


✓ Importing necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.preprocessing import StandardScaler
from scipy import stats
```

✓ Loading the Dataset and Data Preprocessing


```
data = pd.read_csv('/content/uber.csv')
data.head()
```



	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropof
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	

```
data['pickup_datetime'] = pd.to_datetime(data['pickup_datetime'])
data['hour'] = data['pickup_datetime'].dt.hour
data['day_of_week'] = data['pickup_datetime'].dt.dayofweek
```

```
data.head()
```

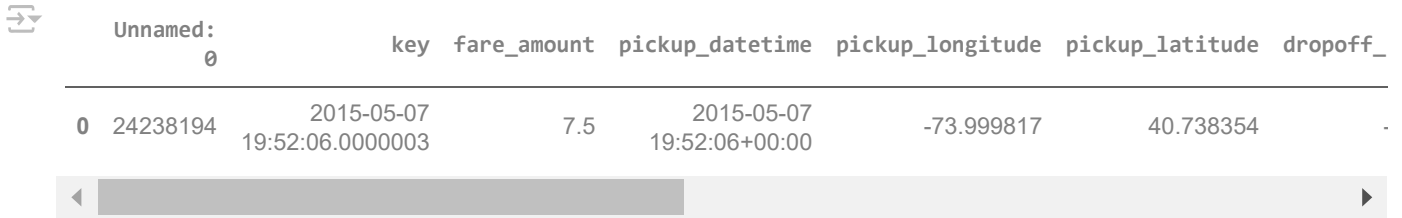


	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropof
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	

```
data.dropna(inplace=True)
```

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

```
data.head(1)
```



	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-

```
data.drop(['key', 'pickup_datetime', 'pickup_latitude', 'pickup_longitude',
          'dropoff_latitude', 'dropoff_longitude'], axis=1, inplace=True)
```

```
data.head(1)
```

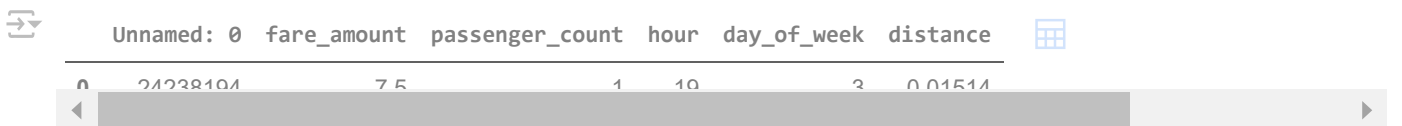


	Unnamed: 0	fare_amount	passenger_count	hour	day_of_week	distance
0	24238194	7.5	1	10	3	0.01514

▼ Identify Outliers

```
z_score = np.abs(stats.zscore(data[['fare_amount', 'distance']]))
threshold = 3
data = data[(z_score < threshold).all(axis=1)]
```

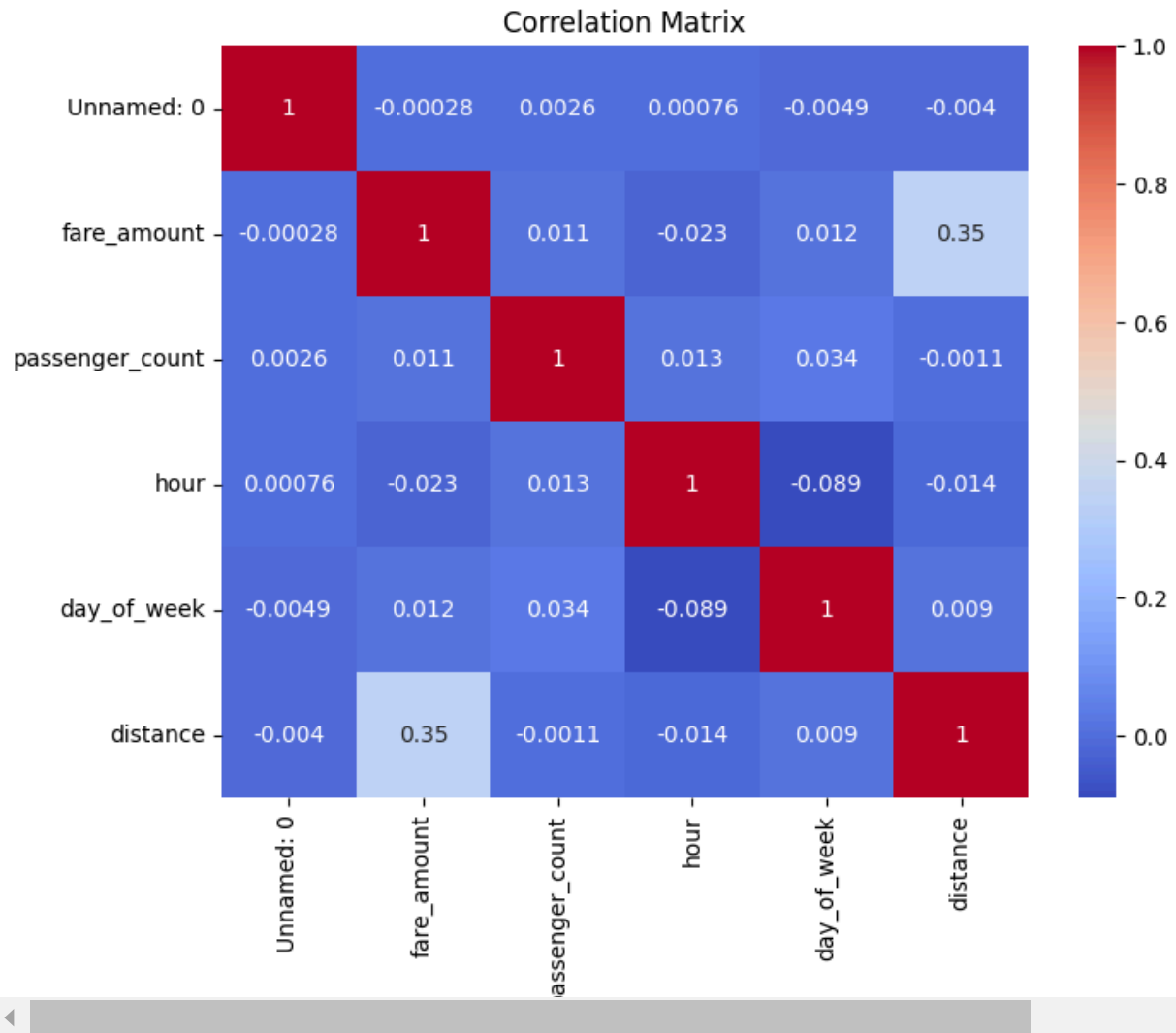
```
data.head(1)
```



	Unnamed: 0	fare_amount	passenger_count	hour	day_of_week	distance
0	24238194	7.5	1	10	3	0.01514

▼ Checking the Correlation

```
plt.figure(figsize=(8, 6))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



▼ Implement Regression Analysis

```
X = data.drop('fare_amount', axis=1)
y = data['fare_amount']
```

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

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
ridge_reg = Ridge(alpha=1.0)
ridge_reg.fit(X_train, y_train)
y_pred_ridge = ridge_reg.predict(X_test)
```

```
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X_train, y_train)
y_pred_lasso = lasso_reg.predict(X_test)
```

```
def evaluate_model(y_true, y_pred, model_name):
    r2 = r2_score(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    print(f"{model_name} - R2: {r2:.2f}, RMSE: {rmse:.2f}")
```

```
evaluate_model(y_test, y_pred, "Linear Regression")
evaluate_model(y_test, y_pred_ridge, "Ridge Regression")
evaluate_model(y_test, y_pred_lasso, "Lasso Regression")
```

```
Linear Regression - R2: 0.00, RMSE: 6.48
Ridge Regression - R2: 0.00, RMSE: 6.48
Lasso Regression - R2: 0.01, RMSE: 6.44
```

Visualizing the predicted results

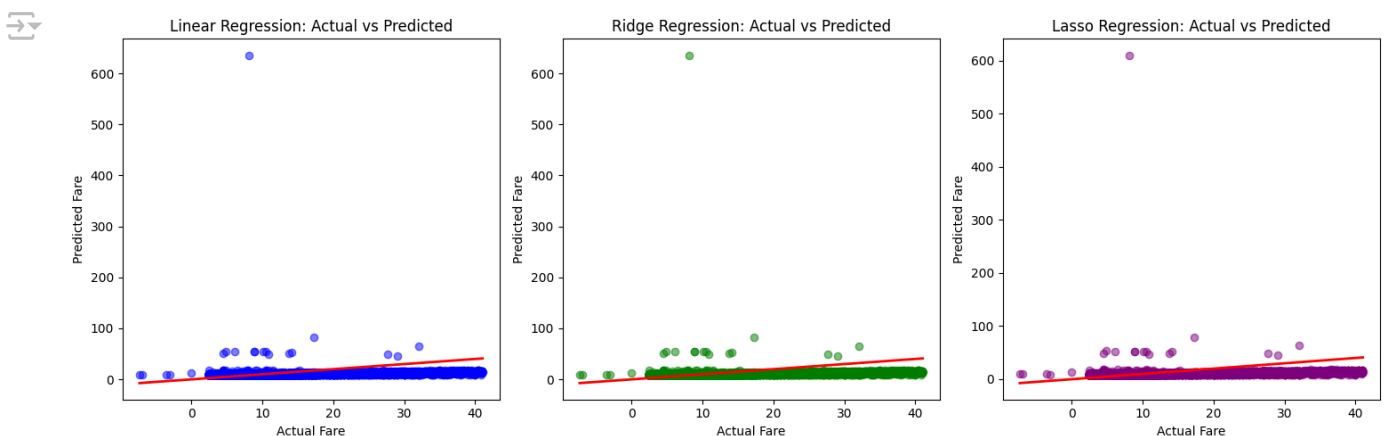
```
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2)
plt.title('Linear Regression: Actual vs Predicted')
plt.xlabel('Actual Fare')
plt.ylabel('Predicted Fare')

plt.subplot(1, 3, 2)
plt.scatter(y_test, y_pred_ridge, color='green', alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2)
plt.title('Ridge Regression: Actual vs Predicted')
plt.xlabel('Actual Fare')
plt.ylabel('Predicted Fare')

plt.subplot(1, 3, 3)
plt.scatter(y_test, y_pred_lasso, color='purple', alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2)
plt.title('Lasso Regression: Actual vs Predicted')
plt.xlabel('Actual Fare')
plt.ylabel('Predicted Fare')

plt.tight_layout()
plt.show()
```



Start coding or generate with AI.