Food Delivery Time Prediction

Data Science With Python Lab Project Report

Bachelor

in

Computer Science

By

KRISHNA VAMSI PAPPALA

AND

CHERUKURI PRAVEEN KUMAR

S200079

S201075

Rajiv Gandhi University Of Knowledge And Technologies

S.M. Puram , Srikakulam -532410

Andhra Pradesh, India

Abstract

Time of delivery plays a vital role in any food delivery platform. Every customer wants to know how long it will take to receive their food after placing an order. This project addresses the concept of Food Delivery Time Prediction. The dataset we have chosen contains records of food delivery timings from the last two years. Our primary objective for this project is to accurately predict the delivery time for a given order. Several factors influence delivery time, including geographical locations, distance ,speed ,common routes between places, and the type of cuisine ordered. The dataset, sourced from Kaggle, includes attributes such as the name of the cuisine, and the latitude and longitude of both the restaurant and the order location. By analyzing this data, we aim to develop a model that can predict delivery times with high accuracy. This will help food delivery platforms enhance their service by providing more precise delivery time estimates, thereby improving customer satisfaction. Moreover, understanding the factors that affect delivery time can help optimize delivery routes and strategies, ultimately leading to a more efficient and reliable service. Our main agenda is to calculate the efficient delivery time by leveraging machine learning techniques. By doing so, we hope to contribute to the optimization of food delivery logistics,benefiting both the platform owners and the customers.

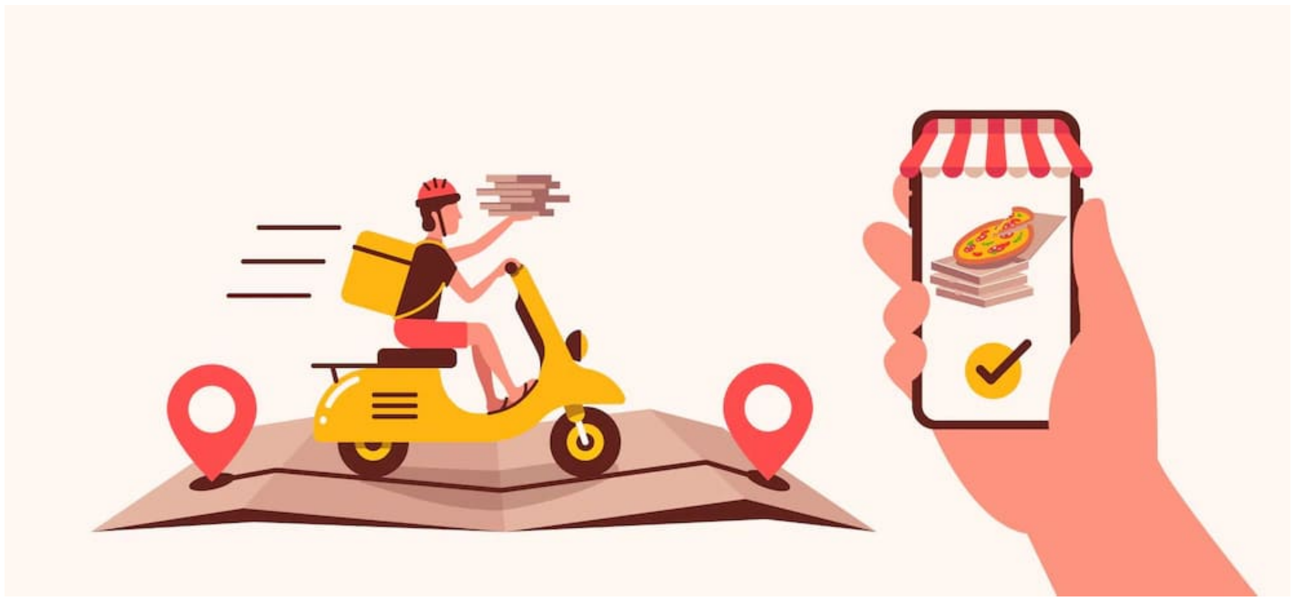
Contents

Chapter 1

Introduction

1.1 Introduction to Your Project

Time prediction of delivery is essential in any product delivery platform. If the product is delivered in a short period, it will give a positive turn in profits. It is a real-life existing project. Some platforms are unable to deliver their items on time, leading to customer dissatisfaction. Our goal is to develop a system that not only estimates delivery times reliably but also enhances the overall efficiency of food delivery operations. These models take into various factors such as customer location, restaurant location, vehicle type used by the delivery partner, and type of cuisine ordered by the customer. Ultimately, our aim is to empower food delivery services with the tools and insights needed to optimize their operations, minimize delivery times, and deliver exceptional customer experiences.



1.2 Application

A. Food Delivery Services:

Predicting delivery times assists food delivery platforms in managing orders efficiently and informing customers about the expected arrival time of their meals. It allows restaurants to optimize their kitchen operations and coordinate deliveries effectively.

B. Parcel and Courier Services:

Time prediction models enable courier companies to estimate delivery windows for packages, improving route planning and resource allocation. Customers benefit from receiving real-time updates on the status of their shipments and estimated delivery times.

C. Retail and Grocery Delivery:

Retailers and grocery stores utilize delivery time predictions to offer convenient and reliable delivery services to their customers. It helps in planning inventory and staffing levels to meet the demand for delivery orders within specified timeframes.

D. E-Commerce:

Time prediction of delivery helps online retailers provide accurate estimated delivery times to customers. It improves customer satisfaction by setting clear expectations regarding when the purchased items will arrive.

1.3 Motivation Towards Your Project

Some platforms are unable to deliver their items on time, leading to customer dissatisfaction. Our goal is to develop a system that not only estimates delivery times reliably but also enhances the overall efficiency of food delivery operations.

1.4 Problem Statement

Our Project is Food Delivery Time Predictor. The project aims to develop a machine learning model that can predict delivery time by using geographical locations and previous data. The dataset for this project is taken from Kaggle. In this, we are going to calculate the distance between the restaurant and customer location, after then we will establish a relationship between the time taken by the delivery partner for the same kind of distance based on past data.

Chapter 2

Approach To Your Project

2.1 Explain About Your Project

This project aims to predict food delivery times, benefiting both sellers and buyers by providing accurate delivery estimates. Platform owners can enhance their services, leading to increased customer satisfaction. Customers can choose their dishes based on delivery times, and platform owners can ensure optimal delivery efficiency.

2.2 Data Set

The dataset is used for Food Delivery Time is taken from kaggle Website. Data set contain various features such as ID,Delivery_person_IDDelivery_person_Age,Delivery_person_Ratings,Restaurant_latitude etc.

ID- Order Id

Delivery_person_ID- Delivery Person Id

Delivery_person_Age Delivery Person Age

Delivery_person_Ratings- Delivery Person Ratings

Restaurant_latitude - Restaurant Latitude Point

Restaurant_longitude-Restaurant longitude point

Delivery_location_latitude - Delivery location latitude point

Delivery_location_longitude- Delivery_location_longitude point

Type_of_order - Type of order

Type_of_vehicle - Type of vehicle

Time_taken(min)- Time taken to deliver food

2.3 Prediction technique

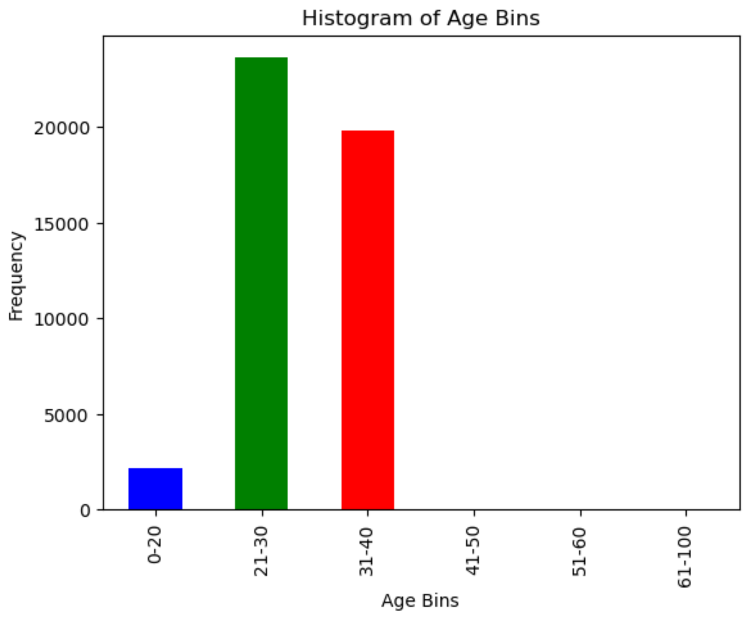
Our prediction techniques are Linear Regression and XGB Regression. Linear Regression is a model that shows the relationship between dependent and independent variables. We selected a suitable regression model for our prediction, which is XGBoost Regression. We use XGBoost because it predicts with high accuracy and runs efficiently even on large datasets.

2.4 Graphs

```
import matplotlib.pyplot as plt
import seaborn as sns
```

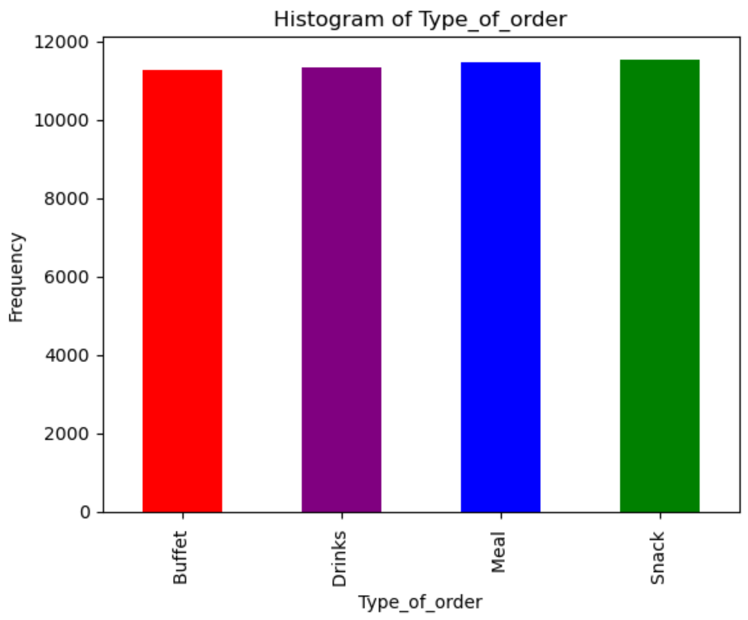
Histogram Plot for Age Bins

```
sorted_data = data['Age_Bins'].value_counts().sort_index()
color_palette = ['blue', 'green', 'red', 'purple']
sorted_data.plot(kind='bar',color=color_palette)
plt.xlabel('Age Bins')
plt.ylabel('Frequency')
plt.title('Histogram of Age Bins')
plt.show()
```



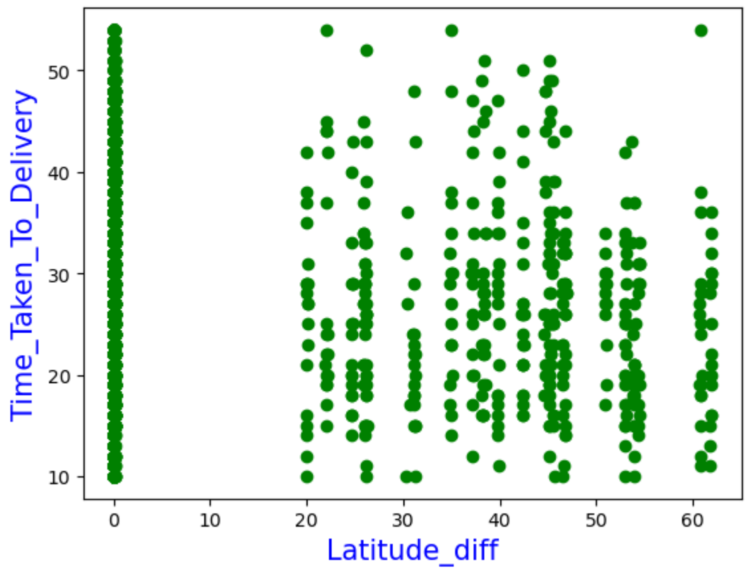
Histogram plot for Type_of_order

```
sorted_data = data['Type_of_order'].value_counts().sort_index()
color_palette = ['red', 'purple','blue', 'green']
sorted_data.plot(kind='bar',color=color_palette)
plt.xlabel('Type_of_order')
plt.ylabel('Frequency')
plt.title('Histogram of Type_of_order')
plt.show()
```



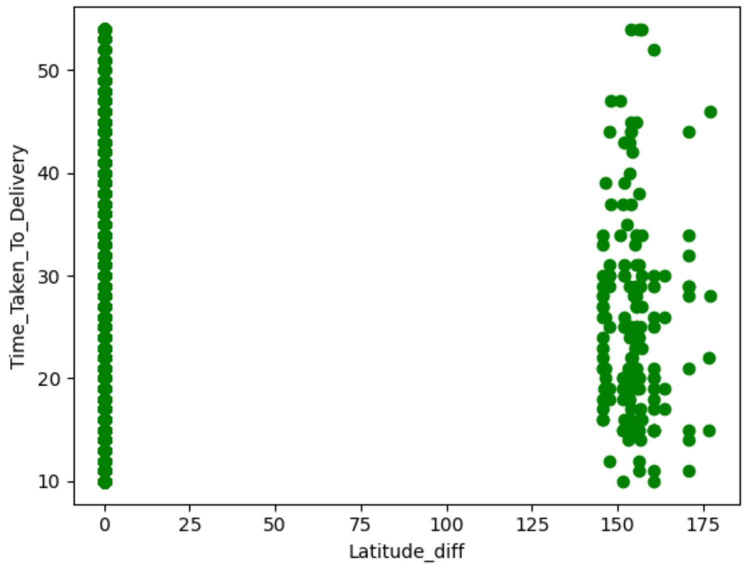
Scatter Plot for Latitude_diff and Time_Taken_To_Delivery

```
sorted_data=data.sort_values(by='Age_Bins')
#scatterplot
plt.scatter(sorted_data['lat_diff'],sorted_data['Time_taken(min)'],color='green')
plt.xlabel('Latitude_diff',color='b',fontsize=15)
plt.ylabel('Time_Taken_To_Delivery',color='b',fontsize=15)
plt.show()
```



Scatter Plot for Latitude_diff and Time_Taken_To_Delivery

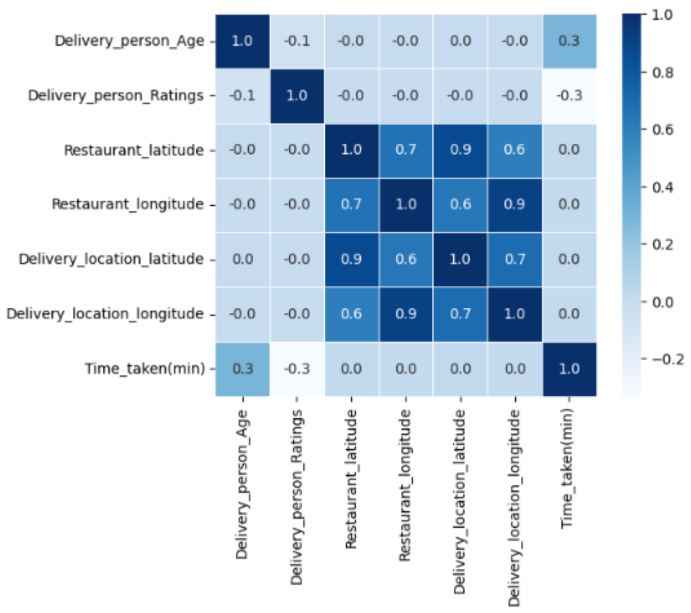
```
plt.scatter(sorted_data['lon_diff'],sorted_data['Time_taken(min)'],color='green')
plt.xlabel('Latitude_diff')
plt.ylabel('Time_Taken_To_Delivery')
plt.show()
```



2.5 Visualization

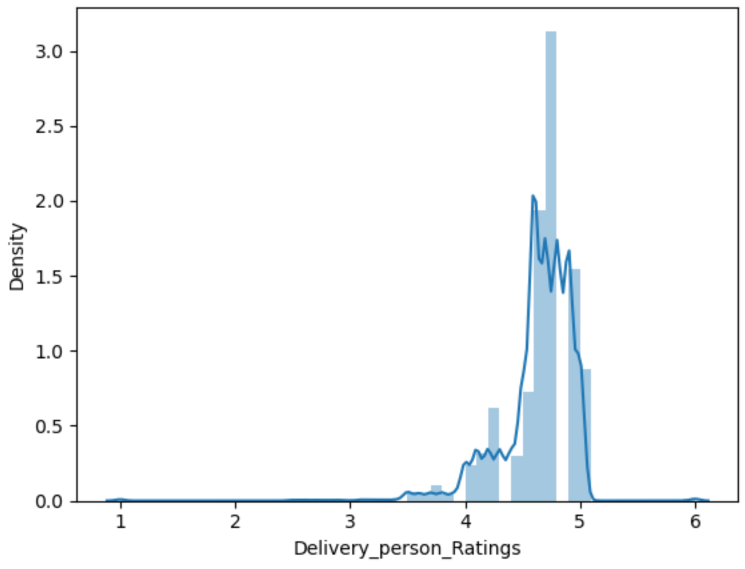
Heatmap

```
sns.heatmap(correlation,cbar=True, square=True, fmt='.1f', annot=True,
annot_kws={'size': 10}, cmap='Blues', linewidths=0.5),
plt.show()
plt.savefig("heatmap")
```



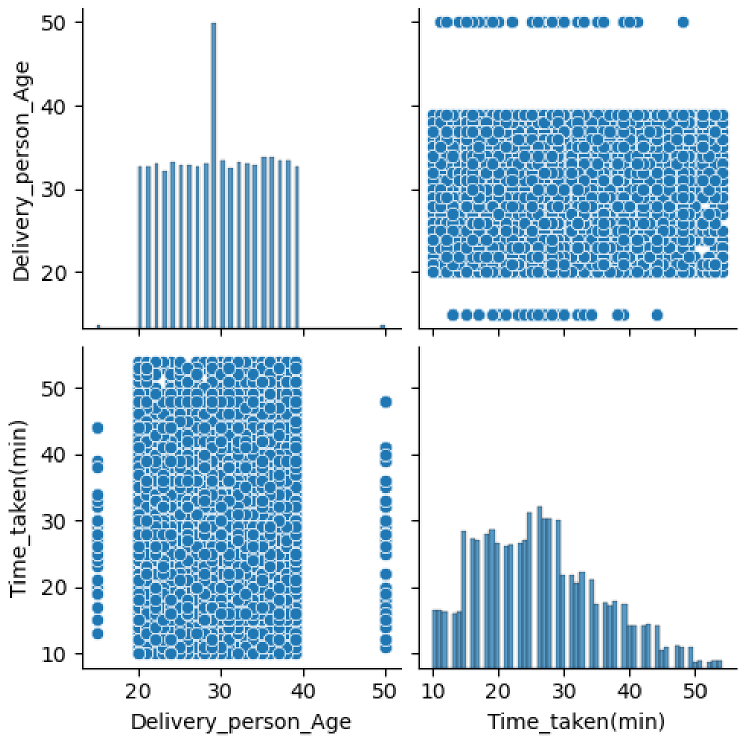
Distplot for delivery_ratings

```
sns.distplot(data['Delivery_person_Ratings'])
```



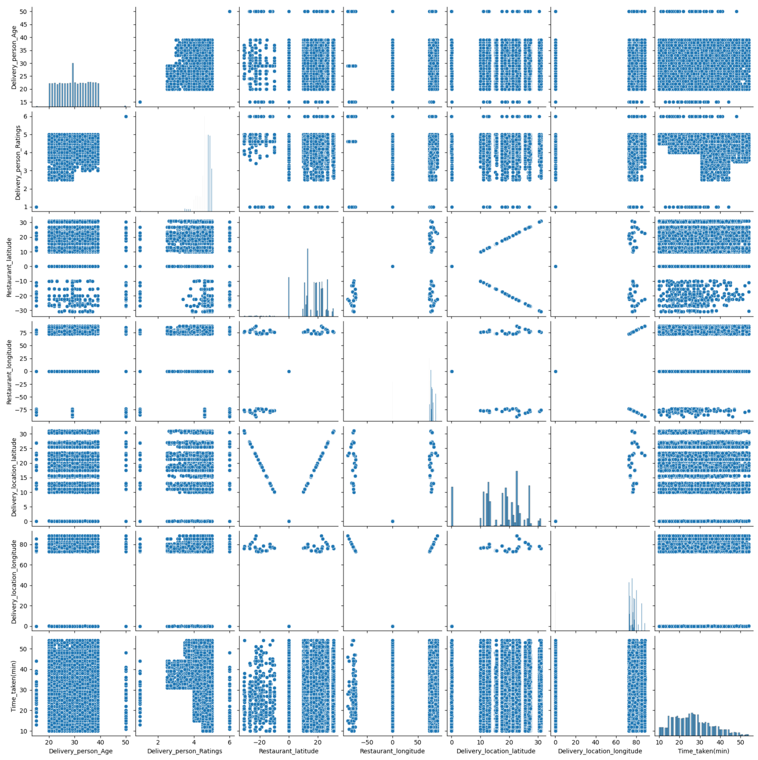
Pairplot for Delivery_person_Age and Time_taken(min)

```
pair_plot_graph=data[['Delivery_person_Age', 'Time_taken(min)']]
sns.pairplot(pair_plot_graph)
```



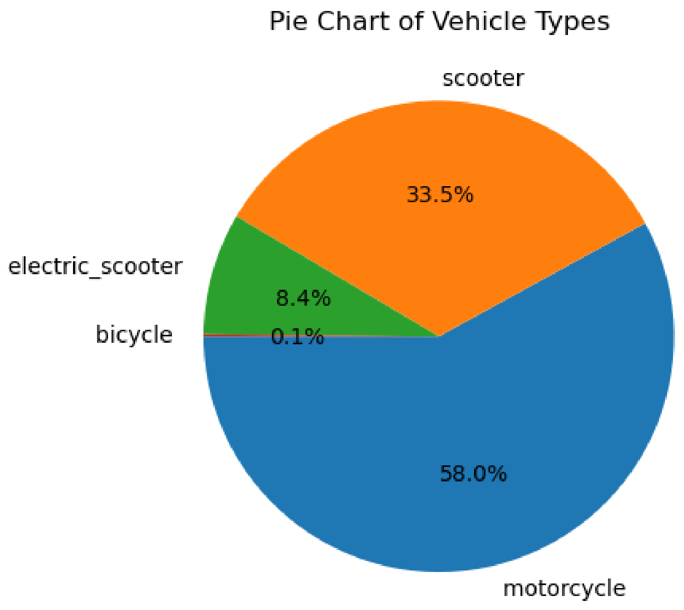
Pairplot

```
sns.pairplot(data)
```



Pie Chart of Vehicle Type

```
un_vehicle = list(data['Type_of_vehicle'].unique())
un_vehicle_count = list(data['Type_of_vehicle'].value_counts())
plt.pie(un_vehicle_count, labels=un_vehicle, startangle=180, explode=[0, 0, 0, 0],
shadow=False, autopct="%2.1f%%")
plt.title("Pie Chart of Vehicle Types")
plt.show()
```



Chapter 3

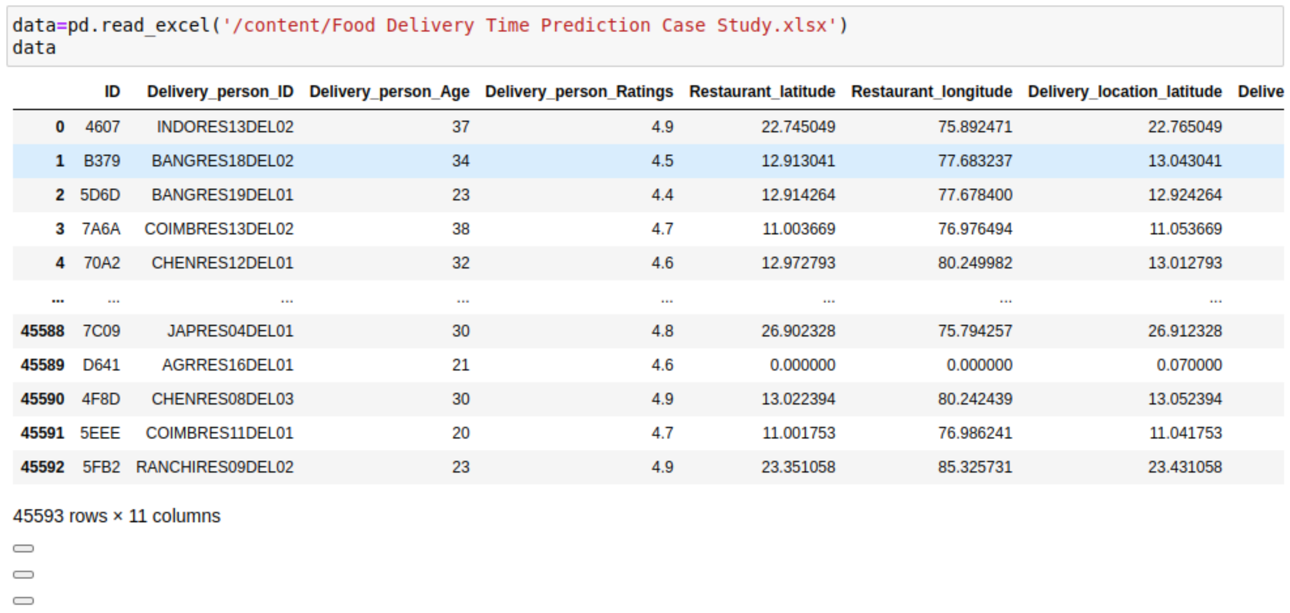
Code

Importing Essential libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Importing csv file to jupyter notebook using Pandas

```
data=pd.read_excel('/home/vamsi/Food Delivery_Time_Prediction Case_Study.xlsx')
```

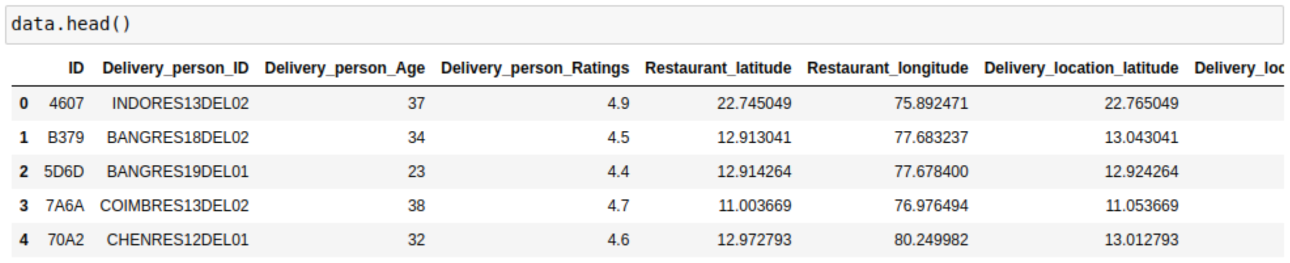


```
data=pd.read_excel('/content/Food Delivery Time Prediction Case Study.xlsx')
data
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delive
0	4607	INDORES13DEL02	37	4.9	22.745049	75.892471	22.765049	
1	B379	BANGRES18DEL02	34	4.5	12.913041	77.683237	13.043041	
2	5D6D	BANGRES19DEL01	23	4.4	12.914264	77.678400	12.924264	
3	7A6A	COIMBRES13DEL02	38	4.7	11.003669	76.976494	11.053669	
4	70A2	CHENRES12DEL01	32	4.6	12.972793	80.249982	13.012793	
...	...	...	...	...	...	...	...	
45588	7C09	JAPRES04DEL01	30	4.8	26.902328	75.794257	26.912328	
45589	D641	AGRRES16DEL01	21	4.6	0.000000	0.000000	0.070000	
45590	4F8D	CHENRES08DEL03	30	4.9	13.022394	80.242439	13.052394	
45591	5EEE	COIMBRES11DEL01	20	4.7	11.001753	76.986241	11.041753	
45592	5FB2	RANCHIRES09DEL02	23	4.9	23.351058	85.325731	23.431058	

45593 rows × 11 columns

Head



```
data.head()
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_loc
0	4607	INDORES13DEL02	37	4.9	22.745049	75.892471	22.765049	
1	B379	BANGRES18DEL02	34	4.5	12.913041	77.683237	13.043041	
2	5D6D	BANGRES19DEL01	23	4.4	12.914264	77.678400	12.924264	
3	7A6A	COIMBRES13DEL02	38	4.7	11.003669	76.976494	11.053669	
4	70A2	CHENRES12DEL01	32	4.6	12.972793	80.249982	13.012793	

Tail

```
data.tail()
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delive
45588	7C09	JAPRES04DEL01	30	4.8	26.902328	75.794257	26.912328	
45589	D641	AGRRES16DEL01	21	4.6	0.000000	0.000000	0.070000	
45590	4F8D	CHENRES08DEL03	30	4.9	13.022394	80.242439	13.052394	
45591	5EEE	COIMBRES11DEL01	20	4.7	11.001753	76.986241	11.041753	
45592	5FB2	RANCHIRES09DEL02	23	4.9	23.351058	85.325731	23.431058	

Checking null values

```
data.isnull().sum()
```

```
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
Type_of_order                  0
Type_of_vehicle                0
Time_taken(min)                0
dtype: int64
```

Checking data types

```
data.dtypes
```

```
ID                              object
Delivery_person_ID              object
Delivery_person_Age              int64
Delivery_person_Ratings        float64
Restaurant_latitude            float64
Restaurant_longitude           float64
Delivery_location_latitude     float64
Delivery_location_longitude    float64
Type_of_order                   object
Type_of_vehicle                 object
Time_taken(min)                  int64
dtype: object
```

Converting some data types into strings

```
#Converting some data types into strings
```

```
data['Type_of_order']=data['Type_of_order'].astype(str,errors='ignore')
```

Columns

```
data.columns
```

```
Index(['ID', 'Delivery_person_ID', 'Delivery_person_Age',
       'Delivery_person_Ratings', 'Restaurant_latitude',
       'Restaurant_longitude', 'Delivery_location_latitude',
       'Delivery_location_longitude', 'Type_of_order', 'Type_of_vehicle',
       'Time_taken(min)'],
      dtype='object')
```

Describing

```
data.describe()
```

	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_location_longitude	Time_t
count	45593.000000	45593.000000	45593.000000	45593.000000	45593.000000	45593.000000	455
mean	29.544075	4.632367	17.017729	70.231332	17.465186	70.845702	
std	5.696793	0.327708	8.185109	22.883647	7.335122	21.118812	
min	15.000000	1.000000	-30.905562	-88.366217	0.010000	0.010000	
25%	25.000000	4.600000	12.933284	73.170000	12.988453	73.280000	
50%	29.000000	4.700000	18.546947	75.898497	18.633934	76.002574	
75%	34.000000	4.800000	22.728163	78.044095	22.785049	78.107044	
max	50.000000	6.000000	30.914057	88.433452	31.054057	88.563452	

Checking Unique Values

```
#To know how many unique values in each row
data.nunique()
```

```
ID                             45355
Delivery_person_ID              1320
Delivery_person_Age               22
Delivery_person_Ratings           28
Restaurant_latitude              657
Restaurant_longitude             518
Delivery_location_latitude      4373
Delivery_location_longitude     4373
Type_of_order                      4
Type_of_vehicle                    4
Time_taken(min)                   45
dtype: int64
```

info()

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ID                           45593 non-null  object
 1   Delivery_person_ID           45593 non-null  object
 2   Delivery_person_Age          45593 non-null  int64
 3   Delivery_person_Ratings      45593 non-null  float64
 4   Restaurant_latitude          45593 non-null  float64
 5   Restaurant_longitude         45593 non-null  float64
 6   Delivery_location_latitude   45593 non-null  float64
 7   Delivery_location_longitude  45593 non-null  float64
 8   Type_of_order                45593 non-null  object
 9   Type_of_vehicle              45593 non-null  object
 10  Time_taken(min)              45593 non-null  int64
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
```

Shape

```
data.shape
```

```
(45593, 11)
```

Correlation Table

```
numeric_df = data.select_dtypes(include=['int64','float'])
correlation=numeric_df.corr()
correlation
```

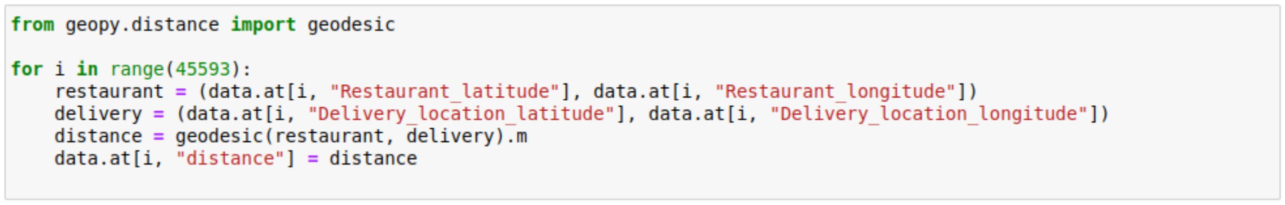
	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_locati
Delivery_person_Age	1.000000	-0.067449	-0.001955	-0.006796	0.002359	
Delivery_person_Ratings	-0.067449	1.000000	-0.004846	-0.011147	-0.010198	
Restaurant_latitude	-0.001955	-0.004846	1.000000	0.661784	0.866378	
Restaurant_longitude	-0.006796	-0.011147	0.661784	1.000000	0.632293	
Delivery_location_latitude	0.002359	-0.010198	0.866378	0.632293	1.000000	
Delivery_location_longitude	-0.000593	-0.013350	0.602713	0.915026	0.690515	
Time_taken(min)	0.292708	-0.331103	0.013981	0.007821	0.014243	

Data Binning

```
bins = [0, 20, 30, 40, 50, 60, 100]  # Define the age ranges for bins
labels = ['0-20', '21-30', '31-40', '41-50', '51-60', '61-100']
data['Age_Bins'] = pd.cut(data['Delivery_person_Age'], bins=bins, labels=labels)
data.head()
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_loc
0	4607	INDORES13DEL02	37	4.9	22.745049	75.892471	22.765049	
1	B379	BANGRES18DEL02	34	4.5	12.913041	77.683237	13.043041	
2	5D6D	BANGRES19DEL01	23	4.4	12.914264	77.678400	12.924264	
3	7A6A	COIMBRES13DEL02	38	4.7	11.003669	76.976494	11.053669	
4	70A2	CHENRES12DEL01	32	4.6	12.972793	80.249982	13.012793	

Adding Distance Column



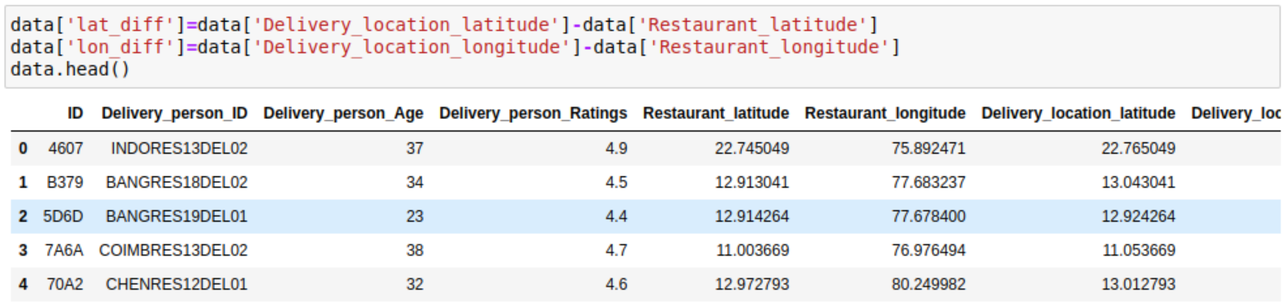
```
from geopy.distance import geodesic

for i in range(45593):
    restaurant = (data.at[i, "Restaurant_latitude"], data.at[i, "Restaurant_longitude"])
    delivery = (data.at[i, "Delivery_location_latitude"], data.at[i, "Delivery_location_longitude"])
    distance = geodesic(restaurant, delivery).m
    data.at[i, "distance"] = distance
```

Adding Speed Column

data['speed'] = data['distance'] / data['Time_taken(min)']

Adding Latitude difference and longitude difference Column



```
data['lat_diff']=data['Delivery_location_latitude']-data['Restaurant_latitude']
data['lon_diff']=data['Delivery_location_longitude']-data['Restaurant_longitude']
data.head()
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_loc
0	4607	INDORES13DEL02	37	4.9	22.745049	75.892471	22.765049	
1	B379	BANGRES18DEL02	34	4.5	12.913041	77.683237	13.043041	
2	5D6D	BANGRES19DEL01	23	4.4	12.914264	77.678400	12.924264	
3	7A6A	COIMBRES13DEL02	38	4.7	11.003669	76.976494	11.053669	
4	70A2	CHENRES12DEL01	32	4.6	12.972793	80.249982	13.012793	

Building Model

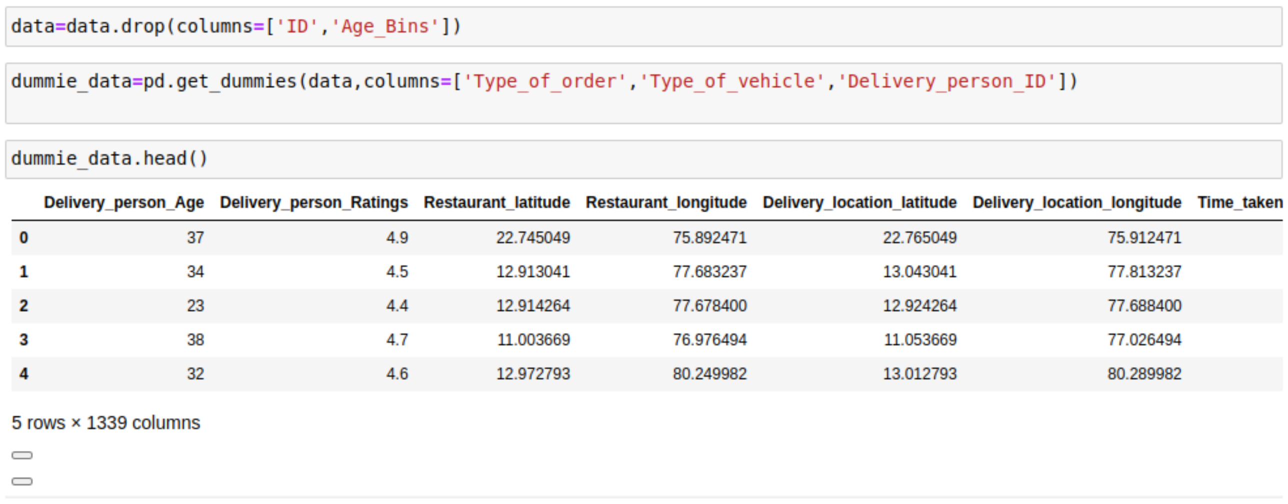
from sklearn.model_selection import train_test_split

from xgboost import XGBRegressor

from sklearn import metrics

from sklearn.linear_model import LinearRegression

Preparing Dummie data for Model



```
data=data.drop(columns=['ID','Age_Bins'])
```

```
dummie_data=pd.get_dummies(data,columns=['Type_of_order','Type_of_vehicle','Delivery_person_ID'])
```

```
dummie_data.head()
```

	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_location_longitude	Time_taken
0	37	4.9	22.745049	75.892471	22.765049	75.912471	
1	34	4.5	12.913041	77.683237	13.043041	77.813237	
2	23	4.4	12.914264	77.678400	12.924264	77.688400	
3	38	4.7	11.003669	76.976494	11.053669	77.026494	
4	32	4.6	12.972793	80.249982	13.012793	80.289982	

5 rows × 1339 columns

XGB Regressor Model

Splitting data into data and Target

X=dummie_data.drop(['Time'],axis=1)

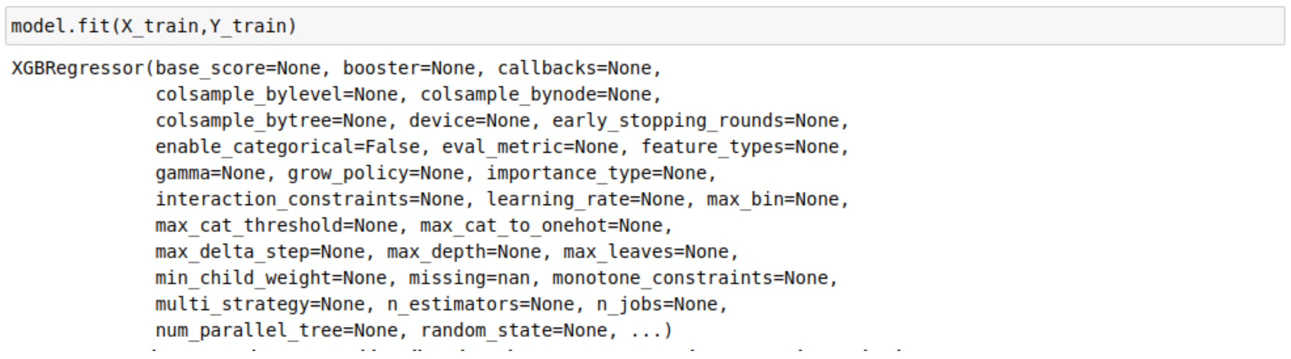
Y=dummie_data['Time']

model.fit(X_train,Y_train)

Dividing dummie data into X_train,X_test,Y_train,Y_test for our model

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)

model=XGBRegressor()



```
model.fit(X_train,Y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

model.score(X_test,Y_test)

0.9883501207168516

Linear Regression

model2 = LinearRegression()

model2.fit(X_train,Y_train)

LinearRegression()

model2.score(X_test,Y_test)

0.28764803085147006

Metrices

```
training_prediction=model.predict(X_train)
model.score(X_test,Y_test)
```

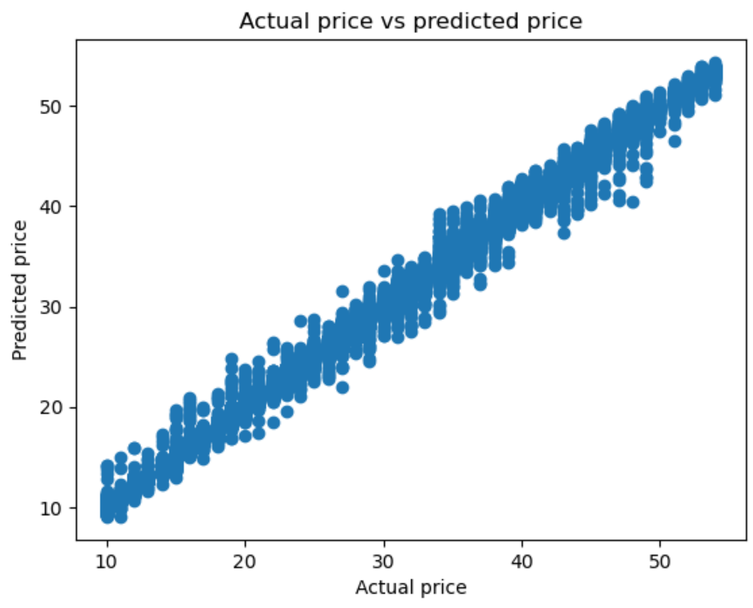
```
0.9883501207168516
```

```
score_1=metrics.r2_score(Y_train,training_prediction)
score_2=metrics.mean_absolute_error(Y_train,training_prediction)
```

```
print('r^2 value is',score_1)
print('mean absolute error',score_2)
```

```
r^2 value is 0.9960717619457725
mean absolute error 0.40036147659321836
```

```
plt.scatter(Y_train,training_prediction)
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual price vs predicted price')
```



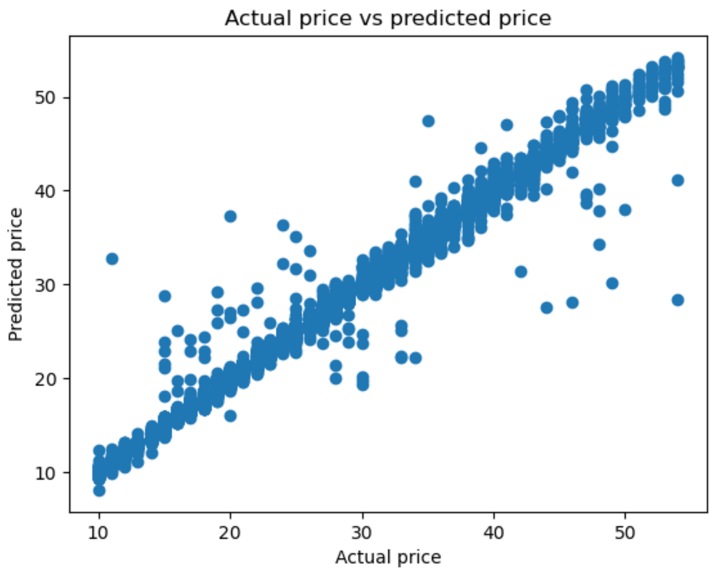
```
test_prediction=model.predict(X_test)
```

```
score_11=metrics.r2_score(Y_test,test_prediction)
score_22=metrics.mean_absolute_error(Y_test,test_prediction)
```

```
print('r^2 value is',score_11)
print('mean absolute error',score_22)
```

```
r^2 value is 0.9883496388104054
mean absolute error 0.4841757195078761
```

```
plt.scatter(Y_test,test_prediction)
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual price vs predicted price')
```



Chapter 4

Conclusion and Future Work

In conclusion, our project aimed to predict delivery time using machine learning models, specifically the XGB Regressor and Linear Regression. By using features such as distance, speed, delivery person age, delivery person ratings, and snack item, we developed a model that successfully predicted delivery time. This model can be helpful to both platform owners and buyers by providing accurate delivery time predictions, allowing for better planning and customer satisfaction. Platform owners can optimize their logistics and improve efficiency, while buyers can have more confidence in the estimated arrival times of their orders. This project demonstrates the potential of machine learning in online food delivery platforms, empowering individuals to make informed decisions about when to order an item. Accurate delivery time predictions can also enhance the overall user experience, leading to increased customer loyalty and trust in the platform. Moreover, by analyzing the factors that influence delivery time, we can gain valuable insights into the delivery process, identify potential areas for improvement, and implement strategies to reduce delivery times.We selected the model with the highest accuracy, which is the XGB Regressor. This model's superior performance, even with large datasets, ensures that our predictions are reliable and efficient. Through this project, we have showcased the power of machine learning in addressing real-world problems and improving service delivery in the food industry.