Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks: 1. Pre-process the dataset.2. Identify outliers. 3. Check the correlation. 4. Implement random forest model. 5. Evaluate the model using R2, RMSE, etc. Use Uber Dataset: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv("C:/Users/Atharva/OneDrive/Documents/BE PROJECT/LPiii/uber.csv")
print(data.head())
```

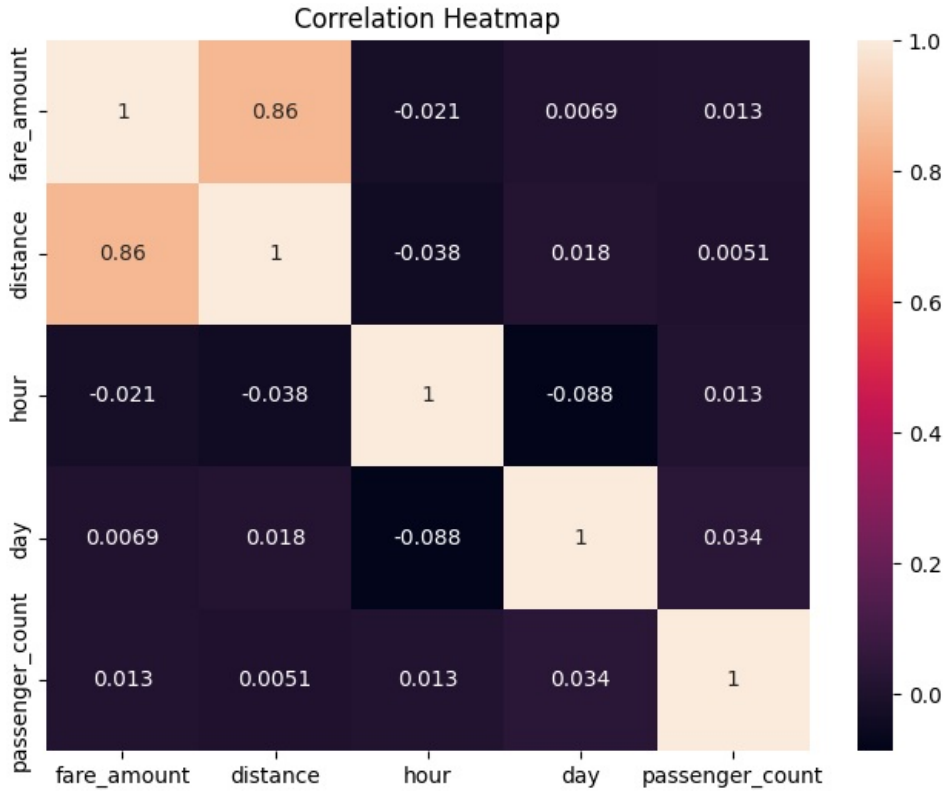
```
   Unnamed: 0                            key  fare_amount  \
0    24238194    2015-05-07 19:52:06.0000003          7.5
1    27835199    2009-07-17 20:04:56.0000002          7.7
2    44984355   2009-08-24 21:45:00.00000061         12.9
3    25894730    2009-06-26 08:22:21.0000001          5.3
4  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

In [3]:
```
# Pre-process dataset
data.dropna(inplace=True)  # Drop missing values
data['pickup_datetime'] = pd.to_datetime(data['pickup_datetime'])
data['hour'] = data['pickup_datetime'].dt.hour
data['day'] = data['pickup_datetime'].dt.dayofweek
```

In [4]:
```
# Calculate distance
data['distance'] = np.sqrt((data['dropoff_longitude'] - data['pickup_longitude'])**2 +
                           (data['dropoff_latitude'] - data['pickup_latitude'])**2)
```

In [5]:
```
# Filter outliers based on distance and fare
data = data[(data['distance'] < data['distance'].quantile(0.99)) &
            (data['fare_amount'] < data['fare_amount'].quantile(0.99))]
```

In [6]:
```
# Check correlation
plt.figure(figsize=(8, 6))
sns.heatmap(data[['fare_amount', 'distance', 'hour', 'day', 'passenger_count']].corr(), annot=True)
plt.title("Correlation Heatmap")
plt.show()
```



In [7]:
```
# Prepare data for training
X = data[['distance', 'hour', 'day', '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)
```

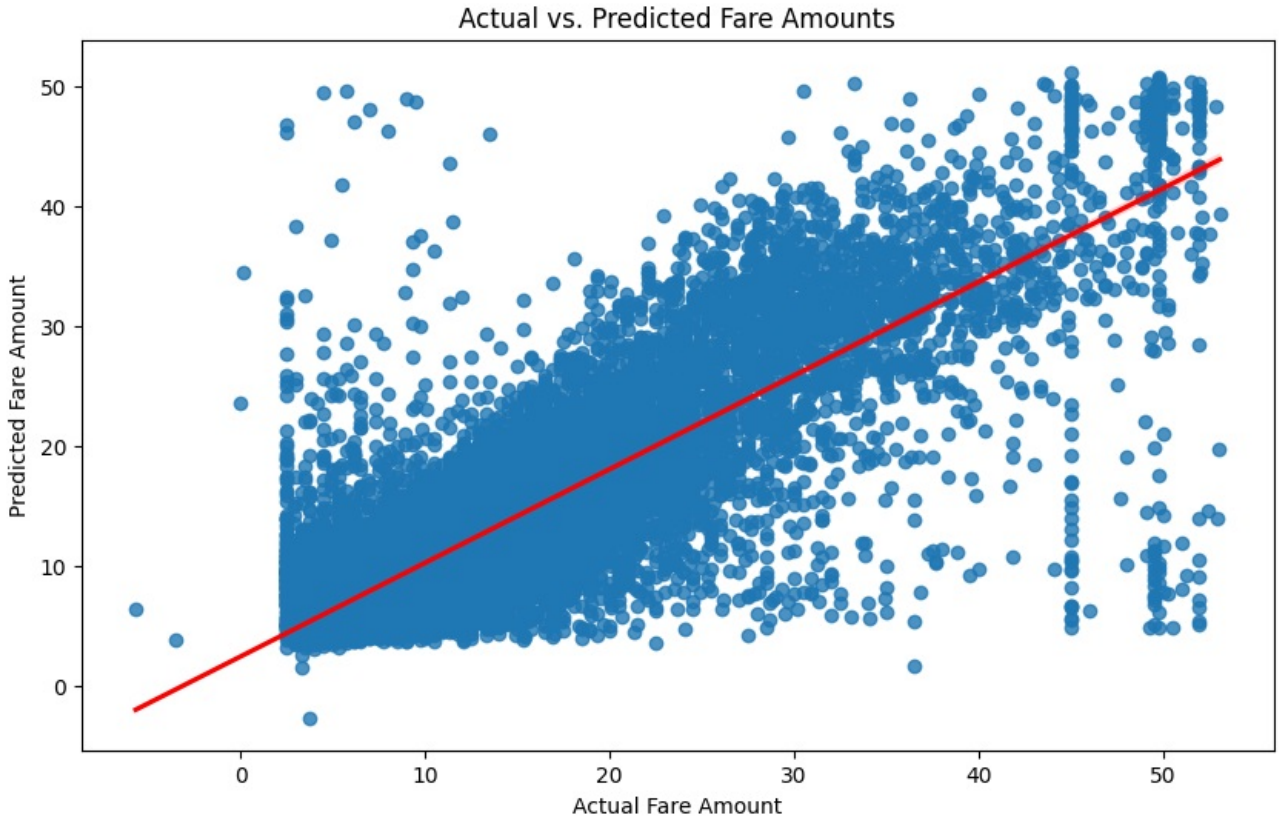
In [8]:
```
# Train Random Forest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[8]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [9]:
```
# Evaluate model
y_pred = model.predict(X_test)
print("R2 Score:", r2_score(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
R2 Score: 0.734234351846624
RMSE: 3.8886530330726283
MAE: 2.2979958923697335
```

In [10]:
```
# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel("Actual Fare Amount")
plt.ylabel("Predicted Fare Amount")
plt.title("Actual vs. Predicted Fare Amounts")
plt.show()
```



In [ ]:

In [ ]: