1Q:-

Consider only the below columns and prepare a prediction model for predicting Price.

Corolla<-Corolla[c("Price","Age\_08\_04","KM","HP","cc","Doors","Gears","Quarterly\_Tax","Weight")]

Model -- model of the car

Price -- Offer Price in EUROs

Age\_08\_04 -- Age in months as in August 2004

Mfg\_Month -- Manufacturing month (1-12)

Mfg\_Year -- Manufacturing Year

KM -- Accumulated Kilometers on odometer

Fuel\_Type -- Fuel Type (Petrol, Diesel, CNG)

HP -- Horse Power

Met\_Color -- Metallic Color? (Yes=1, No=0)

Color -- Color (Blue, Red, Grey, Silver, Black, etc.)

Automatic -- Automatic ( (Yes=1, No=0)

cc -- Cylinder Volume in cubic centimeters

Doors -- Number of doors

Cylinders -- Number of cylinders

Gears -- Number of gear positions

Quarterly\_Tax -- Quarterly road tax in EUROs

Weight -- Weight in Kilograms

Mfr\_Guarantee -- Within Manufacturer's Guarantee period (Yes=1, No=0)

BOVAG\_Guarantee -- BOVAG (Dutch dealer network) Guarantee (Yes=1, No=0)

Guarantee\_Period -- Guarantee period in months

ABS -- Anti-Lock Brake System (Yes=1, No=0)

Airbag\_1 -- Driver\_Airbag (Yes=1, No=0)

Airbag\_2 -- Passenger Airbag (Yes=1, No=0)

Airco -- Airconditioning (Yes=1, No=0)

Automatic\_airco -- Automatic Airconditioning (Yes=1, No=0)

Boardcomputer -- Boardcomputer (Yes=1, No=0)

CD\_Player -- CD Player (Yes=1, No=0)

Central\_Lock -- Central Lock (Yes=1, No=0)

Powered\_Windows -- Powered Windows (Yes=1, No=0)

Power\_Steering -- Power Steering (Yes=1, No=0)

Radio -- Radio (Yes=1, No=0)

Mistlamps -- Mistlamps (Yes=1, No=0)

Sport\_Model -- Sport Model (Yes=1, No=0)

Backseat\_Divider -- Backseat Divider (Yes=1, No=0)

Metallic\_Rim --Metallic Rim (Yes=1, No=0)

Radio\_cassette -- Radio Cassette (Yes=1, No=0)

Tow\_Bar -- Tow Bar (Yes=1, No=0)

**Ans:-**

**import pandas as pd**

**import numpy as np**

**df=pd.read\_csv("ToyotaCorolla.csv",encoding='latin-1')**

**df**

**list(df)**

**y=df['Price']**

**X=df[["Age\_08\_04","KM","HP","cc","Doors","Gears","Quarterly\_Tax","Weight"]]**

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

**from sklearn.linear\_model import LinearRegression**

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

**from sklearn.preprocessing import StandardScaler**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**scaler = StandardScaler()**

**X\_train\_scaled = scaler.fit\_transform(X\_train)**

**X\_test\_scaled = scaler.transform(X\_test)**

**mlr\_model = LinearRegression()**

**mlr\_model.fit(X\_train\_scaled, y\_train)**

**predictions = mlr\_model.predict(X\_test\_scaled)**

**mse = mean\_squared\_error(y\_test, predictions)**

**r2 = r2\_score(y\_test, predictions)**

**print(f'Mean Squared Error: {mse}')**

**print(f'R-squared: {r2}')**

**print('Coefficients:', mlr\_model.coef\_)**

**print('Intercept:', mlr\_model.intercept\_)**

2Q:-

Prepare a prediction model for profit of 50\_startups data.

Do transformations for getting better predictions of profit and

make a table containing R^2 value for each prepared model.

R&D Spend -- Research and devolop spend in the past few years

Administration -- spend on administration in the past few years

Marketing Spend -- spend on Marketing in the past few years

State -- states from which data is collected

Profit -- profit of each state in the past few years

**Ans:-**

**import pandas as pd**

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

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import r2\_score**

**import statsmodels.api as sm**

**df = pd.read\_csv('50\_Startups.csv')**

**df.head()**

**selected\_columns = ['R&D Spend', 'Marketing Spend', 'Profit']**

**data = df[selected\_columns]**

**X = data.drop('Profit', axis=1)**

**y = data['Profit']**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**model = LinearRegression()**

**model.fit(X\_train, y\_train)**

**y\_pred = model.predict(X\_test)**

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

**print(f'R-squared (scikit-learn): {r2}')**

**X\_train = sm.add\_constant(X\_train) # Add constant term for intercept**

**model\_stats = sm.OLS(y\_train, X\_train).fit()**

**r2\_statsmodels = model\_stats.rsquared**

**print(f'R-squared (statsmodels): {r2\_statsmodels}')**

**print(model\_stats.summary())**