1. **Aim**: Program to implement linear techniques using any standard dataset available in the public domain and evaluate its performance.

**Program**

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model

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

df = datasets.load\_diabetes()

print(df['feature\_names'])

# Load the diabetes dataset

diabetes\_X, diabetes\_y = datasets.load\_diabetes(return\_X\_y=True)

print(diabetes\_X.shape)

# Use only one feature

diabetes\_X = diabetes\_X[:, np.newaxis, 2]

print(diabetes\_X.shape)

# Split the data into training/testing sets

diabetes\_X\_train = diabetes\_X[:-20]

diabetes\_X\_test = diabetes\_X[-20:]

# Split the targets into training/testing sets

diabetes\_y\_train = diabetes\_y[:-20]

diabetes\_y\_test = diabetes\_y[-20:]

# Create linear regression object

regr = linear\_model.LinearRegression()

# Train the model using the training sets

regr.fit(diabetes\_X\_train, diabetes\_y\_train)

# Make predictions using the testing set

diabetes\_y\_pred = regr.predict(diabetes\_X\_test)

# The coefficients

print("Coefficients: \n", regr.coef\_)

# The mean squared error

print("Mean squared error: %.2f" % mean\_squared\_error(diabetes\_y\_test, diabetes\_y\_pred))

# The coefficient of determination: 1 is perfect prediction

print("Coefficient of determination: %.2f" % r2\_score(diabetes\_y\_test, diabetes\_y\_pred))

# Plot outputs

plt.scatter(diabetes\_X\_test, diabetes\_y\_test, color="black")

plt.plot(diabetes\_X\_test, diabetes\_y\_pred, color="blue", linewidth=3)

plt.xlabel("age")

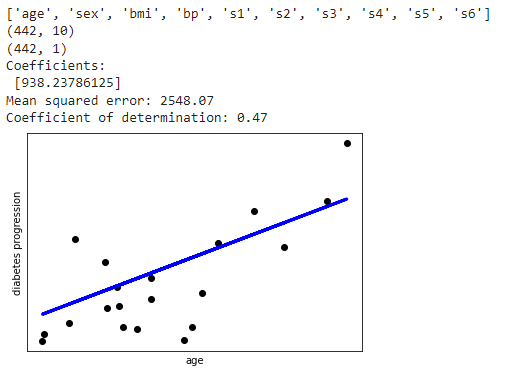
plt.ylabel("diabetes progression")

plt.xticks(())

plt.yticks(())

plt.show()

OUTPUT



1. **Aim**: Program to implement K means clustering using any standard dataset available in the public domain and evaluate its performance.

**Program**

from sklearn import datasets

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cluster import KMeans

iris = datasets.load\_iris()

X = iris.data[:, :2]

y = iris.target

plt.scatter(X[:,0], X[:,1], c=y, cmap='prism')

plt.xlabel('Spea1 Length', fontsize=18)

plt.ylabel('Sepal Width', fontsize=18)

km = KMeans(n\_clusters = 3, init='k-means++', n\_init=10, max\_iter=300, tol=0.0001, verbose=0, random\_state=21, copy\_x=True, algorithm="auto")

km.fit(X)

centers = km.cluster\_centers\_

print(centers)

new\_labels = km.labels\_

print(new\_labels)

print(y)

fig, axes = plt.subplots( 1,2, figsize=(16,8))

axes[0].scatter(X[:, 0], X[:, 1], c=y, cmap='prism',edgecolor='k', s=75)

axes[1].scatter(X[:, 0], X[:, 1], c=new\_labels, cmap='jet',edgecolor='k', s=75)

axes[0].set\_xlabel('Sepal length', fontsize=12)

axes[0].set\_ylabel('Sepal width', fontsize=12)

axes[1].set\_xlabel('Sepal length', fontsize=12)

axes[1].set\_ylabel('Sepal width', fontsize=12)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=15)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=15)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)

Output

