Distance Between Two Locations using Python

Calculating the distance between two locations helps companies like Swiggy and `Zomato` calculate the delivery time for an order. If you want to learn how to calculate the distance between two locations by using their latitude and longitude points. I will working on this project calculating the distance between two locations using Python.

Food Delivery services like Zomato and Swiggy need to show the accurate time it will take to deliver your order to keep transparency with their customers. These companies use Machine Learning algorithms to predict the food delivery time based on how much time the delivery partners took for the same distance in the past. So, if you want to learn how to use Machine Learning for food delivery time prediction

Here's How `Zomato` Uses Data Science

Zomato is an Indian restaurant aggregator and food delivery company. It is known for faster delivery of food from anywhere in your city. Like every successful business today, Data Science is one of the reasons behind the success of Zomato.

Budget `Friendly` Recommendations

Every customer has a spending capacity for a meal. The spending capacity differs from person to person according to their lifestyle and income. So whenever someone searches for food, zomato recommends restaurants based on their spending capacity.

Every time you order something from Zomato, your tastes and preferences are analyzed, and the amount you spend on your meal is recorded. And whenever you're looking for something new to eat, the restaurant recommendations you see are based on:

1. the restaurants that are popular for what you searched for
2. your average spending capacity on zomato
3. reviews and ratings of other users

Food Preparation `Time Prediction`

Zomato is available in 500+ cities in India. To keep transparency with all its customers, Zomato predicts the food preparation time to calculate the delivery time that helps in the overall process of completion of an order.

When you place a food order, the restaurant accepts your order and begins to prepare your food, and if demand is high at that restaurant, your order waits in queued orders. So, predicting the food preparation time when only your food is prepared is an easy task.

But when your order is waiting in queue orders, Zomato predicts the food preparation time considering the preparation time of all food orders in the restaurant before your order.

It helps in showing the accurate time it will take to deliver your order so that the customer can cancel and order from another restaurant if the delivery time is very long.

Analyzing `Food Orders` Demand at all Locations

Another area where Zomato uses Data Science for its business is to analyze the food orders demand in all the areas of the city to improve its food delivery services.

Analyzing food demand in all the areas helps Zomato determine:

1. which areas order more food online
2. demand in which areas increases at what time
3. what kind of food and restaurants are more demanded in which areas

It helps zomato in determining areas with high food demand to employ more delivery partners in those areas to make the complete food delivery process faster.

1. Zomato recommends restaurants based on your average spending capacity to provide budget-friendly recommendations to all customers.
2. Zomato also predicts the food preparation time to show the accurate time it will take to deliver your order so that the customer can cancel and order from another restaurant if the delivery time is very long.

3. Zomato also analyzes food orders demand in all areas of the city to find areas with high food demand to employ more delivery partners in those areas to make the complete food delivery process faster.

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.

```
import pandas as pd
import numpy as np
import plotly.express as px

data = pd.read_csv("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:

```
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:

```
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!

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:

```
# 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(d_lon/2)**2
    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:

```
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   
```

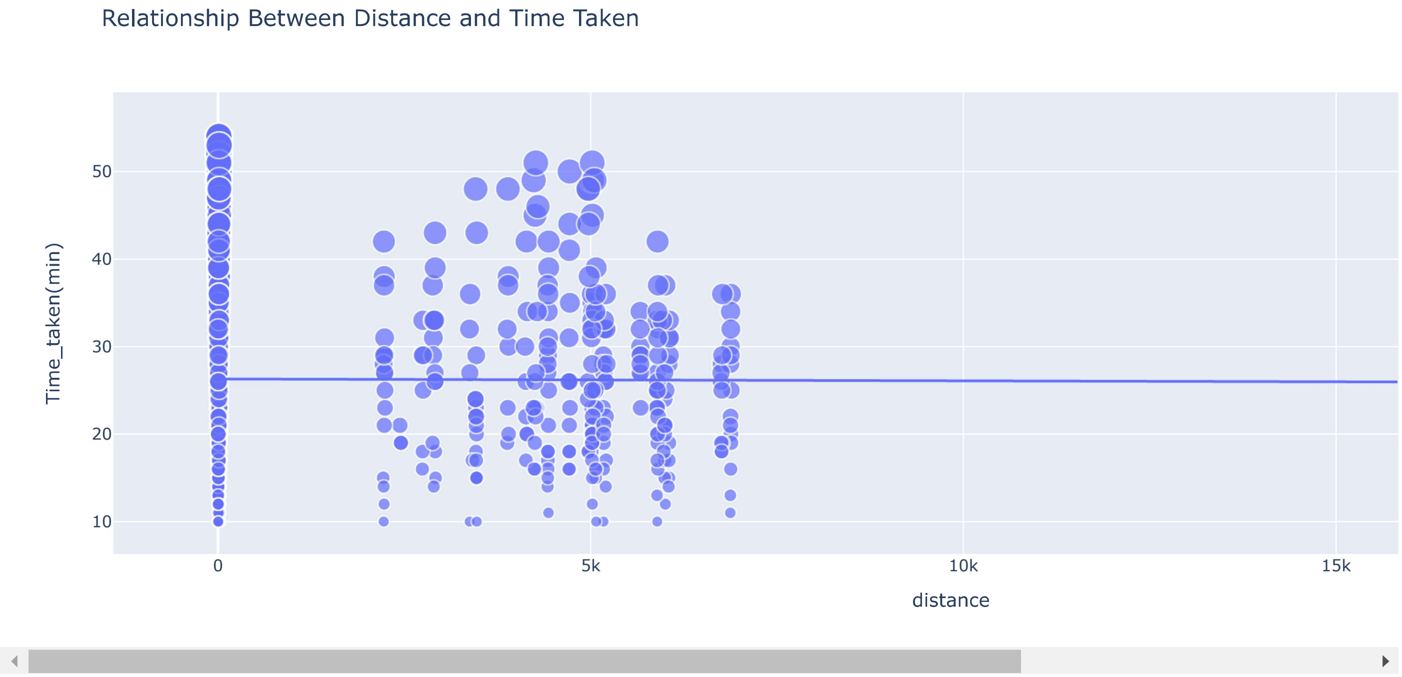
```
4                    80.289982        Snack        scooter                30

     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:

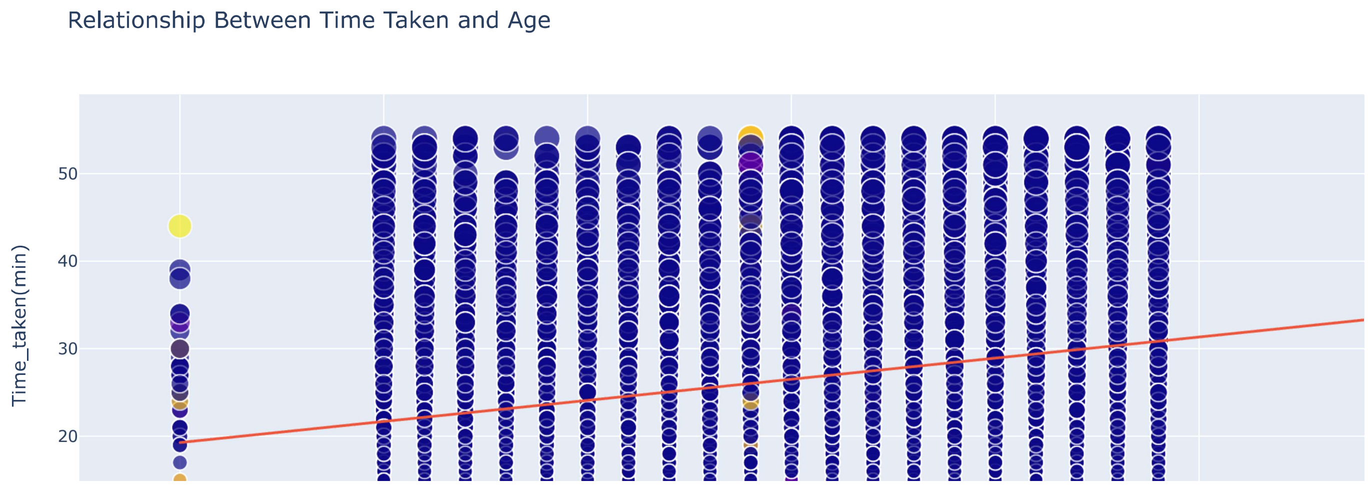
```
figure = px.scatter(data_frame = data,
                    x="distance",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    trendline="ols",
                    title = "Relationship Between Distance and Time Taken")
figure.show()
```



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:

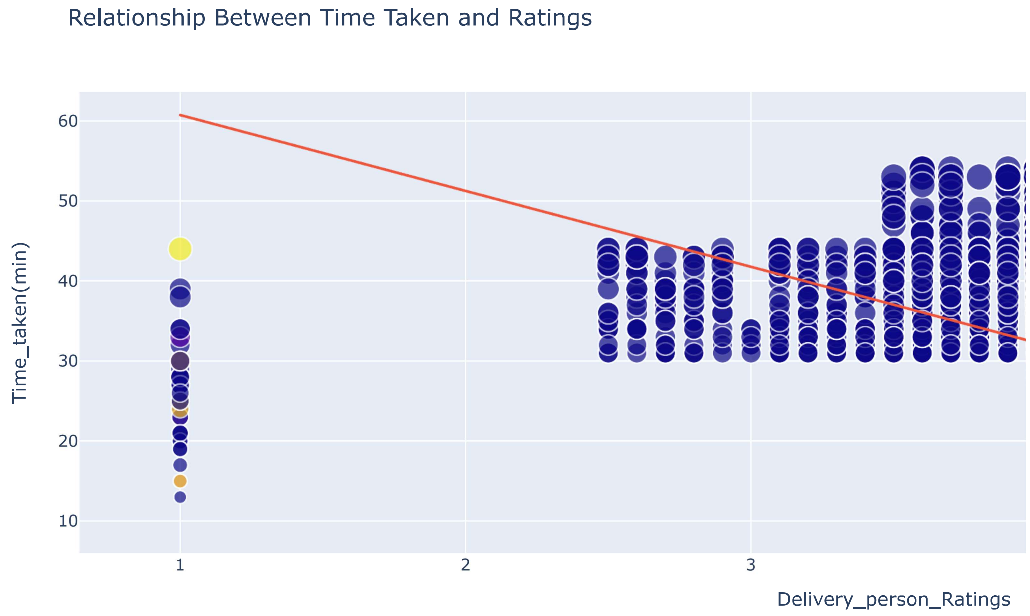
```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Age",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Time 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:

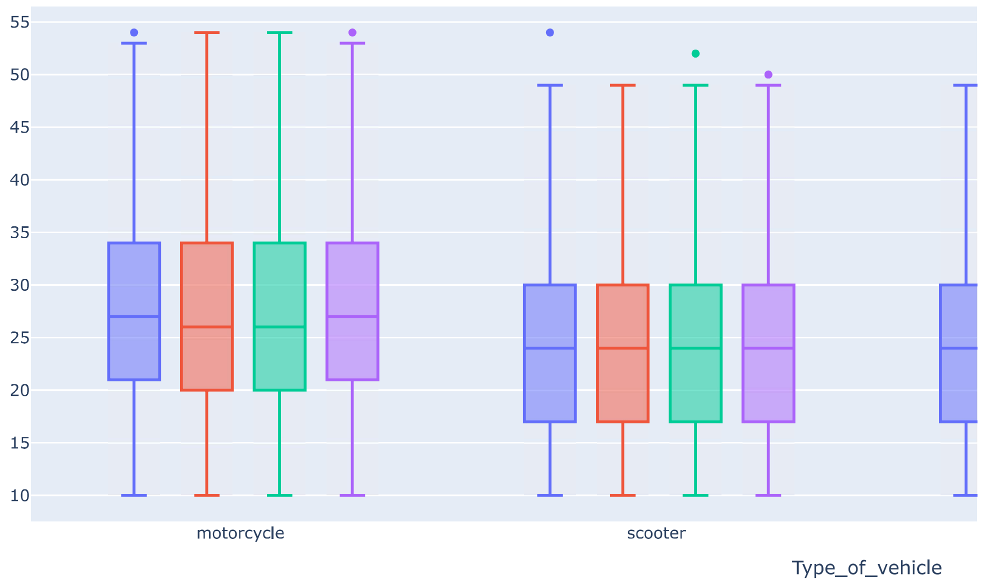
```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Ratings",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between 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:

```
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

1. List item
2. List item

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

```
#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()
```

```
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(xtrain, ytrain, batch_size=1, epochs=9)
```

```
Epoch 1/9
41033/41033 [==============================] - 229s 5ms/step - loss: 69.2229
Epoch 2/9
41033/41033 [==============================] - 221s 5ms/step - loss: 64.0507
Epoch 3/9
41033/41033 [==============================] - 215s 5ms/step - loss: 61.5838
Epoch 4/9
41033/41033 [==============================] - 215s 5ms/step - loss: 60.6159
Epoch 5/9
41033/41033 [==============================] - 209s 5ms/step - loss: 60.1228
Epoch 6/9
41033/41033 [==============================] - 210s 5ms/step - loss: 60.0955
Epoch 7/9
41033/41033 [==============================] - 212s 5ms/step - loss: 59.5841
Epoch 8/9
41033/41033 [==============================] - 213s 5ms/step - loss: 59.3016
Epoch 9/9
41033/41033 [==============================] - 225s 5ms/step - loss: 59.4373
<keras.callbacks.History at 0x7f279b1f2530>
```

```
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: 4.5
Total Distance: 35
1/1 [==============================] - 1s 870ms/step
Predicted Delivery Time in Minutes =  [[33.141262]]
```

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.

✓ 23s completed at 12:57 PM ● ×