Week-1 ML Notes (Kartik Nooney)

Machine Learning:- how to construct computer programs that automatically improve with experience.

The goal of ML is never to make "perfect" guesses, because ML deals in domains where there is no such thing. The goal is to make guesses that are good enough to be useful.

There are two types of machine learning:

1. **Supervised machine learning:** The program is "trained" on a pre-defined set of "training examples", which then facilitate its ability to reach an accurate conclusion when given new data.
2. **Unsupervised machine learning:** The program is given a bunch of data and must find patterns and relationships therein.

ML builds heavily on statistics. For example, when we train our machine to learn, we have to give it a statistically significant random sample as training data. If the training set is not random, we run the risk of the machine learning patterns that aren't actually there. And if the training set is too small (see law of large numbers), we won't learn enough and may even reach inaccurate conclusions.

Data exploration, cleaning and preparation can take up to 70% of your total project time. For more detailed methods in this aspect visit the site (https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/)

Popular Algorithms in ML:-(For definitions https://upxacademy.com/10-popular-machine-learning-algorithms/)

1. Linear Regression
2. Logistic Regression

3. Decision Tree
4. Random Forest
5. Artificial Neural Network
6. Support Vector Machine
7. K-Means Clustering
8. K-Nearest Neighbour
9. Naive Bayes Classifier
10. Ensemble Learning

Deep Learning: is the computing process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.

Data Points :- They are basically the observations in the data or rows in your dataset. They can also be termed as measurements in a statistical population.

Data Frames :- They are just like 2D - Arrays, and can be of any datatype.

Toy Dataset :- It is a sample or a dummy dataset used for testing in local machines.

Classes/Labels :- The class label is usually the **target variable** in classification. Which makes it special from other categorial attributes. In particular, on your actual data it won't exist - it only exist on your training and validation data sets. Its basically the final output categories into which your data is being segregated into. Say for *"Iris Dataset"* there are three classes or labels in the final outcome, i.e., there are three types of classifications of flowers in that dataset.

Difference between arrays lists and tuples:-

An array is a relatively basic sort of data structure, and is generally considered to have the following characteristics:

- It has a fixed size
- It's made up of a contiguous chunk of memory

As a consequence, accessing an arbitrary element inside of an array is an O(1) operation, but resizing that array or adding a new element is an O(n) operation since you need to copy every value over to a new array.

A list is similar to an array, but has a variable size, and does not necessarily need to be made up of a single continuous chunk of memory. While this does make lists more useful in general then arrays, you *do* incur a slight performance penalty due to the overhead needed to have those nicer characteristics.
I should also note that a List is not a specific kind of data structure. Rather, it's an abstract data type -- a description of what a data structure should behave like. There are several different kinds of list implementation you can try using, but the two most common ones are an array list and a linked list. An array list is basically a lightweight wrapper around an array. When the internal array is full, the array list class will automatically allocate and use a bigger array.
A linked list works by chaining together a sequence of "node" objects, each of which contains a value, and points to the next and previous elements in the list. (The benefit of this data structure is that appending new values is an O(1) operation).

Tuples are a little different then both arrays and lists. Arrays and lists are intended, as you said, to hold a sequence of homogenous values, whereas a tuple isn't necessarily intended to be iterated over and can hold heterogeneous values.
(Tuples also are typically immutable -- you cannot change their values or add/remove new elements. That makes them more lightweight then lists).

Tuples have a structure where as lists have an order. Tuples are heterogenous data structures while Lists are homogeneous sequences.

Vector Space :- Vector spaces roughly speaking specifies the number independent directions in space or without much jargon, it just gives us the number dimensions. There can be n-dimensional vector spaces as well. In general a vector space is a collection of objects called vectors which can be added together or can be multiplied or scaled with numbers which are generally scalar in nature.

Feature Vector:- It is basically a n-dimensional vector with numerical features representing some object during mathematical representation of models or objects in machine learning. If say we are representing images using these *feature vectors* then in this scenario feature vectors might hold the pixel values of the corresponding image. Incase if we are representing some amount of text then features vectors can be used to represent occurrence frequencies.

Difference between Feature, Feature-Vector and Feature-Set:-

Feature is an individual measurable property of a phenomenon being observed.

There might be various features corresponding to a phenomenon or an observation and if we put all those features together in a vector then a Feature-Vector is formed. Feature-vector is basically an array.

Say for example, I am representing a human being. Then various features could be the age, weight, height, etc. Now if I place these features in a vector, a feature vector is formed. From what I have got, the feature vector (an array), in this case will be having three elements- The first being the age, then weight and finally the height. Similarly for 'n' individuals, I would be having 'n' feature vectors each having 3 elements.

The difference between a set and a vector is the a set has no inherent order and contains no duplicates, while a vector has order and may contain any value (similar to a 1-D Matrix)

Difference between Sparse-Vector and Dense-Vector :-
Roughly speaking, Sparse-Vectors are n-dimensional vectors having less than O(n) non-zero entries. Most libraries that use linear-algebra data types implement these vectors as arrays of length-n by default. This is called *Dense* representation. If you know that you are going to work with sparse vectors then you can use another kind of representation that uses the advantage of sparsity and actually stores *only non-zero entries*. This is called *Sparse Representation*. You can implement it for example by storing indices and appropriate numbers as key-value pairs.

Support Vector Machine:- An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

Clustering: is the assignment of a set of observations into subsets (called **clusters**) so that observations in the same **cluster** are similar in some sense. **Clustering** is a method of unsupervised **learning**, and a common technique for statistical data analysis used in many fields.

Different types of plots: (https://elitedatascience.com/python-seaborn-tutorial)

- Violin plots are useful alternatives to box plots. They show the distribution (through the thickness of the violin) instead of only the summary statistics. Violin plots are great for visualizing distributions. A violin plot combines the benefits of the previous two plots and simplifies them. Denser regions of the data are fatter, and sparser thiner in a violin plot.
- Heatmaps help you visualize matrix-like data.
- Histograms allow you to plot the distributions of numeric variables.
- Bar plots help you visualize the distributions of categorical

variables.

- Factor plots make it easy to separate plots by categorical classes.
- Density plots display the distribution between two variables.
- Joint distribution plots combine information from scatter plots and histograms to give you detailed information for bi-variate distributions. A seaborn jointplot shows bivariate scatterplots and univariate histograms in the same figure.
- Box Plot and Quartiles, detailed explanation withe example can be found in this website (http://www.purplemath.com/modules/boxwhisk.htm)
- Box and whisker plots tips:- If you've got a wide box and long whiskers, then maybe the data doesn't cluster as you'd hoped (or at least assumed). If your box is small and the whiskers are short, then probably your data does indeed cluster. If your box is small and the whiskers are long, then maybe the data clusters, but you've got some "outliers" that you might need to investigate further — or, as we'll see later, you may want to discard some of your results.

Imbalanced Datasets: Imbalanced data sets are a special case for classification problem where the class distribution is not uniform among the classes. These type of sets suppose a new challenging problem for Data Mining, since standard classification algorithms usually consider a balanced training set.

Probability Density Function (PDF): The PDF is used to specify the probability of the random variable falling *within a particular range of values*, as opposed to taking on any one value. It is a function representing the relative distribution of frequency of a continuous random variable from which parameters such as its mean and variance can be derived.

Cumulative Density Function (CDF): Probability is a measure of the certainty in which an event might occur. This definition is easily implemented when dealing with several distinct events. When a continues random variable is examined, however, it becomes harder to use this definition directly, because the term "event" is somewhat elusive. If we want to define the probability of an event in which the random variable takes a specific value, the probability for this event will usually be zero, because there are infinitely many distinct values in any continues interval. And it is not very useful to define probability as having a value of zero everywhere. Instead, we define the probability of events in which the random variable takes a value within a specific interval. The customary way to do it is by using a cumulative distribution function (CDF).

Difference between PDF & CDF: A PDF answers the question: "How common are samples at exactly this value?" A CDF answers the question "How common are samples that are less than or equal to this value?" The CDF is the integral of the PDF. The slope of the cumulative distribution function is the probability density function .

Long Tail Distribution: In statistics and business, a **long tail** of some distributions of numbers is the portion of the distribution having a large number of occurrences far from the "head" or central part of the distribution. The example of long tail distribution is AMAZON PRIME DAY Sales.

Normal Distribution: The normal distribution is sometimes informally called the **bell curve**. However, many other distributions are bell-shaped. It is symmetric around the point x=μ which is at the same time the mode, the median and the mean of the distribution. (https://en.wikipedia.org/wiki/Normal_distribution)

Central Limit Theorem: The central limit theorem (CLT) is a statistical theory that states that given a sufficiently large sample size from a population with a finite level of variance, the mean of all samples from the same population will be approximately equal to the mean of the population. (https://en.wikipedia.org/wiki/Central_limit_theorem)

Basics of Jupyter :-

- The In[some number]: indicates the index location in cache where the processed input code is saved for future use. Similarly Out[some number]: indicates the index location in cache where the processed output code is saved for future use. If there is In[*] then it means that the kernel is still processing the code.

- The csv files or whatever input files you are reading in, basically have to be in the same directory as that of you ipython notebook. For that you can either create your ipython notebook in the directory where your input files are or you can change the directory using python code in your ipython notebook to point out to the location of the input data reading it. The code for changing the location in the ipython notebook is as follows:

- *import os*
- *os.chdir("give the path of your directory where your file is present, don't include your file name in this path.")*

Pandas: Pandas is great for data manipulation, data analysis, and data visualisation. The Pandas modules uses objects to allow for data analysis at a fairly high performance rate in comparison to typical Python procedures. With it, we can easily read and write from and to CSV files, or even databases. From there, we can manipulate the data by columns, create new columns, and even base the new columns on other column data. Next, we can progress into data visualisation using Matplotlib.

Useful Links for built in functions of pandas:

- Methods related to Dataframe objects (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html)
- Basic cookbook on overall features in pandas (https://pandas.pydata.org/pandas-docs/stable/tutorials.html)
- Pandas complete ***API documentation*** can be found here (http://pandas.pydata.org/pandas-docs/version/0.15.2/api.html)

Sentdex pandas tutorials are helpful. (https://www.youtube.com/playlist?list=PLQVvvaa0QuDfHt4XU7vTm22xDegR0v0fQ)

The following are few properties in Pandas library:-

- *Index* is basically the first column of your data but you can also specify what the index is in your data.
- *Series* in Pandas is a 1D-Array and *DataFrames* is a 2D- Array. Series and data-frames can also act as dictionaries in pandas.
- *Slicing* is selecting specific part of the entire data. You could create new data-frames and assign to them one or more columns from your entire data. And when you print this new data-frame,

you will get the index column and also the columns you specifically sliced into the new data-frame.

- You can do mathematical operations on rows, you can also interchange the position of rows, there are several built in commands to find averages or standard deviation.
- *Graphing* , it uses matplotlib. Graphing means, you can basically take a data-frame and load it in a matplot library. Pandas doesn't have 3D graphing built in, so you have to import that specifically from matplotlib. But pandas has 2D graphing builtin.
- *Conditional Statements,* you can use conditional statements while slicing the data, i.e., it will slice only the specific data in that data-frame for which that conditional statement holds true.
- *Mathematical Operations* such as moving averages (called as ROLLING MEAN in pandas can also be performed on columns. Moving averages though a new term, is just like an arithmetic mean. It is basically used in stock-market. Moving averages are calculated by adding the previous X number of periods value and dividing their sum by the total number of periods. For example a five day moving average is calculated by dividing the previous five days price data and dividing by 5. (A simple arithmetic mean). Each time a new period occurs, the moving average moves forward, dropping the first datapoint and adding the newest one. For more details about moving averages watch the following video tutorial (http://www.investopedia.com/terms/m/movingaverage.asp)
- We could get the description of the entire dataset using the .describe() method, it gives us all the statistical information.
- The .corr() method gives us the correlation between each of the features, such as how one feature changes when another particular feature is changed. It's value ranges from -1 to +1, where the negative sign indicates the inverse proportionality and positive sign indicates the direct proportionality of the features.
- The .head() method gives us the first five rows of the entire data and it also gives us the header row.

Matplotlib: Detailed matplotlib tutorial videos can be found here (https://www.youtube.com/playlist?list=PLQVvvaa0QuDfefDfXb9Yf0la1fPDKluPF)

- Generally the statement to import the matplot library is *import matplotlib.pyplot as plt*

- You can name the imported object as anything you want but plt is sort of the standard convention.
- You can plot as many graphs as you can using pandas and they will be plotted and ready in the background but to display them you need the help of matplot library and the command to display our plotted graphs is *plt.show()*
- If you are not using pandas to plot the graph, in that case you can also use the matplot libraries and you can simply pass in the "x" and "y" parameters into the *plt.plot()* function.
- You could also label your X and Y axes using *plt.xlabel("name")* and similarly for Y axis we have *plt.ylabel("name")*. You can replace that argument "name" to whatever you want to.
- You could also add a title to your graph using *plt.title("name")*
- Legend is a box on your graph displaying which colour line represents which feature. You can play around with legends, like their positions, size and other properties.
- There are ***BAR CHARTS*** *and* ***HISTOGRAMS*** supported by this library. Histograms are meant to show the distribution, whereas bar-charts are representation of one series, but sometimes you might be comparing things using bar-charts also. For clear understanding of Bar-Charts and Histogram (http://stattrek.com/statistics/charts/histogram.aspx?Tutorial=AP)
- plt.bar() is the method for bar charts, it has several parameters associated to that method such as the axis names and label name, color of the bar graph etc.
- ***SCATTER PLOTS*** are simple type of graphs, generally the idea of the scatter plot is to show the correlation between something or distribution of data. But generally it is mostly used to show the relationship between two types of variables. We are specifying two here because it is a 2D Scatter plot. We also have 3D scatter plots where in you can compare three variables at the same time.
- plt.scatter() is the method used for drawing scatter plots, it has several parameters as input just like that of bar-charts. You can also specify the marker and marker size as parameters.
- For 3D plots the main statement to is *import mpl_toolkits.mplot3d import axes3d*
- You can do different 3D plots, such as 3D scatter plot or bar charts or histograms etc. You can apply different styles to your 3D plots. Most of the styles are present as predefined methods so you can lookup the documentation for different styles. For tutorial on 3D plots watch (https://www.youtube.com/watch?

v=ecZZ8CvNQ6M&index=29&list=PLQVvvaa0QuDfefDfXb9Yf0la1fPDKluPF)

Seaborn: Detailed seaborn information can be found here (https://elitedatascience.com/python-seaborn-tutorial)

- All the ***API documentation*** related to seaborn library can be found here (https://seaborn.pydata.org)
- One of Seaborn's greatest strengths is its diversity of plotting functions. For instance, making a scatter plot is just one line of code using the *lmplot()* function.

Seaborn is a library for making attractive and informative statistical graphics in Python. It is built on top of matplotlib and tightly integrated with the PyData stack, including support for numpy and pandas data structures and statistical routines from scipy and statsmodels.
Some of the features that seaborn offers are

- Several built-in themes for styling matplotlib graphics
- Tools for choosing color palettes to make beautiful plots that reveal patterns in your data
- Functions for visualizing univariate and bivariate distributions or for comparing them between subsets of data
- Tools that fit and visualize linear regression models for different kinds of independent and dependent variables
- Functions that visualize matrices of data and use clustering algorithms to discover structure in those matrices
- A function to plot statistical timeseries data with flexible estimation and representation of uncertainty around the estimate
- High-level abstractions for structuring grids of plots that let you easily build complex visualizations