Artificial Intelligence in Smart Grid

ECE 563

Project 0

Abhilash Kashyap Balasubramanyam [abalasubramanyam@hawk.iit.edu](mailto:abalasubramanyam@hawk.iit.edu) A20566944

Spring 2024

Illinois Institute of Technology

Date: 25 January 2024

# Project Overview

The project 0 is conducted to understand and setup python (Programming Language) and the necessary IDE (Integrated Development Environment) as a preparation for the learnings, practice and practical implementation of theoritical concepts for the 'Artificial Intelligence in Smart Grid' (ECE 563) course taught by Prof. Dr. Alexander J. Flueck. during the Spring semester of 2024 at Illinois Tech.

The Project 0 consists of four sub questions each addressing methods of implementations of logics and libraries disucssed during the lecture. They are as follows:

1. Setup and testing of the programming language and IDE.
2. Plotting a Sigmoid function for a set defined range of values.
3. Calculating the Mean and Standard Deviation for a range of pre-defined values then calculating and plotting the Gaussian Distribution function by appropriately marking and labelling wherever necessary.
4. Using Pandas library to import a given .csv (comma seperated values) file onto the python working directory and applying a given condition to filter, sort and display the output result.

The problem statements from the project 0 along with the code and the solutions are documented using Jupyter Notebook and are as follows.

## Problem statement 1

A screenshot of your computer showing your development environment, e.g., IPython shell, Jupyter Lab, Jupyter Notebook, Spyder, etc. with the following commands and their results displayed:

In [1]:

**import** os os**.**getlogin() os**.**getcwd()

Out[1]:

'/Users/Kashyap'

The solution above indicates that the setup is complete and the current working directory is displayed as the output.

## Problem Statement 2

Python code that creates a sigmoid plot from -6 to 6 using matplotlib. Be sure to include grid lines and a title on your plot. (Note: "%matplotlib tk" works for Tcl/Tk; "%matplotlib notebook" works for Jupyter; "%matplotlib inline" works for Jupyter and spyder. To connect to the default backend, use "%matplotlib" without any additional parameter). The plot showing your sigmoid function.

In [2]:

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

*#Range of sigmoid plot definition*

r1 **= -**6

r2 **=** 6

*#Generation of evenly spaced points within the range defined*

x **=** np**.**linspace(r1,r2,100)

*#Sigmoid function formula*

y **=** 1**/**(1**+**np**.**exp(**-**x))

*#Plot and plot properties*

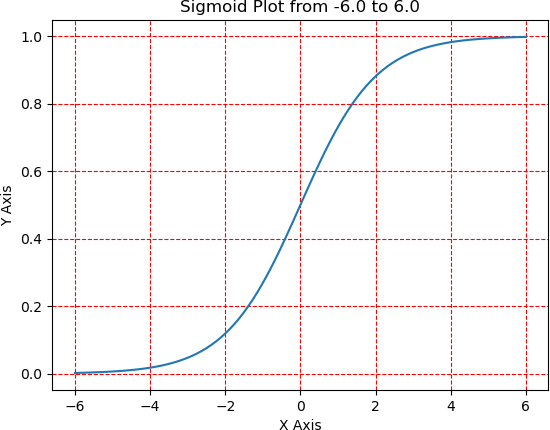
plt**.**plot(x, y)

plt**.**xlabel("X Axis") plt**.**ylabel("Y Axis")

plt**.**title(f"Sigmoid Plot from {r1:.1f} to {r2:.1f}") *#f-string is used to*

plt**.**grid(color **=** 'Red', linestyle **=** '--')

plt**.**show()



The above solution to problem statement 2 displays with a blue curve which is the sigmoid function. The graph range is determined by the start and end point defined in the code as per the requirement from the problem statement.

## Problem Statement 3

Python code that creates two Gaussian probability density function plots with different standard deviation values. Be sure to include grid lines and a title on your plot. Also, be sure to use the marker attribute and the label attribute so that each curve is unique. Finally, include a legend on the plot that clearly explains the difference between the two curves shown on the same axes. To draw two curves on same axes in an interactive console/notebook, separate the plotting commands with a comma, i.e., plt.plot(x1,y1,marker="x"),plt.plot(x2,y2,marker="o"). The plot showing your two Gaussian functions on the same axes in a single figure.

In [3]:

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

*#Generation of distribution range for two sets*

x1 **=** np**.**linspace(**-**550, 310, 10) x2 **=** np**.**linspace(**-**153,1080, 10)

*#Generation of mean*

m1 **=** np**.**mean(x1) m2 **=** np**.**mean(x2)

*#Generation of Standard Deviation*

s1 **=** np**.**std(x1) s2 **=** np**.**std(x2)

*#Printing Mean and Standard Deviation for both the different sets of valu*

print("Mean 1 =", m1) print("Mean 2 =", m2)

print("Standard Deviation 1 =", s1) print("Standard Deviation 2 =", s2)

*#Gaussian Distribution formula for two sets of standard deviations*

gd1 **=** 1 **/** (np**.**sqrt(2 **\*** np**.**pi) **\*** s1 **\*\*** 2) **\*** np**.**exp(**-**((x1**-**m1) **\*\*** 2) **/** (2 **\***

gd2 **=** 1 **/** (np**.**sqrt(2 **\*** np**.**pi) **\*** s2 **\*\*** 2) **\*** np**.**exp(**-**((x2**-**m2) **\*\*** 2) **/** (2 **\***

*#Plot and plot properties*

plt**.**plot(x1, gd1, marker**=**'o', label**=**f'PDF plot for Std Dev = {s1:.2f}') plt**.**plot(x2, gd2, marker**=**'D', label**=**f'PDF plot for Std Dev = {s2:.2f}')

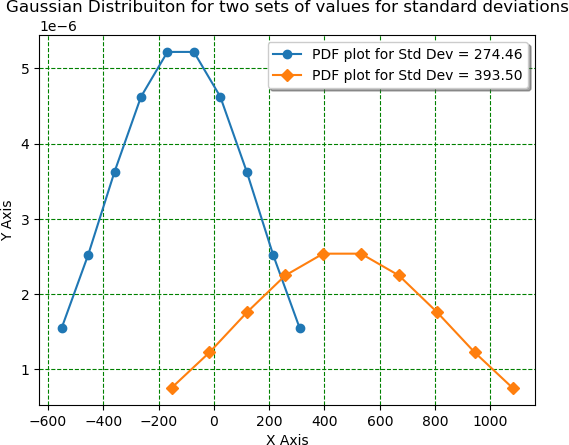
plt**.**title("Gaussian Distribuiton for two sets of values for standard devi plt**.**grid(color **=** 'Green', linestyle **=** '--')

plt**.**xlabel("X Axis") plt**.**ylabel("Y Axis")

plt**.**legend(shadow **= True**, loc **=** "upper right")

plt**.**show()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mean 1 = | -120.0 |  | | |
| Mean 2 = | 463.5 |
| Standard | Deviation | 1 | = | 274.46243755681695 |
| Standard | Deviation | 2 | = | 393.50254128785497 |



The above solution to the problem statement 3 is a console output showcasing the mean and standard distribution values for two ranges of data that is pre-defined. Added to this, the graph output is the plot of these two calculated Gaussian Distribution Functions which are differently colored and labelled in the legend as well to difference between them. The legend also contains the standard deviation of these two plots to identify the difference between them. Furthermore, the program also uses markers for the points along the curve (here circle and diamond is used) and the labelling of the axes with the title of the graph also included.

## Problem Statement 4

Python code that uses pandas to select and print 'Timestamp', 'VALPM:Magnitude', 'VBLPM:Magnitude', 'VCLPM:Magnitude' whenever the 'VCLPM:Magnitude' value is less than 1UU000 in the zipped CSV file of PMU data. The results from your Python code that selects and prints a subset of the PMU data above.

In [4]:

**import** pandas **as** pd

*#Reading the file from the database repository*

df **=** pd**.**read\_csv("//Users/Kashyap/Documents/Files/Academics/Institutions/

*# Filter rows based on the condition*

subset **=** df**.**loc[df["VCLPM:Magnitude"] **<** 199000, ["Timestamp", "VALPM:Magn

*# Save the resulting subset to a CSV file to manually verify the correctn*

subset**.**to\_csv("//Users/Kashyap/Documents/Files/Academics/Institutions/Mas

*# Print the resulting subset*

print(subset)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Timestamp | VALPM:Magnitude | VBLPM:Magnitude | \ |
| 11315 | 2014/07/01 | 02:42:08.966 | 206117.2188 | 197712.6875 |  |
| 11476 | 2014/07/01 | 02:42:14.333 | 205979.9063 | 197424.2969 |  |
| 11477 | 2014/07/01 | 02:42:14.366 | 206323.2344 | 199758.8906 |  |
| 48926 | 2014/07/01 | 03:03:05.133 | 205595.3906 | 206446.8281 |  |
| 48927 | 2014/07/01 | 03:03:05.166 | 204551.6875 | 206062.2969 |  |
| 56064 | 2014/07/01 | 03:07:03.600 | 190640.2500 | 204277.0313 |  |
| 56065 | 2014/07/01 | 03:07:03.633 | 168283.0625 | 199415.5781 |  |
| 56066 | 2014/07/01 | 03:07:03.666 | 185916.1250 | 199470.5000 |  |
| 11315 | VCLPM:Magnitude 198948.6563 | | | | |
| 11476 | 198701.4688 | | | | |
| 11477 | 198509.2031 | | | | |
| 48926 | 198797.5938 | | | | |
| 48927 | 198179.6094 | | | | |
| 56064 | 198440.5313 | | | | |
| 56065 | 195103.4375 | | | | |
| 56066 | 196888.7188 | | | | |

The above solution represents the list of data from the columns defined as per the problem statement. The above list satisfies the given condition to check if VCLPM:Magnitude values are lesser than 1UU000 and prints the correspoding Timestamp, VALPM:Magnitude, VBLPM:Magnitude and VCLPM:Magnitude values. Additionally, to further verify the data manually using find and replace, there is a provision added to the code to save the result in an output .csv file.

# Conclusion

The project 0 has played a formative role to help understand the various issues that can appear during the setup of an environment in a system. It has also helped in understanding that the code tends to behave differently if not tested for robustness across IDEs in the same system. Learning and using python in general and its libraries such as matplotlib, numpy and pandas has indeed enabled confidence in the language and acclimitization towards the concepts.