**100 DAYS OF CODING CHALLENGE**

**DAY 1:**

**Having installed Jupyter notebook and Anaconda,I was surprised to know that arrays are better than python lists in many ways. For a long time, I thought python lists are better than arrays because of dynamic data type. However, having learnt some serious flaws in this dynamic data , I completed learning how to create numpy arrays and was overwhelmed by the countless ways to create numpy arrays.**

**DAY 2:**

**After learning how to create numpy arrays from scratch, I learnt how to manipulate arrays using operations like indexing, slicing, reshaping, splitting arrays and joining arrays. Numpy has some great functions for these operations and it was very easy to learn all of the operations in both single and multi-dimensional arrays. Also it was inferred that unlike python lists numpy lists are shallow copies of the main array.**

**DAY 3:**

**Day 3 focused on learning one of the most important concepts of numpy arrays, which was, ufuncs. With the help of timeit function, I was able to visualize how vectorized ufuncs saved a lot of time performing operations on arrays.I was able to use powerful calculations on arrays such as aggregates and was able to access elements with ease without the need of for loops.**

**DAY 4:**

**I learnt Min and max aggregates which helped me deal with minimum and maximum values of a numpy array.I also learnt how to find sum of elements using sum function and also found sum of elements along rows and columns.**

**DAY 5:**

**Broadcasting was different compared to other concepts I learnt so far because it was based on rules. However, I was able to perform operations on arrays of different sizes which I was very doubtful on how to do on numpy using this interesting concept.**

**DAY 6:**

**This day was probably the day I was most overwhelmed by functions as I started learning an extremely powerful concept of masking and the amazing use of Boolean functions in arrays.I was able to count, modify and display elements based on conditions I give, which was seemingly a hectic task on other data structures. This was the day I realized how truly powerful numpy is.**

**DAY 7:**

**Fancy indexing enabled me to access any element in the array by passing an array as an index which I thought was extremely clever. I saw and practiced some programs combining fancy indexing with masking which truly enhanced the accessing of elements to its peak. Fancy indexing was also versatile enough to be combined with other indexing methods, proving the flexibility of numpy.**

**DAY 8:**

**Sorting of arrays has been something I have doing for a long time in computer science and I wasn’t surprised to see numpy sort function to sort arrays. I was able to sort along rows and columns for multidimensional arrays as well. I, however was surprised to learn a new concept called partial sorting which was something I had never seen before. This handy sorting allowed me to find k minimum values of an array with real ease.**

**DAY 9:**

**Structured array was in my opinion something which sort of destroyed the very concept of arrays being a single data but opened doors to countless possibilities. With structured arrays I was able to make dictionary like operations with multiple data types and accessing each key and value was very easy as well. It was then revealed this was less applicable for the real world but however laid the path for my next package in python: Pandas.**

**DAY 10:**

**Having finished numpy,I started learning on the three main objects in pandas which was Series, dataframes and Index. It was very easy to realize why pandas was much efficient than numpy and previous data structures, mainly because of the customizable indexing and the tabular format in which it displays output. I was also glad to know that the previous topic of Structured arrays was able to be converted to pandas dataframe every easily.**

**DAY 11:**

**I learnt how to properly index arrays and also use dataframe and data selection in pandas. I also got to learn how pandas has utilities like iloc and loc to handle potential confusion in indexing arrays and also how Series object is similar to 1 dimensional array and dataframe is similar to 2-dimensional array.**

**DAY 12:**

**Just like Numpy, Pandas has my favorite ufuncs and I learnt how to add two Series objects and two dataframe objects and also a series and a dataframe object. I also learnt the basis of Nan and Nan values can be customized for missing data when adding pandas objects. This gave me a good foundation for the upcoming concept of dealing with missing data.**

**DAY 13:**

**Python has two data types for missing data none and nan and I learnt a great deal of theory on how missing data was dealt in other languages and pros and cons of each method. I have now learnt how to to add Null values, automatically making python typecast none and add and remove and fill Null values in both Series and dataframe objects in pandas.**

**DAY 14:**

**To deal with Multidimensional Data, Python has MultiIndexing which allowed me to add more levels of index and for the first time, I dealt with real world data by taking petrol prices of 3 states and trying out operations on them. Compared to all the topics I have seen this topic has proved to be the most difficult because it required a great deal of understanding of pandas objects and had too much features packed in it. This one of the topics which I might have to repeatedly revise until I am confident in it.**

**DAY 15:**

**Pandas has concatenation function called concat which enabled me to combine datasets. Within, concat i learnt how to overcome the major problem of dealing with repeating indices by throwing errors, ignoring repeating indices etc. I also ran into some deprecations and learnt how to overcome them by browsing stack overflow**

**DAY 16:**

**While concat enabled combining of datasets in a simple manner, merge and join provided multiple ways to combine datasets by checking common column names, Index names, allowing user to define the columns or indexes in which the concatenation must happen. I also learnt how to obtain both intersection and union when combining datasets and also how to change suffixes so that columns with same name when concatenating datasets can be distinguished.**

**Grouping and Aggregating data was one of the most detailed I have learnt with the practical data of planets dataset which I obtained from seaborn, which is subsequently about to be covered in matplotlib. I learnt how to group data based on keys and also how to change values in dataframes by passing user defined functions, filtering data by defining functions, aggregates from list of operations. As a final test, I also tried grouping data from planet dataset and although I couldn’t complete it on my own,it gave me a decent grip on grouping in python.**

**DAY 17:**

**While working with grouping in pandas, I thought it was cumbersome every time I had to pass a list of indexes and unpack the pandas object to display result. With pivot tables I was able to group on the basis of multiple criteria and also had multiple rows and columns to my dataframe.I again worked with a practical data of information of titanic passengers and the grouping I did using pivot tables surprisingly allowed me to draw so many conclusions from raw data accurately. In the end I saw how this is feature was also closely related to matplotlib as well.**

**Vectorized string operations in pandas allowed me to use all the extremely efficient string functions in python with ease. It was a great nostalgia to use all string functions again which I learnt long back ago. I also learnt other handy string functions and also learnt that numpy couldn’t support vectorized string functions and pandas has ensured the use of these vectorized string functions. I also practiced with a database on online recipe with which I aggregated data based on string functions.**

**I never expected that time would play a role in pandas. However, to my surprise time has an important role in pandas mainly because of the use of pandas for finance which requires calculations across certain time periods. Numpy and python had their own time objects which had advantages and disadvantages and I was impressed how pandas managed to be the best of both worlds by combining all the advantages. I also learnt how to resample data, shift time and rolling data with help of apple stock market data. However, I found the last practice problems to be extremely difficult mainly because of not knowing matplotlib and the complex use of time series objects and precision calculations. So, I currently skipped those problems now, something which I didn’t do till now.**

**I had seen the word query ad eval pop up sometime I the previous exercises and this chapter mainly focused on the efforts to reduce the computation time of arrays and the drawback of creation of intermediate arrays in vectorized operations. Using eval I was able to do most of the vectorized functions at almost half the time taken by traditional methods and with the use of query, masking became extremely easy in dataframes.I also learnt when to use all the three computational functions depending on data size, memory and speed of the system. With this I have covered the basics of the daunting Pandas. The next topic I am going to delve is plotting and graphical representation of data by a very special python library: Matplotlib**

**DAY 18:**

**Having installed matplotlib,I studied briefly the history of matplotlib and learnt the different modes and platforms matlplotlib can be used. Matplotlib also allows the saving of graphs in various image formats and currently I am saving them in “.png” format. Matplotlib has two approaches: one similar to MATLAB and one of object-oriented type. I learnt how to plot simple y=f(x) functions in matplotlib and basic adjusting of axes, legend, labels and plot colors. It was nice to see visual outputs compared to the dull tables and arrays given by pandas and numpy.**

**After line plotting, I learnt how to display scatter plots in matplotlib by using both the plot and scatter library. I also learnt how to combine both scatter and line plots, how to customize markers and how to map colors in scatter plot. The understanding of fundamental differences between plot and scatter methods proved that plot was efficient for larger databases than scatter. It helped me to understand better with help of generating random arrays for selecting colors and sizes.**

**Visualization of errors helps us understand the data with much precision and matplotlib has error bars to show errors in data. I learnt how to customize error bars colors, size and the range it could cover. For continuous data with thousands of points, rather than constructing error bars for each point I understood how to use fill error region around the points with a particular color so that it would much easier to understand than a messy graph with many error bars. I revised how to use the fill function as well to color graphs in matplotlib.**

**Contours was a topic I had to do additional work since it was a mathematical graphing method which I never learnt and had no idea how it works. After watching videos from YouTube, I learnt what a contour was and started plotting three-dimensional data in a plane in matplotlib using contour and contour methods. Although the concept was confusing in theory it was very easy to implement in matplotlib because of the great library system and easy syntax. I also learnt how to interpret a dimensional grid as a image with plt.imshow() and used it for obtaining contours in which colors were more uniform rather than being discrete in plt.contour()**

**DAY 19:**

**Histograms are a great way of representing data and I learnt how to plot histograms using the hist function in matplotlib. The hist function also allowed for multiple customizations and enabled me the specify the number of bins, transparency and color mapping to be used for representing data. Matplotlib also has a hexagonal representation of 2-dimensional data which although I understood very well in matplotlib, found it confusing in theory as I never came across 2D histograms before. Having watched other resources, I was able to understand how 2D histograms were plotted. I also found how to represent data in non-visual forms with help of np.histogram and np.histogram2d.In the end I practiced a exercise on Kernel Density Estimation which the author said would be covered in the upcoming chapters where I had to plot 2d histogram of a Gaussian function.**

**Legend customization was the next topic I studied. In legend customization I learnt how to customize the legend frame by turning on the frame, transparency, text padding, defining number of columns and the location of the legend. I also learnt how to make a legend for points with different sizes by creating dummy legend plots by plotting empty lists with size of the point. I learnt this through an extremely practical example of plotting population of California cities. Lastly, with the help of Legend module and a lower-level function add\_artist I learnt how to have multiple legends in a same plot.**

**DAY 20:**

**In Colormaps customization I learnt some of the commonly used colormaps in matplotlib and also three broad categories of colormaps.I learnt how to change colormaps for a colorbar and also when and where to use appropriate colormaps and strengths and weakness of each colormaps. I also saw a look at how color mapping can be made discrete rather than continuous, how limits can be set for colorbars and how out of bounds can be represented in colorbars, with application of representing data subject to noise signals.**

**Subplots are particularly used for handling multiple plots and I learnt four methods to use subplots. The first method was to use axes and specify the parameters of subplot via a list of four arguments. The second method was to add subplots using subplot() while specifying number of rows and columns. The third method was to use subplots() by which we can return a numpy array of axes instances and also label all subplots with same x and y axis. The last and best Method is to use Gridspec which creates an interface recognized by plt.subplot() which allows us to access various subplots with ease and draw separate graphs.**

**Adding text and annotations was the next thing I learnt. I learnt how to add text in matplotlib at specific points by adding text in a graph of US birth rates. I learned about transforms which enabled me to assign text which doesn’t change despite the data being changed and I learnt all the three types in it. Drawing arrows in Matplotlib was the time I realized the drawbacks in matplotlib and I learnt that despite having an arrow function I couldn’t use to because of plot dependent SVG objects and used the annotate functions and ran through huge options regarding the arrows which I mostly glanced and brushed aside.**

**DAY 21:**

**I observed how to customize ticks both the bigger and the smaller ones using formatter and locator classes and also learnt how almost everything in matplotlib is aimed to be object oriented like python. I learnt how to change ticks, how to remove ticks and how to create fancy ticks by passing custom functions and also leant about how to evenly space ticks and other tick manipulation in matplotlib.**

**In customization of Matplotlib it was possible for me to change the color of graphs, bar color and edges, removing ticks, setting color of ticks and so on both by hand and by using rcParams to make them as default settings. I also learnt how to use various stylesheets such as grayscale, dark\_background, and seaborn to make my plots look more visually aesthetic. The author has suggested seaborn to be the best tool so far as I observed it to be the last concept of matplotlib in the book.**

**Matplotlib has tools for plotting 3D plots as I plotted 3d shapes,3D contours and also plotted graphs based on wireframe. I also plotted scatter plots on 3d planes and saw the adjustments on transparency that matplotlib does to ensure better visualization. Later I learnt how visualization in wireframe graphs can be made by changing view points and can be made better by filling the faces with colormaps and also saw the power of triangular plots rather than cartesian. As a summary I plotted a twisted loop (Mobius strip) with matplotlib and used triangulation for the best possible visualization of the shape.**

**Geographical maps were possibly the most horrendously difficult part in matplotlib as it was a great struggle for me because of requiring a great deal of understanding cartography and to even get the basemap imported after a great deal of struggle I finally imported basemap. Then I ran into a bug of values being out of range and hence the image of earth not being displayed properly. I learnt how to plot places with latitude and longitude by marking my own native place in a map. I learnt how to use shaded relief to display maps of various projections such a cylinder, cone, pseudo cylinder and perspective view. I again ran into a serious bug of a white image appearing in pseudo cylindrical projection and ortho projection. I however understood maps in a very well in the final exercises given in the notebook where I marked coastal areas, states and also customized legends and made population analysis**

**DAY 22:**

**I next stumbled upon the seaborn package. Seaborn allowed my to do incredible data visualization and made it so easy to visualize data compared to matplotlib. With seaborn I was able to plot joint plots, pair plots, hexagonal plots with histograms. I also learnt about kernel density estimation and plotted my own pandas dataframe using kde.I then plotted faceted histograms and catplots. Having run into a lot of deprecations, I could see the matplotlib team constantly updating the seaborn package. Then it was time to practice seaborn with the data of marathon athletes and although extremely difficult, it gave a great insight on how to use timeseries in matplotlib and also add new columns of data for drawing accurate conclusion. With the illustration it was possible for me to plot violin plots and linear regression plots. Having conquered Seaborn, I have completed the basics of matplotlib and was ready for the next topic, the prime concept of data science: Machine learning**

**DAY 23:**

**I carefully followed the major theory of machine learning which involved learning sophisticated definitions of machine learning and classifications. I understood the two broad categories of supervised and unsupervised machine learning and also so sub categories of the above two classifications. I also explored some simple machine learning models for classification models, regression models, clustering models and dimensionality reduction models.**

**Scikit was an amazing package in python and in immediately started working on implementing all the model types I learnt in theory the previous session. I learnt how to use classification using Gaussian naïve byes classification, linear regression in supervised learning. With the iris dataset, it was possible for me to try out dimensionality reduction techniques using manifold and also do clustering of data. I learnt what was a feature matrix, target array, accuracy score, training dataset, testing dataset and confusion matrix. Lastly, I practiced all of the concepts by analyzing handwritten digits dataset in scikit learn.**