Scenario: Food Delivery services like Mr D food and Uber Eats 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.

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.

In [ ]:

```
```

In [ ]:

```
```

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 matplotlib.pyplot as plt
import plotly.express as px
```

In [2]:
```python
data = pd.read_csv('deliverytime.xlsx')
data.head()
```

Out[2]:

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

In [ ]:

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

In [3]:
```python
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
```

In [ ]:

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

In [4]:
```
data.isna().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
```

In [5]:
```
# The dataset does not have any null values. Let’s continue
```

In [ ]:

Calculating Distance Between Two Latitudes and Longitudes

The dataset we just exracted doesn't have any information or features that gives us the differnce between the restaurant and the delivery location. But we have the Latitude and Longititude points of the Resturant and the Delivery loaction, so we can use that information to calculate the distance use the haversine formula.

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes.

Lets see 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 [6]:
```python
# Set the earth's radius (in kilometers)

r = 6371

# Function to convert degrees to radians

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

# Function to calculate the distance using haversine formula between two points

def calculate_distance(rest_latitude, rest_longitude, delivery_latitude, delivery_longitude):
    distance_latitude = degree_to_radian(delivery_latitude - rest_latitude)
    distance_longitude = degree_to_radian(delivery_longitude - rest_longitude)

    a = np.sin(distance_latitude / 2)**2 + np.cos(degree_to_radian(rest_latitude))* np.cos(degree_to_radian(d
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))

    return r * c

# Calculate the distance between each pair of points(I will add a distance column aswell)

data['Distance'] = np.nan

for i in range(len(data)):
    data.loc[i, 'Distance'] = calculate_distance(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 column which is 'distance' a new feature. Let’s look at the dataset again:

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

Out[7]:

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

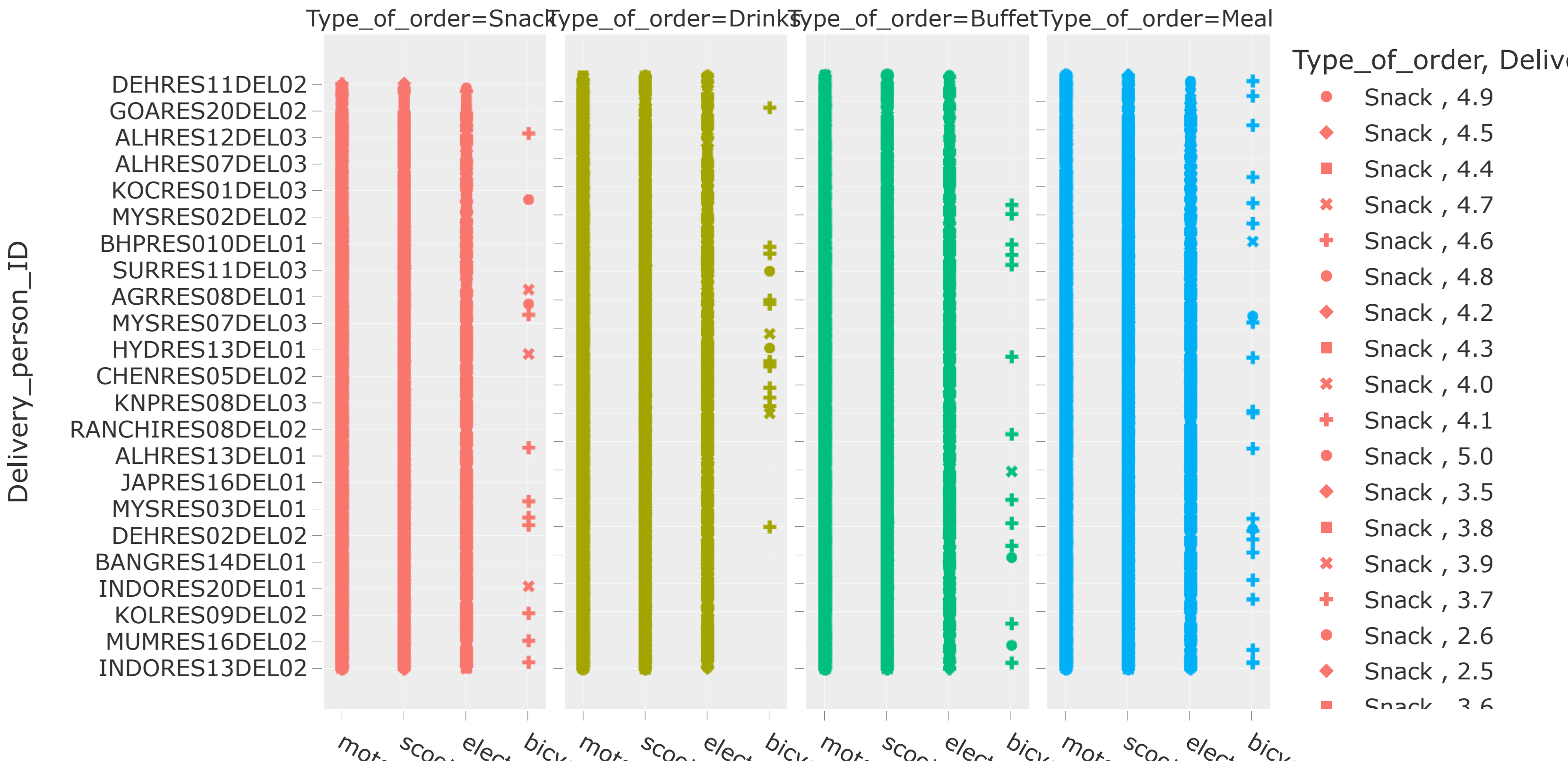
In [ ]:

Data Exploration

Now let’s go deep into the data and find relationships between the features. Its best I start by looking at the relationship between the Type of vehicle, Delivery person ID, based on Type of order and Delivery Ratings:

In [8]:

```
for template in ["ggplot2"]:

    figure = px.scatter(data,
                        x="Type_of_vehicle",
                        y="Delivery_person_ID",
                        color='Type_of_order',
                        symbol='Delivery_person_Ratings',
                        facet_col='Type_of_order',

                        template=template)
    figure.show()
```

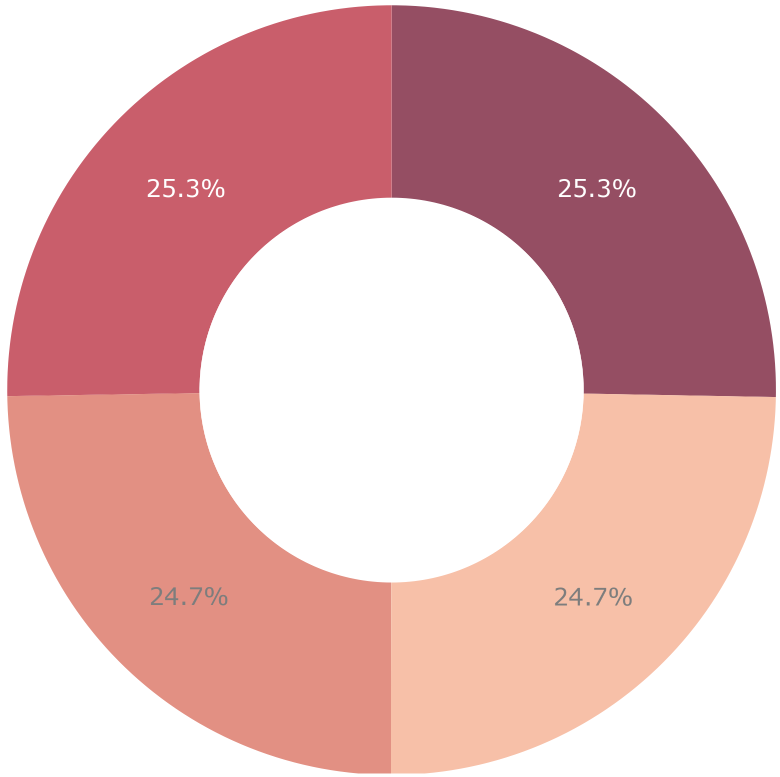


With this data we can see different ratings based on the type of order, and type of vehicle, the Bicycle isn't used the most but if we check its ratings they're far much better than other vehicles

```
In [ ]:
```

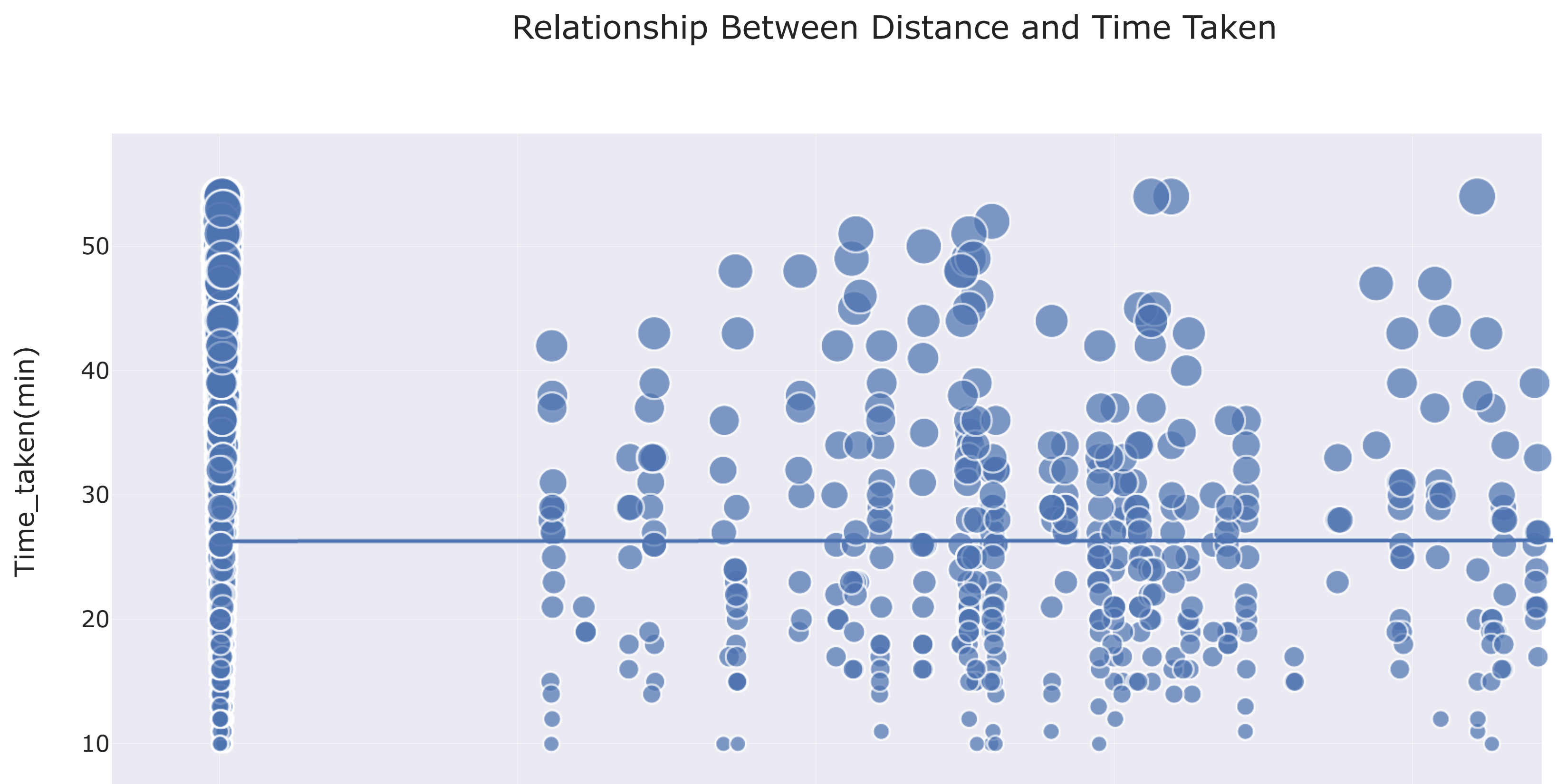
This pie chart shows us that both Drinks and Buffet takes less time to be delivered compared to Snack and Meal.

In [9]:
```
fig = px.pie(data, values="Time_taken(min)", names="Type_of_order",
             color_discrete_sequence=px.colors.sequential.RdBu,
             opacity=0.7, hole=0.5)
fig.show()
```



Now let’s go deep into the data and find relationships between the features. Its best I start by looking at the relationship between the distance and time taken to deliver the food:

In [10]:
```
for template in ["seaborn"]:

    figure = px.scatter(data,
                        x="Distance",
                        y="Time_taken(min)",
                        size="Time_taken(min)",
                        hover_data=['Delivery_person_Age'],
                        trendline="ols",
                        template=template,
                        title = "Relationship Between Distance and Time Taken")
    figure.show()
```



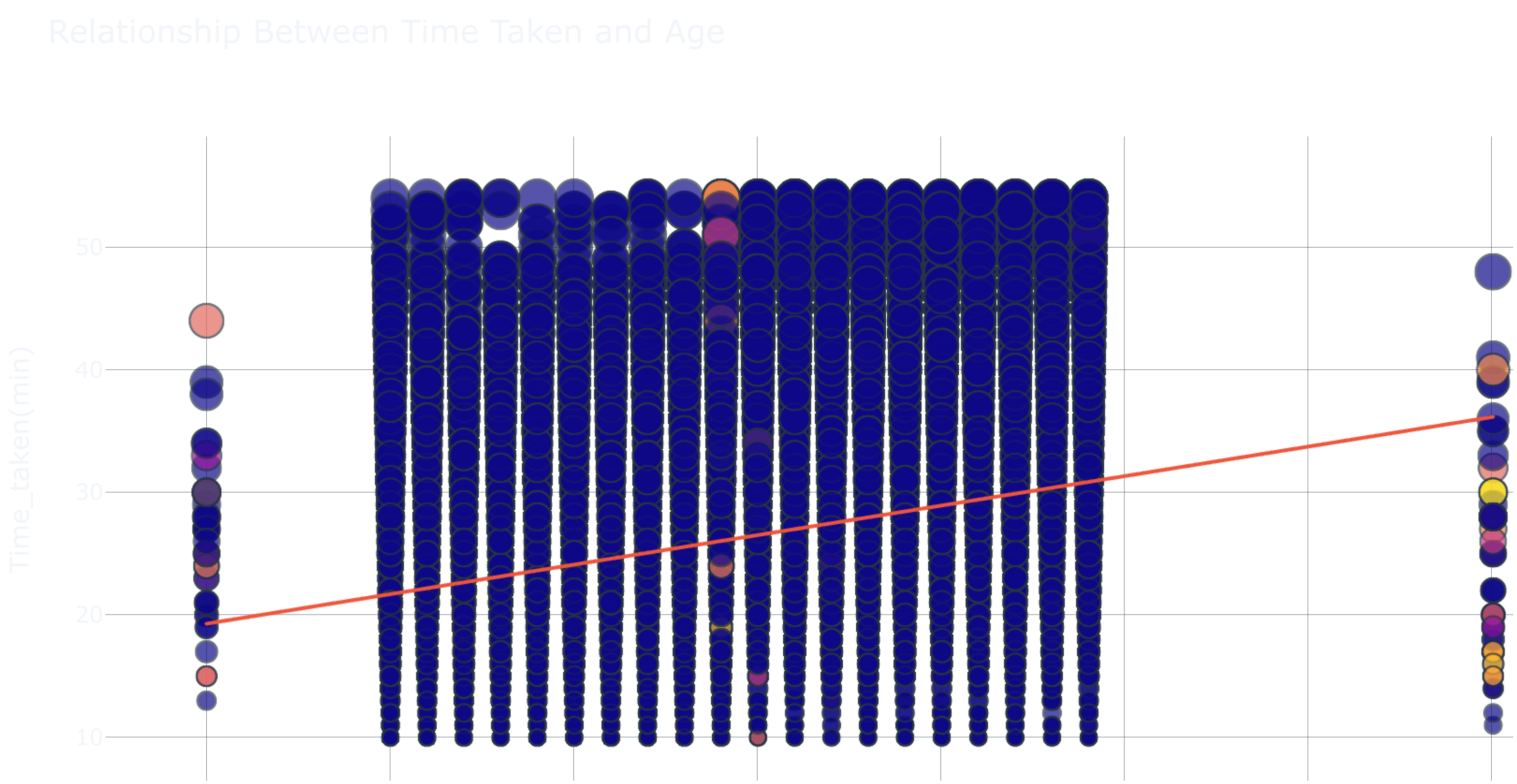
We use 'Ols' trendline to show us the overall trend between the time taken and the distance travelled to deliver the food. So we can see that it takes 25 - 30min for food to be delivered regardless the distance.

In [ ]:

```

```

Lets say we now want to see the time taken to deliver food and the age of the delivery partner:

In [11]:
```
for template in ["plotly_dark"]:

    figure = px.scatter(data_frame = data,

                       x="Delivery_person_Age",
                       y="Time_taken(min)",
                       size="Time_taken(min)",
                       color="Distance",
                       trendline="ols",
                       template=template,
                       title="Relationship Between Time Taken and Age")
    figure.show()
```

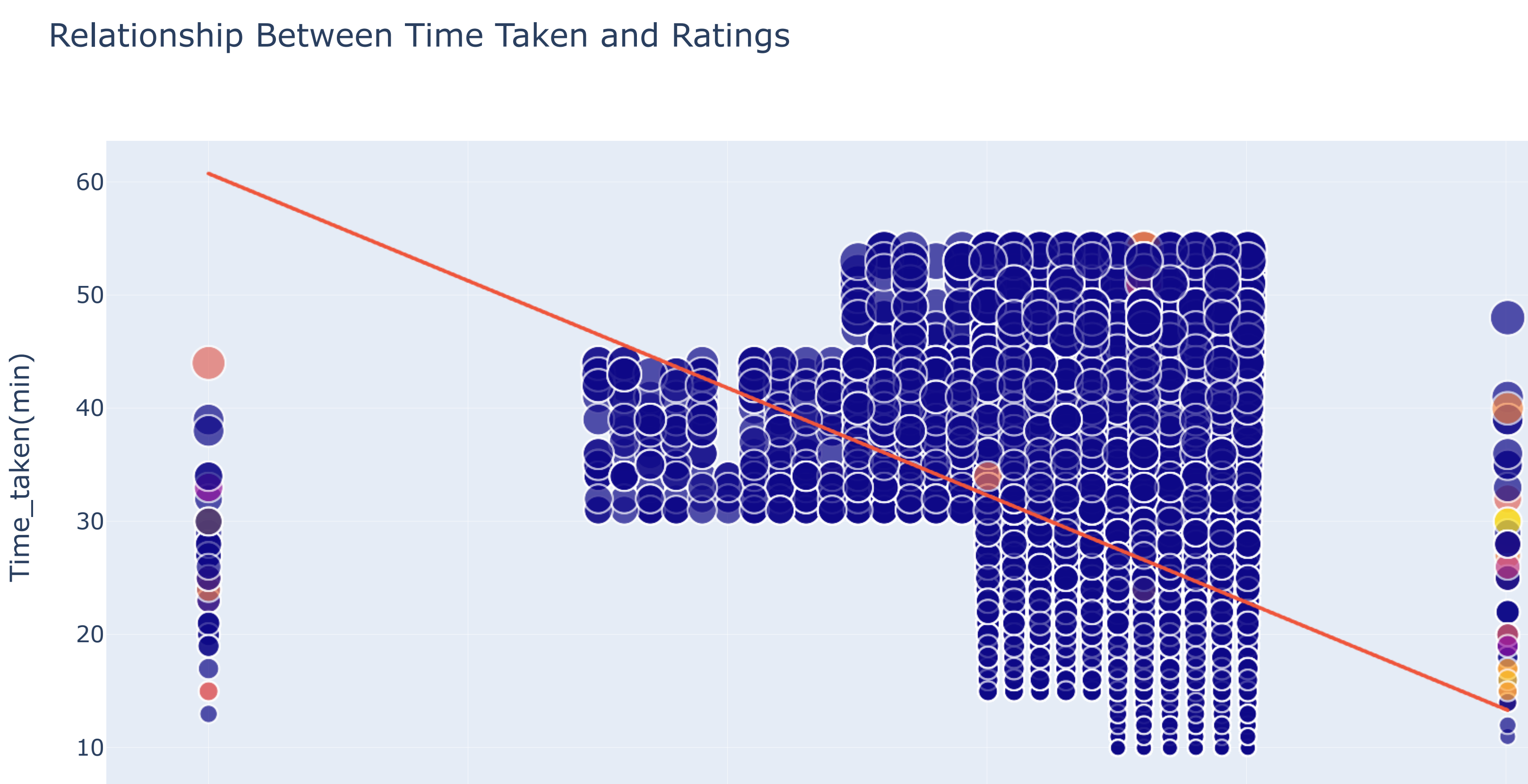


There is a linear relationship between the time taken to deliver the food and the age of the delivery partner. It means young delivery partners take less time to deliver the food compared to the elder partners.

```
In [ ]:
```

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 [12]:
```
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()
```



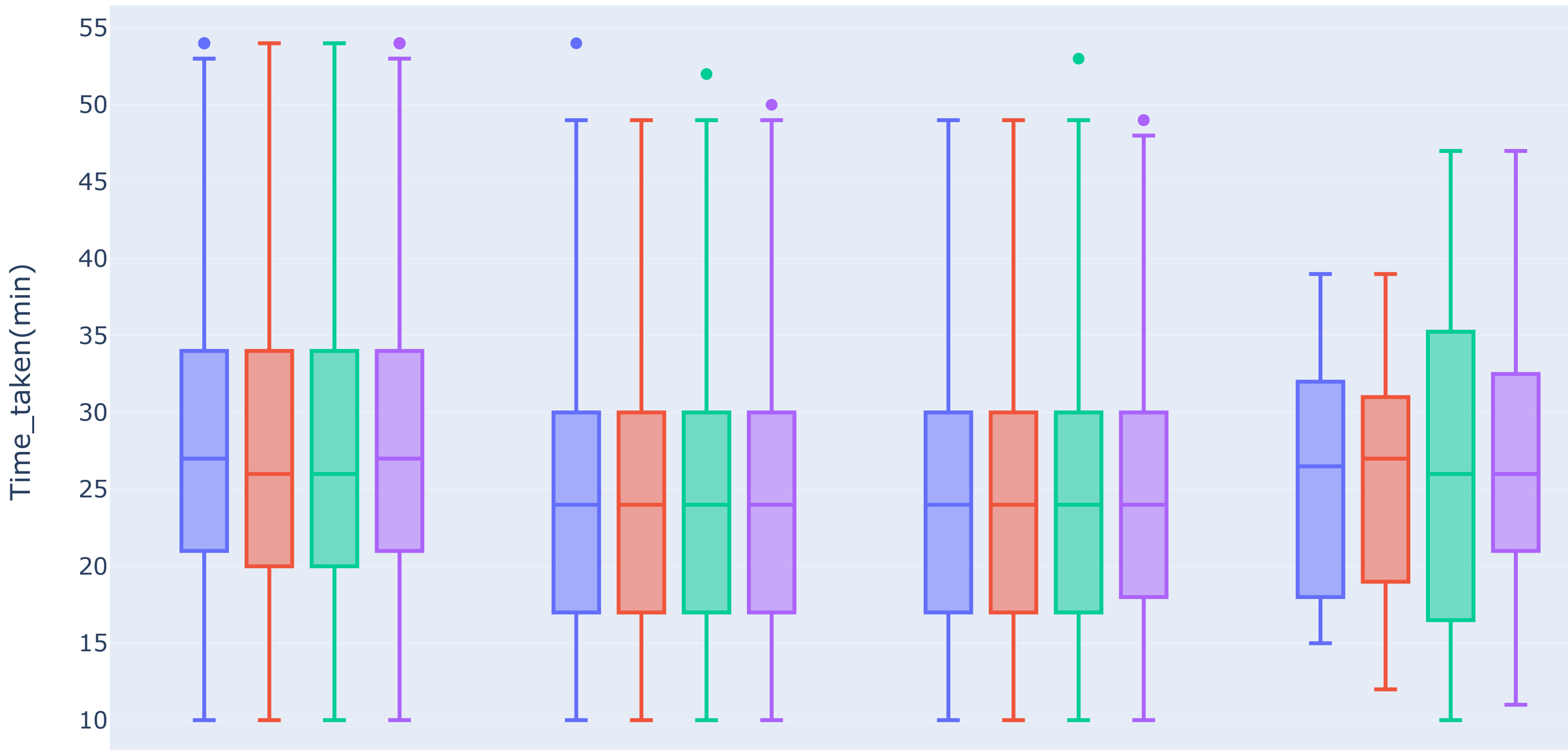
There is an inverse linear relationship between the time taken to deliver the food and the ratings of the delivery partner. It means delivery partners with higher ratings take less time to deliver the food compared to partners with low ratings.

In [ ]:

```

```

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:

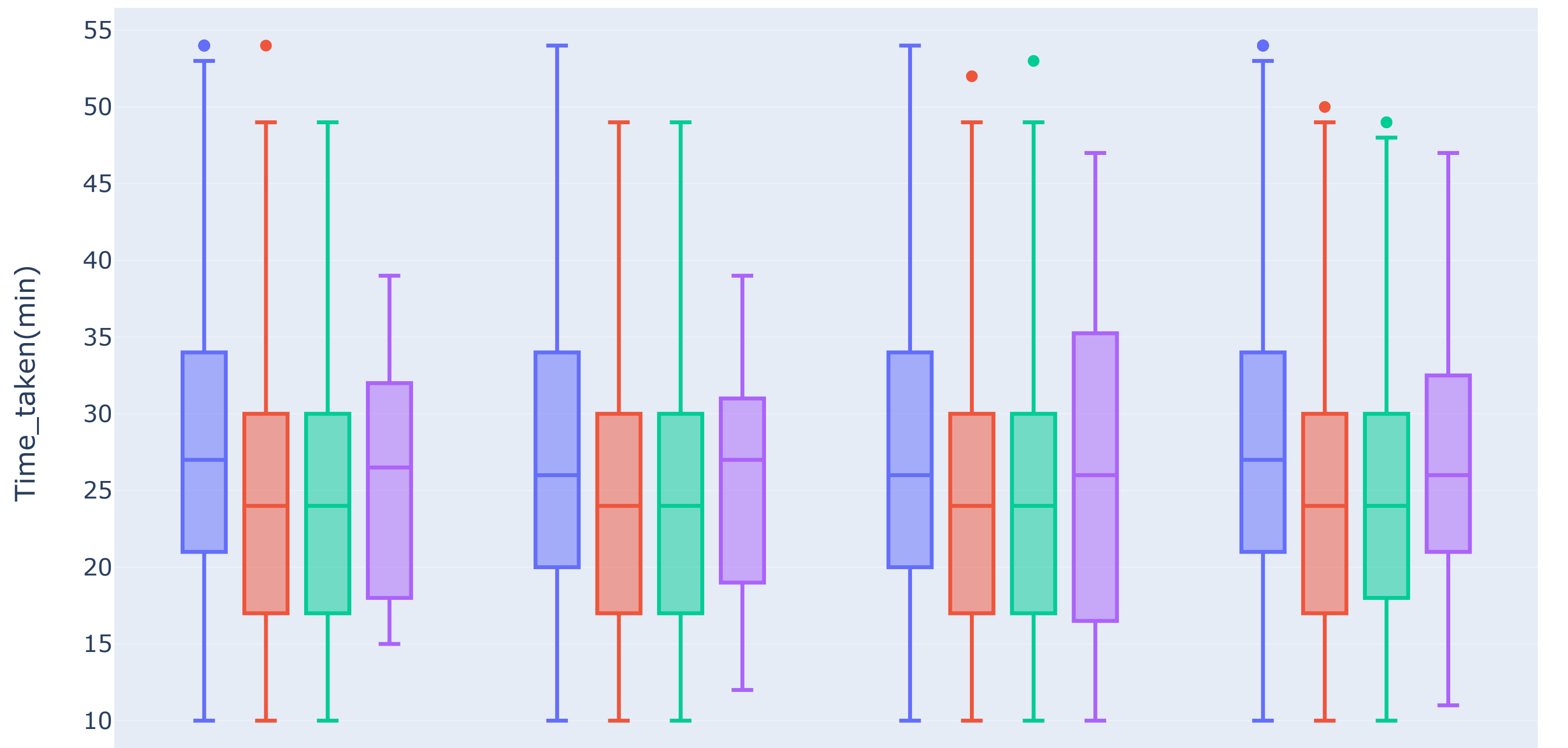
In [13]:
```
figure = px.box(data,
               x="Type_of_vehicle",
               y="Time_taken(min)",
               color="Type_of_order")
figure.show()
```



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.

But before I end, we can also look at this data in a different way by making "Type of vehicle" the distance and the "Type of order" as the x axis:

In [14]:
```
figure = px.box(data,
                x="Type_of_order",
                y="Time_taken(min)",
                color="Type_of_vehicle")
figure.show()
```



Now we can see a few differences, the bicycle take more time to make deliveries compared to ther vehicles..that's another way you can read the data, provide 2 options can help the company make descison based on that..Just a thought though.

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

*Also the type of order in relationship of vehicle type(optional)

In [ ]:

```
```

We are now done analysing the data, cleaning, manipulating and visaulising the data. Please not this process can be different based on what the company wants.

In [ ]:

```
```

Food Delivery Time Prediction Model

Now this is my favourite part, I will train a Machine Learning model for food delivery time prediction.

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

In [15]:
```
# Splitting the 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:

In [16]:
```
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
_________________________________________________________________
```

In [ ]:

Training the model:

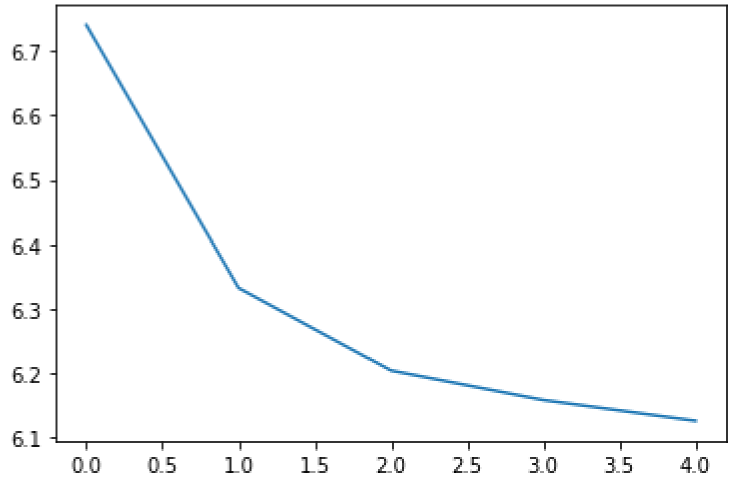
In [17]:
```
model.compile(optimizer='adam', loss='mean_absolute_error', metrics=['accuracy'])
history= model.fit(xtrain, ytrain, batch_size=1, epochs=5, validation_data=(xtest,ytest))
```

```
Epoch 1/5
41033/41033 [==============================] - 177s 4ms/step - loss: 6.7407 - accuracy: 0.0000e+00 - val_los
s: 6.7924 - val_accuracy: 0.0000e+00
Epoch 2/5
41033/41033 [==============================] - 179s 4ms/step - loss: 6.3324 - accuracy: 0.0000e+00 - val_los
s: 6.0172 - val_accuracy: 0.0000e+00
Epoch 3/5
41033/41033 [==============================] - 172s 4ms/step - loss: 6.2039 - accuracy: 0.0000e+00 - val_los
s: 6.1142 - val_accuracy: 0.0000e+00
Epoch 4/5
41033/41033 [==============================] - 175s 4ms/step - loss: 6.1580 - accuracy: 0.0000e+00 - val_los
s: 5.9485 - val_accuracy: 0.0000e+00
Epoch 5/5
41033/41033 [==============================] - 174s 4ms/step - loss: 6.1257 - accuracy: 0.0000e+00 - val_los
s: 6.6038 - val_accuracy: 0.0000e+00
```

In [18]:
```
results = model.predict(xtest)
```

```
143/143 [==============================] - 2s 3ms/step
```

In [19]:
```
plt.plot(history.history['loss'])
plt.show()
```



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

In [20]:
```
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: 23
Ratings of Previous Deliveries: 4
Total Distance: 7
1/1 [==============================] - 0s 15ms/step
Predicted Delivery Time in Minutes =  [[32.082256]]
```

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

In [ ]:

```

```

Power BI Report with Plotly

With Power BI Desktop, you can connect to multiple different sources of data, and combine them (often called modeling) into a data model. This data model lets you build visuals, and collections of visuals you can share as reports, with other people inside your organization. Most users who work on business intelligence projects use Power BI Desktop to create reports, and then use the Power BI service to share their reports with others.

The most common uses for Power BI Desktop are as follows:

*Connect to data.

*Transform and clean data to create a data model.

*Create visuals, such as charts or graphs that provide visual representations of the data.

*Create reports that are collections of visuals on one or more report pages.

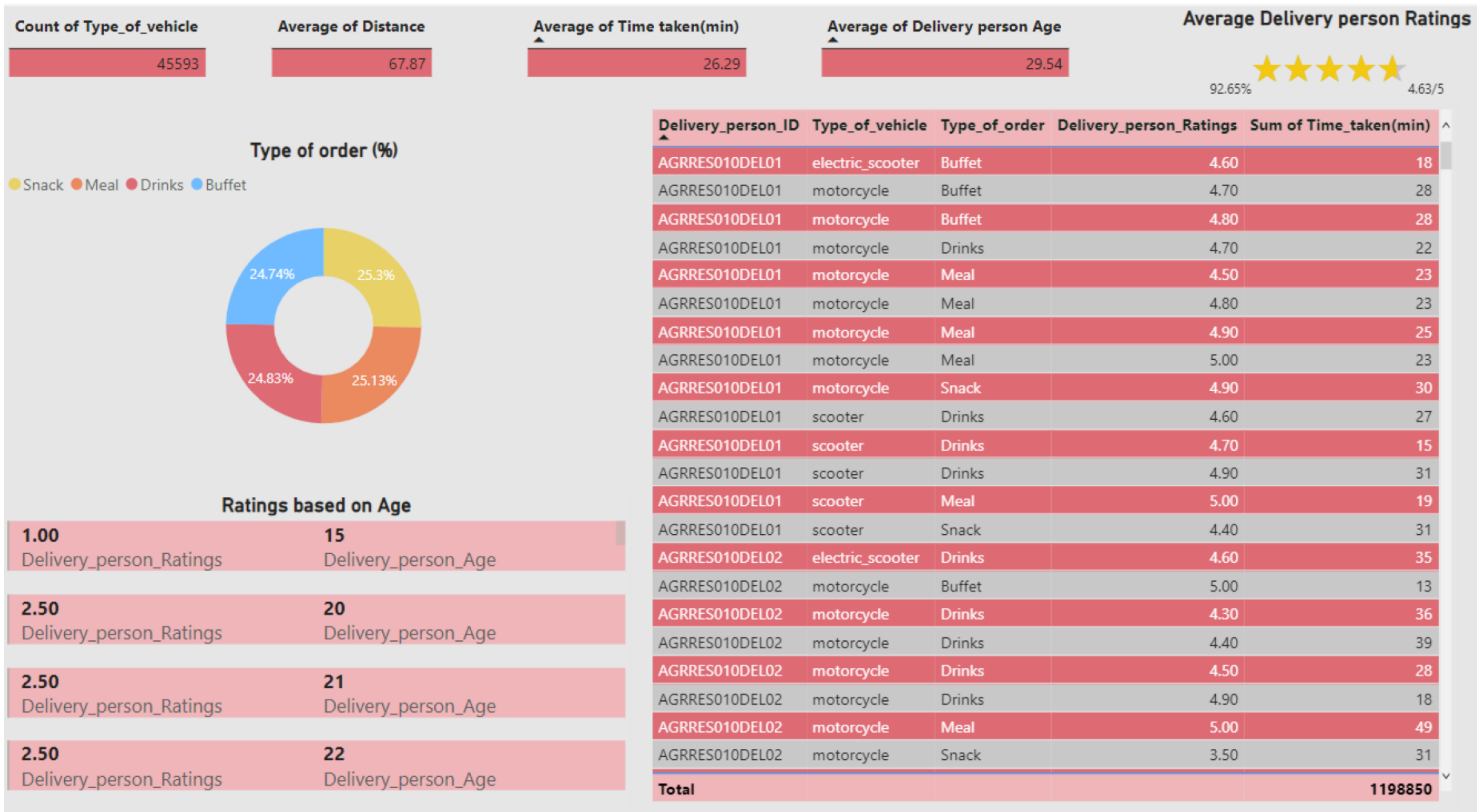
*Share reports with others by using the Power BI service.

NB!! I will visaul data just to give us different insight, optional if you already satisfied with the data you got.

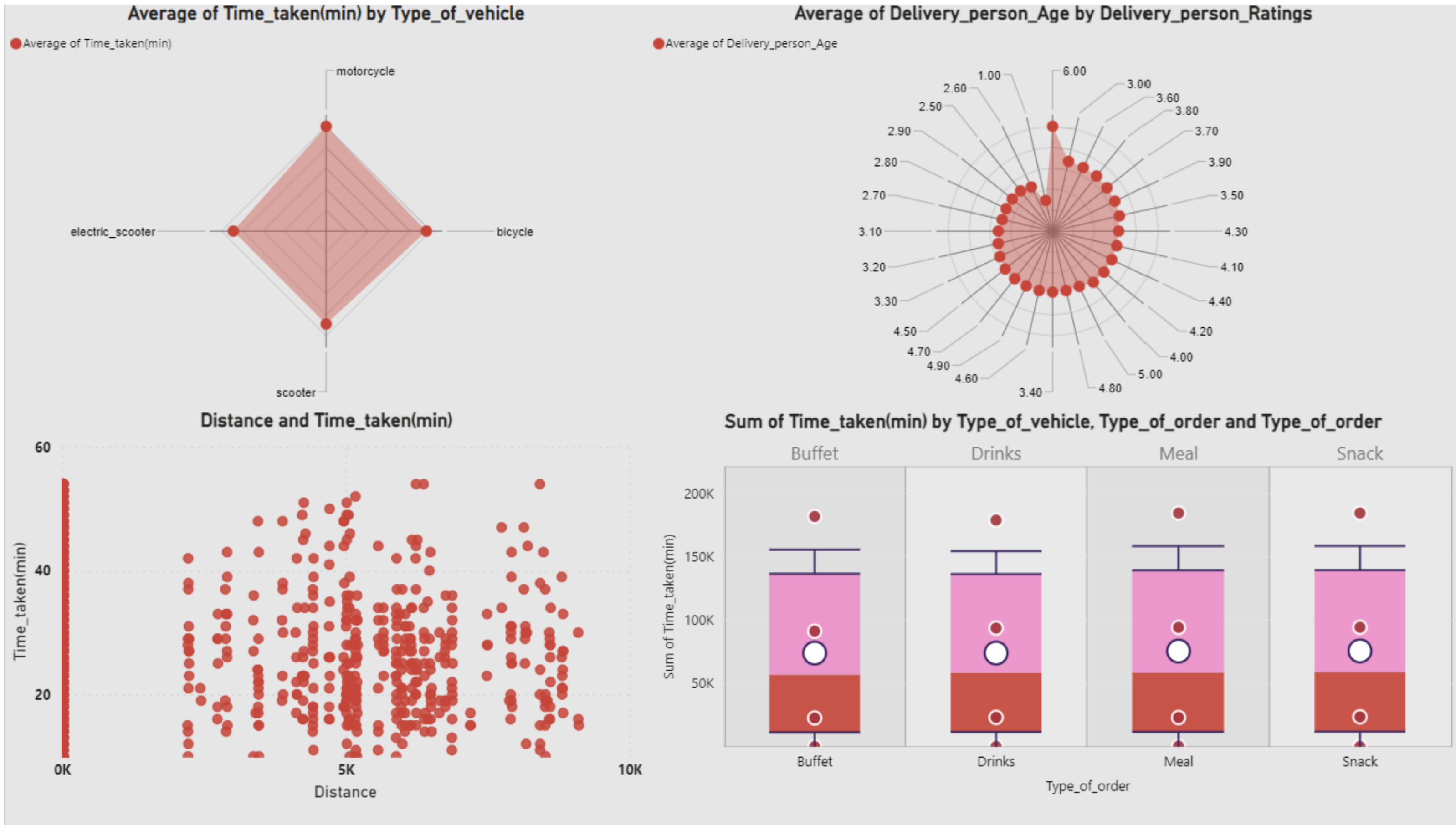
In [21]:

```
# Save the dataset and create a new updated data set.
data.to_csv("new_data.csv", index = False)
```

In [21]:
```
from skimage import io
import matplotlib.pyplot as plt
img = io.imread("Nelio.PNG")
plt.figure(dpi=400)
plt.imshow(img)
plt.axis('off')
plt.show()
```



In [22]:
```
img = io.imread("Lino.PNG")
plt.figure(dpi=400)
plt.imshow(img)
plt.axis('off')
plt.show()
```



Thank you!!

In [ ]:

```

```