_person_Ratings        0
Restaurant_latitude            0
Restaurant_longitude           0
```

```
Delivery_location_latitude     0
Delivery_location_longitude    0
Order_Date                     0
Time_Orderd                    0
Time_Order_picked              0
Weatherconditions              0
Road_traffic_density           0
Vehicle_condition              0
Type_of_order                  0
Type_of_vehicle                0
multiple_deliveries            0
Festival                       0
City                           0
Time_taken                     0
dtype: int64
```

Test:

```
ID                             0
Delivery_person_ID             0
Delivery_person_Age            0
Delivery_person_Ratings        0
Restaurant_latitude            0
Restaurant_longitude           0
Delivery_location_latitude     0
Delivery_location_longitude    0
Order_Date                     0
Time_Orderd                    0
Time_Order_picked              0
Weatherconditions              0
Road_traffic_density           0
Vehicle_condition              0
Type_of_order                  0
Type_of_vehicle                0
multiple_deliveries            0
Festival                       0
City                           0
dtype: int64
```

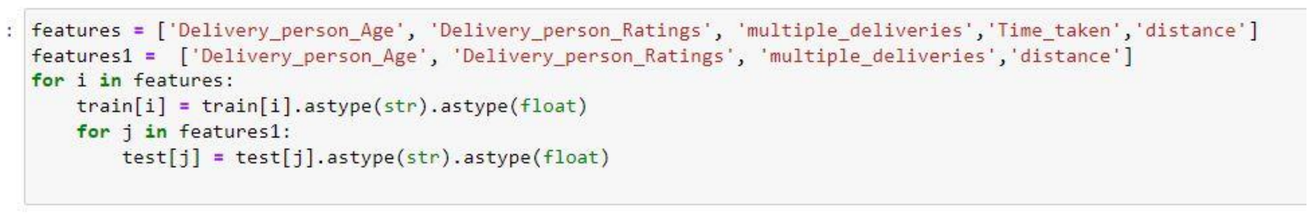
b) Data Reduction:-

Attribute Subset Selection –

The highly relevant attributes should be used, rest all can be discarded. Variable selection method had been tried but the p-values of each features is very negligible which is difficult to compare with so the proper method has to be applied with proper visualisations to see which features are important. For seeing this we had applied feature importance method i.e

---Feature importance method:-

feature importance is a measure of how much influence a specific predictor variable (a feature) has on the accuracy of the model prediction. You can use feature importance to prune your model and reduce overfitting by dropping the low performing features.

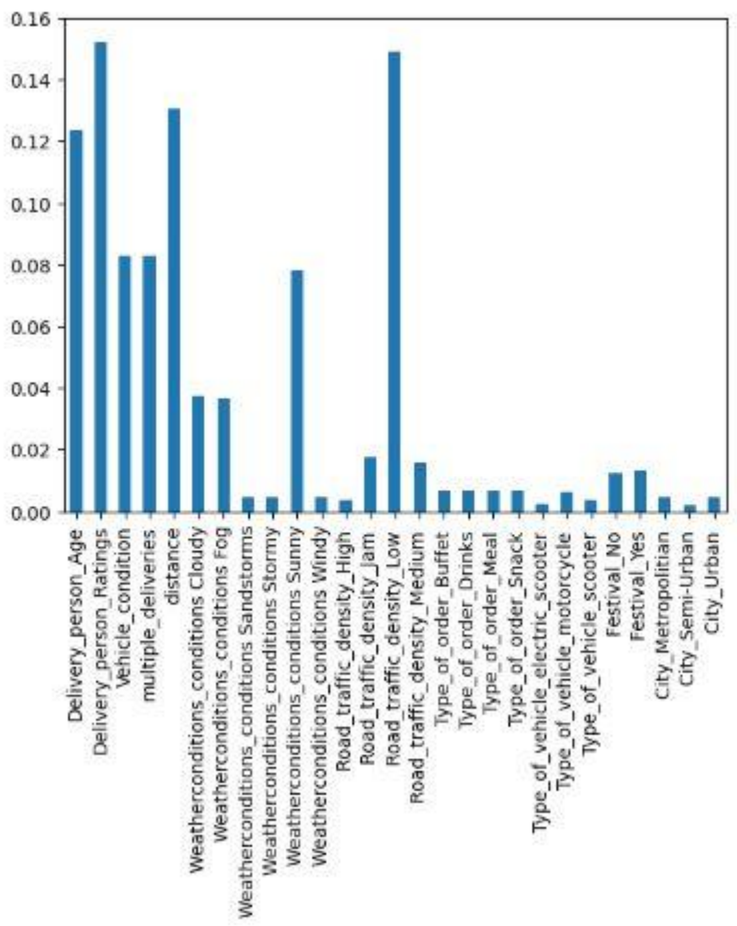


```
features = ['Delivery_person_Age', 'Delivery_person_Ratings', 'multiple_deliveries','Time_taken','distance']
features1 =  ['Delivery_person_Age', 'Delivery_person_Ratings', 'multiple_deliveries','distance']
for i in features:
    train[i] = train[i].astype(str).astype(float)
    for j in features1:
        test[j] = test[j].astype(str).astype(float)
```

It can be used for the continuous as well as categorical data

```
from sklearn.ensemble import ExtraTreesRegressor

model=ExtraTreesRegressor()

model.fit(x,y)

model.feature_importances_

s=pd.Series(model.feature_importances_,index=x.columns)

s.plot(kind='bar')
```

Here we get the graph which shows the importance of the features:



This bar graph shows that how much each features are important by using the probabilities. Here we can see all the features except the latitude ,longitude, ID, restaurant Id ,time of order and delivery,age location are not important at all as they have small probabilities so we removed.

```
Train.drop([‘ID’,’Delivery_person_ID',’Restaurant_latitude','
Restaurant_longitude','Delivery_location_latitude','Delivery_
location_longitude','Order_Date','Time_Orderd','Time_Order_pi
cked'],axis=1,inplace=True)


test.drop(['ID','Delivery_person_ID','Restaurant_latitude','R
estaurant_longitude','Delivery_location_latitude','Delivery_l
ocation_longitude','Order_Date','Time_Orderd','Time_Order_pic
ked'],axis=1,inplace=True)
```

---- Features in the model important are:

1)Delivery_person_age

2)Delivery_person_ratings

3)Vehicle condition

4)Multiple Deliveries

5)Weathercondition

6)Road Traffic

7)Type of Order

8)Type of vehicle

9)Festival(yes/no)

10)City

11)Distance

MODEL SELECTION

Here we have fitted two models to see which fits the best these two models are:

1)Linear Regression

2)Random Forest

Let's see how we make model selection ---

First of all, we have to make train and test splitting for finding the function on the training data and see if the model works on the testing data. It follows as—

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=

0.2,random_state=0)
```

1)Linear Regression: -

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables.

For fitting the model:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()

lr.fit(x_train,y_train)

lr.coef_

lr.intercept_

lr.score(x_train,y_train)
lr.score(x_test,y_test)
y_pred=lr.predict(x_test)
y_pred
```

Intercept:- 54.59904710850824

Lr.score of train :- 0.5981211799912576

Lr.score of test :- 0.5980667969245608

Therefore here we can conclude that no overfit or underfit of the data is present as the lr score of both train and test data is almost same.

For checking the accuracy of the model on the test data:

```
from sklearn.metrics import mean_squared_error,

mean_absolute_error,r2_score

MAE=mean_absolute_error(y_test,y_pred)

RMSE=np.sqrt(mean_squared_error(y_test,y_pred))

r2_score(y_test,y_pred)
```

R2score that is coefficient of determination of the model is 59.8%, that is 59.8% of the model is effective.

Hence, Accuracy of trained and testing model is 59.8% and 59.8 % respectively.

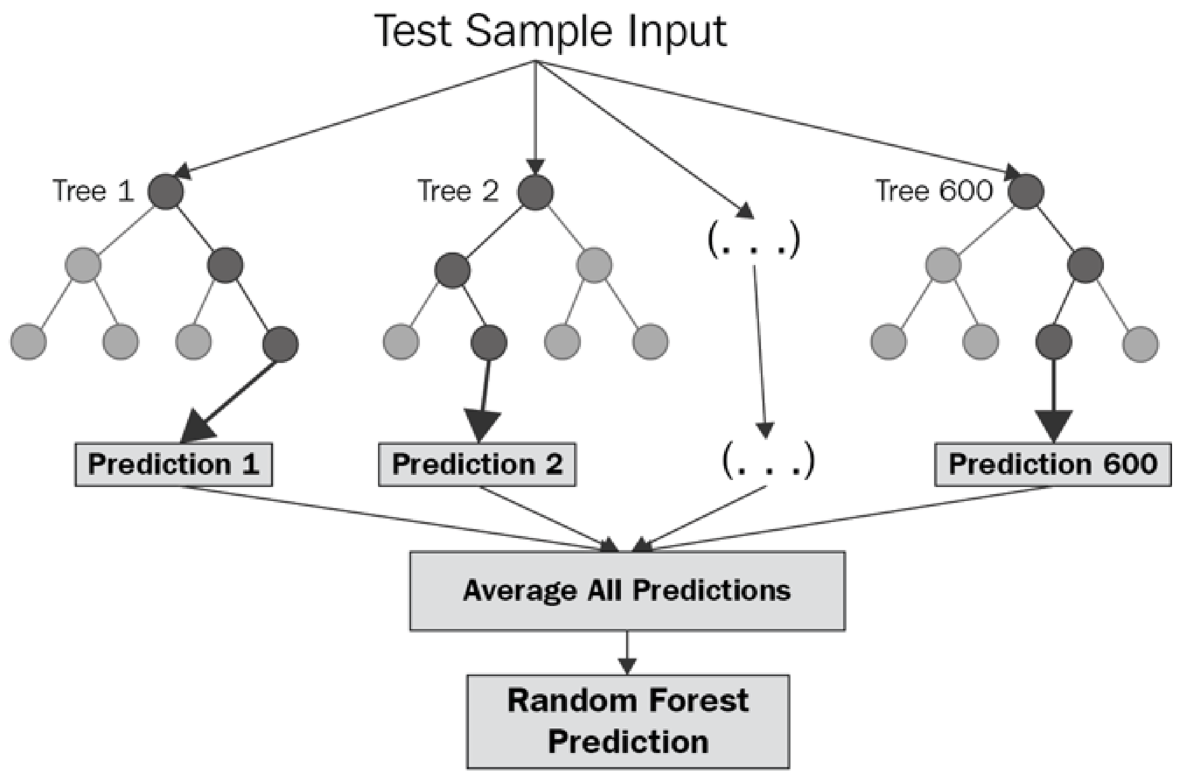
2)**Random Forest:**

What is random forest?

Random forest is a commonly-used machine learning algorithm which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

Decision trees

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes, and leaf nodes.



In our data Random Forest is also applied to make comparison which model fits well the data. It follows as following.

```
x = train. Drop('Time taken', axis=1)
y = train['Time taken']
```

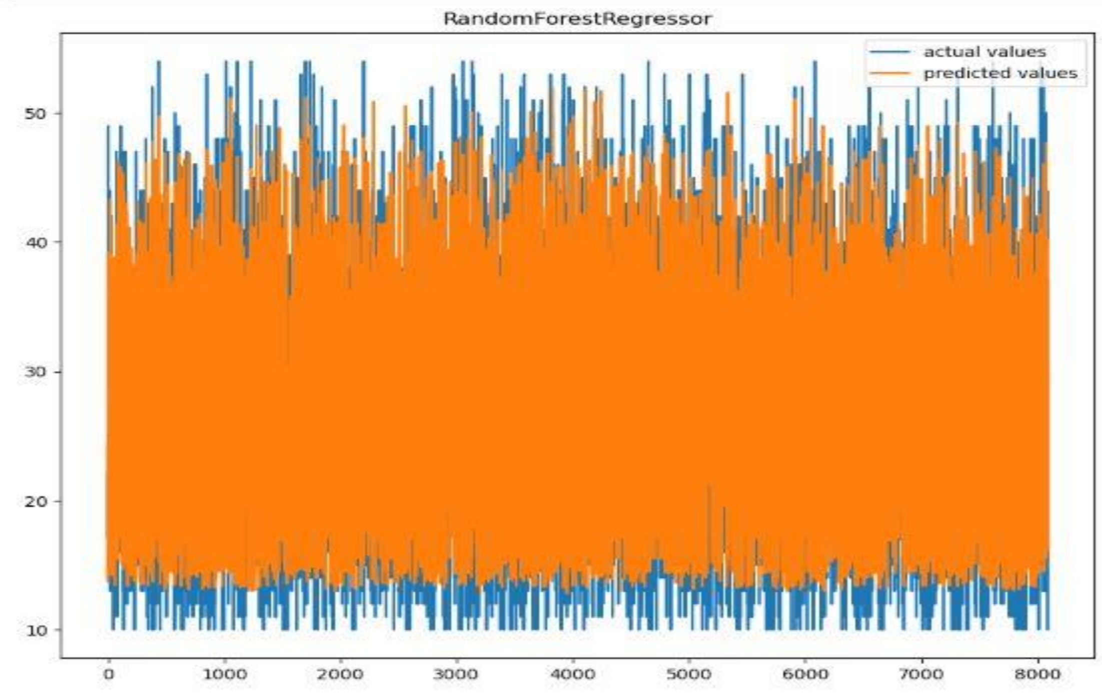
Fitting the Random Forest Model: -

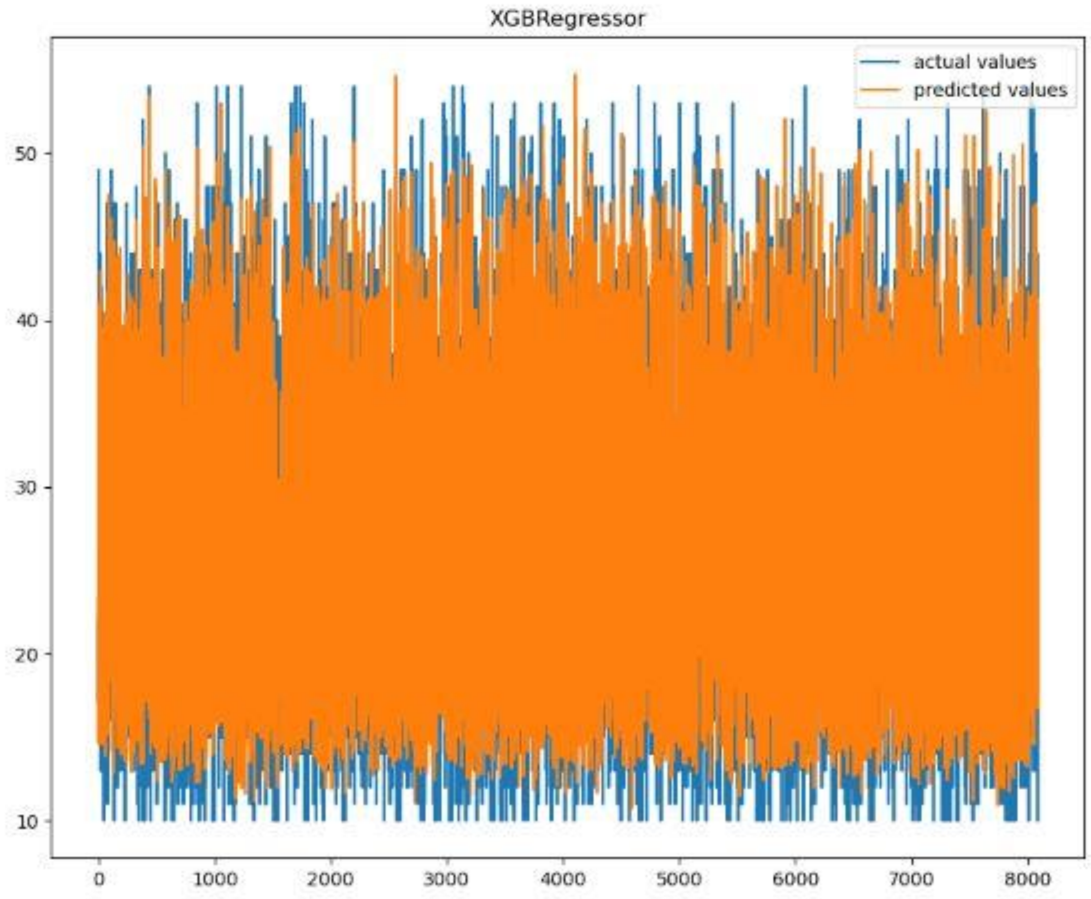
```
rf = RandomForestRegressor(max_depth=10, estimators=1000, random

state=123).fit(x_train, y_trian)

xgb=

XGBRegressor(n_estimators=1000,learning_rate=0.1,alpha=0.1,seed=123).fit(

x_train, y_trian)

rf_pred = rf_pred(x_test)

xgb_predict= xgb.predict(x_test)
```

For checking the accuracy for the model using rmse and r2 score:

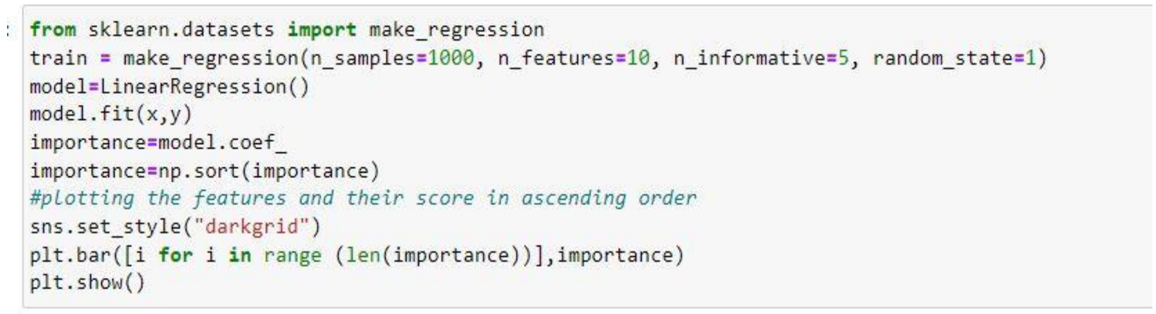
```
def rmse (pred, title):

    mse = metrics.mean_squared_error(pred, y_test)

    rmse = np.sqrt(mse)

    print('RMSE of '+ title +str(':'))

    print(np.round(rmse,3))

rmse(rf_pred, 'RandomForestRegressor')

rmse(xgb_pred, 'XGBRegressor')


def r2(pred, title):

    r2 = metrics.r2_score(pred, y_test)

    print('R2 of ' + title + str(':'))

    print(r2)

r2(rf_pred, 'RandomForestRegressor')

r2(xgb_pred, 'XGBRegressor')
```

Rmse for the pred value is 3.83 and for xgb regressor is 3.838.
R2 score for pred value is 79.5% and for xgb regressor is 80%.

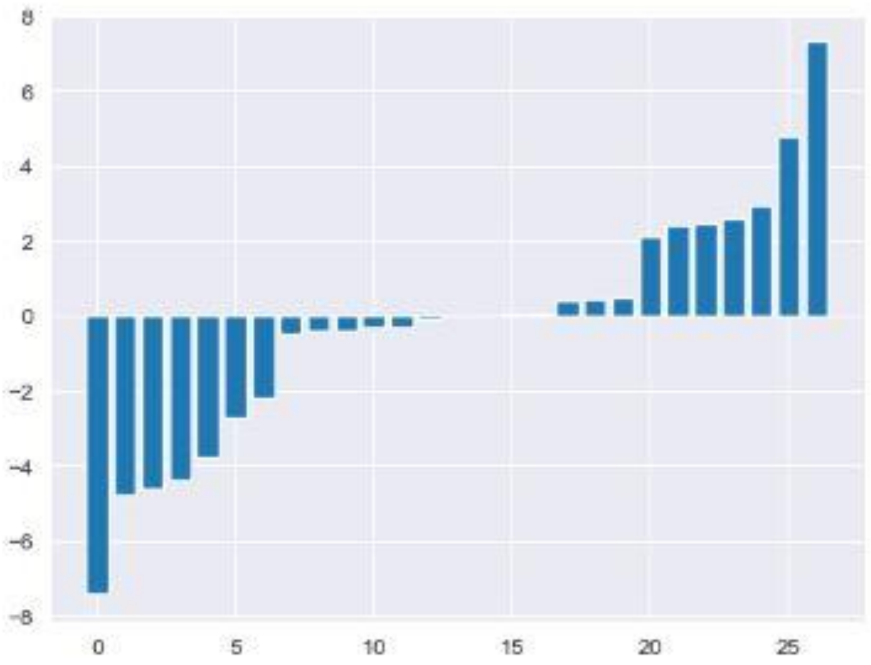

RandomForestRegressor
actual values
predicted values
10
20
30
40
50
0
1000
2000
3000
4000
5000
6000
7000
8000


XGBRegressor
actual values
predicted values
10
20
30
40
50
0
1000
2000
3000
4000
5000
6000
7000
8000

Plotting the features and their score in ascending order



```
from sklearn.datasets import make_regression
train = make_regression(n_samples=1000, n_features=10, n_informative=5, random_state=1)
model=LinearRegression()
model.fit(x,y)
importance=model.coef_
importance=np.sort(importance)
#plotting the features and their score in ascending order
sns.set_style("darkgrid")
plt.bar([i for i in range (len(importance))],importance)
plt.show()
```



Model Selection

The process of selecting the machine learning model most appropriate for a given issue is known as model selection.

Model selection is a procedure that may be used to compare models of the same type that have been set up with various model hyperparameters and models of other types. The R2 score is one of the performance evaluations measures for regression-based machine learning models. It is also known as the coefficient of determination.

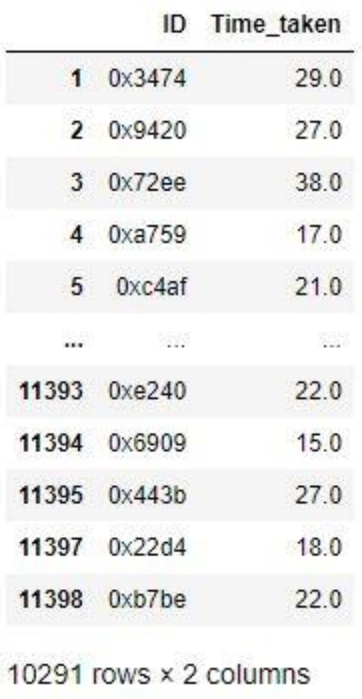
Here we can see above r2 score for the Linear regression is 59% which is smaller than the Random Forest model whose R2 score is 80%.

DATA PREDICTION

Hence the Random Forest model is selected and now the prediction for the testing data must be done which can be done as:

```
pred_result = np.round(xgb.predict(test))

result = pd.DataFrame({'ID':ID, 'Time_taken': pred_result})

result
```

We get the predicted values as:

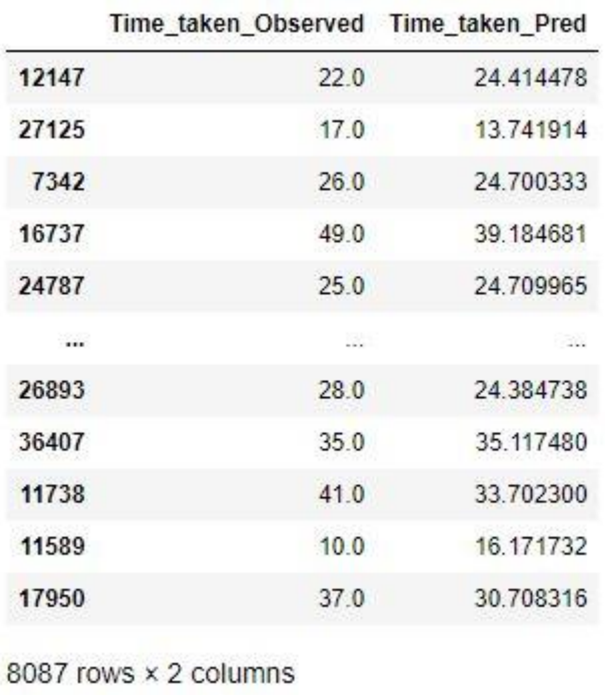


	ID	Time_taken
1	0x3474	29.0
2	0x9420	27.0
3	0x72ee	38.0
4	0xa759	17.0
5	0xc4af	21.0
...	...	...
11393	0xe240	22.0
11394	0x6909	15.0
11395	0x443b	27.0
11397	0x22d4	18.0
11398	0xb7be	22.0

10291 rows × 2 columns

As the data is large only head and tailed values can be seen.

The comparison of the predicted values and the given values can be seen:



	Time_taken_Observed	Time_taken_Pred
12147	22.0	24.414478
27125	17.0	13.741914
7342	26.0	24.700333
16737	49.0	39.184681
24787	25.0	24.709965
...	...	...
26893	28.0	24.384738
36407	35.0	35.117480
11738	41.0	33.702300
11589	10.0	16.171732
17950	37.0	30.708316

8087 rows × 2 columns

Here we didn't get the appropriate results which is limitation of our project. Observed value is 22 min and predicted is 24 min approx.
2 min error contains where a limitation is with time. We didn't have the error more than 10 min.

Total Data File: https://github.com/parharx/food-delivery-time

CONCLUSION

The factors like delivery persons age, traffic conditions, weather conditions, multiple deliveries, type of order, delivery person rating, type of vehicles, festival is there or not, type of city(urban ,rural, semi urban) are contributing towards the delivery time. Older personnel, personnel with multiple deliveries, High traffic and bad weather conditions leads to more time in delivery completion.

After using models – Linear Regression and Random Forest Regression and checking accuracies we have found out that the accuracy for Linear Regression is 59% while for Random Forest Regression it comes out to be 80%. By using the model of Random Forest further the prediction has been done for the testing data. With less error as possible. More precise data has to been studied for an accurate prediction of the delivery time.