Problem Statement :)- Food Delivery Time Prediction

Description:

To predict the food delivery time in real-time, we need to calculate the distance between the food preparation point and the point of food consumption. After finding the distance between the restaurant and the delivery locations, we need to find relationships between the time taken by delivery partners to deliver the food in the past for the same distance.

1. Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
```

2. Dataset Information

The dataset you are given here is a cleaned version of the original dataset submitted by Gaurav Malik on Kaggle. Below are all the features in the dataset:

- DataSet Link (Click Me::)

2.1 Dataset Feature

1. ID: order ID number
2. Delivery_person_ID: ID number of the delivery partner
3. Delivery_person_Age: Age of the delivery partner
4. Delivery_person_Ratings: ratings of the delivery partner based on past 5. deliveries
5. Restaurant_latitude: The latitude of the restaurant
6. Restaurant_longitude: The longitude of the restaurant
7. Delivery_location_latitude: The latitude of the delivery location
8. Delivery_location_longitude: The longitude of the delivery location
9. Type_of_order: The type of meal ordered by the customer
10. Type_of_vehicle: The type of vehicle delivery partner rides
11. Time_taken(min): The time taken by the delivery partner to complete the order

2.3 Reading Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun
```

```
data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/DS_PROJECT/FOOD_DELIVERY_TIME_PREDICTION/deliverytime.txt")
data.head()
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Res
0	4607	INDORES13DEL02	37	4.9	
1	B379	BANGRES18DEL02	34	4.5	
2	5D6D	BANGRES19DEL01	23	4.4	
3	7A6A	COIMBRES13DEL02	38	4.7	
4	70A2	CHENRES12DEL01	32	4.6	

2.4 Data Exploration

```
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
```

```
data.describe()
```

	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Resta
count	45593.000000	45593.000000	45593.000000	
mean	29.544075	4.632367	17.017729	
std	5.696793	0.327708	8.185109	
min	15.000000	1.000000	-30.905562	
25%	25.000000	4.600000	12.933284	
50%	29.000000	4.700000	18.546947	
75%	34.000000	4.800000	22.728163	
max	50.000000	6.000000	30.914057	

2.5 Null Value

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

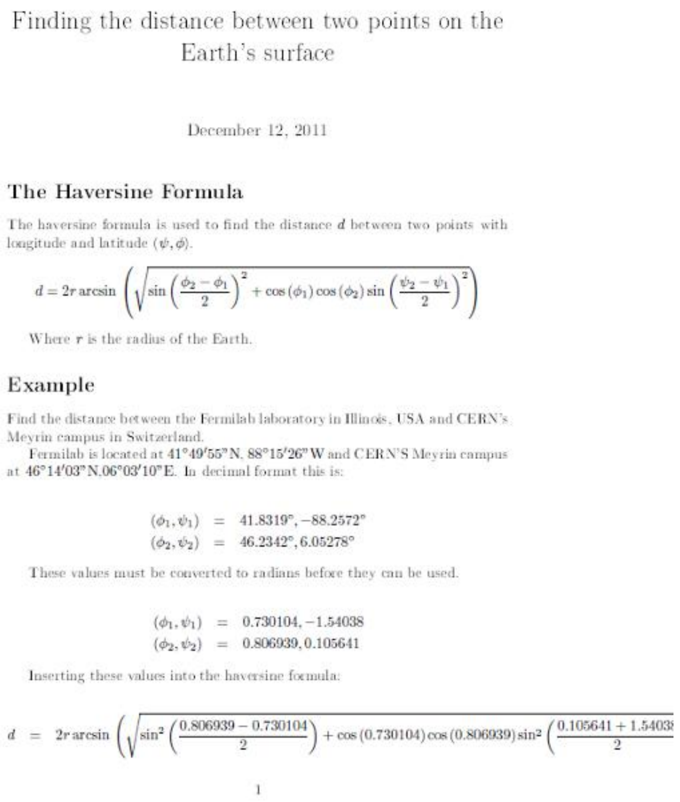
```
ID                             0
Delivery_person_ID             0
Delivery_person_Age            0
Delivery_person_Ratings        0
Restaurant_latitude            0
Restaurant_longitude           0
Delivery_location_latitude     0
Delivery_location_longitude    0
Type_of_order                  0
Type_of_vehicle                0
Time_taken(min)                0
dtype: int64
```

The dataset does not have any null values. Let’s move further!

3. Calculating Distance Between Two Latitudes and Longitudes

The dataset doesn’t have any feature that shows the difference between the restaurant and the delivery location. All we have are the latitude and longitude points of the restaurant and the delivery location. We can use the haversine formula to calculate the distance between two locations based on their latitudes and longitudes.

Harversine Formula



Finding the distance between two points on the Earth's surface

December 12, 2011

The Haversine Formula

The haversine formula is used to find the distance d between two points with longitude and latitude (ψ, ϕ).

$$d = 2r \arcsin\left(\sqrt{\sin\left(\frac{\phi_2 - \phi_1}{2}\right)^2 + \cos(\phi_1)\cos(\phi_2)\sin\left(\frac{\psi_2 - \psi_1}{2}\right)^2}\right)$$

Where r is the radius of the Earth.

Example

Find the distance between the Fermilab laboratory in Illinois, USA and CERN's Meyrin campus in Switzerland.

Fermilab is located at 41°49′55″N, 88°15′26″W and CERN'S Meyrin campus at 46°14′03″N,06°03′10″E. In decimal format this is:

$$\begin{aligned}(\phi_1, \psi_1) &= 41.8319°, -88.2572° \\ (\phi_2, \psi_2) &= 46.2342°, 6.05278°\end{aligned}$$

These values must be converted to radians before they can be used.

$$\begin{aligned}(\phi_1, \psi_1) &= 0.730104, -1.54038 \\ (\phi_2, \psi_2) &= 0.806939, 0.105641\end{aligned}$$

Inserting these values into the haversine formula:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{0.806939 - 0.730104}{2}\right) + \cos(0.730104)\cos(0.806939)\sin^2\left(\frac{0.105641 + 1.5403}{2}\right.}\right.$$

```
# set earth raduis(in K.M)
r=6371
# convert degree to radians
def deg_to_rad(degree):
  return degree*(np.pi/180)

# function to calaculate distance between two point
def dist_calculate(lat1,lon1,lat2,lon2):
  d_lat=deg_to_rad(lat2-lat1) # difference b/w two latitiude
  d_lon=deg_to_rad(lon2-lon1) # diffrenece b/w two longitude
  # usign above formula
  x=np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1))* np.cos(deg_to_rad(lat2) ) * np.sin(d_lon/2)**2
  y=2*np.arctan2(np.sqrt(x),np.sqrt(1-x))
  return r*y

# calculate distance between eah pair of points
data['distance']=np.nan

for i in range(len(data)):
  data.loc[i,'distance']=dist_calculate(data.loc[i,'Restaurant_latitude'],
                                        data.loc[i, 'Restaurant_longitude'],
                                        data.loc[i, 'Delivery_location_latitude'],
                                        data.loc[i, 'Delivery_location_longitude'])
```

We have now calculated the distance between the restaurant and the delivery location. We have also added a new feature in the dataset as distance. Let's look at the dataset again:

```
data.head()
```

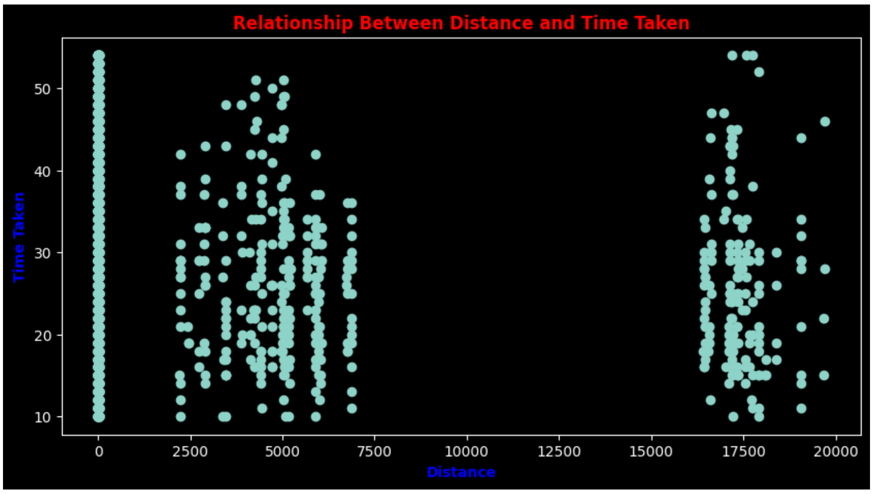
	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Res
0	4607	INDORES13DEL02	37	4.9	
1	B379	BANGRES18DEL02	34	4.5	
2	5D6D	BANGRES19DEL01	23	4.4	
3	7A6A	COIMBRES13DEL02	38	4.7	
4	70A2	CHENRES12DEL01	32	4.6	

4. Data Visualization

```
data.columns.unique()
```

```
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)', 'distance'],
      dtype='object')
```

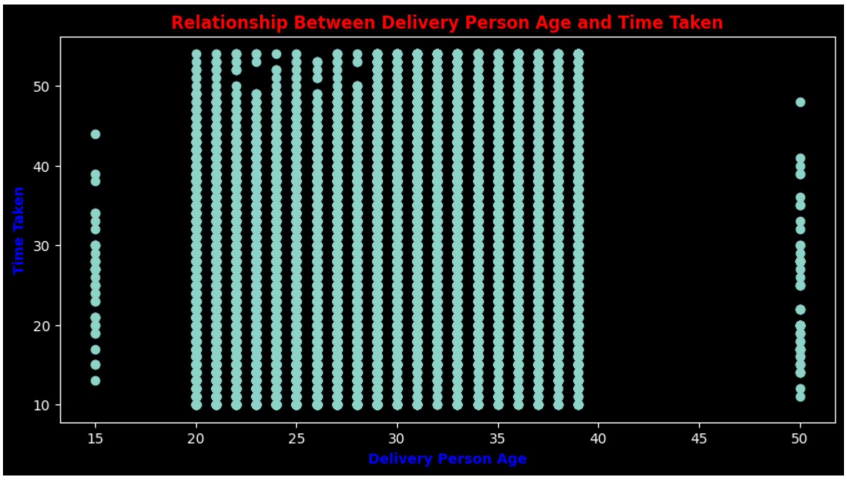
```
plt.style.use('dark_background')
plt.rcParams.update({'text.color':'white'})
plt.figure(figsize=(10,5))
plt.scatter(x=data['distance'],y=data['Time_taken(min)'])
plt.title("Relationship Between Distance and Time Taken",weight='bold',color='red')
plt.xlabel("Distance",weight='bold',color='blue')
plt.ylabel("Time Taken",weight='bold',color='blue')
plt.show()
```



- **There is a consistent relationship between the time taken and the distance travelled to deliver the food. It means that most delivery partners deliver food within 25-30 minutes, regardless of distance.**

- **Now let's have a look at the relationship between the time taken to deliver the food and the age of the delivery partner:**

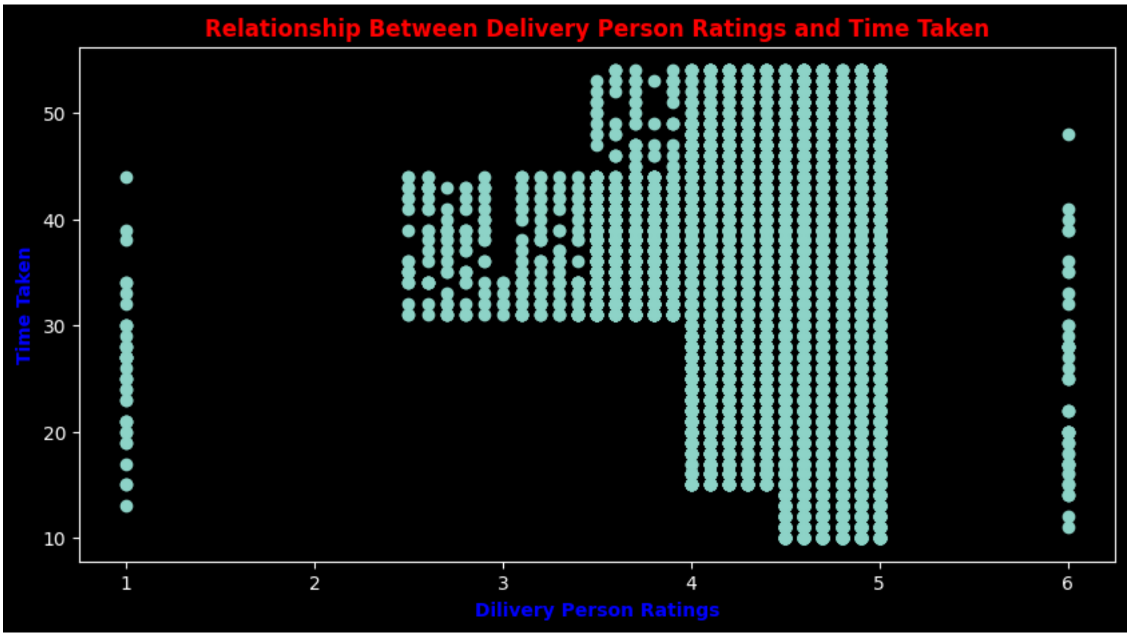
```
plt.style.use('dark_background')
plt.rcParams.update({'text.color':'white'})
plt.figure(figsize=(10,5))
plt.scatter(x=data['Delivery_person_Age'],y=data['Time_taken(min)'])
plt.title("Relationship Between Delivery Person Age and Time Taken",weight='bold',color='red')
plt.xlabel("Delivery Person Age",weight='bold',color='blue')
plt.ylabel("Time Taken",weight='bold',color='blue')
plt.show()
```



- **There is a linear relationship between the time taken to deliver the food and the age of the delivery partner. It means young delivery partners take less time to deliver the food compared to the elder partners.**

- **Now let's have a look at the relationship between the time taken to deliver the food and the ratings of the delivery partner:**

```
plt.style.use('dark_background')
plt.rcParams.update({'text.color':'white'})
plt.figure(figsize=(10,5))
plt.scatter(x=data['Delivery_person_Ratings'],y=data['Time_taken(min)'])
plt.title("Relationship Between Delivery Person Ratings and Time Taken",weight='bold',color='red')
plt.xlabel("Dilivery Person Ratings",weight='bold',color='blue')
plt.ylabel("Time Taken",weight='bold',color='blue')
plt.show()
```



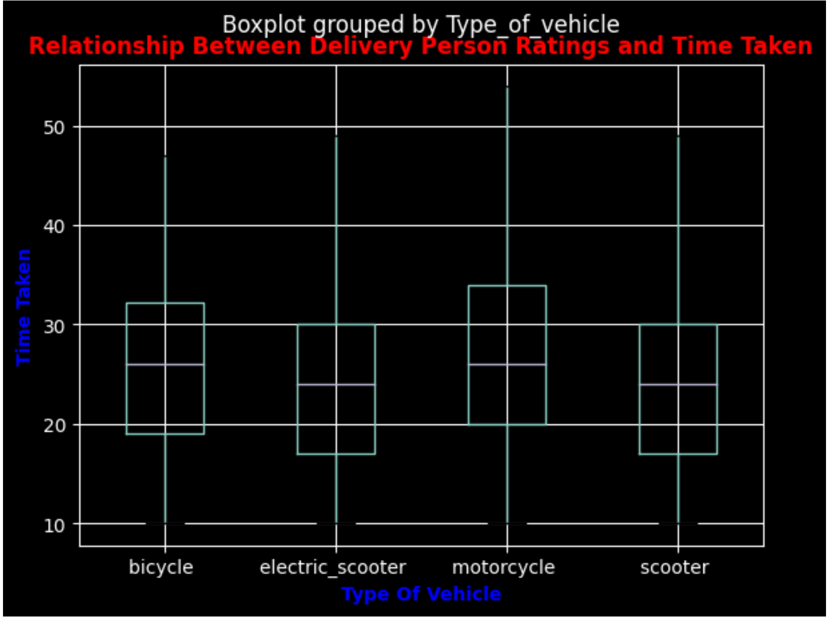
- **There is an inverse linear relationship between the time taken to deliver the food and the ratings of the delivery partner. It means delivery partners with higher ratings take less time to deliver the food compared to partners with low ratings.**

- **Now let's have a look if the type of food ordered by the customer and the type of vehicle used by the delivery partner affects the delivery time or not:**

```
plt.style.use('dark_background')
plt.rcParams.update({'text.color':'white'})
plt.figure(figsize=(10,5))
fig, ax = plt.subplots()
bp = data.boxplot(column=['Time_taken(min)'], by='Type_of_vehicle',ax=ax , showfliers=False)
# Set colors for each box
#colors = ['red', 'green', 'blue']
#for i, box in enumerate(bp['boxes']):
```

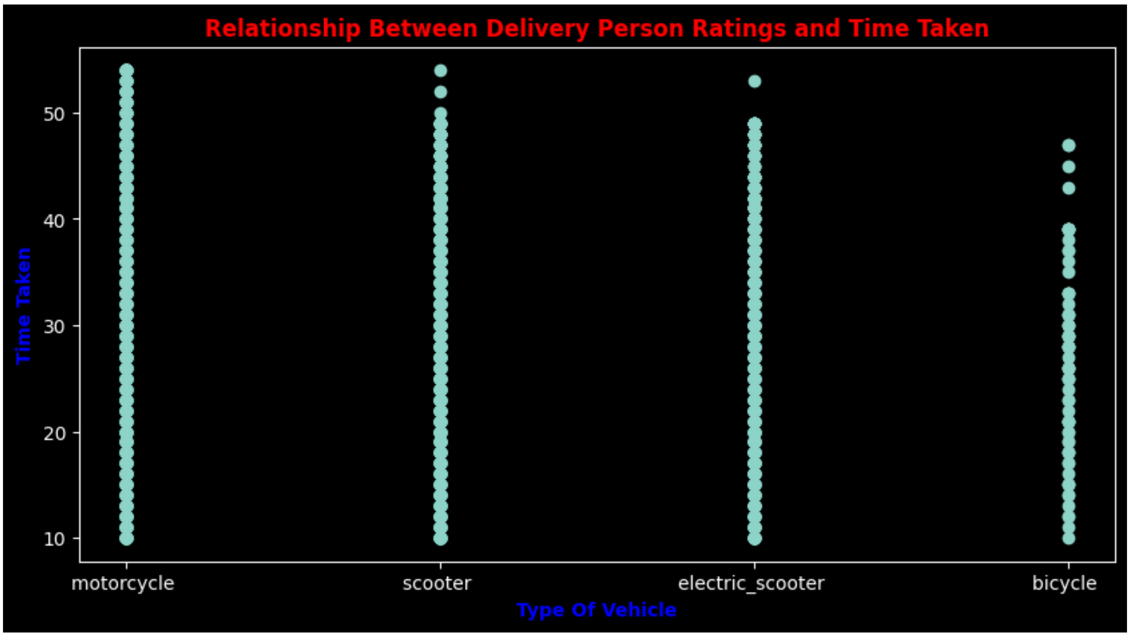
```
#   box.set(facecolor=colors[i])
plt.title("Relationship Between Delivery Person Ratings and Time Taken",weight='bold',color='red')
plt.xlabel("Type Of Vehicle",weight='bold',color='blue')
plt.ylabel("Time Taken",weight='bold',color='blue')
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```



```
plt.style.use('dark_background')
plt.rcParams.update({'text.color':'white'})
plt.figure(figsize=(10,5))
plt.scatter(x=data['Type_of_vehicle'],y=data['Time_taken(min)'])

plt.title("Relationship Between Delivery Person Ratings and Time Taken",weight='bold',color='red')
plt.xlabel("Type Of Vehicle",weight='bold',color='blue')
plt.ylabel("Time Taken",weight='bold',color='blue')
plt.show()
```



- **So there is not much difference between the time taken by delivery partners depending on the vehicle they are driving and the type of food they are delivering.**

So the features that contribute most to the food delivery time based on our analysis are:

- **age of the delivery partner**
- **ratings of the delivery partner**
- **distance between the restaurant and the delivery location**

5. Model Training

```
#splitting data
from sklearn.model_selection import train_test_split
x = np.array(data[["Delivery_person_Age",
                   "Delivery_person_Ratings",
                   "distance"]])
y = np.array(data[["Time_taken(min)"]])
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.10, random_state=42)
```

```
# creating the LSTM neural network model
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

```
# training the model
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size=1, epochs=9)
```

```
Epoch 1/9
41033/41033 [==============================] - 229s 5ms/step - loss: 69.2354
Epoch 2/9
41033/41033 [==============================] - 219s 5ms/step - loss: 63.8000
Epoch 3/9
41033/41033 [==============================] - 222s 5ms/step - loss: 61.1632
Epoch 4/9
41033/41033 [==============================] - 220s 5ms/step - loss: 60.8064
Epoch 5/9
41033/41033 [==============================] - 221s 5ms/step - loss: 60.3719
Epoch 6/9
41033/41033 [==============================] - 224s 5ms/step - loss: 60.1123
Epoch 7/9
41033/41033 [==============================] - 219s 5ms/step - loss: 59.5761
Epoch 8/9
41033/41033 [==============================] - 219s 5ms/step - loss: 59.2000
Epoch 9/9
41033/41033 [==============================] - 224s 5ms/step - loss: 58.8433
<keras.callbacks.History at 0x7f82e459cd90>
```

6. **Testing Model**

- **Now let’s test the performance of our model by giving inputs to predict the food delivery time:**

```
print("Food Delivery Time Prediction")
a = int(input("Age of Delivery Partner: "))
b = float(input("Ratings of Previous Deliveries: "))
c = int(input("Total Distance: "))

features = np.array([[a, b, c]])
print("Predicted Delivery Time in Minutes = ", model.predict(features))
```

```
Food Delivery Time Prediction
Age of Delivery Partner: 29
Ratings of Previous Deliveries: 2.9
Total Distance: 6
```

```
1/1 [==============================] - 1s 781ms/step
Predicted Delivery Time in Minutes =  [[36.173397]]
```

- **So this is how you can use Machine Learning for the task of food delivery time prediction using the Python programming language.**

Summary

To predict the food delivery time in real time, you need to calculate the distance between the food preparation point and the point of food consumption. After finding the distance between the restaurant and the delivery locations, you need to find relationships between the time taken by delivery partners to deliver the food in the past for the same distance. I hope you liked this article on food delivery time prediction with Machine Learning using Python. Feel free to ask valuable questions in the comments section below.

Reference:

- Aman Kharwal
- Google.com
- Wikipedia
- Kaggle

✓ 12s completed at 02:42