Assignment No. : 1

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 linear regression and random forest regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc. Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset

```
# import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#load  the dataset
data = pd.read_csv('/content/uber.csv')

data.head(1)
```

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitud
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.73835

Pre-process the dataset.

```
data.isnull().sum()
```

	0
Unnamed: 0	0
key	0
fare_amount	0
pickup_datetime	0
pickup_longitude	0
pickup_latitude	0
dropoff_longitude	1
dropoff_latitude	1
passenger_count	0

```
# missing value fill
data['dropoff_longitude'].fillna(data['dropoff_longitude'].mean(), inplace=True)
```

```
data['dropoff_latitude'].fillna(data['dropoff_latitude'].mean(), inplace=True)
```

```
data.describe()
```

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dro
count	2.000000e+05	200000.000000	200000.000000	200000.000000	200000.000000	
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	
std	1.601382e+07	9.901776	11.437787	7.720539	13.117375	
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	
75%	4.155530e+07	12.500000	-73.967154	40.767158	-73.963658	
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  200000 non-null  float64
 7   dropoff_latitude   200000 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

```
data.columns
```

```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

```
data.shape
```

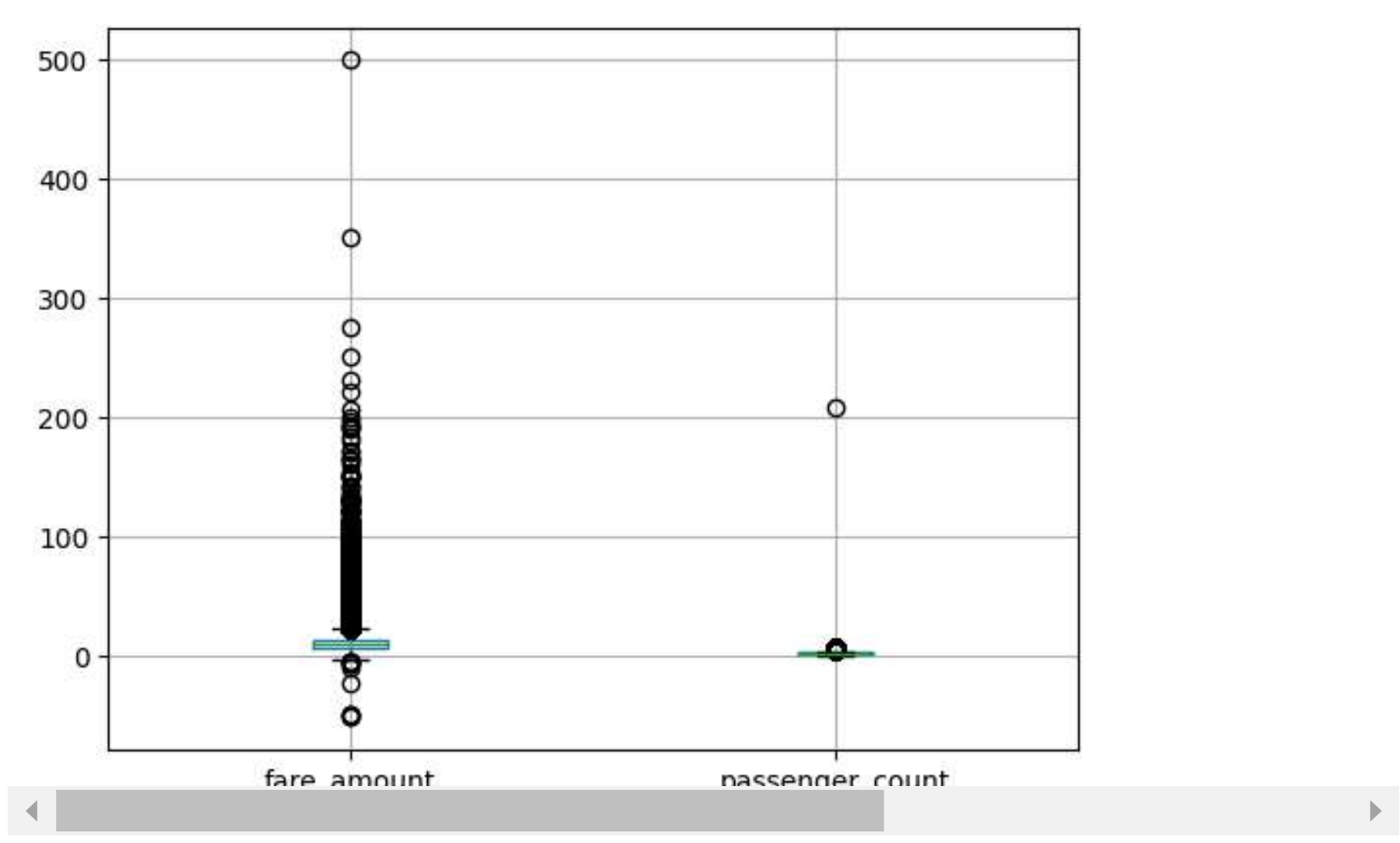
```
(200000, 9)
```

Identify outliers

```
import matplotlib.pyplot as plt

# Analyze numerical features
data.boxplot(column=["fare_amount", "passenger_count"])
plt.show()

# Handle outliers (e.g., capping, removing)
data = data[data["fare_amount"] < 3 * data["fare_amount"].quantile(0.95)]
```



Check the correlation.

```
# Calculate correlation for numerical features only
correlation = data.select_dtypes(include=['number']).corr()
```

linear regression

```
# Now define x and y
x = data.select_dtypes(include=['number']).drop("fare_amount", axis=1)
y = data["fare_amount"]

# Split data into training and testing sets.
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size=0.2, random_state=1)

# Initialize the linear regression model.
model = linear_model.LinearRegression()

# Train the model on the training data.
model.fit(xTrain, yTrain)
```

```
LinearRegression
LinearRegression()
```

+ Code + Text

```
# model evaluation
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(xTrain, yTrain)
lr_pred = lr.predict(xTest)
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
lr_rmse = np.sqrt(mean_squared_error(yTest, lr_pred))
lr_r2 = r2_score(yTest, lr_pred)
lr_mae = mean_absolute_error(yTest, lr_pred)
print(f" The RMSE of Linear regression is {lr_rmse}")
print(f" The R2 score of Linear regression is {lr_r2}")
print(f" The MAE of Linear regression is {lr_mae}")
```

```
The RMSE of Linear regression is 9.308045744639495
The R2 score of Linear regression is 0.0002925775801029262
The MAE of Linear regression is 5.925918920808279
```

Random Forest Regression Model

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 100, random_state = 101)
rf.fit(xTrain, yTrain)
rf_pred = rf.predict(xTest)


rf_rmse = np.sqrt(mean_squared_error(yTest, rf_pred))
rf_r2 = r2_score(yTest, rf_pred)
print(f" The RMSE of random forest model is {rf_rmse}")
print(f" The R2 score of random forest model is {rf_r2}")
```

```
The RMSE of random forest model is 4.231392124468747
The R2 score of random forest model is 0.793403744661945
```

Random Forest model outperforms the Linear Regression model in terms of both prediction accuracy and model fit