

As the decision tree regression and random forest regressor got more accuracy. A model is made using random forest regression to predict the TRIP FARE by taking no.of passengers, trip distance, pick up longitude, pick up latitude, drop off longitude, drop off latitude.

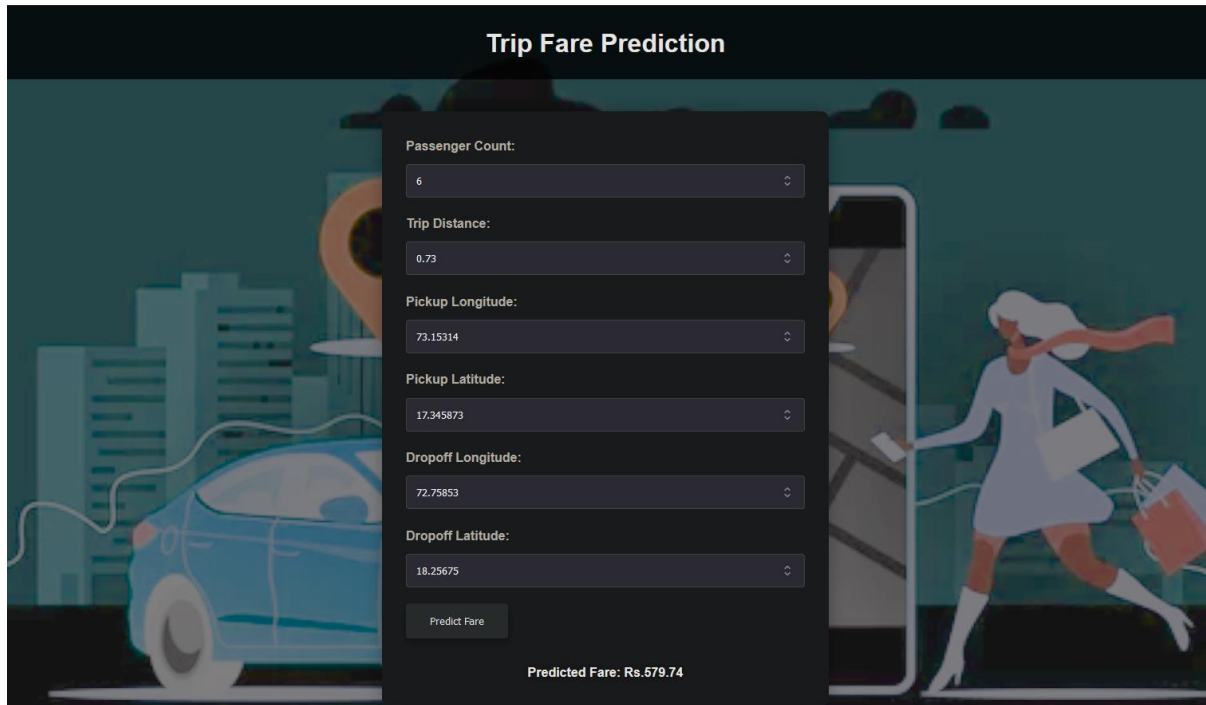
```
1 import pandas as pd
2 from sklearn.ensemble import RandomForestRegressor
3 from sklearn.feature_selection import SelectKBest, f_regression
4 from sklearn.pipeline import Pipeline
5 from sklearn.compose import ColumnTransformer
6 from sklearn.preprocessing import StandardScaler, OneHotEncoder
7 from sklearn.impute import SimpleImputer
8 import joblib
9
10 # Load the dataset
11 data = pd.read_csv('uber_data.csv')
12
13 # Convert 'payment_type' column to numeric, handling errors by coercing to NaN
14 data['payment_type'] = pd.to_numeric(data['payment_type'], errors='coerce')
15
16 # Drop rows with missing values
17 data.dropna(inplace=True)
18
19 # Define features and target variable
20 X = data.drop(columns=['total_amount']) # Features
21 y = data['total_amount'] # Target variable
22
23 # Define numerical and categorical features
24 numeric_features = ['passenger_count', 'trip_distance']
25 categorical_features = ['pickup_latitude', 'pickup_longitude', 'dropoff_latitude', 'dropoff_longitude']
26
27 # Define preprocessing steps for numerical and categorical features
28 numeric_transformer = Pipeline(steps=[
29     ('imputer', SimpleImputer(strategy='mean')),
30     ('scaler', StandardScaler())
31 ])
32
33 categorical_transformer = Pipeline(steps=[
34     ('imputer', SimpleImputer(strategy='most_frequent')),
35     ('onehot', OneHotEncoder(handle_unknown='ignore'))
36 ])
37
38 # Combine preprocessing steps for numerical and categorical features
39 preprocessor = ColumnTransformer(
40     transformers=[
41         ('num', numeric_transformer, numeric_features),
42         ('cat', categorical_transformer, categorical_features)
43     ])
44
45 # Define the model
46 model = Pipeline(steps=[('preprocessor', preprocessor),
47     ('feature_selection', SelectKBest(score_func=f_regression, k=5)),
48     ('regressor', RandomForestRegressor())])
49
50 # Train the model
51 model.fit(X, y)
52
53 # Save the trained model to a file
54 joblib.dump(model, 'model.pkl')
55
56 # Take user input for prediction
57 passenger_count = int(input("Enter the number of passengers: "))
58 trip_distance = float(input("Enter the trip distance: "))
59 pickup_latitude = float(input("Enter the pickup latitude: "))
60 pickup_longitude = float(input("Enter the pickup longitude: "))
61 dropoff_latitude = float(input("Enter the dropoff latitude: "))
62 dropoff_longitude = float(input("Enter the dropoff longitude: "))
63
64 # Make predictions on new data
65 new_data = pd.DataFrame({
66     'passenger_count': [passenger_count],
67     'trip_distance': [trip_distance],
68     'pickup_latitude': [pickup_latitude],
69     'pickup_longitude': [pickup_longitude],
70     'dropoff_latitude': [dropoff_latitude],
71     'dropoff_longitude': [dropoff_longitude]
72 })
73
74 # Predict trip fare
75 predicted_fare = model.predict(new_data)
76 print("Predicted Trip Fare:", (predicted_fare)*80)
77
```

The trained model was saved in the form of pkl (python pickle). It's often used to save trained models to disk so that they can be reused later without needing to retrain them every time. When a POST request is made to this endpoint, it expects JSON data containing the features required for prediction. It then converts this JSON data into a panda Data Frame and passes it through the trained model pipeline to make predictions. Finally, it returns the prediction as a JSON response.

```
1 from flask import Flask, render_template, request, jsonify
2 import pandas as pd
3 from sklearn.ensemble import RandomForestRegressor
4 from sklearn.pipeline import Pipeline
5 from sklearn.compose import ColumnTransformer
6 from sklearn.preprocessing import OneHotEncoder, StandardScaler
7 import joblib
8
9 app = Flask(__name__, template_folder='template')
10
11 # Load the trained model
12 model_pipeline = joblib.load('model.pkl')
13
14 # Load the HTML page
15 @app.route('/')
16 def home():
17     return render_template('index.html')
18
19 # Predict fare
20 @app.route('/predict', methods=['POST'])
21 def predict():
22     data = request.get_json(force=True)
23     new_data = pd.DataFrame(data, index=[0])
24
25     fare_prediction = model_pipeline.predict(new_data)
26
27     return jsonify(float(fare_prediction[0])*80)
28
29 if __name__ == '__main__':
30     app.run(debug=True)
```

OUTPUT:

The final output of the project is a web page that asks passenger count, trip distance, pickup longitude, pickup latitude, drop off longitude, drop off latitude and predicts the trip fare using the random forest regressor model.



The screenshot shows a web application titled "Trip Fare Prediction". It features a dark-themed interface with a background illustration of a city street at night, including a blue car, a person walking, and a smartphone. The application has several input fields for user data:

- Passenger Count: 6
- Trip Distance: 0.73
- Pickup Longitude: 73.15314
- Pickup Latitude: 17.345873
- Dropoff Longitude: 72.75853
- Dropoff Latitude: 18.25675

Below the input fields is a "Predict Fare" button. At the bottom of the form, the predicted fare is displayed as "Predicted Fare: Rs.579.74".

CONCLUSION:

In this project we first performed preprocessing and EDA steps on dataset and then we used four machine learning algorithms and selected the algorithm with highest accuracy for training the model by splitting the dataset. After training the model we saved it in .pkl file for future use. We then made a flask based web page which uses the saved pkl model for prediction of TRIP FARE.