Food Delivery Time Prediction

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.
So, for this task, we need a dataset containing data about the time taken by delivery partners to deliver food from the restaurant to the delivery location. I found an ideal dataset with all the features for this task. You can download the dataset from **here**
In the section below, I will take you through the task of Food Delivery Time Prediction with Machine Learning using Python.

Food Delivery Time Prediction using Python

I will start the task of food delivery time prediction by importing the necessary Python libraries and the **dataset** :

In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
```

In [2]:
```
data = pd.read_csv('~/Downloads/Delivery time/deliverytime.txt')
print(data.head())
```

```
     ID Delivery_person_ID  Delivery_person_Age  Delivery_person_Ratings  \
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

   Restaurant_latitude  Restaurant_longitude  Delivery_location_latitude  \
0            22.745049             75.892471                   22.765049
1            12.913041             77.683237                   13.043041
2            12.914264             77.678400                   12.924264
3            11.003669             76.976494                   11.053669
4            12.972793             80.249982                   13.012793

   Delivery_location_longitude Type_of_order Type_of_vehicle  Time_taken(min)
0                    75.912471        Snack      motorcycle                24
1                    77.813237        Snack         scooter
```

```
33
2                    77.688400       Drinks      motorcycle
26
3                    77.026494       Buffet      motorcycle
21
4                    80.289982        Snack         scooter
30
```

Let’s have a look at the column insights before moving forward:

```
In [3]: 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
```

Now let’s have a look at whether this dataset contains any null values or not:

```
In [4]: data.isnull().sum()
```

```
Out[4]:

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!

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.

Below is how we can find the distance between the restaurant and the delivery location based on their latitudes and longitudes by using the haversine formula:

```
In [5]: R = 6371

def deg_to_rad(degrees):
    return degrees * (np.pi/180)
```

```
In [6]: # Set the earth's radius (in kilometers)
        R = 6371

# Convert degrees to radians
def deg_to_rad(degrees):
    return degrees * (np.pi/180)

# Function to calculate the distance between two points using the haversine formula
def distcalculate(lat1, lon1, lat2, lon2):
    d_lat = deg_to_rad(lat2-lat1)
    d_lon = deg_to_rad(lon2-lon1)
    a = np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1)) * np.cos(deg_to_rad(lat2)) * np.sin
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    return R * c

# Calculate the distance between each pair of points
data['distance'] = np.nan

for i in range(len(data)):
    data.loc[i, 'distance'] = distcalculate(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:

```
In [7]: print(data.head())
```

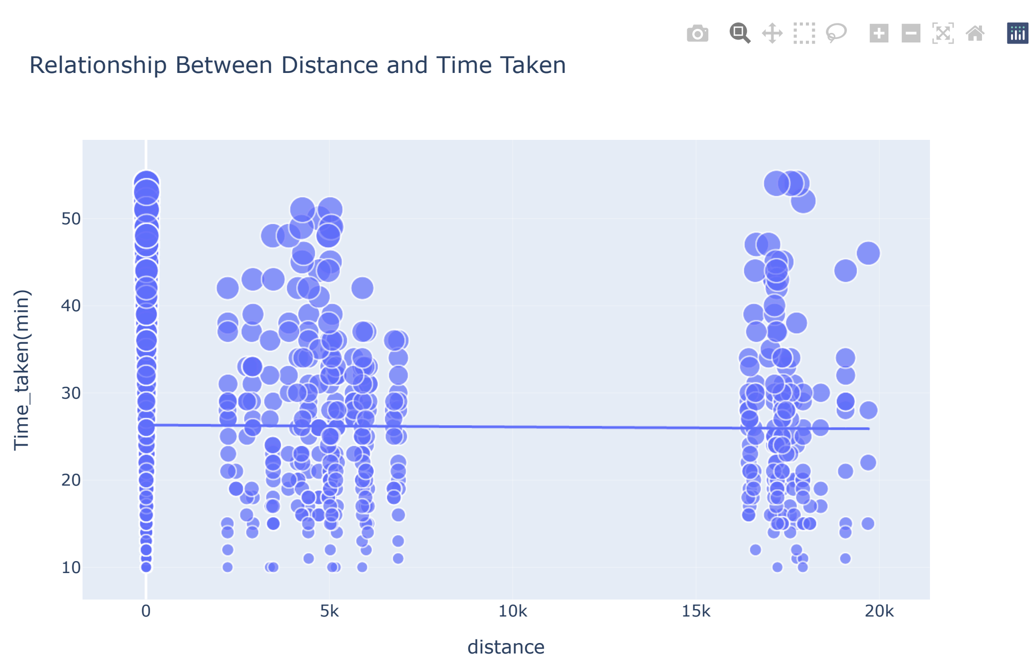
```
     ID Delivery_person_ID  Delivery_person_Age  Delivery_person_Ratings  \
0  4607     INDORES13DEL02                   37                      4.9
1  B379     BANGRES18DEL02                   34                      4.5
2  5D6D     BANGRES19DEL01                   23                      4.4
3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

   Restaurant_latitude  Restaurant_longitude  Delivery_location_latitude  \
0            22.745049             75.892471                   22.765049
1            12.913041             77.683237                   13.043041
2            12.914264             77.678400                   12.924264
3            11.003669             76.976494                   11.053669
```

```
4           12.972793            80.249982                     13.012793

   Delivery_location_longitude Type_of_order Type_of_vehicle  Time_taken(m
in)  \
0                    75.912471         Snack      motorcycle              
24
1                    77.813237         Snack         scooter              
33
2                    77.688400        Drinks      motorcycle              
26
3                    77.026494        Buffet      motorcycle              
21
4                    80.289982         Snack         scooter              
30

    distance
0   3.025149
1  20.183530
2   1.552758
3   7.790401
4   6.210138
```

Data Exploration

Now let’s explore the data to find relationships between the features. I’ll start by looking at the relationship between the distance and time taken to deliver the food:

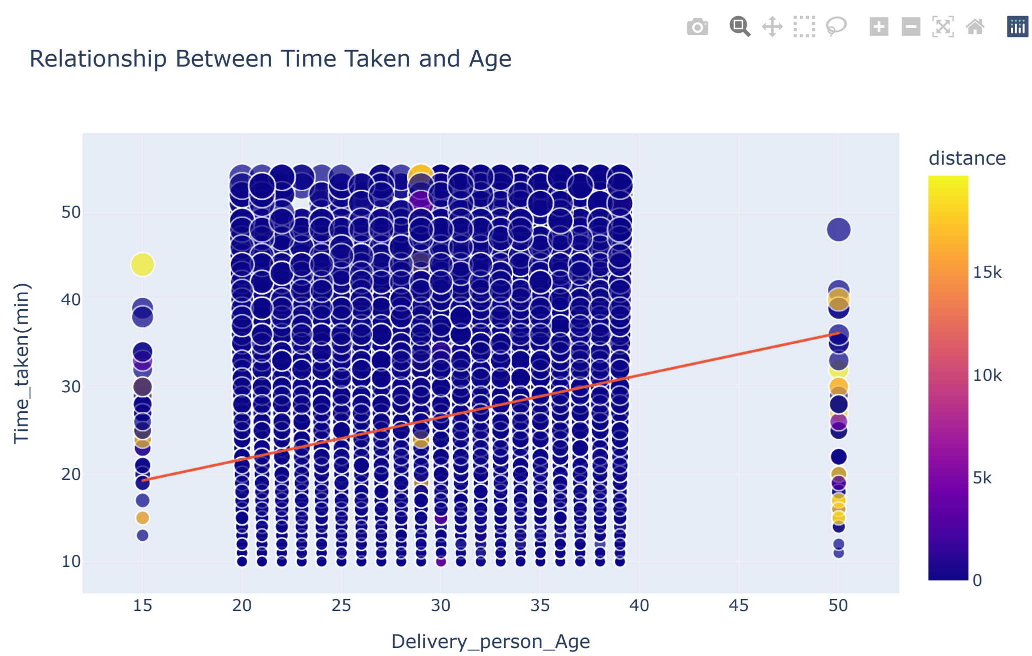
```
In [8]: figure = px.scatter(data_frame = data,
                                        x="distance",
                                 y="Time_taken(min)",
                                 size="Time_taken(min)",
                                 trendline="ols",
                                 title = "Relationship Between Distance and Time Taken")
figure.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.

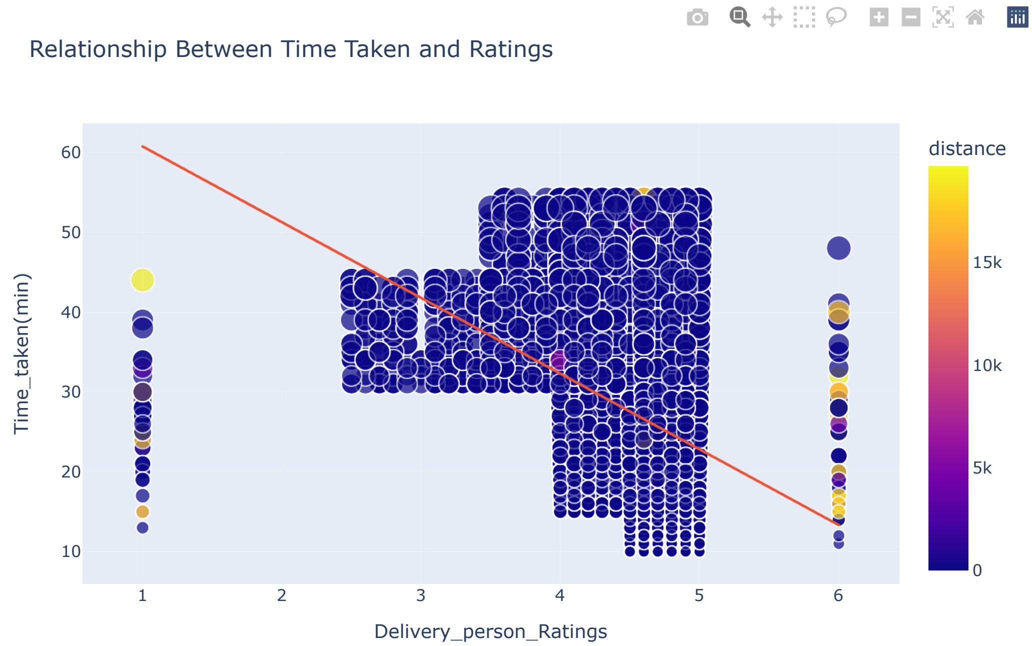
Now let's have a look at the relationship between the time taken to deliver the food and the age of the delivery partner:

```
In [9]: figure = px.scatter(data_frame = data,
                    x="Delivery_person_Age",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Age")
figure.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:

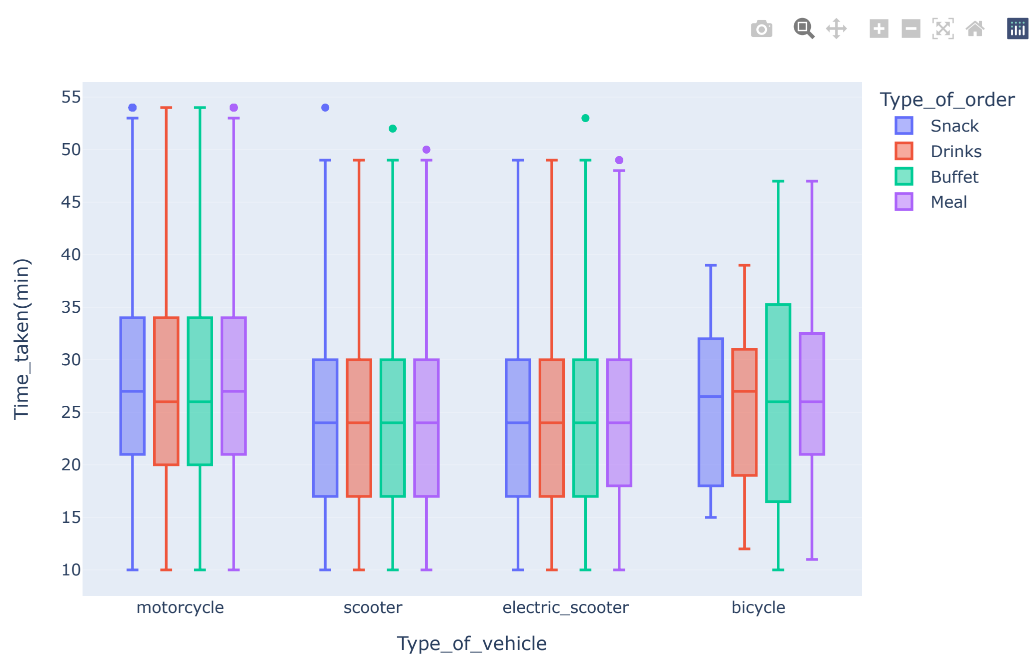
```
In [10]: figure = px.scatter(data_frame = data,
                    x="Delivery_person_Ratings",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Ratings")
figure.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:

```
In [11]: fig = px.box(data,
                      x="Type_of_vehicle",
               y="Time_taken(min)",
               color="Type_of_order")
fig.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

In the section below, I will take you through how to train a Machine Learning model for food delivery time prediction.

Food Delivery Time Prediction Model

Now let’s train a Machine Learning model using an LSTM neural network model for the task of food delivery time prediction:

```
In [12]: #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)"]])
xtrain, xtest, ytrain, ytest = 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= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
2024-11-01 19:25:21.218746: I tensorflow/core/platform/cpu_feature_guard.c
c:210] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebui
ld TensorFlow with the appropriate compiler flags.
/Users/mac/anaconda3/lib/python3.11/site-packages/keras/src/layers/rnn/rn
n.py:204: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When using S
equential models, prefer using an `Input(shape)` object as the first layer
in the model instead.
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 3, 128)	66,560
lstm_1 (LSTM)	(None, 64)	49,408
dense (Dense)	(None, 25)	1,625
dense_1 (Dense)	(None, 1)	26

Total params: 117,619 (459.45 KB)

Trainable params: 117,619 (459.45 KB)

Non-trainable params: 0 (0.00 B)

```
In [13]: # training the model
         model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=9)
```

```
Epoch 1/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 214s 5ms/step - loss: 76.2033
Epoch 2/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 212s 5ms/step - loss: 64.1248
Epoch 3/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 229s 6ms/step - loss: 61.9977
Epoch 4/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 208s 5ms/step - loss: 61.3661
Epoch 5/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 198s 5ms/step - loss: 59.9033
Epoch 6/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 214s 5ms/step - loss: 60.0627
Epoch 7/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 243s 6ms/step - loss: 59.1287
Epoch 8/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 212s 5ms/step - loss: 59.0088
Epoch 9/9
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 193s 5ms/step - loss: 58.4231
Out[13]:
```

```
<keras.src.callbacks.history.History at 0x13cc680d0>
```

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

```
In [14]: 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: 20
Ratings of Previous Deliveries: 4.3
Total Distance: 10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Predicted Delivery Time in Minutes =  [[25.476435]]
```

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