Uber Fare Prediction

Team Members:
507- Rajasi Barapatre
510- Sanika Chavan
513- Akansha Dahikar

Problem Statement :- This case study is to predict the price of the Uber ride from a given pickup point to the agreed dropoff location. Evaluating the models & comparing their respective scores RMSE

Introduction:- The project is about the world's largest taxi company Uber inc. In this project, we're looking to predict the fare for their future transactional cases. Uber delivers service to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. Eventually, it becomes really important to estimate the fare prices accurately according to the distance, and time.

The dataset contains the following fields:

key - a unique identifier for each trip

fare_amount - the cost of each trip in usd

pickup_datetime - date and time when the meter was engaged

passenger_count - the number of passengers in the vehicle (driver entered value)

pickup_longitude - the longitude where the meter was engaged

pickup_latitude - the latitude where the meter was engaged

dropoff_longitude - the longitude where the meter was disengaged

dropoff_latitude - the latitude where the meter was disengaged

Pickup - the pickup coordinate of latitude and longitude

Dropoff- the dropoff coordinate of latitude and longitude

Time- time column which is assigned 0-3 values according to the traffic hours

Distance- calculated using Haversine Formula

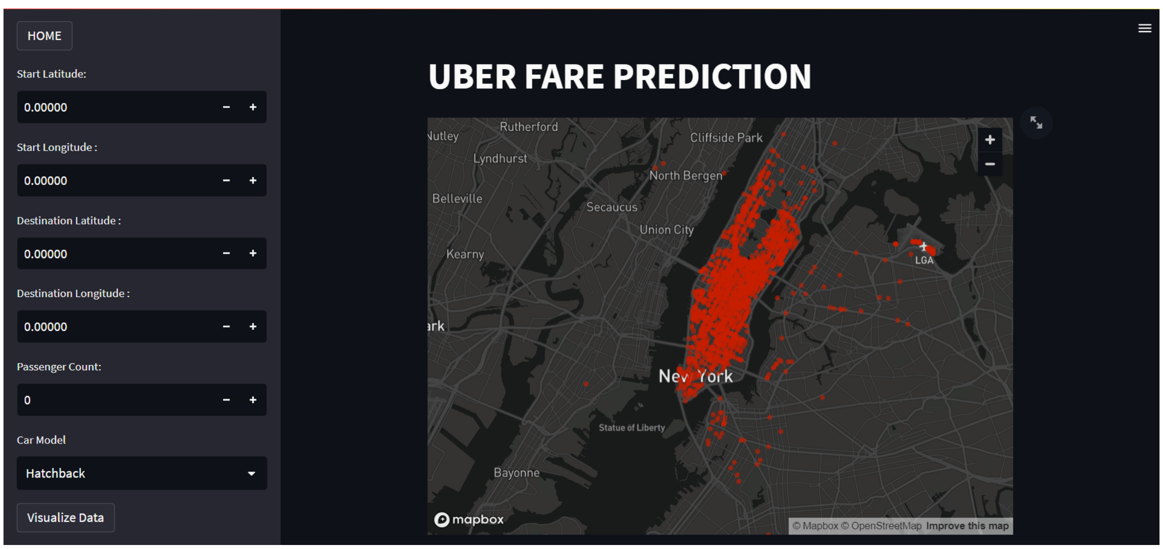
Csv File link:

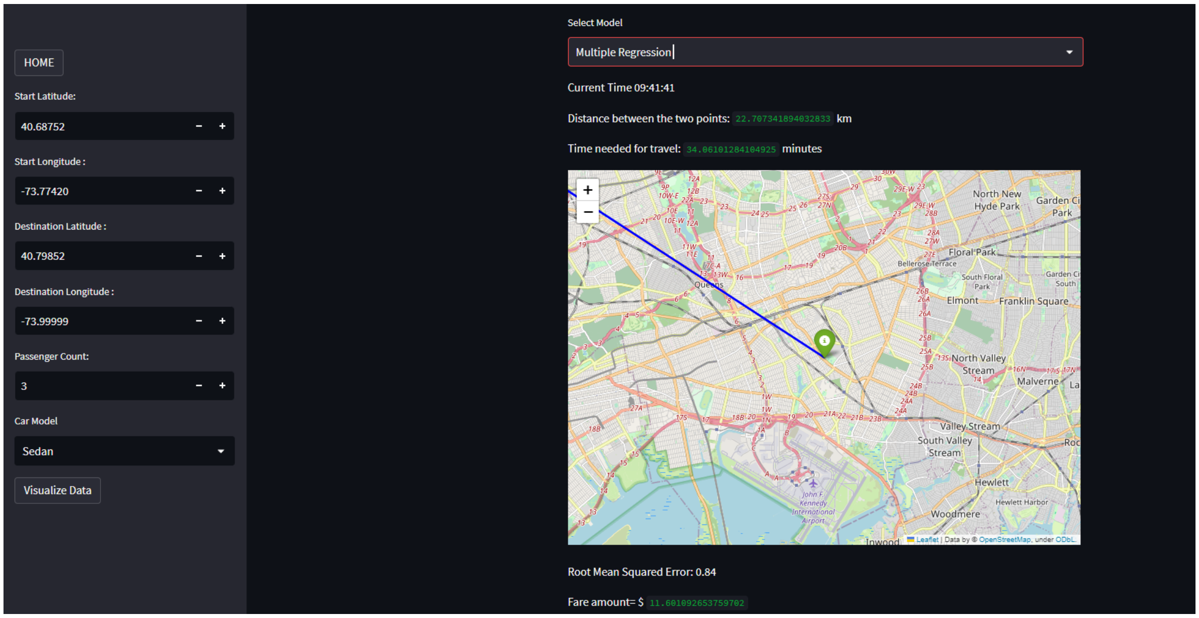
https://drive.google.com/file/d/1wAxmLjmwO0T6kWsGrY8I4UaO1QWrLArw/view?usp=sharing

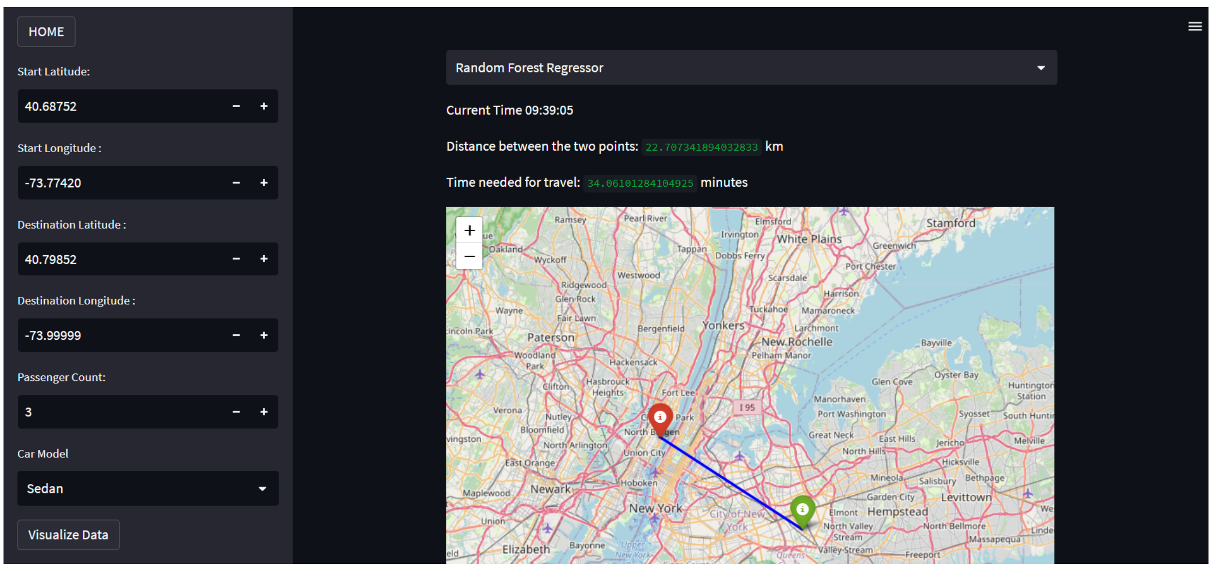
Library Used :

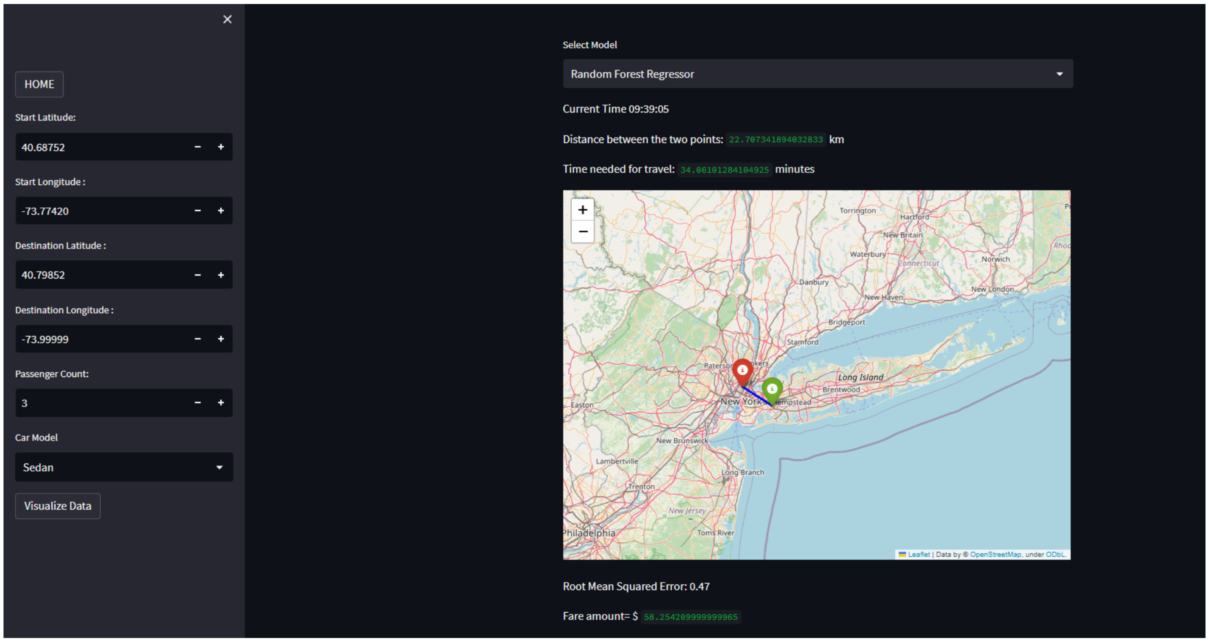
Pandas, sklearn,numpy,seaborn,matplotlib

Snippets of the website:

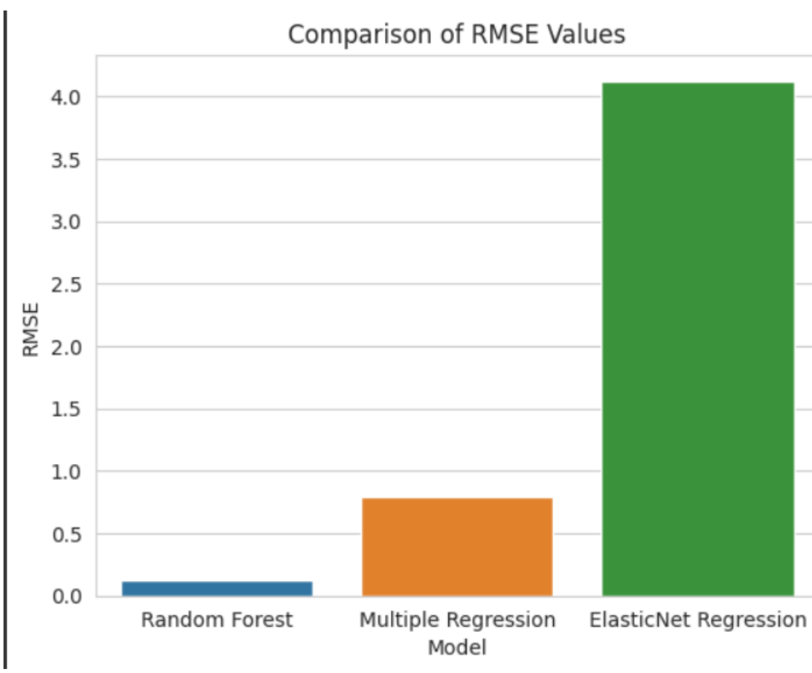


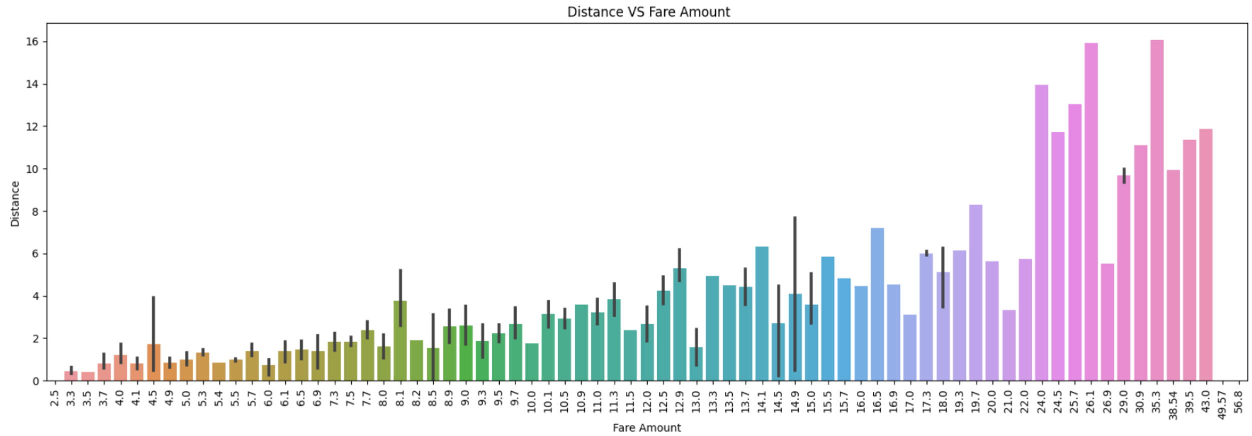

HOME
Start Latitude:
40.68752
Start Longitude :
-73.77420
Destination Latitude :
40.79852
Destination Longitude :
-73.99999
Passenger Count:
3
Car Model
Sedan
Visualize Data
Select Model
Multiple Regression
Current Time 09:41:41
Distance between the two points: 22.707341894032833 km
Time needed for travel: 34.06101284104925 minutes
Root Mean Squared Error: 0.84
Fare amount= $ 11.601092653759702

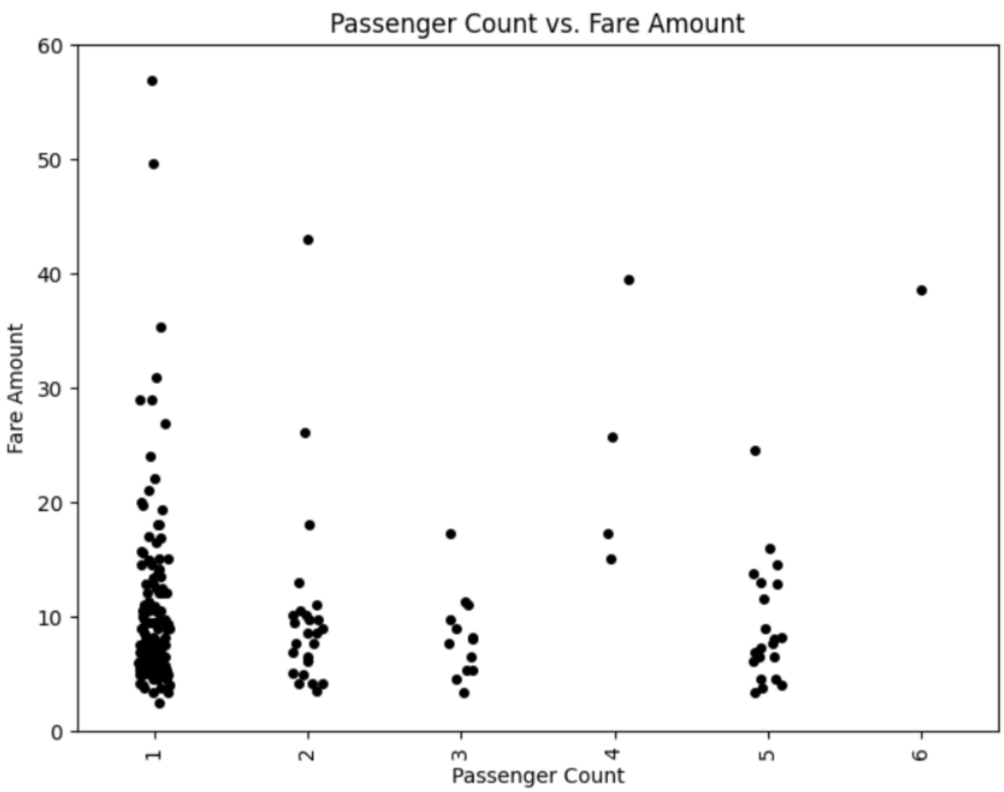

HOME
Start Latitude:
40.68752
Start Longitude :
-73.77420
Destination Latitude :
40.79852
Destination Longitude :
-73.99999
Passenger Count:
3
Car Model
Sedan
Visualize Data
Random Forest Regressor
Current Time 09:39:05
Distance between the two points: 22.707341894032833 km
Time needed for travel: 34.06101284104925 minutes

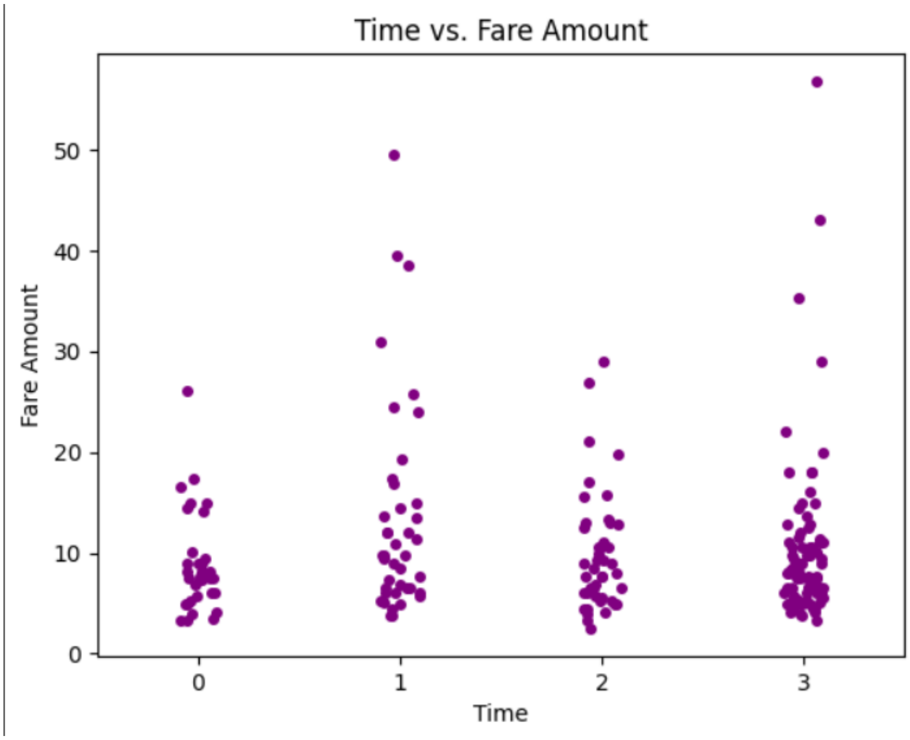

HOME
Start Latitude:
40.68752
Start Longitude :
-73.77420
Destination Latitude :
40.79852
Destination Longitude :
-73.99999
Passenger Count:
3
Car Model
Sedan
Visualize Data
Select Model
Random Forest Regressor
Current Time 09:39:05
Distance between the two points: 22.707341894032833 km
Time needed for travel: 34.06101284104925 minutes
Root Mean Squared Error: 0.47
Fare amount= $ 58.254209999999965

Graphs:


Comparison of RMSE Values
RMSE
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Random Forest
Multiple Regression Model
ElasticNet Regression


Distance VS Fare Amount
Distance
0
2
4
6
8
10
12
14
16
Fare Amount
2.5 3.3 3.5 3.7 4.0 4.1 4.5 4.9 5.0 5.3 5.4 5.5 5.7 6.0 6.1 6.5 6.9 7.3 7.5 7.7 8.0 8.1 8.2 8.5 8.9 9.0 9.3 9.5 9.7 10.0 10.1 10.5 10.9 11.0 11.3 11.5 12.0 12.5 12.9 13.0 13.3 13.5 13.7 14.1 14.5 14.9 15.0 15.5 15.7 16.0 16.5 16.9 17.0 17.3 18.0 19.3 19.7 20.0 21.0 22.0 24.0 24.5 25.7 26.1 26.9 29.0 30.9 35.3 38.54 39.5 43.0 49.57 56.8


Passenger Count vs. Fare Amount
Fare Amount
0
10
20
30
40
50
60
Passenger Count
1
2
3
4
5
6



CODE:

```
import streamlit as st
import pandas as pd
import folium
from geopy import distance
from streamlit_folium import folium_static
from datetime import datetime, timedelta
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LinearRegression
from math import radians,sin, cos, sqrt, asin
from sklearn.linear_model import ElasticNet
from sklearn.datasets import make_regression
from PIL import Image
from sklearn.metrics import mean_squared_error, r2_score
```

```
st.title("UBER FARE PREDICTION")

# Load the data from CSV file
data =
pd.read_csv(r"C:\Users\91846\OneDrive\Desktop\Uber\Uber_Fare.csv")

# Extract latitude and longitude columns from the data
lat = data['pickup_latitude']
lon = data['pickup_longitude']

# Create a DataFrame with latitude and longitude columns
locations = pd.DataFrame({'latitude': lat, 'longitude': lon})
#button_style = "background-color: blue; color: white;"
home=st.sidebar.button("HOME")

if(home==True):
    # Plot the locations on a map using Streamlit's map function
    st.map(locations)

# Get user input for latitude and longitude
lat1 = st.sidebar.number_input("Start Latitude:",format="%.5f")
lon1 = st.sidebar.number_input("Start Longitude :",format="%.5f")
lat2 = st.sidebar.number_input("Destination Latitude
:",format="%.5f")
lon2 = st.sidebar.number_input("Destination Longitude
:",format="%.5f")
passenger = st.sidebar.number_input("Passenger Count:",step=1)
car=st.sidebar.selectbox( "Car Model",('Hatchback','Sedan','SUV'))

# Calculate the distance between the two points using geopy.distance
coord1 = (lat1, lon1)
coord2 = (lat2, lon2)
dist = distance.distance(coord1, coord2).km

currentDateAndTime = datetime.now()
currentTime = currentDateAndTime.strftime("%H:%M:%S")

cur=0
if '00:00:00' < currentTime < '07:00:00':
    cur=1
elif '07:00:00'<= currentTime < '12:00:00':
    cur=2
elif '12:00:00' <=currentTime < '17:00:00':
    cur=3
else:
    cur=4

speed=0
if cur==1:
    speed=70
elif cur==2:
```

```
    speed=40
elif cur==3:
    speed=50
else:
    speed=40

time_needed = dist / speed

time=[]
for i in range(0,1999):
  temp=data['pickup_datetime'][i].split(' ')[1]
  time.append(temp)

data['Time']=time
data.drop(['key'], axis=1, inplace= True)

# convert 'Time' column to datetime objects
data['Time'] = pd.to_datetime(data['Time'], format='%H:%M:%S')
for i in range(len(data)):

  if datetime.strptime('00:00:00', '%H:%M:%S') < data.loc[i, 'Time']
< datetime.strptime('06:00:00', '%H:%M:%S'):
    data.loc[i, 'Time'] = 0
  elif datetime.strptime('06:00:00', '%H:%M:%S') <= data.loc[i,
'Time'] < datetime.strptime('12:00:00', '%H:%M:%S'):
    data.loc[i, 'Time'] = 1
  elif datetime.strptime('12:00:00', '%H:%M:%S') <=data.loc[i,
'Time'] < datetime.strptime('17:00:00', '%H:%M:%S'):
    data.loc[i, 'Time'] = 2
  else:
    data.loc[i, 'Time'] = 3

# convert 'Time' column back to integers
data['Time'] = data['Time'].astype(int)

data.drop(['pickup_datetime'], axis=1, inplace= True)

pickup=[]
for i in range(0,1999):
  p=[data['pickup_latitude'][i],data['pickup_longitude'][i]]
  pickup.append(p)

dropoff=[]
for i in range(0,1999):
  p=[data['dropoff_latitude'][i],data['dropoff_longitude'][i]]
  dropoff.append(p)

data['Drop_off']=dropoff
data['Pick_up']=pickup

# Haversine Formula
distance=[]
```

```
R = 6371.0
for x in range(0,1999):
  p=radians(data['Pick_up'][x][0])
  q=radians(data['Pick_up'][x][1])
  r=radians(data['Drop_off'][x][0])
  s=radians(data['Drop_off'][x][1])
  dlon = q - s
  dlat = p - r
  a = (sin(dlat/2))**2 + cos(p) * cos(r) * (sin(dlon/2))**2
  c = 2 * asin(sqrt(a))
  dis= R * c
  distance.append(dis)
data['Distance']=distance



option=st.selectbox("Select Model",('','Random Forest
Regressor','Multiple Regression','Elastic Net Regression'))
if option == 'Random Forest Regressor':
#DISPLAY

# Display the distance between the two points
    st.write("Current Time",currentTime)
    st.write("Distance between the two points:", dist, "km")

    if time_needed >= 1:
        st.write("Time needed for travel:",time_needed, "hours")
    elif time_needed <1:
        st.write("Time needed for travel:",time_needed*60, "minutes")

    # Create a map using folium
    m = folium.Map(location=[lat1, lon1], zoom_start=12)

    # Add markers for the two points
    folium.Marker(location=[lat1, lon1], tooltip="Point 1",
icon=folium.Icon(color="green")).add_to(m)
    folium.Marker(location=[lat2, lon2], tooltip="Point 2",
icon=folium.Icon(color="red")).add_to(m)

    # Add a line connecting the two points
    folium.PolyLine(locations=[(lat1, lon1), (lat2, lon2)],
color='blue').add_to(m)

    # Display the map
    folium_static(m)


    X = data[['Distance','Time','passenger_count']]
    Y = data['fare_amount']
    X_train,X_test,Y_train,Y_test=train_test_split(X, Y,
test_size=0.25, shuffle= True)
    random=RandomForestRegressor()
```

```
    random.fit(X_train,Y_train)
    Y_pred=random.predict(X_test)

    mse = mean_squared_error(Y_test, Y_pred)
    r2 = r2_score(Y_test, Y_pred)
    rmse_rf = np.sqrt(mse) / np.mean(Y_test)

    # Print results
    st.write("Root Mean Squared Error: {:.2f}".format(rmse_rf ))

    sample=np.array([[dist,time_needed,passenger]])
    predict1=random.predict(sample)

    if car=='Hatchback':
        st.write("Fare amount= $",predict1[0])
    elif car=='Sedan':
        st.write("Fare amount= $",predict1[0]+predict1[0]*0.05)
    elif car=='SUV':
        st.write("Fare amount= $",predict1[0]+predict1[0]*0.1)

elif option=='Multiple Regression':
    st.write("Current Time",currentTime)

# Display the distance between the two points
    st.write("Distance between the two points:", dist, "km")

    if time_needed >= 1:
        st.write("Time needed for travel:",time_needed, "hours")
    elif time_needed <1:
        st.write("Time needed for travel:",time_needed*60, "minutes")

    # Create a map using folium
    m = folium.Map(location=[lat1, lon1], zoom_start=12)

    # Add markers for the two points
    folium.Marker(location=[lat1, lon1], tooltip="Point 1",
icon=folium.Icon(color="green")).add_to(m)
    folium.Marker(location=[lat2, lon2], tooltip="Point 2",
icon=folium.Icon(color="red")).add_to(m)

    # Add a line connecting the two points
    folium.PolyLine(locations=[(lat1, lon1), (lat2, lon2)],
color='blue').add_to(m)

    # Display the map
    folium_static(m)

    a = data[['Distance', 'Time','passenger_count']]
    b=data['fare_amount']
    a_train,a_test,b_train,b_test=train_test_split(a, b,
test_size=0.25, shuffle= True)
    lin=LinearRegression()
```

```
    lin.fit(a_train,b_train)
    a_pred=lin.predict(a_test)
    # cd=lin.score(a_train,b_train)
    # st.write("Coefficient of Determination : ",cd)
    mse = mean_squared_error(b_test, a_pred)
    r2 = r2_score(b_test, a_pred)
    rmse_mr = np.sqrt(mse) / np.mean(b_test)

    # Print results
    st.write("Root Mean Squared Error: {:.2f}".format(rmse_mr ))
    sample=np.array([[dist,time_needed,passenger]])
    predict2=lin.predict(sample)
    if car=='Hatchback':
        st.write("Fare amount= $",predict2[0])
    elif car=='Sedan':
        st.write("Fare amount= $",predict2[0]+predict2[0]*0.05)
    elif car=='SUV':
        st.write("Fare amount= $",predict2[0]+predict2[0]*0.1)

elif option =='Elastic Net Regression':
    st.write("Current Time",currentTime)
# Display the distance between the two points
    st.write("Distance between the two points:", dist, "km")

    if time_needed >= 1:
        st.write("Time needed for travel:",time_needed, "hours")
    elif time_needed <1:
        st.write("Time needed for travel:",time_needed*60, "minutes")

    # Create a map using folium
    m = folium.Map(location=[lat1, lon1], zoom_start=12)

    # Add markers for the two points
    folium.Marker(location=[lat1, lon1], tooltip="Point 1",
icon=folium.Icon(color="green")).add_to(m)
    folium.Marker(location=[lat2, lon2], tooltip="Point 2",
icon=folium.Icon(color="red")).add_to(m)

    # Add a line connecting the two points
    folium.PolyLine(locations=[(lat1, lon1), (lat2, lon2)],
color='blue').add_to(m)

    # Display the map
    folium_static(m)

    P= data[['Distance', 'Time','passenger_count']]
    Q=data['fare_amount']
    # P, Q = make_regression(n_features=2, random_state=0)
    regr = ElasticNet()
    P_train,P_test,Q_train,Q_test=train_test_split(P, Q,
test_size=0.25, shuffle= True)
    regr.fit(P_train, Q_train)
```

```
    Q_pred=regr.predict(P_test)
    mse = mean_squared_error(Q_test, Q_pred)
    r2 = r2_score(Q_test, Q_pred)
    rmse_en = np.sqrt(mse) / np.mean(Q_test)

    # Print results
    st.write("Root Mean Squared Error: {:.2f}".format(rmse_en))
    sample=np.array([[dist,time_needed,passenger]])
    predict3=regr.predict(sample)
    if car=='Hatchback':
        st.write("Fare amount= $",predict3[0])
    elif car=='Sedan':
        st.write("Fare amount= $",predict3[0]+predict3[0]*0.05)
    elif car=='SUV':
        st.write("Fare amount= $",predict3[0]+predict3[0]*0.1)

graphs=st.sidebar.button("Visualize Data ")
if graphs==True:
    df_image=Image.open('D _ F.png')
    st.image(df_image, caption='DISTANCE VS FARE AMOUNT')
    st.write("")
    pf_image=Image.open('P _F.png')
    st.image(pf_image, caption='PASSENGER COUNT VS FARE AMOUNT')
    st.write("")
    tf_image=Image.open('T_F.png')
    st.image(tf_image, caption='TIME VS FARE AMOUNT')
    comp_image=Image.open('Compare.png')
    st.image(comp_image, caption='COMPARISON')
```

Conclusion:

In conclusion, our Uber Fare Prediction model has been trained and evaluated using three different regression models: Random Forest, Multiple Regression, and ElasticNet Regression. Based on our model training, we have found that the Random Forest and Multiple Regression models provide better accuracy than the ElasticNet Regression model in predicting Uber fares. This suggests that the Random Forest and Multiple Regression models are better suited to our problem domain and are more effective in capturing the complexities of the dataset. These models may be more useful in real-world applications, as they can more accurately predict Uber fares for different scenarios and help users make informed decisions about their travel plans.

Overall, the results of our model training demonstrate the importance of selecting the right machine learning algorithm for the task at hand and the value of thorough experimentation and evaluation to ensure the accuracy and reliability of our models.

References:

[1] https://docs.streamlit.io/library/get-started/main-concepts

[2] https://scikit-learn.org/stable/getting_started.html

[3] Vinod Chandra S. S., Anand Hareendran S., 'Artificial Intelligence and machine learning', PHI, (2014), ISBN 978-81-