Name : Devkumar Biswas

Class: BE(AI&DS)

Div: B

Subject : ML(CL-I Lab)

Roll no. : BEAD21267

**Assignment No. - 2**

**Problem Statement** : Regression Analysis:(Any one)

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 ridge, Lasso 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

Code: -

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression, Ridge, Lasso

from sklearn.metrics import r2\_score, mean\_squared\_error

from sklearn.impute import SimpleImputer # Import SimpleImputer

import seaborn as sns

import matplotlib.pyplot as plt

data = pd.read\_csv('uber.csv')

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 = data.drop(columns=['Unnamed: 0', 'key', 'pickup\_datetime'])

imputer = SimpleImputer(strategy='mean')

data\_imputed = pd.DataFrame(imputer.fit\_transform(data), columns=data.columns)

X = data\_imputed.drop(columns=['fare\_amount'])

y = data\_imputed['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)

data

plt.figure(figsize=(8, 6))

sns.boxplot(data=data, x='fare\_amount')

plt.title('Box Plot of Fare Amount')

plt.show()

correlation\_matrix = data.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm")

plt.title('Correlation Matrix Heatmap')

plt.show()

# Linear Regression

lr = LinearRegression()

lr.fit(X\_train, y\_train)

y\_pred\_lr = lr.predict(X\_test)

r2\_lr = r2\_score(y\_test, y\_pred\_lr)

rmse\_lr = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_lr))

# Ridge Regression

ridge = Ridge(alpha=1.0) # You can adjust the alpha parameter

ridge.fit(X\_train, y\_train)

y\_pred\_ridge = ridge.predict(X\_test)

r2\_ridge = r2\_score(y\_test, y\_pred\_ridge)

rmse\_ridge = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_ridge))

# Lasso Regression

lasso = Lasso(alpha=1.0) # You can adjust the alpha parameter

lasso.fit(X\_train, y\_train)

y\_pred\_lasso = lasso.predict(X\_test)

r2\_lasso = r2\_score(y\_test, y\_pred\_lasso)

rmse\_lasso = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_lasso))

# Print results

print("Linear Regression - R2:", r2\_lr, "RMSE:", rmse\_lr)

print("Ridge Regression - R2:", r2\_ridge, "RMSE:", rmse\_ridge)

print("Lasso Regression - R2:", r2\_lasso, "RMSE:", rmse\_lasso)

Output : -









