```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
from geopy.distance import geodesic

data = pd.read_csv("uber.csv")  # Replace with the path to your data
file

data.head()
```

```
   Unnamed: 0                             key  fare_amount  \
0    24238194    2015-05-07 19:52:06.0000003          7.5
1    27835199    2009-07-17 20:04:56.0000002          7.7
2    44984355   2009-08-24 21:45:00.00000061         12.9
3    25894730    2009-06-26 08:22:21.0000001          5.3
4    17610152  2014-08-28 17:47:00.000000188         16.0

           pickup_datetime  pickup_longitude  pickup_latitude  \
0  2015-05-07 19:52:06 UTC        -73.999817        40.738354
1  2009-07-17 20:04:56 UTC        -73.994355        40.728225
2  2009-08-24 21:45:00 UTC        -74.005043        40.740770
3  2009-06-26 08:22:21 UTC        -73.976124        40.790844
4  2014-08-28 17:47:00 UTC        -73.925023        40.744085

   dropoff_longitude  dropoff_latitude  passenger_count
0         -73.999512         40.723217                1
1         -73.994710         40.750325                1
2         -73.962565         40.772647                1
3         -73.965316         40.803349                3
4         -73.973082         40.761247                5
```

```
def calculate_distance(row):
    pickup = (row['pickup_latitude'], row['pickup_longitude'])
    dropoff = (row['dropoff_latitude'], row['dropoff_longitude'])
    return geodesic(pickup, dropoff).km

# Filter out rows where latitude values are invalid
data = data[(data['pickup_latitude'] >= -90) &
(data['pickup_latitude'] <= 90)]
data = data[(data['dropoff_latitude'] >= -90) &
(data['dropoff_latitude'] <= 90)]

data['distance_km'] = data.apply(calculate_distance, axis=1)

X = data[['distance_km', 'passenger_count']]
y = data['fare_amount']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

```
Mean Squared Error: 103.9774775531441
```

```
r2 = r2_score(y_test,y_pred)
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test,y_pred)

print(f"  R2 Score:{r2}")
print(f"  R2 Score:{mse}")
print(f"  R2 Score:{rmse}")
print(f"  R2 Score:{mae}")
```

```
  R2 Score:0.0009140854780474994
  R2 Score:103.9774775531441
  R2 Score:10.196934713586437
  R2 Score:6.05294545238674
```

```
new_ride = {
    'distance_km': 5.0,  # Example distance in kilometers
    'passenger_count': 1  # Example passenger count
}
new_ride_df = pd.DataFrame([new_ride])

predicted_fare = model.predict(new_ride_df)
print("Predicted Fare:", predicted_fare[0])
```

```
Predicted Fare: 11.31596978434554
```

```
#random forest model

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

rf_model.fit(X_train,y_train)
```