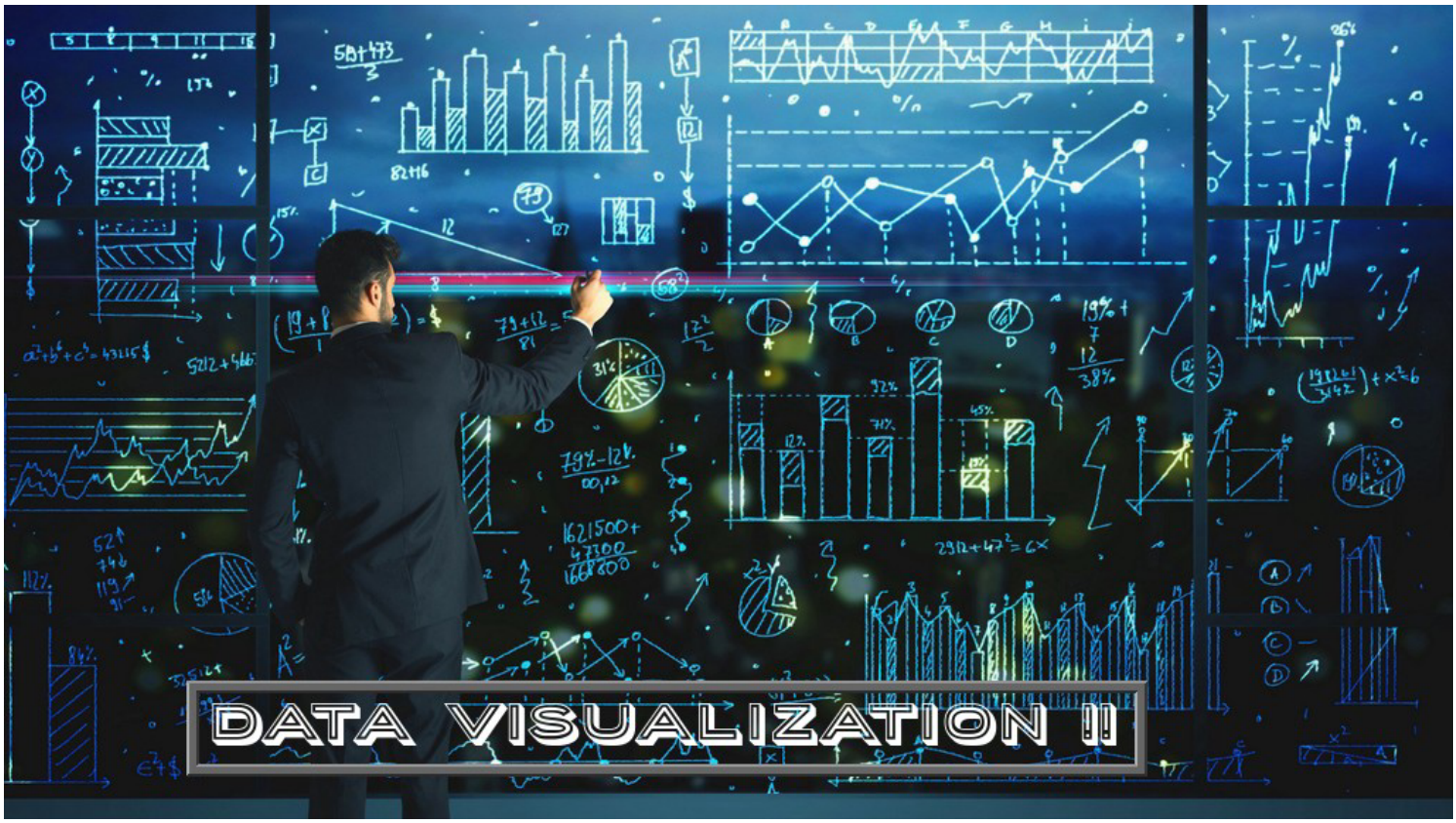
Data Visualization using Python for Machine Learning and Data science :

Sanat Follow
Dec 15, 2018 · 21 min read



Good day Folks,

> *Myself is **Sanat** currently working in IBM Chennai and I'm an aspiring Machine learning engineer/Data Scientist who loves to make my hand dirty on playing with a huge amount of data. Let me cut short my boring intro here :P and move towards the topic of the day **"Data Visualization".***

Note : This blog was made keeping in mind that the person reading is just aware of very basics on data visualizations and some topics may be too naive. So please skip the topics that you knew already and enjoy the rest. I am open to all your comments regarding the blog.

As we all knew that there is a huge buzz going over the term **data**, like Big data, Data science, Data Analysts, Data Warehouse,Data mining etc. which emphasize that, In the current era data plays a major role in influencing day to day activities of the mankind. Everyday we are generating more than 2.5 quintillion(10^{18}) bytes of data(Link) ranging from our Text messages, Images, emails, till data from autonomous cars, IOT devices etc. With such huge amount of data being available on hand, leveraging useful information from this data can help each and every organization very much, for getting a clear insight on several areas like, what can bring a boost for their organization's revenue, which field needs more focus, how to seek more customer's attention etc. Machine learning(ML), Data science are some of the interrelated areas of Artificial Intelligence(AI) where this task of learning from data is done in a huge extent on these recent days.

From above intro, now you may be fascinated with the term "**Data**" and will be planning to make use of these available data much effectively, to get as much useful information as possible. Do you think it is very easy to interpret these collected data? **May be No :(** . *The data will be in raw format and there are several steps that needs to be performed to convert this raw data to an useful information, just like how a goldsmith prepares an ornament by doing several preprocessing steps on raw gold*. This is where a Data scientist comes in handy, You throw him/her a raw data, they can narrate you a great story from that, which would not be as easy for others when provided with same data. There will be several stages when those data scientists write their story (I mean working with the data :P) like Data acquisition, Data cleaning, Data Visualization, building a model that can be used to predict the future information etc. Among them one key stage is **Data Visualization**. This is the first and foremost step where they will get a high level statistical overview on how the data is. and some of its attributes like the underlying distribution, presence of outliers, and several more useful features.

> *"Data is the new oil? No, data is the new soil." — David McCandless*

. . .

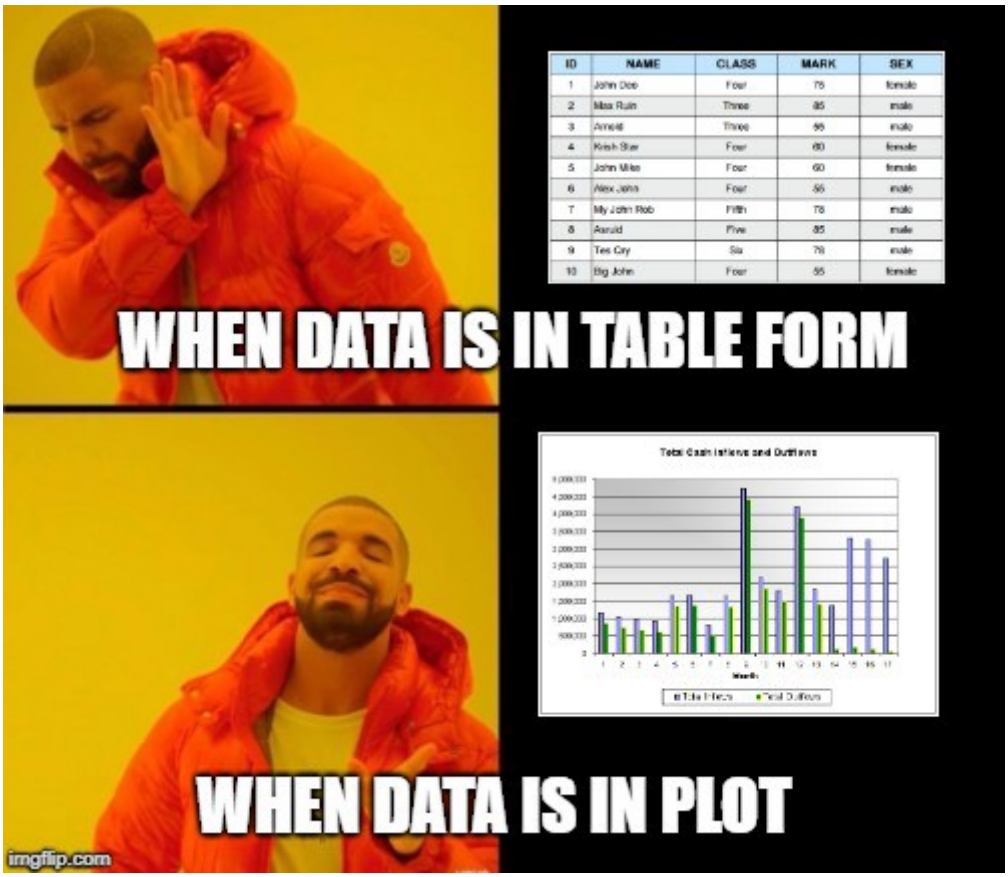
Why Visualization?

Do you think giving you the data of lets say 1 Million points in a table/Database file and asking you to provide your inferences by just seeing the data on that table is feasible? Unless you're a super human its not possible. This is when we make use of Data visualization, wherein all the data will be transformed into some form of plots and

analyzed further from that. As being a human, we are more used to grasp a lot of info from diagrammatic representation than the counterparts.

> *Kids should learn how to read and create graphics from a very young age. Most kids are natural-born artists, as* ***we humans are a visual species****. We should take advantage of that, as graphics and illustrations can be such a powerful weapon for understanding and communicating* — Prof. **Alberto Cairo**,

Okay ! So since it is said that we need to convert the data from a boring table into an interesting pictorial form like a scatter plot or Bar chart, You may wonder, How can I do it? Do I need to write my own code for that ? **NO!** Actually we can make use of very good packages from some popular programming languages which are readily available and can make the work pretty much simple with just a single line of code. That's the power of modern programming :D !



Meme created using imgflip.com

As a human, we can just visualize anything in either in 2-d or 3-d. But trust me almost of the data that you obtain in real world won't be this way. As a Machine learning engineer,

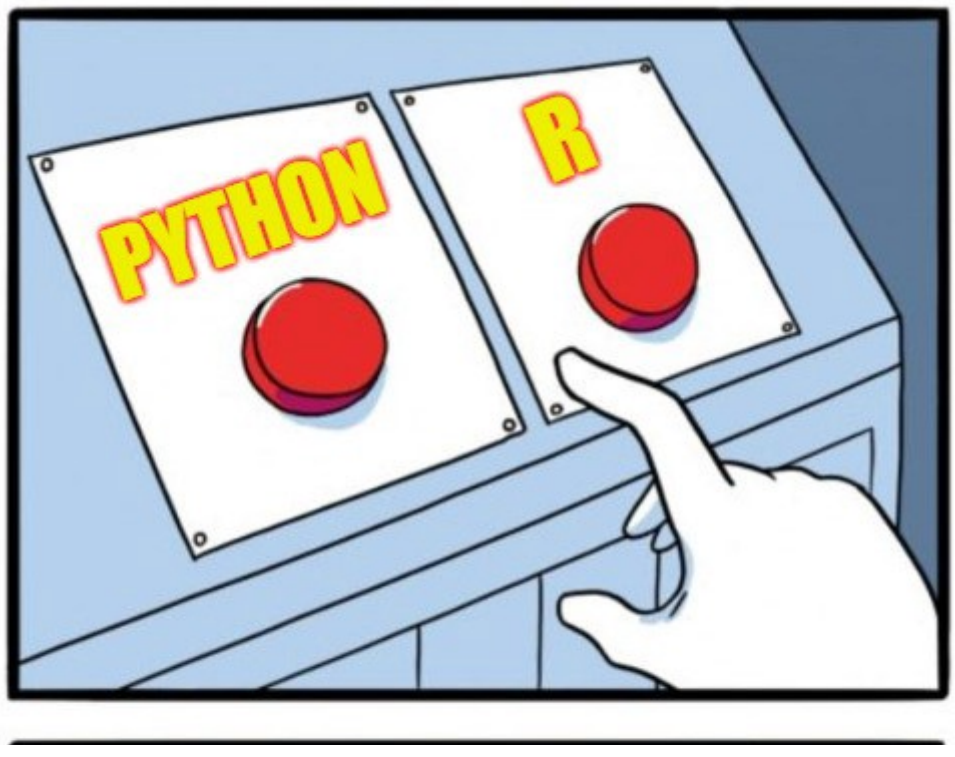
working with more than 1000-dimensional data is very common. So what can we do in such cases where data is more than 3D ? *There are some **Dimensionality Reduction(DR)** techniques like PCA, TSNE , LDA etc which helps you to convert data from a higher dimension to a 2D or 3D data in order to visualize them.* There may be some loss of information with each DR techniques, but only they can help us visualize very high dimensional data on a 2d plot. TSNE is one of the state of the art DR technique employed for visualization of high dimensional data.

From perspective of building models , By visualizing the data we can find the hidden patterns, Explore if there are any clusters within data and we can find if they are linearly separable/too much overlapped etc. From this initial analysis we can easily rule out the models that won't be suitable for such a data and we will implement only the models that are suitable, without wasting our valuable time and the computational resources.

This part of data visualization is a predominant one in initial **Exploratory Data Analysis (EDA)** on the field of Data science/ML.

. . .

Which language can I use?



Meme created using imgflip.com

There may be several languages on which we can perform the Data Visualization, but the ones that are much widely used in the field of Data Science are Python & R. So your next question may be, ***Which one to learn and which one has a better scope ?. The answer is simple! It's purely your choice*** **;)** . 'R' is way more statistical language and has several great packages for Data science applications, whereas Python on the other hand is widely used in general purpose programming as well as for Data science and ML related applications. I am pretty comfortable with python and so I will continue the rest of the blog with python codes and also it has several good packages like Scikit,Matplotlib,seaborn etc which helps us a lot and a special thanks to those developers who made our work simple.

. . .

Plotting data with Python :

As mentioned above, Python has several good packages to plot the data and among them **Matplotlib** is the most useful one. **Seaborn** is also a great package which offers a lot more appealing plot and even it uses matplotlib as its base layer. There are also many similar type of plots available in Pandas when the entire data is stored in a pandas

dataframe. In this blog we are going to discuss about the different types of plots that are under this 2 special packages and let us explore them in detail.

. . .

Matplotlib Architecture:

There are overall 3 different layers in the architecture of matplotlib as follows.

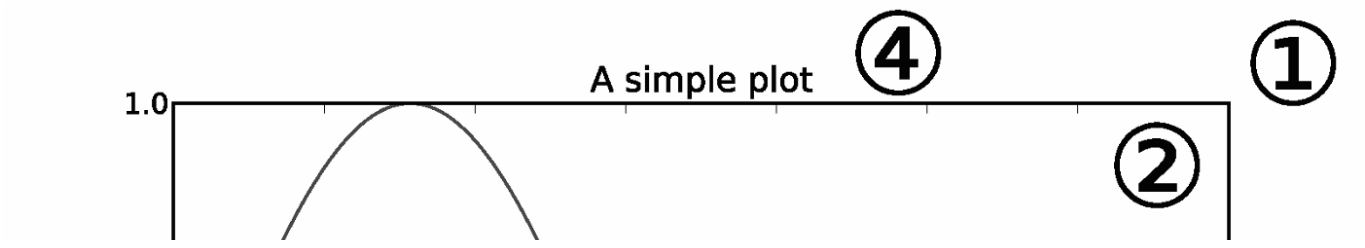
1.) Backend layer.
2.) Artist layer.
3.) Scripting layer.

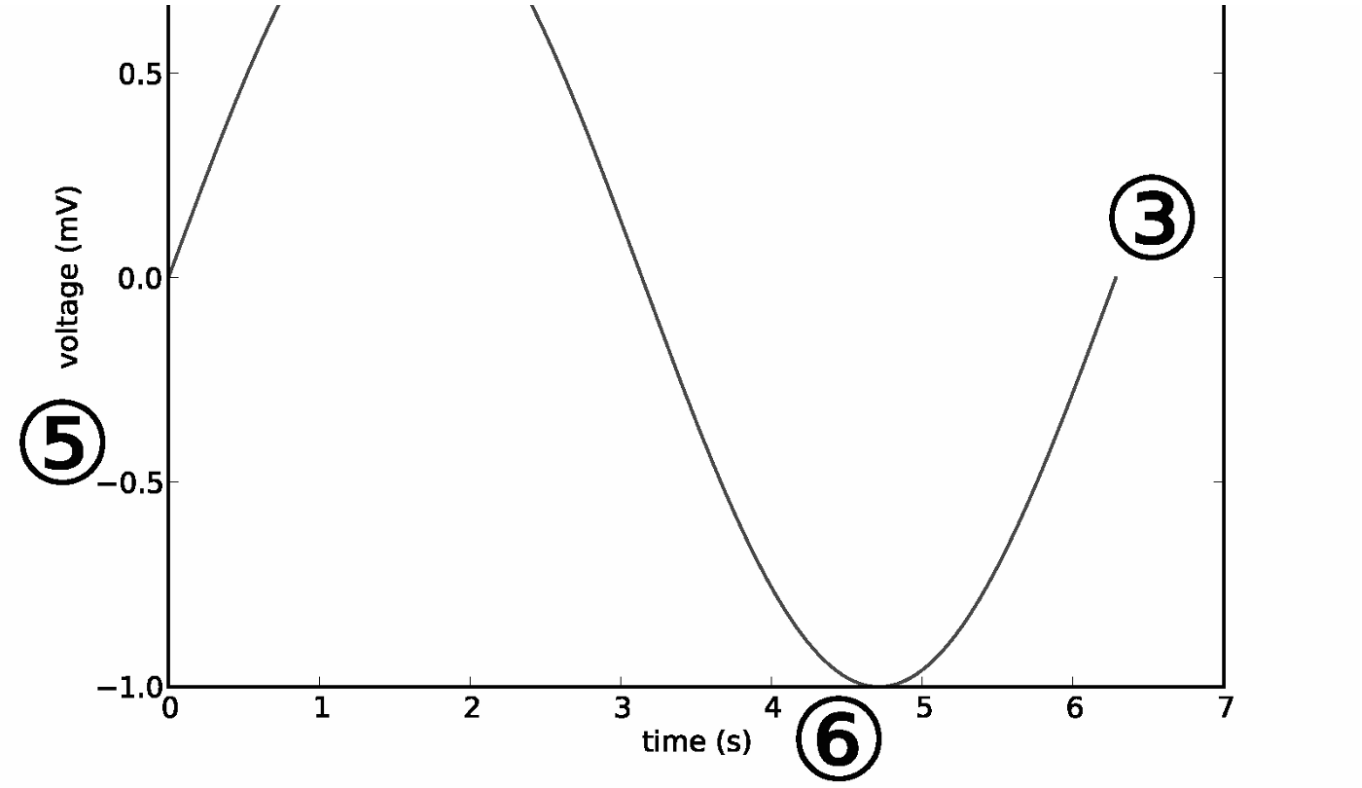
- **Backend layer:**

This is the bottom most layer of a figure which contains implementation of several functions that are required for plotting. There are 3 main classes from the backend layer **FigureCanvas** (the layer/Surface on which the figure will be drawn), **Renderer** (the classn that takes care of the drawing on the surface) and **Event** (to handle the mouse and keyboard events). We don't work much with the Backend layer as compared to the counterparts.

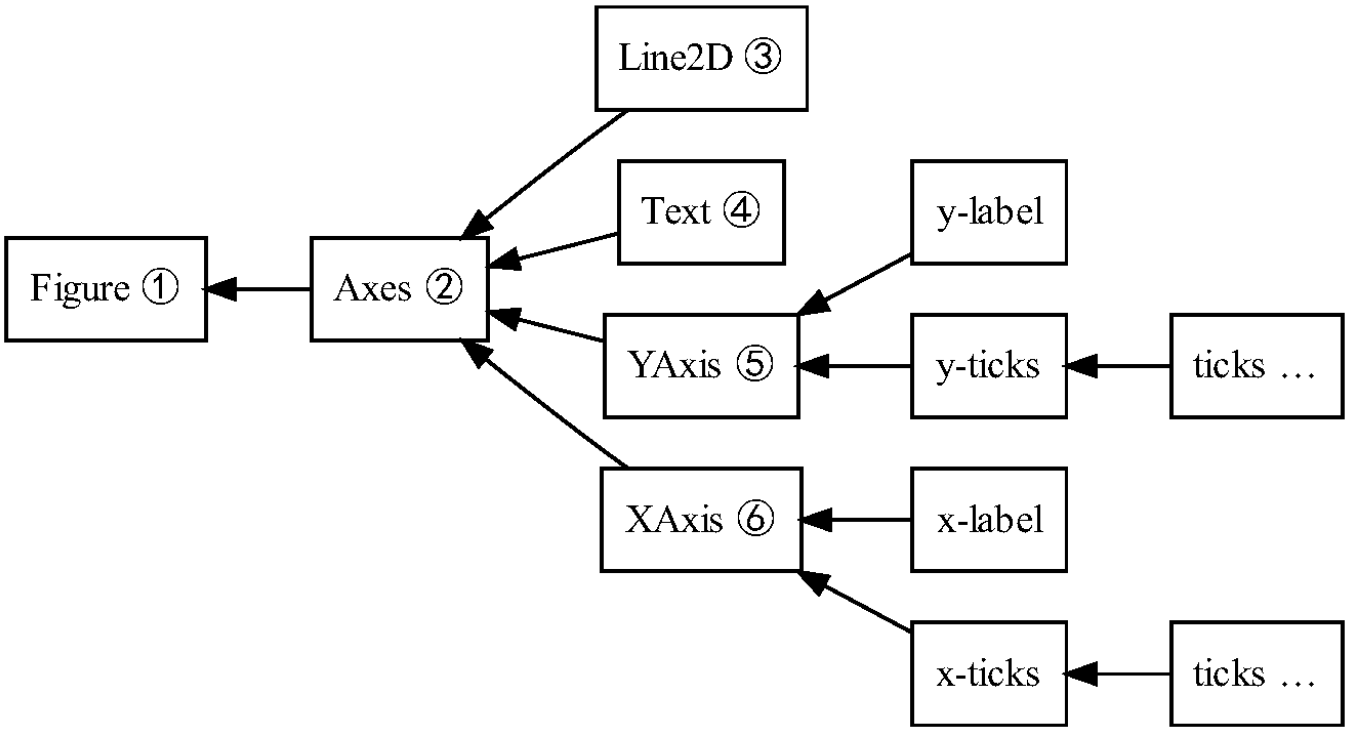
- **Artist Layer:**

This is the second/middle most layer in the architecture. It is what does most of the duty on plotting the various functions, like axis which coordinates on how to use the renderer on the Figure canvas. ***To put it simple, lets consider Paper as the Figure Canvas and Sketch pen as renderer. Then the hand of the painter is the Artist layer*** which has certain functions, knows how to sketch to get the exact figure. There are several classes available on artist layer and a few important ones are Figure, Axes and Axis.





Machine learning/Data Science



The 2 images above explain the hierarchy between various classes in the artist layer. Figure is the topmost one and a figure can contain multiple number of axes upon which the plot is done. Moving on, under each axes we can add multiple plots. This provides a

lot more additional functionality to improve the plots and this is the place where most of the heavy lifting works takes place.

- **Scripting layer:**

This is the topmost layer on which majority of our codes will play around. For day to day exploratory works, we almost rely on this scripting layer of matplotlib. Pyplot is the scripting layer that provides almost simlar functionality as that of Matlab in python. The methods in scripting layer, almost automatically takes care of the other layers and all we need to care about is the current state(figure & Subplot). Hence it is also called as ***stateful interface***.

Please visit the link for reference.

. . .

Matplotlib Terms:

Here lets have a short glance on some of the commonly used terms in data visualization using Matplotlib.

> *P.S: Wherever plt.(Some function) is used, it means that I imported matplotlib.pyplot as plt in my program and sns means that i imported seaborn just for the ease of coding.*

```
import matplotlib.pyplot as plt
import seaborn as sns
```

If you don't have the packages already installed in your system. Please install Python 3 and use the below code on your command prompt.

```
pip3 install matplotlib
pip3 install seaborn
```

Axes:

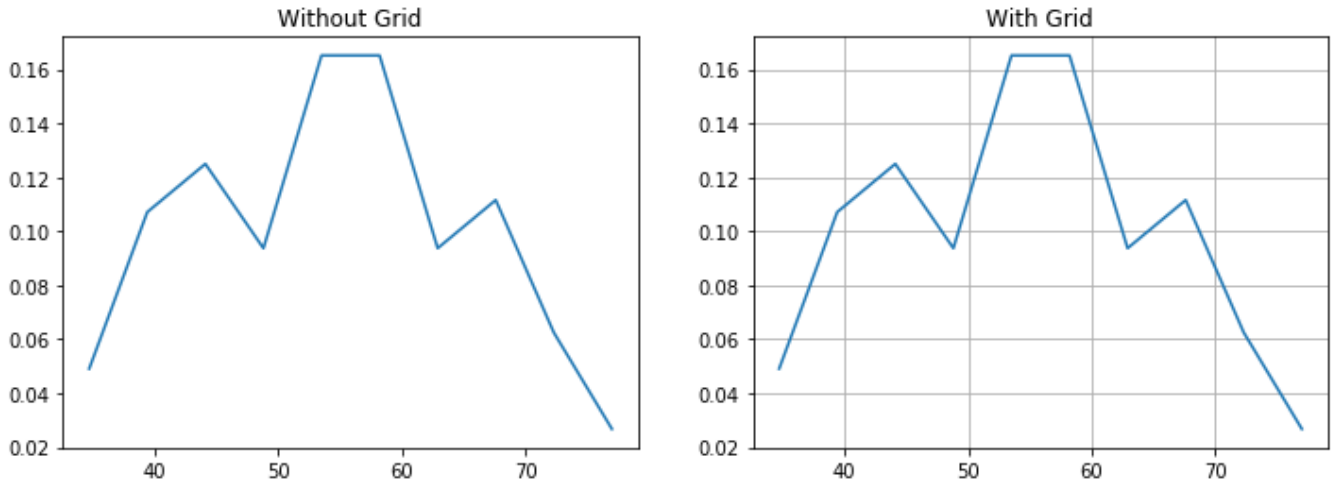
Axes is the entire area of a single plot in the figure. It is the class that contains several attributes needed to draw a plot such as like adding title, giving labels, selecting bin values for different types of plots on each axes etc.

We can have multiple axes in a single plot, by which we can combine multiple plots into a single figure. For e.g: If we want both PDF and CDF curves in a same figure we can create 2 axes and draw each of them in different axes. Then we can combine them into a single figure.

Link

Grid:

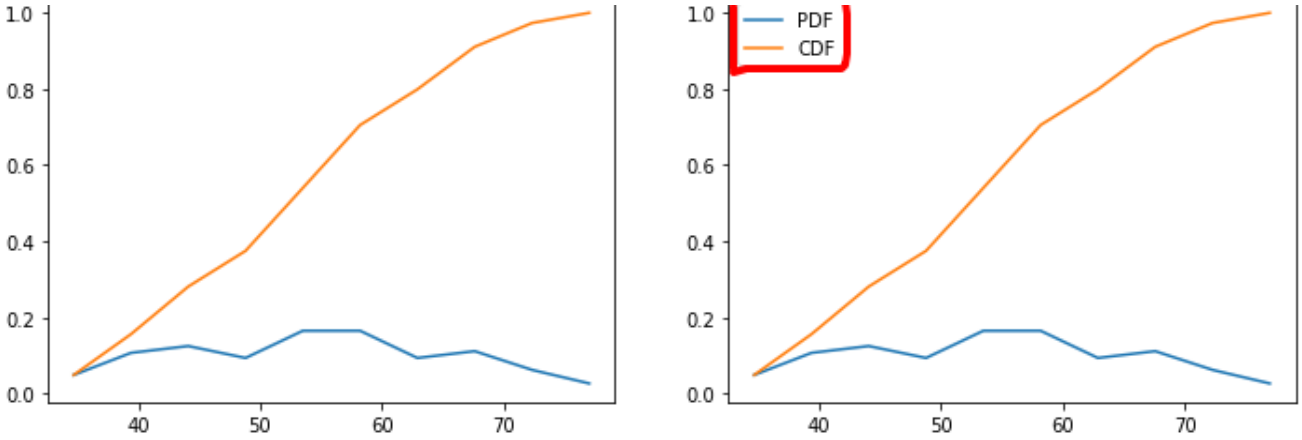
When grid is enabled in a plot, a set of horizontal and vertical lines will be added at the background layer of the plot.This can be enabled using plt.grid(). It can be useful for a rough estimation of value at a particular coordinate, just by looking at the plot.



Ref: Link

Legend:

Legends are nothing but labeled representation of the different plots available in a figure. i.e when there are multiple plots in a single image(e.g : Iris dataset), Legends will help us to identify the correct name of the different colored plots in a single figure.



Ref: Link

Subplots:

When we want 2 or more plots in a single image, we can make use of Subplot in Matplotlib, plt.subplot (xyz). xyz is a 3 digit integer where x-No of rows, y =No of columns, z= index number of that plot. This is one of the most useful feature when we need to compare two or more plots hand to hand instead of having them in separate images.

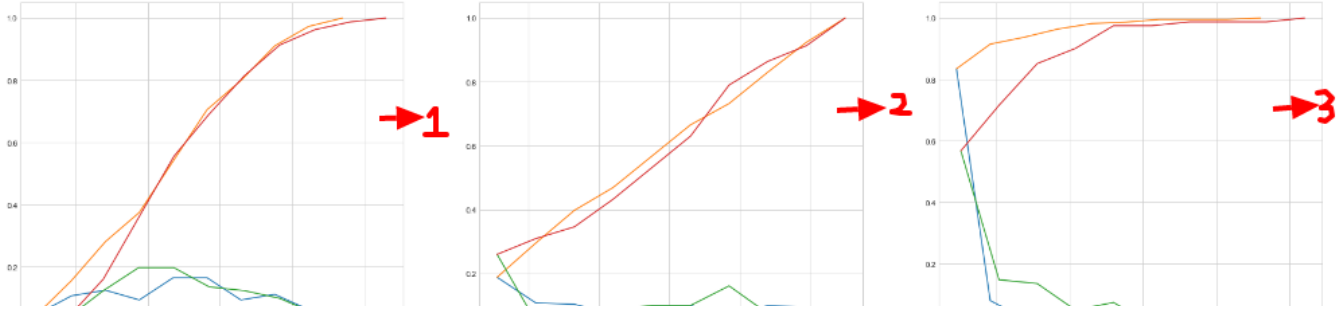
```
plt.figure(1,figsize=(30,8))

plt.subplot(131)
 #Code for fig1.

plt.subplot(132)
#code for fig2

plt.subplot(133)
#code for fig3.

plt.show()
```



Subplots

In addition to Subplot try using gridspec , which can help us split the plots in subplot more effectively and easier.

Ref: Link

Title:

We can set a title to a plot using plt.title()

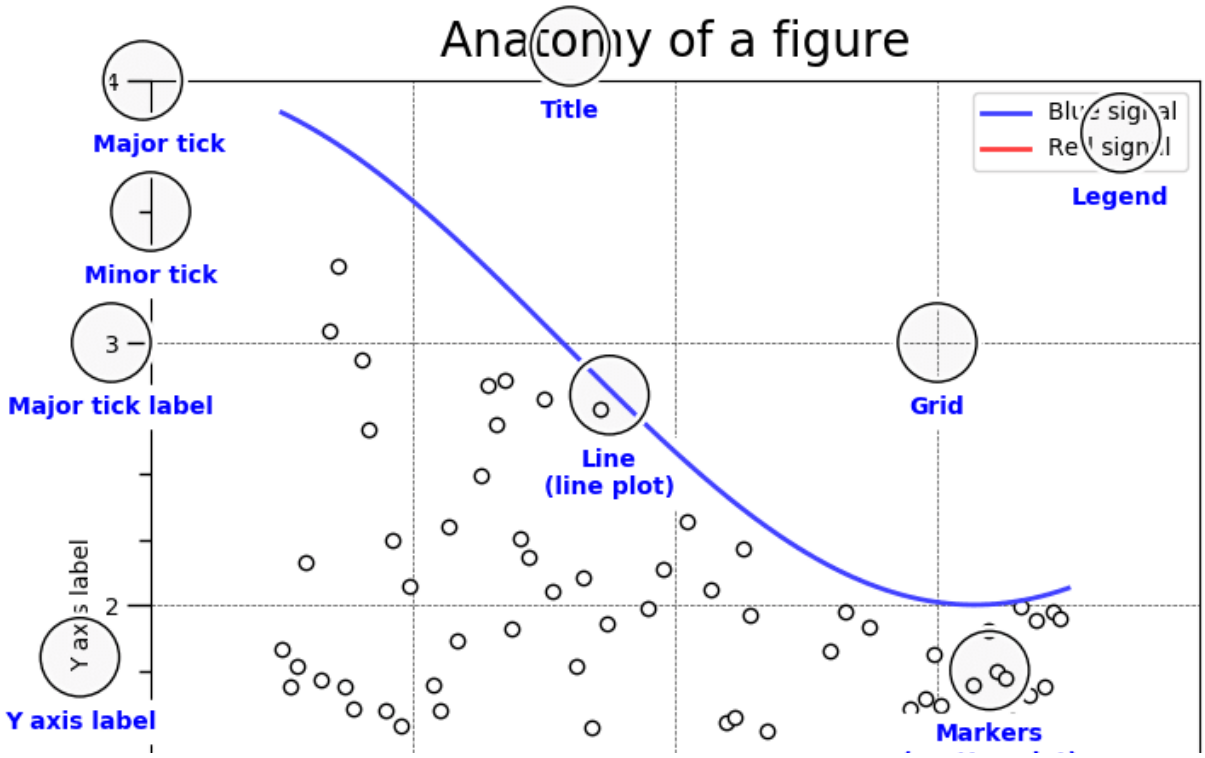
xlabel:

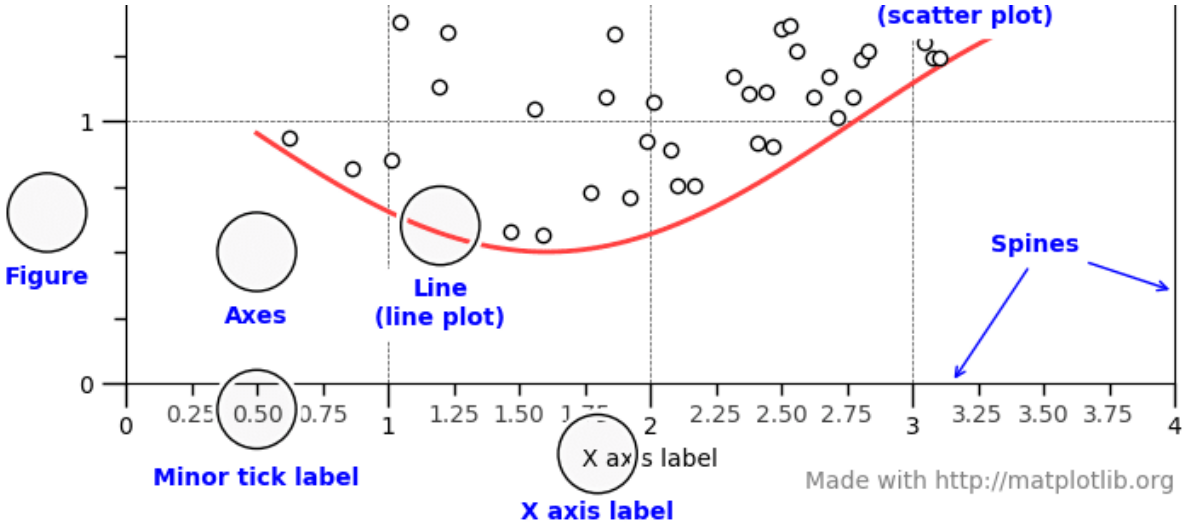
plt.xlabel() is the command to set the label for x axis

ylabel:

plt.ylabel() is the command to set the label for y axis

The below image very well explains each and every parts in visualizing data as a figure.





. . .

Different types of analysis:

There are different types of analysis as mentioned below.

1. **Univariate** : In univariate analysis we will be using a single feature to analyze almost of its properties.
2. **Bivariate** : When we compare the data between exactly 2 features then its called bivariate analysis.
3. **Multivariate**: Comparing more than 2 variables is called as Multivariate analysis.

In the below blog, for each plot I will mark them as (U),(B) & (M) to represent them as Univariate, Bivariate and Multivariate plots correspondingly.

. . .

Plots discussed:

The below are the list of plots that I am going to explain in the subsequent topics.

i) Scatter plot (B)

ii) Pair plot (M)

iii) Box plot (U)

iv) Violin plot(U)

v) Distribution plot (U)

vi) Joint plot(U) & (B)

vii) Bar chart (B)

viii) Line plot(B)

. . .

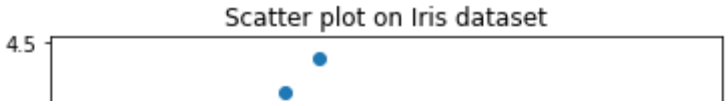
i. Scatter plot:

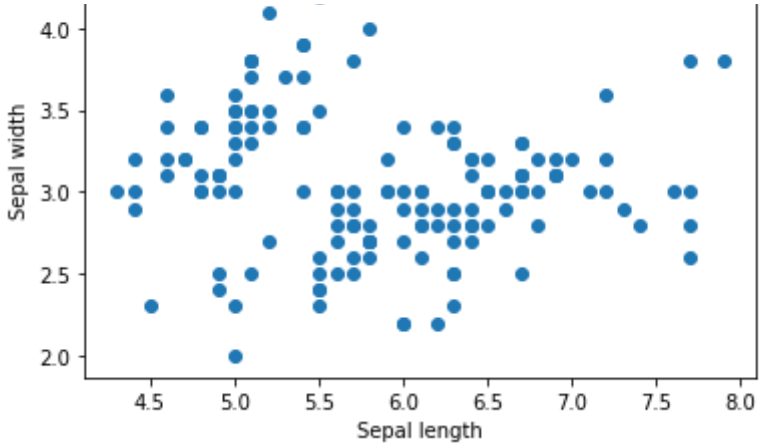
As far as Machine learning/Data Science is concerned, one of the most commonly used plot for simple data visualization is scatter plots. This plot gives us a representation of where each points in the entire dataset are present with respect to any 2/3 features(Columns). Scatter plots are available in 2D as well as 3D . The 2D scatter plot is the important/common one, where we will primarily find patterns/Clusters and separability of the data. The code snippet for using a scatter plot is as shown below.

```
plt.scatter(x,y)
```

When we use scatter from Matplotlib directly we will get a plot similar to the one below. I used Iris dataset to explain simple scatter plot.

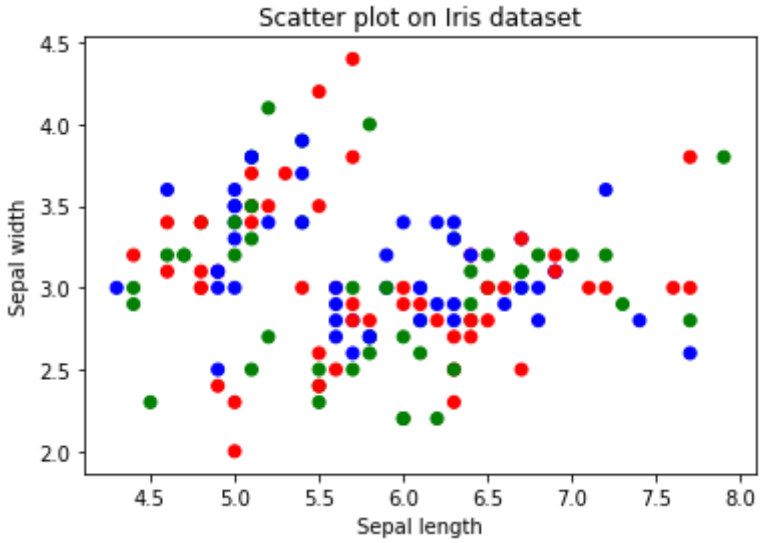
```
plt.scatter(iris['sepal_length'],iris['sepal_width'])
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('Scatter plot on Iris dataset')
```





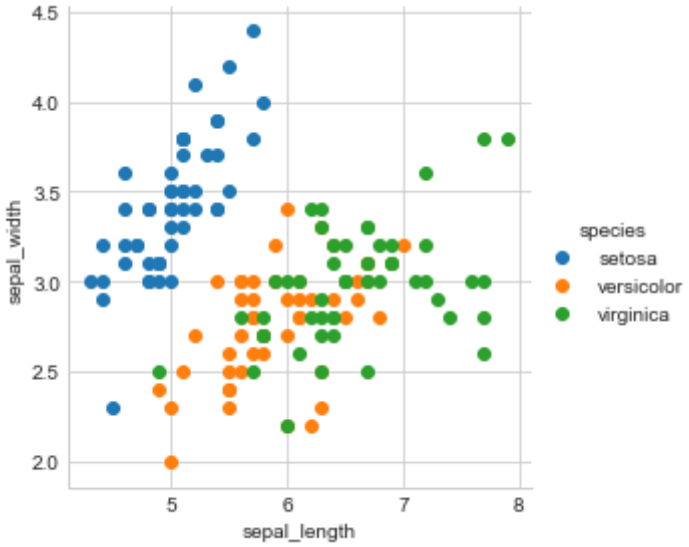
Here we can see that all the points are marked on their corresponding position with respective to their values of x and y. Lets tweak around to see if we can get points with different colours.

```
plt.scatter(iris['sepal_length'],iris['sepal_width'],color=
['r','b','g'])
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('Scatter plot on Iris dataset')
```



This one seems fine! but the colors are assigned to points on basis of how they were present in dataset. Now how it would be if we can color the points as per their class label. Here the class labels are Setosa, Virginica and Veriscolor. This is where Seaborn pops up, with its effective visualization stuffs.

```
sns.set_style("whitegrid")
sns.FacetGrid(iris, hue="species", size=4) \
   .map(plt.scatter, "sepal_length", "sepal_width") \
   .add_legend()
plt.show()
```



Hurray!! The plot seems a lot cooler now, just simply by visualizing it, we can conclude like setosa flowers are well separated from the other 2 classes and also there are some overlaps between virginica and versicolor. The parameter **hue**, in the facetGrid decides the color of each datapoints. Here we used Species column(dependent feature/Class) in hue, so that we got this plot colored in this manner.

This makes seaborn a bit more superior than Matplotlib when it comes to visualization. But not to forget that still we are using plt.scatter from Matplotlib, on the map function in seaborn. So seaborn is just making the visual more appealing. For 3d scatter plots, we can use plot.ly to achieve that. But there is a hack that we can try like plotting points between 2 variables and setting the size of points with respective to the third variable where we did analysis of 3 features.

Documentation link of Scatterplot.

Datset used : Iris dataset

. . .

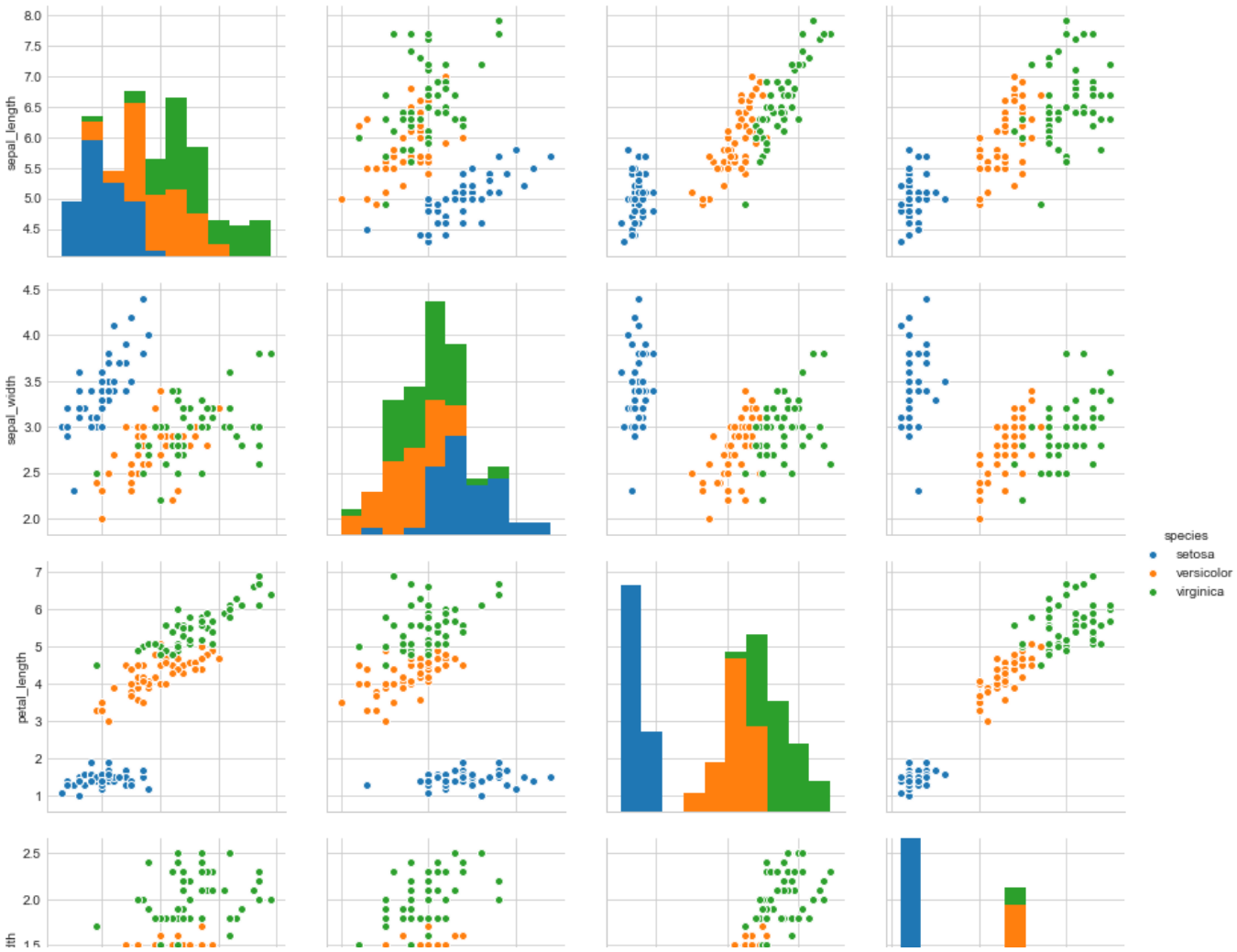
ii. Pair plots:

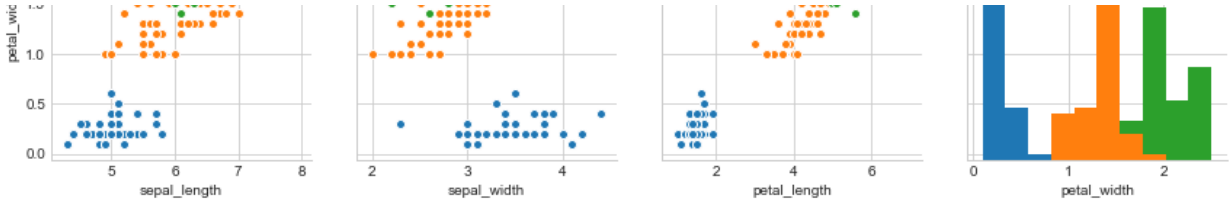
We can use scatter plots for 2d with Matplotlib and even for 3D, we can use it from plot.ly. What to do when we have 4d or more than that? This is when Pair plot from seaborn package comes into play.

Lets say we have n number of features in a data, Pair plot will create us a (n x n) figure where the diagonal plots will be histogram plot of the feature corresponding to that row and rest of the plots are the combination of feature from each row in y axis and feature from each column in x axis.

The code snippet for pair plot implemented for Iris dataset is provided below.

```
sns.set_style("whitegrid");
sns.pairplot(iris, hue="species", size=3);
plt.show()
```





By getting a high level overview of plots from pair plot, we can see which two features can well explain/separate the data and then we can use scatter plot between those 2 features to explore further. From the above plot we can conclude like, Petal length and petal width are the 2 features which can separate the data very well.

Since we will be getting n x n plots for n features, pairplot may become complex when we have more number of feature say like 10 or so on. So in such cases, the best bet will be using a dimensionality reduction technique to map data into 2d plane and visualizing it using a 2d scatter plot.

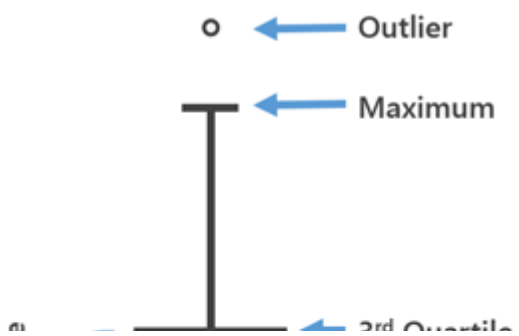
Documentation link of Pairplots.

Datset used : Iris dataset

. . .

iii. Box plots:

This is the type of plot that can be used to obtain more of the statistical details about the data. The straight lines at the maximum and minimum are also called as **whiskers**. Points outside of whiskers will be inferred as an outliers. The box plot gives us a representation of 25th, 50th ,75th quartiles.From box plot we can also see the Interquartile range(IQR) where maximum details of the data will be present. It also gives us a clear overview of outlier points in the data.



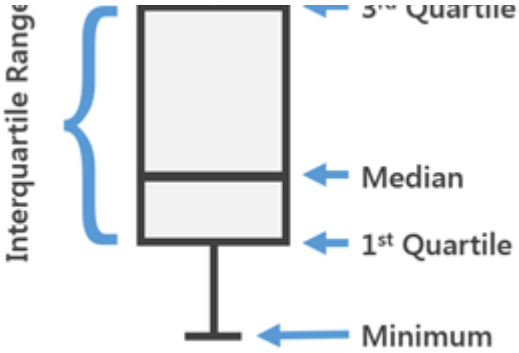
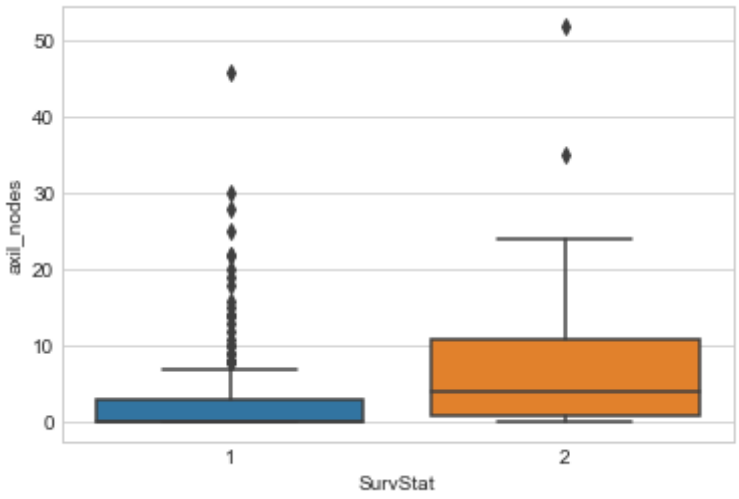


Image taken from link

Boxplot is available in seaborn library. Lets jump on to the code part. Here x is the variable to be predicted and y is the independent feature. These **box plots comes under univariate analysis**, which means that we are exploring data only with one variable i.e here we are just checking influence of feature axil_nodes on the class Survival status and not between any two independent features.

```
sns.boxplot(x='SurvStat',y='axil_nodes',data=hb)
```



Using the box plot, as mentioned above we can see how much of data is present in 1st quartile and how much points are outliers etc. From the above plot for the **class 1** we can see that there are very few/no data is present between median and the 1st quartile. Also there are more number of outlier points for **class 1** in feature **axil_nodes**. Such details about outliers and so on will help us to well prepare the data before sending it to a model, as outlier impacts a lot of Machine learning models.

Documentation link for box plots.

Dataset used: Haberman Dataset.

. . .

iv. Violin plots:

The violin plots can be inferred as a combination of Box plot at the middle and distribution plots(Kernel Density Estimation) on both side of the data. This can give us the details of distribution like whether the distribution is mutimodal, Skewness etc. It also give us the useful info like 95% confidence interval. The below image help us grasp some important parts from a violin plot.

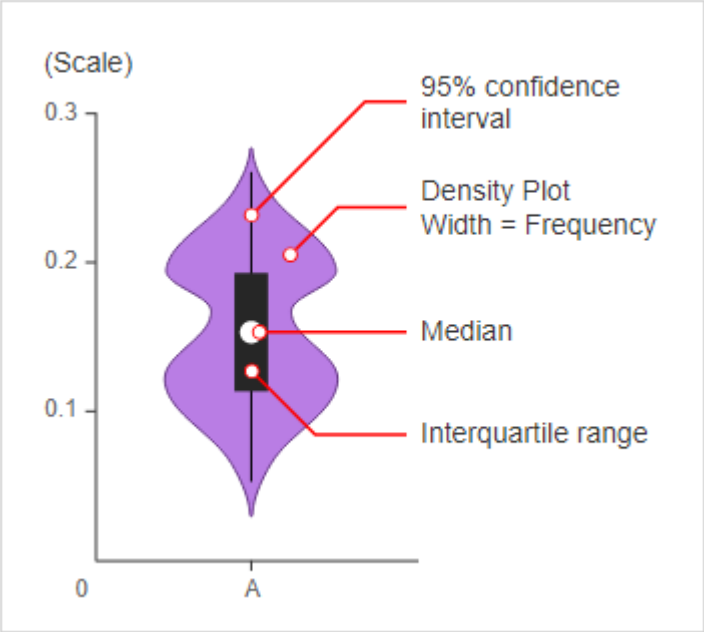
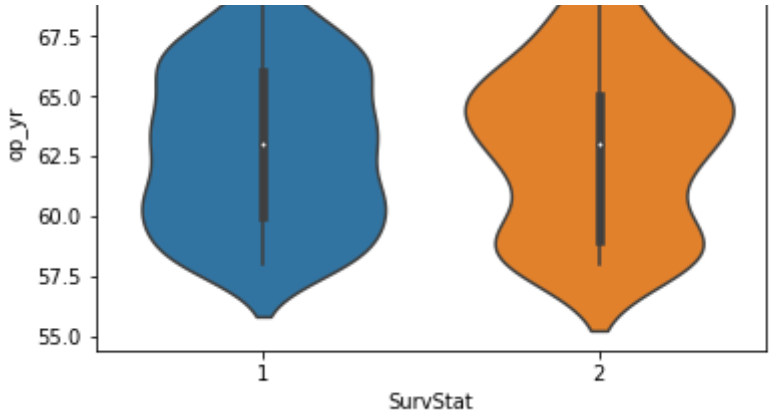


Image taken from Link

Violin plot is also from seaborn package. The code is simple and as follows.

```
sns.violinplot(x='SurvStat',y='op_yr',data=hb,size=6)
```



From the above violin plot we can infer that median of both the classes are around 63 and also the maximum number of persons with class 2 has op_yr value as 65 whereas for persons in class1, the maximum value is around 60.The 3rd quartile to median has lesser points than the median to 1st quartile and so on.

Documentation link for violin plot.

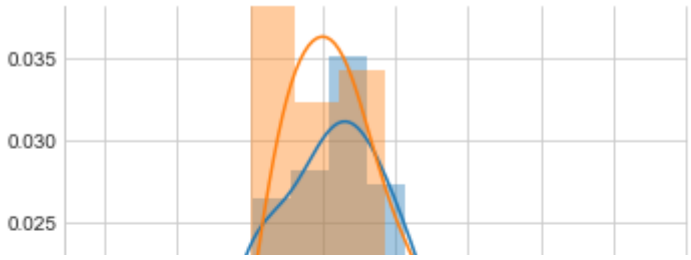
Dataset used: Haberman Dataset.

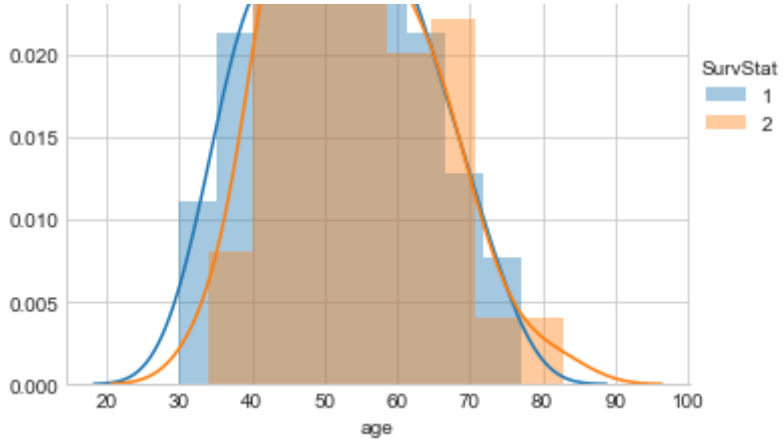
. . .

v. Distribution plot:

This is one of the best univariate plot to know about the distribution of data. When analyzing effect on dependent variable(output) with respective to a single feature(input), we use distribution plots a lot. It is also readily available in seaborn package. This plot gives us a combination of pdf and histogram in a single figure.

```
sns.FacetGrid(hb,hue='SurvStat',size=5).map(sns.distplot,'age').add_
legend()
```





From the above plot, we can see that we created a distribution plot on the feature 'Age'(input feature) and we used different colors for the Survival status(dependent variable/output) as it is the class to be predicted and we can see there is a huge overlap between their PDFs. The sharp block like structures are histograms and the smoothened curve is called Probability density function(PDF). The pdf of a curve can help us to identify the underlying distribution of that feature which is one major takeaway from Data visualization/EDA.

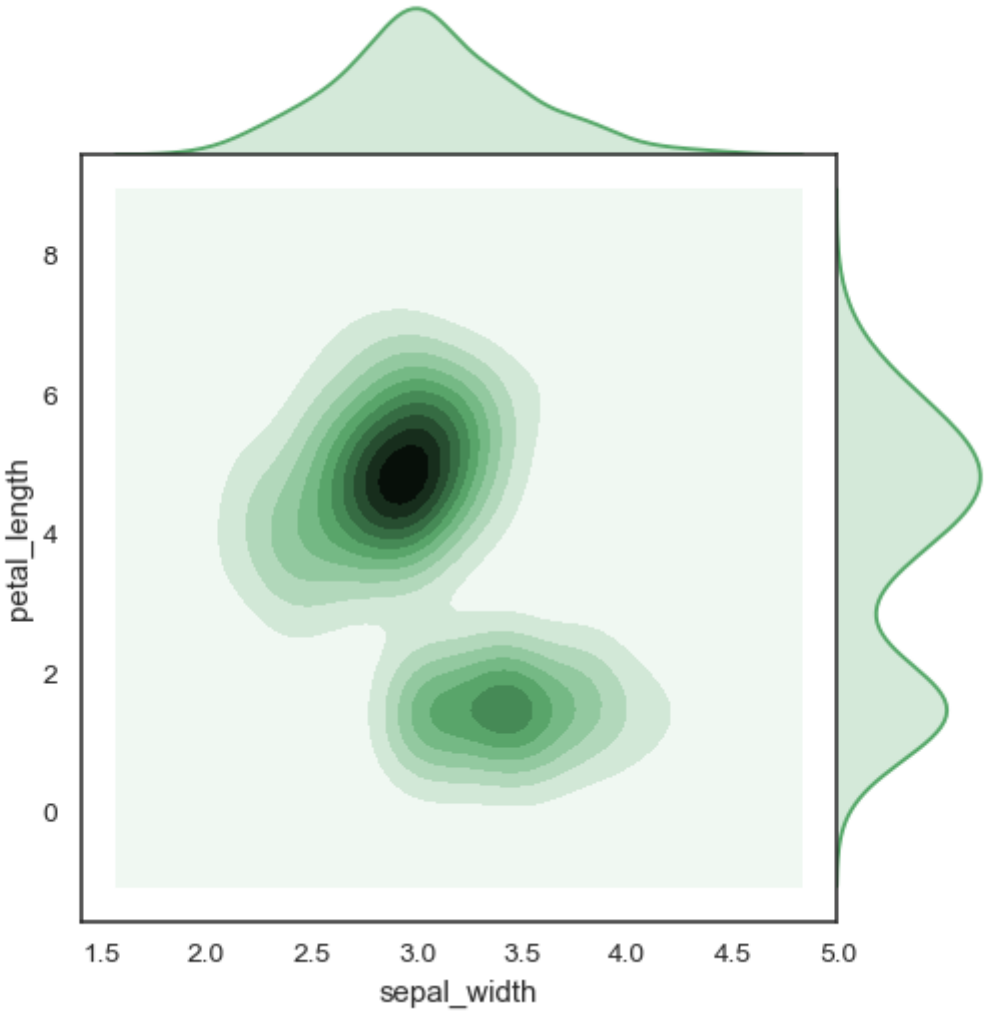
Documentation link for distplot.

Dataset used: Haberman Dataset.

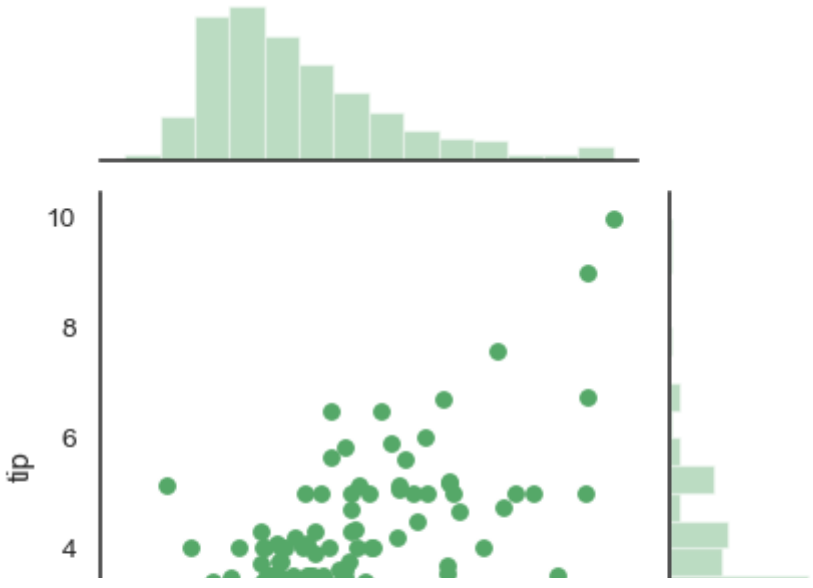
. . .

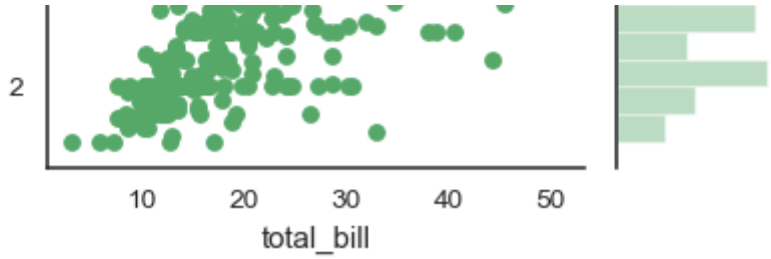
vi. Joint plot :

It's one of my favourite and the great thing about joint plots is, in a single figure we can do both univariate as well as bivariate analysis. The main plot will give us a bivariate analysis, whereas on the top and right side we will get univariate plots of both the variables that were considered. There are variety of option you can choose from, which can be tuned using **kind** parameter in seaborn's jointplot function. The one shown below is of kind as KDE(Kernel Density Estimation) and it is represented in a contour diagram, All points with same boundary will have the same value and the colour at a point depends on number of datapoints i.e it will be light color when very few points had that value and will be darker with more points. This is why at center it is darker and will be pale/lighter at the ends for this dataset.



The 2 most important plots we use will be scatter plot for bivariate and distribution plot for univariate and since we are getting both in a single plot as shown below, it will make our work a lot easier.





Documentation Link

Datset used : Iris dataset

. . .

vii. Bar chart:

This is one of the widely used plot, that we would have saw multiple times not just in data analysis, but wherever there is a trend analysis in many fields. Though it may seem simple it is powerful in analyzing data like sales figure every week, revenue from a product, Number of visitors to a site on each day of a week etc.

The code is pretty straight forward, We will be using bar function from Matplotlib to achieve it. The below code will give you a bar plot, where the position of bars are mentioned by values in x and length/height of the bar as values in y.

```
plt.bar(x,y)
```

I will show you a chart that I created as a part of my current work, where as a lead member in projecting retrospective details to my management, I need to specify the alert trend analysis which was done earlier with a simple tabular data. Then by using bar plots for the same was more presentable and easily interpretable by my audience.

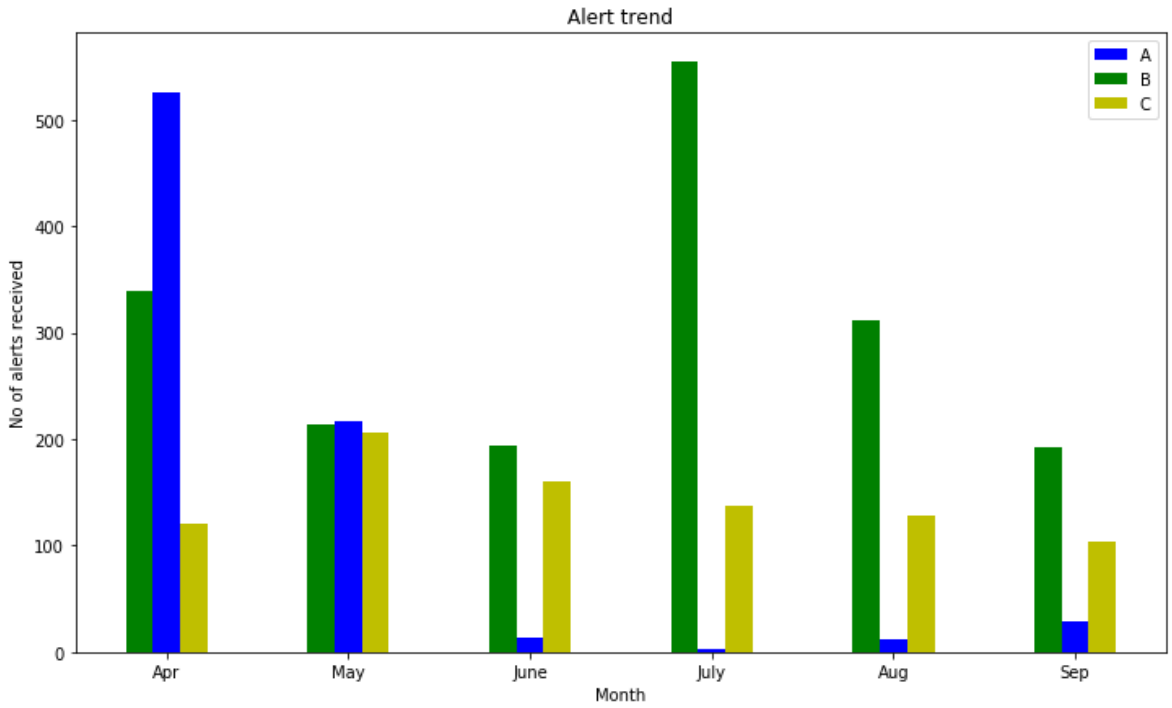
```
a=np.arange(6)
w=0.15

fig,ax=plt.subplots(figsize=(12,7),edgecolor='k')
p1=ax.bar(a,d,w,color='b')
```

```
p2=ax.bar(a-w,c,w,color='g')
p3=ax.bar(a+w,e,w,color='y')
ax.set_xticks(a)
ax.set_xticklabels(('Apr','May','June','July','Aug','Sep'))
ax.set_title('Alert trend')
ax.legend((p1[0],p2[0],p3[0]),('A','B','C'))
plt.xlabel('Month')
plt.ylabel('No of alerts received')

#plt.grid()

plt.show()
```



So this way, We can view the data in a cool plot and can convey the details straight forward to others. This plot may be simple and clear but its not much frequently used in Data science applications. I will provide the documentation of Bar plot below, please play around with the several parameters to get plot of your desire.

Documentation Link for Bar plot.

. . .

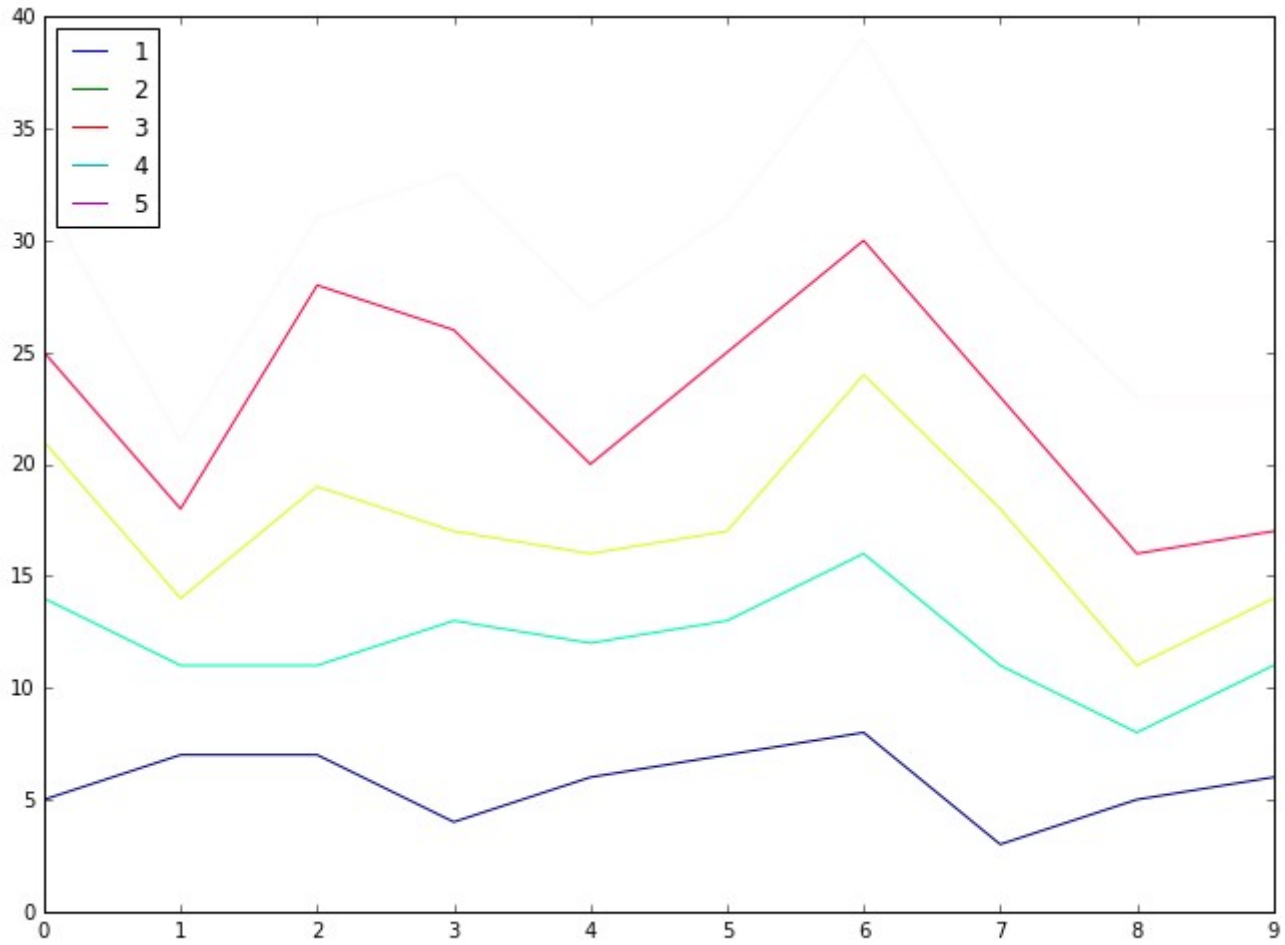
viii. Line plots:

This is the plot that you can see in nook and corners of any sort of analysis between 2 variables. The line plots are nothing but the values on series of datapoints will be connected with straight lines. The plot may seem very simple but it has more number of applications not only in machine learning but in many other areas.

The code is pretty much simple and the plot function in matplotlib does the duty of line graphs pretty well.

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

We call plot multiple lines inside a single figure as shown below where you need to add mutliple plt.plot() commands with each line represening a different color parameter.



The line charts are used right from performing distribution comparision using QQplots till CV tuning using elbow method and analysing performance of a model using AUC

curve.

Documentation link

. . .

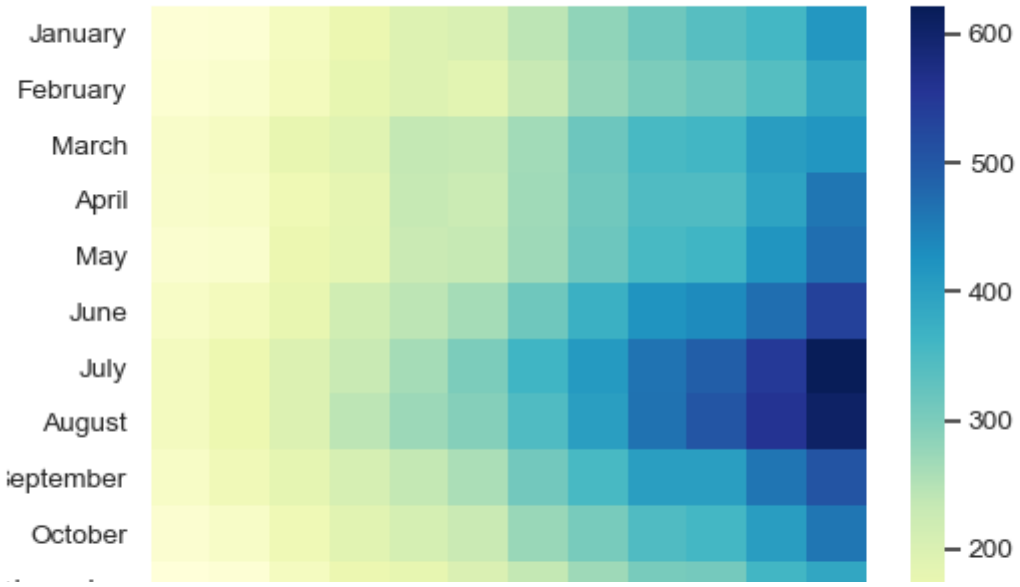
So far we saw some widely employed methods that are used to extract useful information/insight from the data.

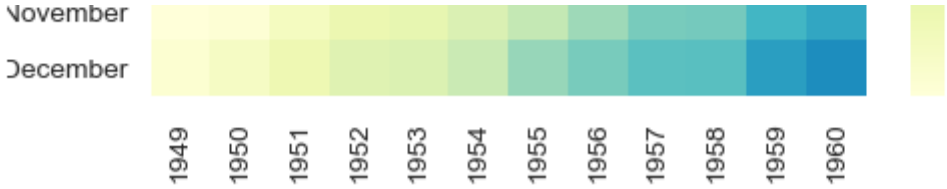
We can also use some visualization tools to convey our final information/inference in form of plots to the audience. Lets see a few commonly used ones among them.

. . .

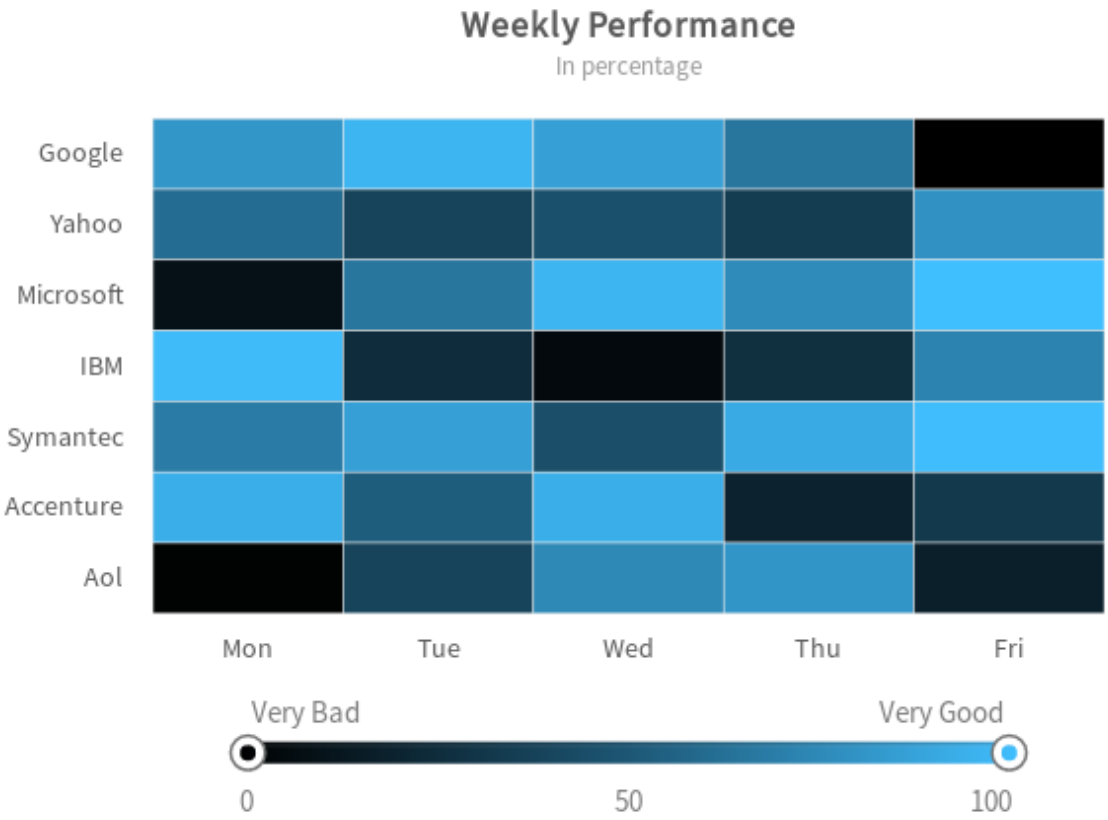
Heatmap:

Heatmap is one good visualization technique used to compare any 2 variables/features with respective to the values. The heatmap from seaborn library will create a grid like plot along with an optional color bar. We provide a 2D input matrix with certain values on each element to the heatmap and it exactly reproduces the output plot in the same shape as that of input matrix and each tile are colored based on the values provided in each elements of matrix to their corresponding tile.





This can be much useful in cases where there will be a bigger matrix and we want to find which value is higher, lower and so on by simply looking at the different colour tones used. For eg. Weekly performance chart of different companies can be plotted using heatmaps



In Machine learning applications, it can be used in representing Confusion matrix of a model, used in hyperparameter tuning to plot error values between 2 different hyperparameters etc.

seaborn documentation link

. . .

Wordcloud:

Wordcloud is nothing but creating an image that contains all the words in a passage/string with different size, as per the frequency of occurrence in that passage. The word will appear bigger and bolder if it occurs more number of times in a given passage.

Enter the below line in your command prompt to install wordcloud.

```
pip3 install wordcloud
```

This is recently used in many place, where by seeing the wordcloud we can conclude like what's discussed in a paragraph without reading it much. The most repeating word will appear bold and bigger than the least occuring ones.

It can be used a lot with text data analysis like, finding the common topics in a cluster while performing sentiment analysis and SEO optimizations etc.

Documentation link

. . .

Graphviz:

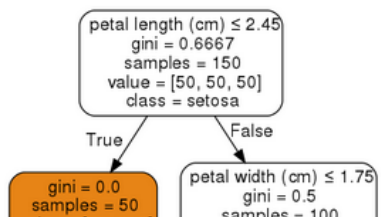
The Decision tree algorithms are one of the popular non linear model. It build a tree where the condition/feature on each splits will be selected on the basis of information gain or Gini impurity value.

If you want to view a linear model like linear regression you can do it simply using matplotlib/seaborn, whereas to visualize trees, we use a special tool called Graphviz. U can install it using below command as how we install other python packages.

```
pip3 install graphviz
```

The Sklearn has a good implementation of a function called export_graphviz which can help us to convert the decision tree model into dot format, which is supported by graphviz tool.

Reference : Sklearn link



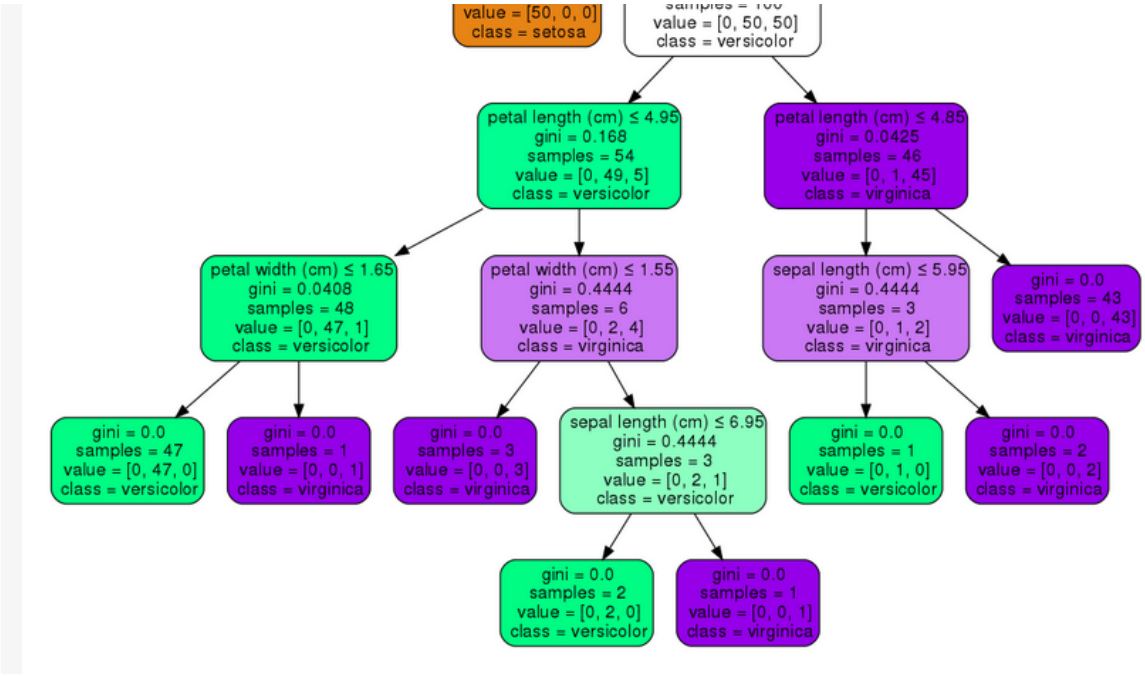
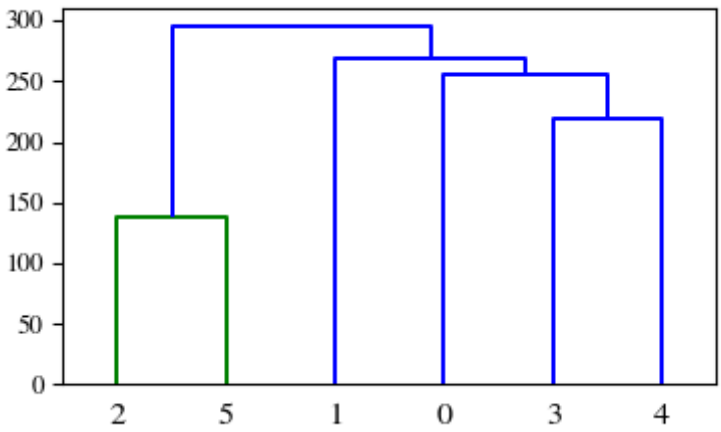


Image taken from Sklearn

Dendograms:

This is the method used to visualize the clusters formed when using the hierarchical clustering technique like Agglomerative Clustering.



Here each chain/link between points, means that they fall under same cluster. The dendogram structure may be too much complex when the number of datapoints are large.

Reference : Link on procedure to plot dendogram

. . .

Think it is enough for this single blog and there are variety of visualization methods discussed. If there is something which I feel missing, I will try to edit or create a part 2 if there is a need..

Thanks you very much if you read my blog until this conclusion and I am really sorry if I took too much of your precious time!! . ***Show your love on claps if you found it useful****. If you still have 2 more minutes, please leave me a feedback that I can use to improve myself in the upcoming days. This is my* ***first blog*** *ever on Internet and as mentioned earlier all your constructive feedbacks are most welcomed.*

> *"I think it's very important to have a feedback loop, where you're constantly thinking about what you've done and how you could be doing it better."*
> ***– Elon Musk (My inspiration ;))***

Have a great day, Happy Coding :) !!!

> *- **Sanat***
>
> *Email ID : smartersanat@gmail.com*
>
> *Linked In : www.linkedin.com/in/sanatmpa*

Edit : Thanks much Applied AI Team for selecting my blog as one of the top 20 special mentioned blog from overall 118 blogs submitted. Link

References:

These below ones helped me a lot for the above blog and would like to honour them by providing reference.

1.) https://dev.to/skotaro/artist-in-matplotlib---something-i-wanted-to-know-before-spending-tremendous-hours-on-googling-how-tos--31oo

2.) https://www.aosabook.org/en/matplotlib.html

3.) https://www.fusioncharts.com/resources/chart-primers/heat-map-chart

4.) Sklearn library.

5.) AppliedAICourse.

6.) Wikipedia.

7.) Imgflip.com for creating meme images.