Artificial Intelligence Lab
AL-2002

Lab 01

Instructor: Muhammad Saood Sarwar

Semester: Spring 2024

Artificial Intelligence Lab 01

Objectives

The objective of this session is to get exposure to Python programming, and write some simple code to access data and plot it using some key Python libraries.

Learning Outcomes

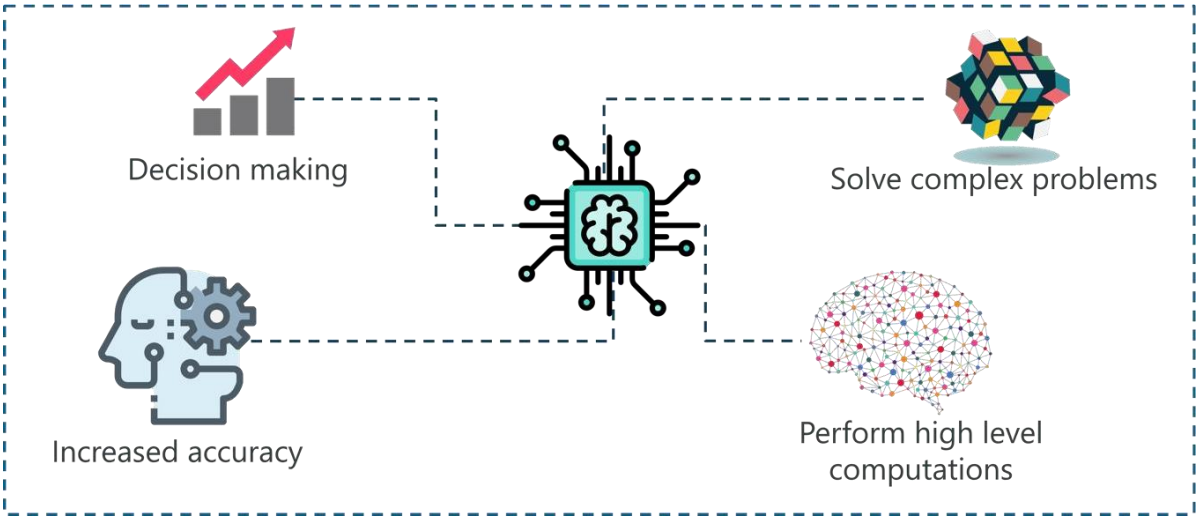
1. Write simple Python code to create random numbers and frequency histogram using the NumPy library.
2. Create graphs in Python using the matplotlib library.
3. Understand and work with numeric data in the library pandas.

Table of Contents

What is Artificial Intelligence?

The father of Artificial Intelligence, John McCarthy, defines it as

"The science and engineering of making intelligent machines, especially intelligent computer programs"



Artificial Intelligence can also be defined as the development of computer systems that are capable of performing tasks that require human intelligence, such as decision making, object detection, solving complex problems and so on.

Types Of Learning in Artificial Intelligence

1. Artificial Narrow Intelligence
2. Artificial General Intelligence
3. Artificial Super Intelligence

Artificial Narrow Intelligence (ANI)

Also known as Weak AI, ANI is the stage of Artificial Intelligence involving machines that can perform only a narrowly defined set of specific tasks. At this stage, the machine does not possess any thinking ability, it just performs a set of pre-defined functions.

Examples of Weak AI include Siri, Alexa, Self-driving cars, Alpha-Go, Sophia the humanoid and so on. Almost all the AI-based systems built till this date fall under the category of Weak AI.

Artificial General Intelligence (AGI)

Also known as Strong AI, AGI is the stage in the evolution of Artificial Intelligence wherein machines will possess the ability to think and make decisions just like us humans.
There are currently no existing examples of Strong AI, however, it is believed that we will soon be able to create machines that are as smart as humans.

Strong AI is considered a threat to human existence by many scientists, including Stephen Hawking who stated that:

"The development of full artificial intelligence could spell the end of the human race…. It would take off on its own, and re-design itself at an ever-increasing rate. Humans, who are limited by slow biological evolution, couldn't compete and would be superseded."

Artificial Super Intelligence (ASI)

Artificial Super Intelligence is the stage of Artificial Intelligence when the capability of computers will surpass human beings. ASI is currently a hypothetical situation as depicted in movies and science fiction books, where machines have taken over the world.

Types Of Artificial Intelligence

Based on the functionality of AI-based systems, AI can be categorized into the following types:

1. Reactive Machines AI
2. Limited Memory AI
3. Theory Of Mind AI
4. Self-aware AI

Reactive Machine AI

This type of AI includes machines that operate solely based on the present data, taking into account only the current situation. Reactive AI machines cannot form inferences from the data to evaluate their future actions. They can perform a narrowed range of pre-defined tasks.

An example of Reactive AI is the famous IBM Chess program that beat the world champion, Garry Kasparov.

Limited Memory AI

Like the name suggests Limited Memory AI, can make informed and improved decisions by studying the past data from its memory. Such an AI has a short-lived or a temporary memory that can be used to store past experiences and hence evaluate future actions.

Self-driving cars are Limited Memory AI, that uses the data collected in the recent past to make immediate decisions. For example, self-driving cars use sensors to identify civilians crossing the road, steep roads, traffic signals and so on to make better driving decisions. This helps to prevent any future accidents.

Theory Of Mind AI

The Theory Of Mind AI is a more advanced type of Artificial Intelligence. This category of machines is speculated to play a major role in psychology. This type of AI will focus mainly on emotional intelligence so that human believes and thoughts can be better comprehended.

The Theory of Mind AI has not yet been fully developed but rigorous research is happening in this area.

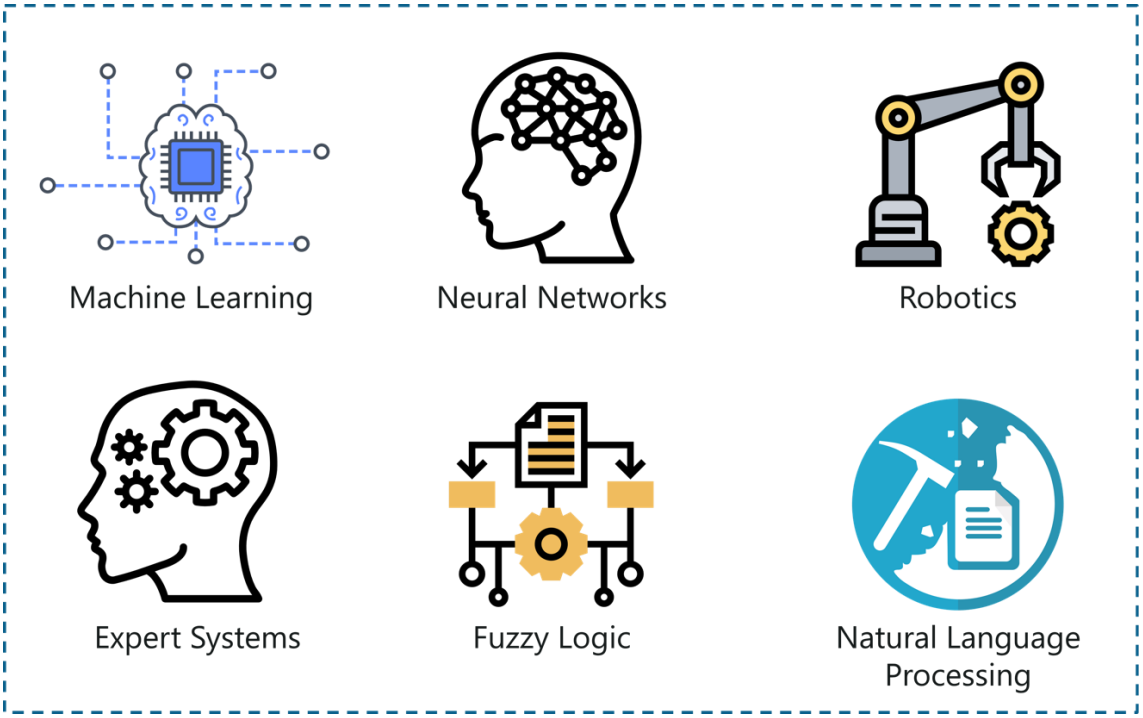
Self-Aware AI

Let's just pray that we don't reach the state of AI, where machines have their own consciousness and become self-aware. This type of AI is a little far-fetched given the present circumstances. However, in the future, achieving a stage of superintelligence might be possible.

Branches Of Artificial Intelligence

Artificial Intelligence can be used to solve real-world problems by implementing the following processes/ techniques:

1. Machine Learning
2. Deep Learning
3. Natural Language Processing
4. Robotics
5. Expert Systems
6. Fuzzy Logic



Python

When it comes to programming, adopting a language depends on what you want to accomplish. For example, if you want to write a code to solve complex numerical equations, you may use C++ or Fortran. Similarly, if you are interested in statistics, R may be a good choice. Accordingly, when it comes to implementation of pattern recognition or quick scripting for artificial intelligence problems, python is widely used as it is a higher-level language that is open source, cross-platform, and is easy to learn and code. Additionally, standard libraries for clustering, optimization, and classification are available for direct use in Python. You can find more about python at http://www.python.org/doc/ and http://www.diveintopython.org.

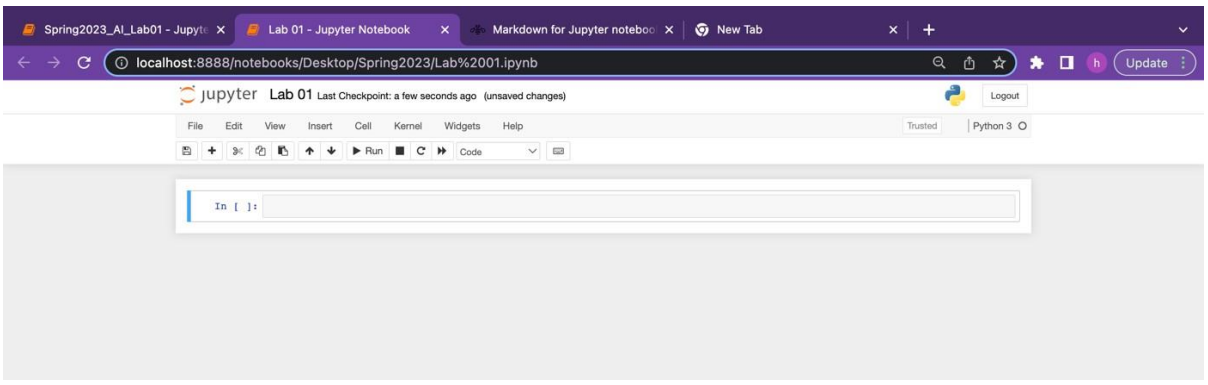
Python Libraries

A library is basically a collection of standard code or sub-routines that are frequently used in programming. A library helps a coder (you) to accomplish a programming task in a few lines of code by borrowing all the lines stored in a function within a library. One of the greatest strengths of Python is its large library that provides access to tools for numerical computationsas well as GIS applications. Some common libraries that we will be using in this class are briefly introduced below, along with a sample code that you can try in your Jupyter Notebook.

NumPy

If you are familiar with Matlab, numpy provides Matlab like functionality within Python. Using NumPy you can solve multi-dimensional arrays and matrices and perform simple to complex mathematical operations including statistics. Now, let's see how to use numpy in Jupyter Notebook to generate 1000 random numbers from a normal distribution with mean = 100 and standard deviation = 15.

When you open the notebook, you will have an empty module as shown below.



Write the first line of code as below:

```
In [ ]: import numpy as np
```

Using this single statement, we have imported or borrowed the numpy library and referenced it as np. This is how we will import any library. You can use any name to reference a library. In this case we used np as it is an abbreviation of numpy. You can use nmy, npy or anything. You get the point! Use something logical!

Now we are going to define two variables for the mean and standard deviation, and assign them the value of 100 and 15, respectively. Again, you can pick any name for defining a variable, but it is a good practice to use something that is self-explanatory. We use greek letters and for mean and standard deviation, respectively. Let's write our second line of codeto define these two variables as below.

```
In [1]: mu, sigma = 100, 15
```

Here we defined both variables in a single statement. We could also do mu = 100 on one line and sigma = 15 on another line. Once we have the mean and standard deviation, we can then generate values using the following statement.

```
In [4]: x = mu + sigma*np.random.randn(1000)
```

The randn function above generates random numbers from a standard normal distribution (mean = 0 and variance = 1), which is then scaled to our mu and sigma, and stored in variable x. Basically, x is an array to store 1000 random values from the specified normal distribution.

Q. What is an array?

We defined an array named x to store our random values. You can see the results by calling this x array. Simply type x in the code cell, and press shift + Enter. You should then see all the random values as shown below.

```
In [5]: x
Out[5]: array([125.74778438, 106.05457151, 110.41717146, 104.49064374,
               103.98835956,  91.32390617,  93.19642946,  97.73418627,
               106.82632139, 127.73862018, 127.38517572, 111.96190802,
               108.74744086,  92.70483936,  69.55039126,  79.27132812,
                85.52034697,  93.79844929,  73.41706201,  98.91848123,
               128.98488356,  84.40134551,  97.91311519, 143.7124608 ,
                98.40521043,  86.57875479, 100.14573388,  99.14475364,
                91.53563163, 108.14778573, 108.50175109,  66.2054245 ,
               103.17965796,  99.65424029, 102.06573622, 110.92723013,
                79.73718764,  81.26886801,  76.46451517,  85.20781582,
               100.23588532,  96.58204401, 110.8617336 , 110.39618399,
               107.93596818, 103.48009708,  96.51623319,  89.50731864,
                91.09443905,  93.25103096,  93.48550106, 105.86834394,
               120.82575941,  96.99696513, 107.34856441,  86.99196365,
               105.19671413, 114.45413102,  91.64654301,  91.72619932,
                93.48427107, 112.41083958,  86.23397592,  76.44844388,
                74.76584302, 113.62904744, 119.6899374 ,  84.32781577,
                65.5066625 ,  78.96169213,  99.27913046, 105.48583105,
                92.17849697, 112.42813965,  91.86177831,  81.01410599,
```

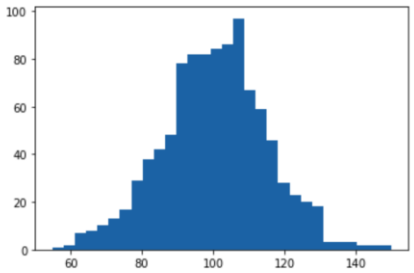
Matplotlib

Matplotlib is a plotting library for the Python programming language, and it can be used tomake plots. Let's use this library to create a histogram from the x array we just created. Import the pyplot sub-library from matplotlib and name it as plt.

Write the code in your code cell as shown in the fig below.

(Note: you can either write this statement after your last statement, but it is a good practice to have all imports on the top so you know what libraries are available in the code.)

```
In [10]: import matplotlib.pyplot as plt
         plt.hist(x,bins=30)
         plt.show()
```



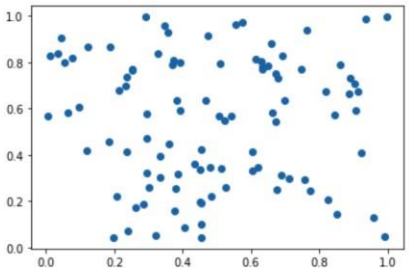
Create the histogram of x by using plt. Code `plt.hist(x,bins=30)` will generate the histogram by dividing the array into 30 bins. Bins size is a parameter so you can choose some other value if you wish. After creating the histogram, plot the histogram by using the `plt.show()`. Now run the code, and you should see something similar.

If you want, you can play around with the bin size to see how the plot changes. Lets use pyplot to create scatter plot between two random variables. Define two random variables a and b as shown below. Remember we used randn earlier to create random numbers from a standard normal distribution. The random function will generate purely random numbers. The number inside the parenthesis defines the size.

```
In [11]: a = np.random.random(100)
         b = np.random.random(100)
```

Now plot a and b on a scatter plot by using the code below. After you type plt, you will see a list of associated functions so pick scatter and provide a and b as two variables as shown below. Then show the plot.

```
In [12]: plt.scatter(a,b)
         plt.show()
```



Pandas

Python Data Analysis Library, or Pandas, is a Python package providing fast, flexible, and expressive data structures designed to make working with is “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

As usual, lets first import the pandas library as pd
(Note: All import statements will be at the beginning of your code)

```
In [13]: import pandas as pd
```

Now read the csv file from your working directory and store it in data as shown below:

```
In [1]: import pandas as pd
```

```
In [2]: path = "Auto85.csv"
        data = pd.read_csv(path, header = None) #assumes data has header

        data.head(5)

        headers = ["symboling","normalized-losses","make","fuel-type",
                   "aspiration", "num-of-doors", "body-style", "drive-wheels",
                   "engine-location", "wheel-base", "length", "width", "height",
                   "curb-weight", "engine-type", "num-of-cylinders", "engine-size",
                   "fuel-system", "bore", "stroke", "compression-ratio", "horsepower",
                   "peak-rpm", "city-mpg", "highway-mpg", "Price"]

        data.columns = headers

        data.head()

        path = "up_data.csv"

        data.to_csv(path)
```

The first line sets a variable 'path' to the file name "Auto85.csv", which is the csv file that we want to read. The second line reads the CSV file using the 'pd.read_csv()' function and assigns the data to a variable 'data'. The 'header = None' parameter tells the function that the file does not have a header row. The third line uses the 'head()' function to display the first n=5 rows of the data.

The next block of code creates a list of headers, which will be used to set the column names for the data. The following line sets the column names for the dataframe using the 'columns' attribute and the list of headers. The next line uses the 'head()' function to display the first five rows of the dataframe with the new column names. The last line writes the dataframe to a new CSV file using the 'to_csv()' function and the new file path.

Data before adding headers:

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

Data after adding headers:

Out[6]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115

5 rows × 26 columns

Other Supported formats

Data Format	Read	Write
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
Excel	pd.read_excel()	df.to_excel()
Sql	pd.read_sql()	df.to_sql()

Now we can plot information from data by using matplotlib

```
In [22]: data["Price"].replace("?", np.nan, inplace = True)
         data["Price"] = pd.to_numeric(data["Price"])
```

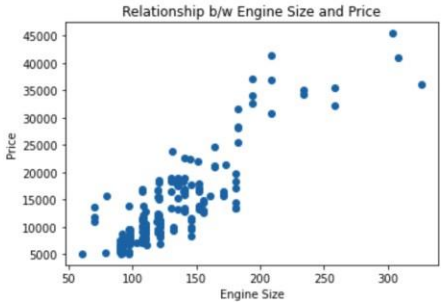
The first line uses the 'replace()' function to replace all instances of "?" in the "Price" column with a 'NaN' value (Not a Number). The 'inplace = True' parameter tells the function to make the changes to the original dataframe rather than returning a new one.

The second line uses the 'pd.to_numeric()' function to convert the values of the "Price" column to numeric data type. This is necessary because the "Price" column was read in as a string data type and it needs to be converted to a numeric data type for any mathematical operations to be performed on it.

```
In [23]: import matplotlib.pyplot as plt

         plt.scatter(data["engine-size"], data["Price"])
         plt.title("Relationship b/w Engine Size and Price")
         plt.xlabel("Engine Size")
         plt.ylabel("Price")
```

```
Out[23]: Text(0, 0.5, 'Price')
```



To visualize the relationship between "engine-size" and "Price" we are using scatter plot. The first line uses the 'scatter()' function to create a scatter plot with the "engine-size" column on the x-axis and the "Price" column on the y-axis. The second line sets the title of the plot as "Relationship b/w Engine Size and Price" using the 'title()' function. The third line sets the label for the x-axis as "Engine Size" using the 'xlabel()' function. The fourth line sets the label for the y-axis as "Price" using the 'ylabel()' function.

The scatter plot can be used to find whether there is a positive or negative correlation between engine size and price of the car. This can be determined by the slope of the plotted points. If

the slope is positive, then the engine size has a positive correlation with the price, if the slope is negative, then the engine size has a negative correlation with the price, if the slope is zero, then the engine size has no correlation with the price.

NetworkX

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks. It is used to study large complex networks represented in the form of graphs with nodes and edges. Using networkx we can load and store complex networks. We can generate many types of random and classic networks, analyze network structure, build network models, design new network algorithms, and draw networks.