Name : Tavhare Ruchita Sharad

class : BE AI&DS

Roll No.: 61

Subject: Computer Laboratory-I(ML)

Title: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 ridge, Lasso regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
```

In [3]:
```
# 1. Load and Pre-process the Dataset
df = pd.read_csv('uber.csv')  # Change the name to the correct csv file
```

In [6]:
```
print("Initial Data Info:")
print(df.info())
print(df.head())
```

```
Initial 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  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
None
   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 [9]:
```
# Drop rows with missing values
df = df.dropna()
```

In [11]:
```
# Convert datetime columns
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
```

In [12]:
```
# Feature Engineering: Extract hour, day, month
df['hour'] = df['pickup_datetime'].dt.hour
df['day'] = df['pickup_datetime'].dt.day
df['month'] = df['pickup_datetime'].dt.month
```

In [13]:
```
# Remove unneeded columns
df = df.drop(columns=['key', 'pickup_datetime'])
```

In [17]:

```python
# Calculate distance (Haversine formula)
def haversine(lat1, lon1, lat2, lon2):
    R = 6371  # Earth radius in km
    phi1, phi2 = np.radians(lat1), np.radians(lat2)
    dphi = np.radians(lat2 - lat1)
    dlambda = np.radians(lon2 - lon1)
    a = np.sin(dphi/2)**2 + np.cos(phi1)*np.cos(phi2)*np.sin(dlambda/2)**2
    return 2 * R * np.arcsin(np.sqrt(a))

df['distance_km'] = haversine(df['pickup_latitude'], df['pickup_longitude'],
                              df['dropoff_latitude'], df['dropoff_longitude'])
```

In [19]:

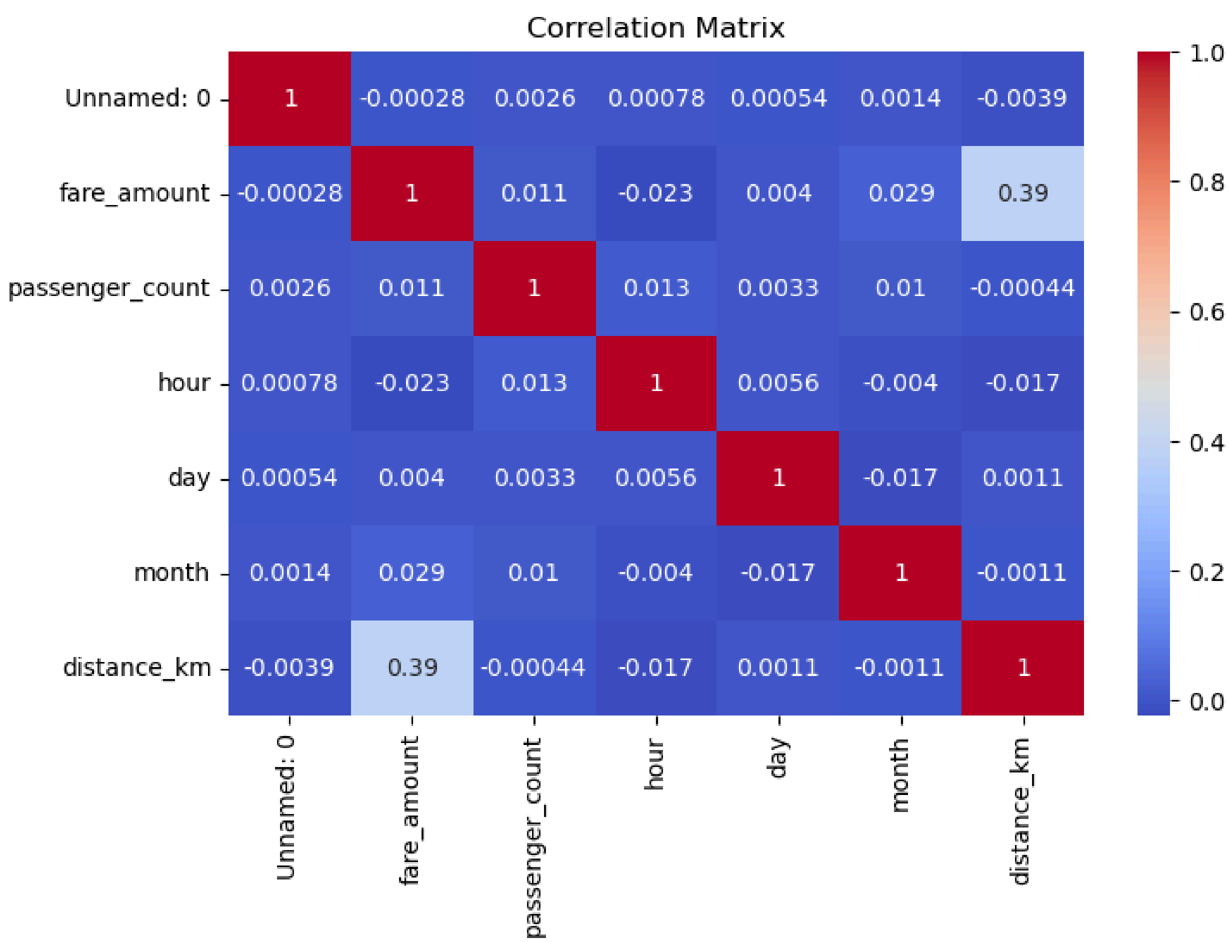
```python
# Drop original lat/lon columns
df = df.drop(columns=['pickup_latitude', 'pickup_longitude', 'dropoff_latitude
```

In [21]:

```python
# 2. Identify Outliers (using z-score on fare_amount and distance_km)
from scipy.stats import zscore
df = df[(np.abs(zscore(df[['fare_amount', 'distance_km']])) < 3).all(axis=1)]
```

In [23]:

```python
# 3. Check Correlation
plt.figure(figsize=(8, 5))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



In [24]:
```python
# 4. Regression Models
X = df.drop(columns=['fare_amount'])
y = df['fare_amount']
```

In [27]:
```python
# One-hot encode categorical columns (if any)
X = pd.get_dummies(X, drop_first=True)
```

In [29]:
```python
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [31]:
```python
# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

In [33]:
```python
# Ridge Regression
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_pred_ridge = ridge.predict(X_test)
```

In [35]:
```python
# Lasso Regression
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)
```

In [37]:
```python
# 5. Model Evaluation
def print_scores(model_name, y_true, y_pred):
    print(f"\n{model_name}")
    print(f"R2 Score: {r2_score(y_true, y_pred):.4f}")
    print(f"RMSE: {np.sqrt(mean_squared_error(y_true, y_pred)):.4f}")
```

In [39]:
```python
print_scores("Linear Regression", y_test, y_pred_lr)
print_scores("Ridge Regression", y_test, y_pred_ridge)
print_scores("Lasso Regression", y_test, y_pred_lasso)
```

```
Linear Regression
R2 Score: 0.1917
RMSE: 5.8158

Ridge Regression
R2 Score: 0.1917
RMSE: 5.8158

Lasso Regression
R2 Score: 0.1909
RMSE: 5.8187
```

In [41]:
```python
# Optional: Compare visually
plt.figure(figsize=(8,5))
plt.plot(y_test.values, label='True')
plt.plot(y_pred_lr, label='Linear Regression Predicted')
plt.plot(y_pred_ridge, label='Ridge Predicted')
```

```python
plt.plot(y_pred_lasso, label='Lasso Predicted')
plt.title('Model Predictions vs True Values (Sample)')
plt.xlabel('Sample')
plt.ylabel('Fare Amount')
plt.legend()
plt.show()
```