UPES
UNIVERSITY WITH A PURPOSE

Pattern & Anomaly Detection Lab

Experiment 3

Submitted By:
Dhruv Singhal
500075346
R177219074
AIML B3(SEM 5)

Submitted To:
Dr. Gopal Phartiyal
Professor
SOCS
UPES

AIM:
To perform Linear Regression (OLS Model)

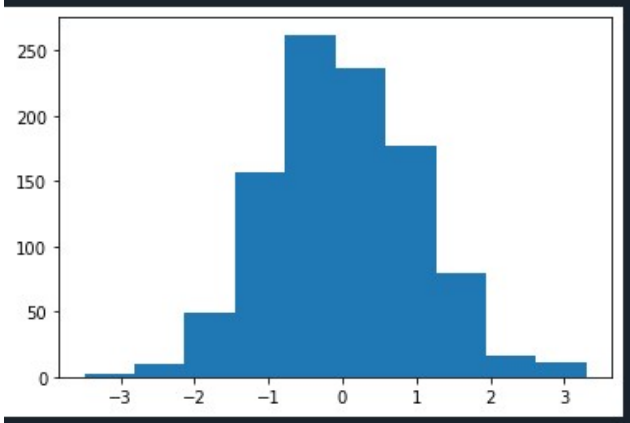
CODE:

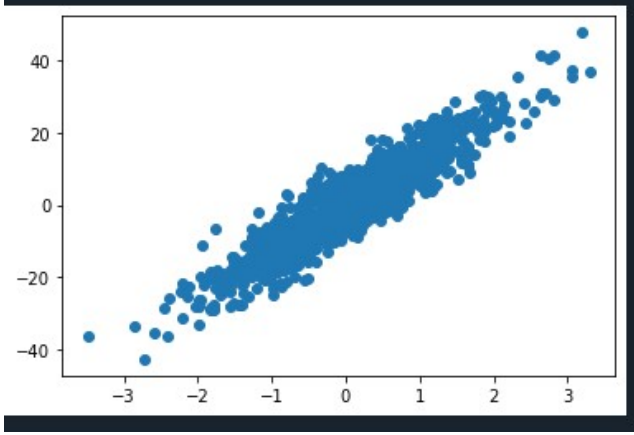
```
#%%
# import required libraries
from sklearn.datasets import make_regression
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error ,mean_absolute_error
import numpy as np
import seaborn as sns
from math import sqrt
import warnings
warnings.filterwarnings('ignore')
#%%
#generate dataset for regression
X,y = make_regression(n_samples=1000, n_features = 1, shuffle = True, noise = 5,random_state=86)
#%%
#Statistics
stdX = np.std(X)
meanX = np.mean(X)

stdY = np.std(y)
meanY = np.mean(y)
print(meanX)
print(stdX)
print(meanY)
print(stdY)
#%%
# plot generated data
plt.hist(X)
plt.show()
#%%
plt.scatter(X,y)
plt.show()
#%%
# As mean of this dataset is already near to 0, so standard scaling is not required
#%%
# Preprocessing
#%%
plt.hist(X)
plt.show()
#%%
#Plot for data after preprocessing
plt.scatter(X,y)
plt.show()
#%%
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10, random_state=42)
#%%
sns.distplot(y_train)
plt.show()
#%%
sns.distplot(y_test)
plt.show()
#%%
#Performing Linear Regression || Using Traditional Method
```

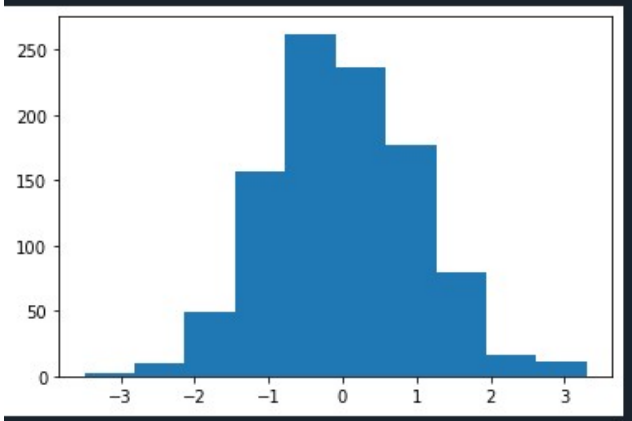
```
#Initializing variables
n = len(X_train)
num = 0
denom = 0
#%%
#Calculating slope & intercept
for i in range(n):
    num += (X_train[i]-meanX) * (y_train[i]-meanY)
    denom += (X_train[i] - meanX)**2
m = num/denom
c = meanY - (m*meanX)
print(m, ', ', c)

min_X = 1 - np.min(X_train)
max_X = 1 - np.max(X_train)
print(min_X, ', ', max_X)
#%%
#calculating the values of x & y
x = np.linspace(min_X, max_X,1000)
y = (m*x)+c
#%%  Plot Regression Line
plt.scatter(X_train,y_train,color='b')
plt.plot(x,y,color='r')
plt.title('Simple Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
#%%
#calculating error
y_pred = []
sum_pred = 0
sum_actual = 0
for i in range(len(X_test)):
    val = (m*X_test[i]+c)
    y_pred.append(val)
mse = mean_squared_error(y_test, y_pred)
print("MSE", mse)
r2 = sqrt(mse)
print("R2",r2)
mae = mean_absolute_error(y_test, y_pred)
print("MAE", mae)
#%%
# Perform Linear Regression || Using sklearn
# Build Model
LRmodel = LinearRegression()
LRmodel.fit(X_train, y_train)
#%%
# Error Calculation
print("-------------------------------------")
y_pred1 = LRmodel.predict(X_test)
mse1 = mean_squared_error(y_test, y_pred1)
print("MSE with sklearn", mse1)
r2new = sqrt(mse1)
print("R2 with sklearn",r2new)
mae1 = mean_absolute_error(y_test, y_pred1)
print("MAE with sklearn", mae1)
```

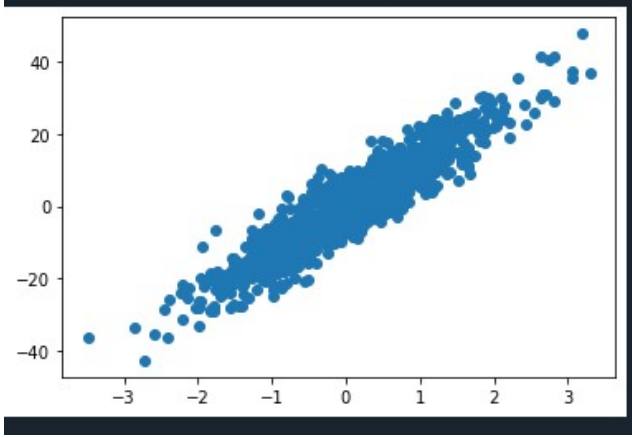
Output:

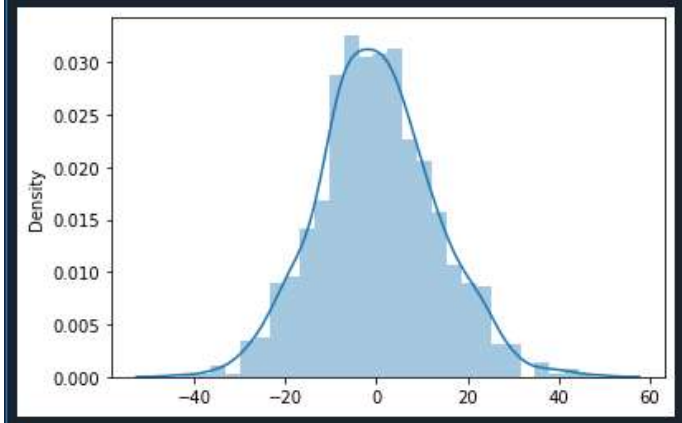
```
In [1]: runfile('B:/3rd year/5th sem/P&AD/LinearRegression.py', wdir='B:/3rd year/5th sem/P&AD')
0.007610682926160853
0.9942474804457474
-0.05835119770320271
13.08071673806697
```

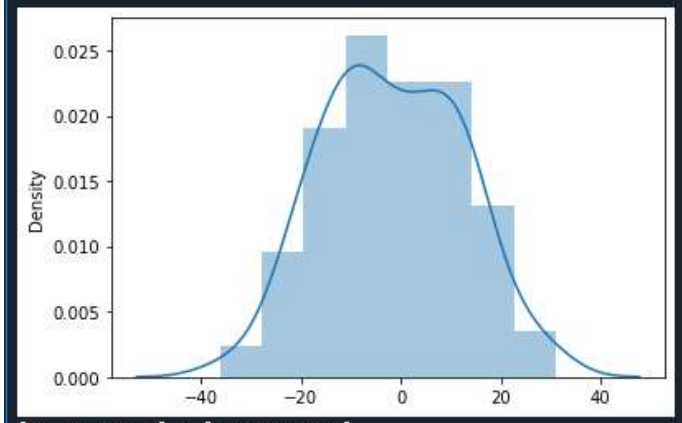




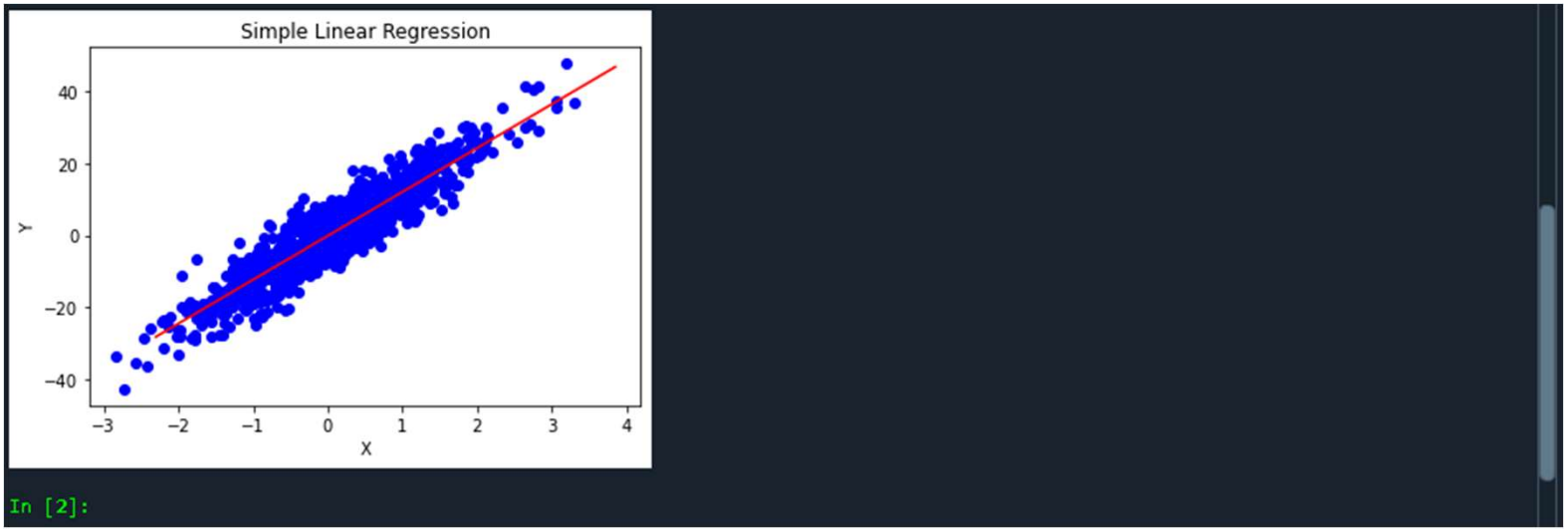

250
200
150
100
50
0
−3
−2
−1
0
1
2
3


40
20
0
−20
−40
−3
−2
−1
0
1
2
3






```
[12.21409321] , [-0.15130879]
3.8487665053713127 , -2.303677618501437
MSE 20.304967238809418
R2 4.506103332016413
MAE 3.611272677585965
--------------------------------------
MSE with sklearn 20.33856770371379
R2 with sklearn 4.509830119163447
MAE with sklearn 3.610421754936301
```




```
In [2]:
```