**Practical No. 12**

**A.1 Aim:**

To solve the demand prediction problem.

**A.2 Prerequisite:**

Python, Jupiter notebook, demo in xlsx

**A.3 Outcome:**

The program is written in Python to solve demand prediction.

**A.4 Theory:**

To be discussed during the lecture hours.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per the following segments within four hours of the practical. The soft copy must be uploaded on the portal or emailed to the concerned lab in charge faculties at the end of the practical in case there is no portal access available)***

|  |  |
| --- | --- |
| Roll. No. Ds2201 | Name: abhishek michael |
| Class | Batch: |
| Date of Experiment: | Date of Submission: 27/02/2024 |
| Grade: | |

**B.1 Task to do:**

Refer the problem statement mentioned in the following URL and solve it using Python.

<https://www.youtube.com/watch?v=kUS9RgWNhBw>

**B.2 Output program**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from scipy.optimize import curve\_fit**

**from scipy.stats import linregress**

**# Generating random data points**

**np.random.seed(0)**

**x\_data = np.linspace(1, 10, 10)**

**y\_data = 2 \* x\_data + 3 \* np.random.randn(10)**

**# Define regression functions**

**def linear\_func(x, a, b):**

**return a \* x + b**

**def exp\_func(x, a, b):**

**return a \* np.exp(b \* x)**

**def log\_func(x, a, b):**

**return a \* np.log(x) + b**

**def power\_func(x, a, b):**

**return a \* np.power(x, b)**

**def polynomial\_func(x, a, b, c):**

**return a \* x\*\*2 + b \* x + c**

**# Perform curve fitting for each model**

**popt\_linear, \_ = curve\_fit(linear\_func, x\_data, y\_data)**

**popt\_exp, \_ = curve\_fit(exp\_func, x\_data, y\_data)**

**popt\_log, \_ = curve\_fit(log\_func, x\_data, y\_data)**

**popt\_power, \_ = curve\_fit(power\_func, x\_data, y\_data)**

**coefficients\_poly = np.polyfit(x\_data, y\_data, 2)**

**# Calculate R-squared values for each model**

**def calculate\_r\_squared(y\_true, y\_pred):**

**ss\_res = np.sum((y\_true - y\_pred)\*\*2)**

**ss\_tot = np.sum((y\_true - np.mean(y\_true))\*\*2)**

**return 1 - (ss\_res / ss\_tot)**

**r\_squared\_values = []**

**r\_squared\_values.append(calculate\_r\_squared(y\_data, linear\_func(x\_data, \*popt\_linear)))**

**r\_squared\_values.append(calculate\_r\_squared(y\_data, exp\_func(x\_data, \*popt\_exp)))**

**r\_squared\_values.append(calculate\_r\_squared(y\_data, log\_func(x\_data, \*popt\_log)))**

**r\_squared\_values.append(calculate\_r\_squared(y\_data, power\_func(x\_data, \*popt\_power)))**

**r\_squared\_values.append(calculate\_r\_squared(y\_data, np.polyval(coefficients\_poly, x\_data)))**

**# Choose the model with the highest R-squared value**

**best\_fit\_index = np.argmax(r\_squared\_values)**

**best\_fit\_models = ['Linear', 'Exponential', 'Logarithmic', 'Power', 'Polynomial']**

**# Plotting**

**plt.figure(figsize=(10, 6))**

**plt.scatter(x\_data, y\_data, label='Data')**

**if best\_fit\_index == 0:**

**plt.plot(x\_data, linear\_func(x\_data, \*popt\_linear), 'r-', label='Best Fit: Linear Regression')**

**elif best\_fit\_index == 1:**

**plt.plot(x\_data, exp\_func(x\_data, \*popt\_exp), 'g--', label='Best Fit: Exponential Regression')**

**elif best\_fit\_index == 2:**

**plt.plot(x\_data, log\_func(x\_data, \*popt\_log), 'b-.', label='Best Fit: Logarithmic Regression')**

**elif best\_fit\_index == 3:**

**plt.plot(x\_data, power\_func(x\_data, \*popt\_power), 'm:', label='Best Fit: Power Regression')**

**else:**

**plt.plot(x\_data, np.polyval(coefficients\_poly, x\_data), 'k', label='Best Fit: Polynomial Regression')**

**plt.xlabel('X')**

**plt.ylabel('Y')**

**plt.title('Best Fit Model: {}'.format(best\_fit\_models[best\_fit\_index]))**

**plt.legend()**

**plt.grid(True)**

**plt.show()**

**# Define the number of future values to predict**

**num\_future\_values = 5**

**# Generate x values for the future predictions**

**x\_future = np.linspace(x\_data[-1] + 1, x\_data[-1] + num\_future\_values, num\_future\_values)**

**# Predict the future values using the best fit model**

**if best\_fit\_index == 0:**

**y\_future = linear\_func(x\_future, \*popt\_linear)**

**elif best\_fit\_index == 1:**

**y\_future = exp\_func(x\_future, \*popt\_exp)**

**elif best\_fit\_index == 2:**

**y\_future = log\_func(x\_future, \*popt\_log)**

**elif best\_fit\_index == 3:**

**y\_future = power\_func(x\_future, \*popt\_power)**

**else:**

**y\_future = np.polyval(coefficients\_poly, x\_future)**

**print("Predicted future values:")**

**for i in range(num\_future\_values):**

**print("X = {}, Y = {:.2f}".format(x\_future[i], y\_future[i]))**

**# Concatenate original and predicted x values**

**x\_all = np.concatenate((x\_data, x\_future))**

**# Concatenate original and predicted y values**

**if best\_fit\_index == 0:**

**y\_all = np.concatenate((linear\_func(x\_data, \*popt\_linear), linear\_func(x\_future, \*popt\_linear)))**

**elif best\_fit\_index == 1:**

**y\_all = np.concatenate((exp\_func(x\_data, \*popt\_exp), exp\_func(x\_future, \*popt\_exp)))**

**elif best\_fit\_index == 2:**

**y\_all = np.concatenate((log\_func(x\_data, \*popt\_log), log\_func(x\_future, \*popt\_log)))**

**elif best\_fit\_index == 3:**

**y\_all = np.concatenate((power\_func(x\_data, \*popt\_power), power\_func(x\_future, \*popt\_power)))**

**else:**

**y\_all = np.concatenate((np.polyval(coefficients\_poly, x\_data), np.polyval(coefficients\_poly, x\_future)))**

**# Plotting**

**plt.figure(figsize=(10, 6))**

**plt.scatter(x\_data, y\_data, label='Original Data', color='blue')**

**plt.plot(x\_all, y\_all, label='Predicted Data', color='red')**

**plt.xlabel('X')**

**plt.ylabel('Y')**

**plt.title('Original vs Predicted Data')**

**plt.legend()**

**plt.grid(True)**

**plt.show()**

**B.4 Conclusion:**



