**Machine Learning**

Involves the application and science of algorithms that make sense of data. In the second half of the twentieth century, machine learning evolved as a subfield of Artificial Intelligence (AI) that involved self-learning algorithms that derived knowledge from data in order to make predictions. Instead of requiring humans to manually derive rules and build models from analyzing large amounts of data, machine learning offers a more efficient alternative for capturing the knowledge in data to gradually improve the performance of predictive models and make data-driven decisions. Thanks to machine learning, we enjoy robust email spam filters, convenient text and voice recognition software, reliable web search engines, challenging chess-playing programs, and, hopefully soon, safe and efficient self-driving cars.

**The three different types of machine learning**

1. **Supervised-**The machine learning program is both given the input data and the corresponding labelling. This means that the learn data has to be labelled by a human being beforehand.
2. **Unsupervised learning-**No labels are provided to the learning algorithm. The algorithm has to figure out a clustering of the input data.
3. **Reinforcement Learning**- A computer program dynamically interacts with its environment. This means that the program receives positive and/or negative feedback to improve its performance.

In this handout, let us discuss some common statistical techniques. For the purposes of this lesson, we will use only two python modules i.e numpy (Numerical Python), matplotlib. You can also use other such as pandas, sklearn and their dependencies. Alternatively, you can pip install the modules. These modules are called whenever you want to manipulate your data sets. A data set is any collection of data for instance an array of testmarks. Let us consider the following dataset for marks.

Testmarks = [98,78, 68, 73, 72, 97, 88, 60, 94, 95, 80, 73, 82, 80, 99, 91, 74, 88, 70, 94, 86, 81, 100, 99, 84, 93, 94, 79]

If you have an array of marks like the one above, you may want to calculate the mean (average), median (middle value) and mode (mark which appears most). Thanks to machine learning as it makes all these calculations easy.

Remember, most data fall into one of two groups: numerical or categorical or ordinal types.

**Numerical data**- These data have meaning as a measurement, such as a person’s height, weight, IQ, or blood pressure; or they’re a count, such as the number of stock shares a person owns, how many teeth a dog has, or how many pages you can read of your favorite book before you fall asleep. (Statisticians also call numerical data quantitative data. Numerical data can be further broken into two types: discrete and continuous. **Discrete data** represent items that can be counted; they take on possible values that can be listed out. e.g 1, 2, 3, 4 etc.

**Continuous data**-numbers that are of infinite value. Example: The price of an item, or the size of an item.

**Categorical data:** Categorical data represent characteristics such as a person’s gender, marital status, hometown, or the types of movies they like. Categorical data can take on numerical values (such as “1” indicating male and “2” indicating female), but those numbers don’t have mathematical meaning.

**Ordinal data** mixes numerical and categorical data. The data fall into categories, but the numbers placed on the categories have meaning. For example, rating a restaurant on a scale from 0 (lowest) to 4 (highest) stars gives ordinal data.

**Measures of central tendency**

We are going to calculate mean, mode and median of the Testmarks data set below.

Testmarks = [98,78, 68, 73, 72, 97, 88, 60, 94, 95, 80, 73, 82, 80, 99, 91, 74, 88, 70, 94, 86, 81, 100, 99, 84, 93, 94, 79]

For mean and median, numpy module works. As for mode, which the value with more occurrences, we will import SciPy module and use mode () method to find the number that appears the most.

Try the following:

meanmarks= np.mean(Testmarks) *#calculating mean*medianmark=np.median(Testmarks) *# calculating median*modemarks=stats.mode(Testmarks) #calculating mode (value with highest frequencies)

When you print the outcome, the value will be printed and number of occurrences. (remember the number of occurrences challenge. Take note that these methods are quite hand when you have very big data sets. Imagine you have 5000 marks and you want to find mean, median and mode.

Now let us move on to standard deviation.

**Measures of dispersion**

**Standard Deviation**

In statistics, the standard deviation is a measure of the amount of variation or dispersion of a set of values. std refers to how spread the values within a dataset are. Standard deviation shows how much variation (dispersion, spread, scatter) from the mean exists. In most real-world applications, consistency is a great advantage. In statistical data analysis, less variation is often better. A low standard deviation means that most of the numbers are close to the mean (average) value. A high standard deviation means that the values are spread out over a wider range.

**Try the following**

Outliers are not just greatest and least values, but values that are very different from the pattern established by the rest of the data. Outliers affect the mean. When outliers are present it is best to use the median as the measure of central tendency.

Testmarks = [98,78, 68, 73, 72, 97, 88, 60, 94, 95, 80, 73, 82, 80, 99, 91, 74, 88, 70, 94, 86, 81, 100, 99, 84, 93, 94, 79]  
myStd=np.std(Testmarks)  
mymean=np.mean(Testmarks)

You will notice that the mean is 84,64. The standard deviation is 11.0. that is the variation with most of the values. However, we see that 100 is outside the standard deviation and these are called outliers.

**Variance**

Variance measures how far a set of data is spread out. A variance of zero indicates that all of the data values are identical. All non-zero variances are positive. A small variance indicates that the data points tend to be very close to the mean, and to each other. A high variance indicates that the data points are very spread out from the mean, and from one another.

**Percentiles**

A percentile is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations falls.

twentyfivep=np.percentile(Testmarks, 25)

The answer is 77. meaning that 25% of the people got 77% or below.

We can run the same and use 50 percentiles and we get 85 meaning that 50% got 85% or below.

**Data Distribution**

For data visualisation, Python allows use of Matplotlib. Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonot, Tkinter.

**Bar graph**

A bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally. Let us consider rainfall recorded in the different towns in mm for four cities in South Africa as below.

Rainfall= [140, 200, 120, 157]

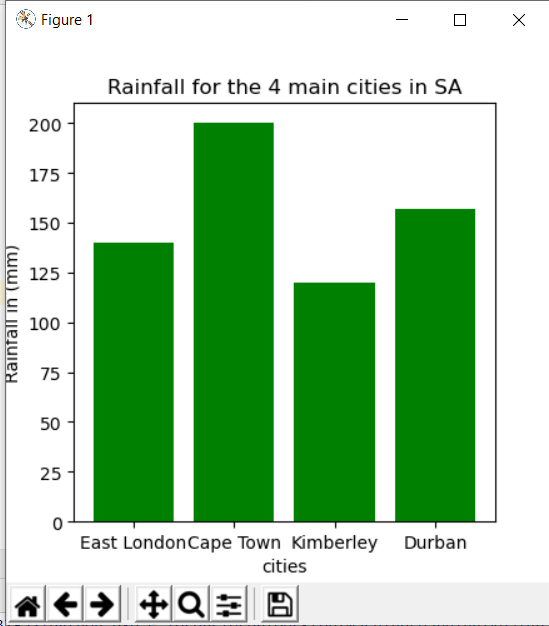
Cities=[“East London”, “Cape Town”, “Kimberley”, “Durban]

## For us to present the following diagrammatically, we need matplotlib.pyplot. Matplotlib. pyplot provides a MATLAB like plotting framework. The plt.bar function, however, takes a list of positions and values, the for x are then provided by plt.xticks().

## Try the following code and see how the graphs are presented.

**import** matplotlib.pyplot **as** plt  
cities=[**'East London'**, **'Cape Town'**, **'Kimberley'**, **'Durban'**]  
rainfall= [140, 200, 120, 157]  
x\_pos = [i **for** i, \_ **in** enumerate(cities)] *#labels on the x-axis  
#labeling and visuals*plt.bar(x\_pos, rainfall, color=**'green'**)  
plt.xlabel(**"cities"**)  
plt.ylabel(**"Rainfall in (mm)"**)  
plt.title(**"Rainfall for the 4 main cities in SA"**)  
plt.xticks(x\_pos, cities)  
plt.show()

You will have an output as show below



Take note that the image allows you to save it and you can use it for your presentations.

**Box Plots**

Boxplot is probably one of the most common type of graphic. It gives a nice summary of one or several numeric variables. The line that divides the box into 2 parts represents the median of the data. The end of the box shows the upper and lower quartiles. The extreme lines show the highest and lowest value excluding outliers. Remember we spoke of outliers at the beginning.

We are going to use testMarks data set for our boxplots.

Testmarks = [98,78, 68, 73, 72, 97, 88, 60, 94, 95, 80, 73, 82, 80, 99, 91, 74, 88, 70, 94, 86, 81, 100, 99, 84, 93, 94, 79]