Iris_Species_Data_Set

November 30, 2018

1 Plotting For Exploratory Data Analysis (EDA)

- exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

Basic Terminology

- What is EDA?
- Data-point/vector/observation where vector is n dimentional numerical array
- Data-set = Table
- Features/Variables/Input-variable/Independent variable
- Lable/Independent variable/output variable
- vector: 2-D,3-D,4-D....n-D

2 2D scatter plot

- seaborn: statistical data visualization. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Q1: While performing this exercise, I am confused between seaborn and matplotlib.pyplot. In many videos in this chapter, seaborn (i.e. sns) and pyplot (i.e. plt) are used jointly to draw a desired plot. What is difference between sns and plt? When to use which one?

Q2: I can't understand the purpose of Facetgrid. I tried documentation. Pls explain or pls. suggest some good link to understand simple usage Facetgrid. When should we use Facetgrid?

Ans: = 1.Seaborn builds on top of Matplotlib and introduces additional plot types. It also makes your traditional Matplotlib plots look a bit prettier.

You should be using both at the same time. Seaborn comes with a number of customized themes and a high-level interface for controlling the look of matplotlib figures.

2. FacetGrid object takes a dataframe as input and the names of the variables that will form the row, column, or hue dimensions of the grid.

The variables should be categorical and the data at each level of the variable will be used for a facet along that axis.

```
In [5]: import warnings
        warnings.filterwarnings("ignore")
```

```
In [4]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np

        iris = pd.read_csv('iris.csv')
        iris.head()

        # we have four features i.e. 'sepal_length', 'sepal_width', 'petal_length', 'petal_wid
```

```
Out[4]:    sepal_length  sepal_width  petal_length  petal_width species
        0           5.1          3.5           1.4          0.2  setosa
        1           4.9          3.0           1.4          0.2  setosa
        2           4.7          3.2           1.3          0.2  setosa
        3           4.6          3.1           1.5          0.2  setosa
        4           5.0          3.6           1.4          0.2  setosa
```

```
In [2]: # (Q) how many data-points and features?
        print(iris.shape)
```

```
(150, 5)
```

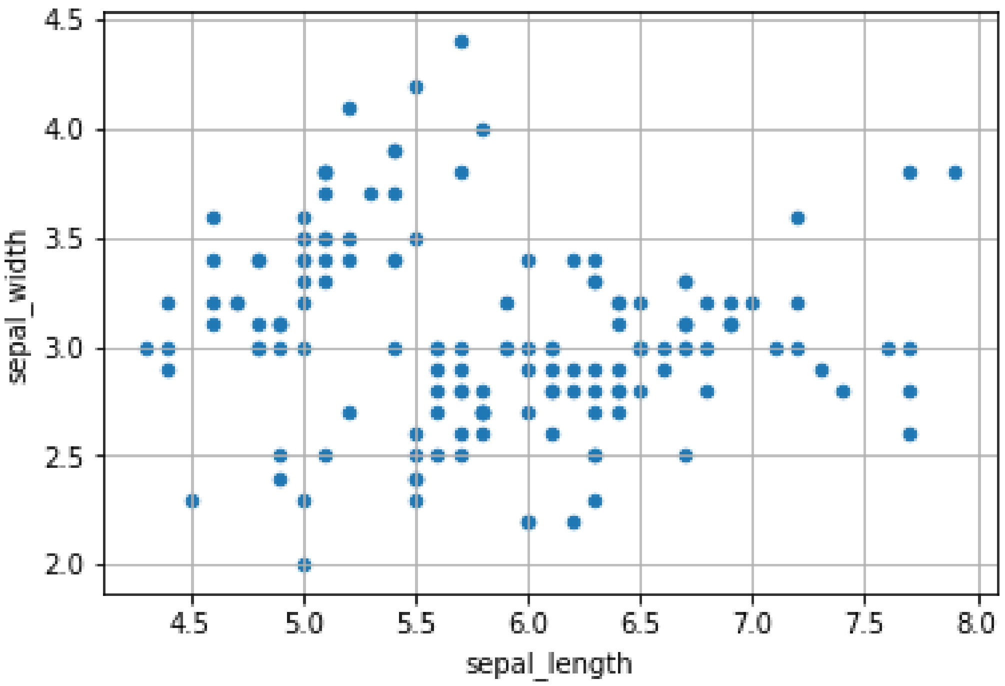
```
In [5]: #(Q) What are the column names in our dataset?
        print(iris.columns)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

```
In [8]: #(Q) How many data points for each class are present?
        #(or) How many flowers for each species are present?

        iris['species'].value_counts()

        # balanced-dataset vs imbalanced-dataset
        # balanced-dataset is almost same from both side
        # imbalanced-dataset is huge point of difference between two data-set

        # iris is a balanced-dataset
```

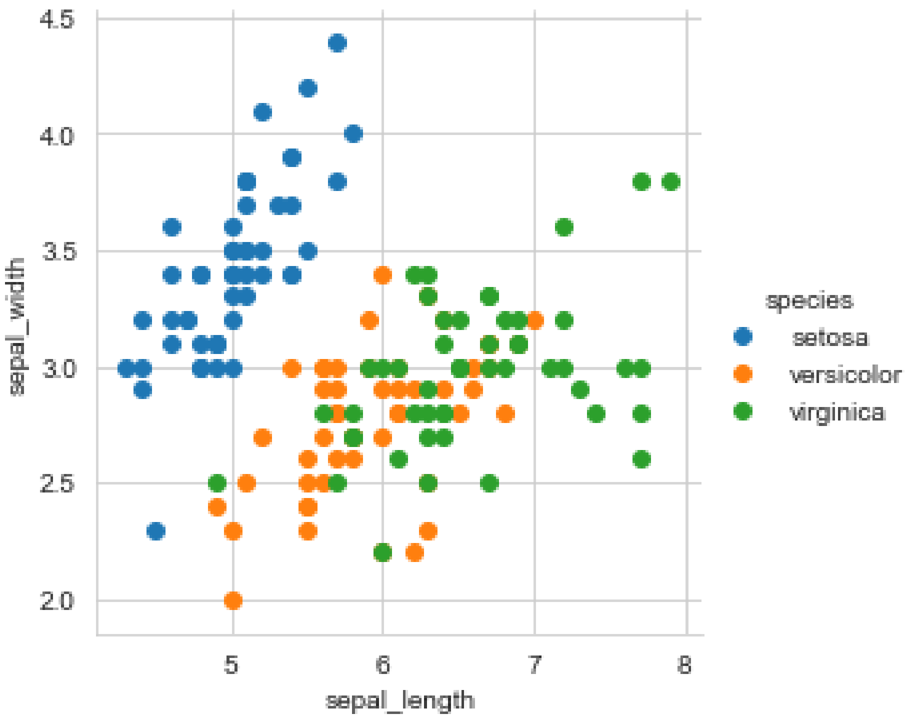
```
Out[8]: virginica     50
        versicolor    50
        setosa        50
        Name: species, dtype: int64
```

```
In [15]: #2-D scatter plot:
         #ALWAYS understand the axis: labels and scale.

         iris.plot(kind="scatter",x="sepal_length",y="sepal_width");
         plt.grid()
         plt.show()

         #cannot make much sense out it.
         #What if we color the points by thier class-label/flower-type.
```



- Remember that here origin is not 0.0 like graph. Its depends on the given points.

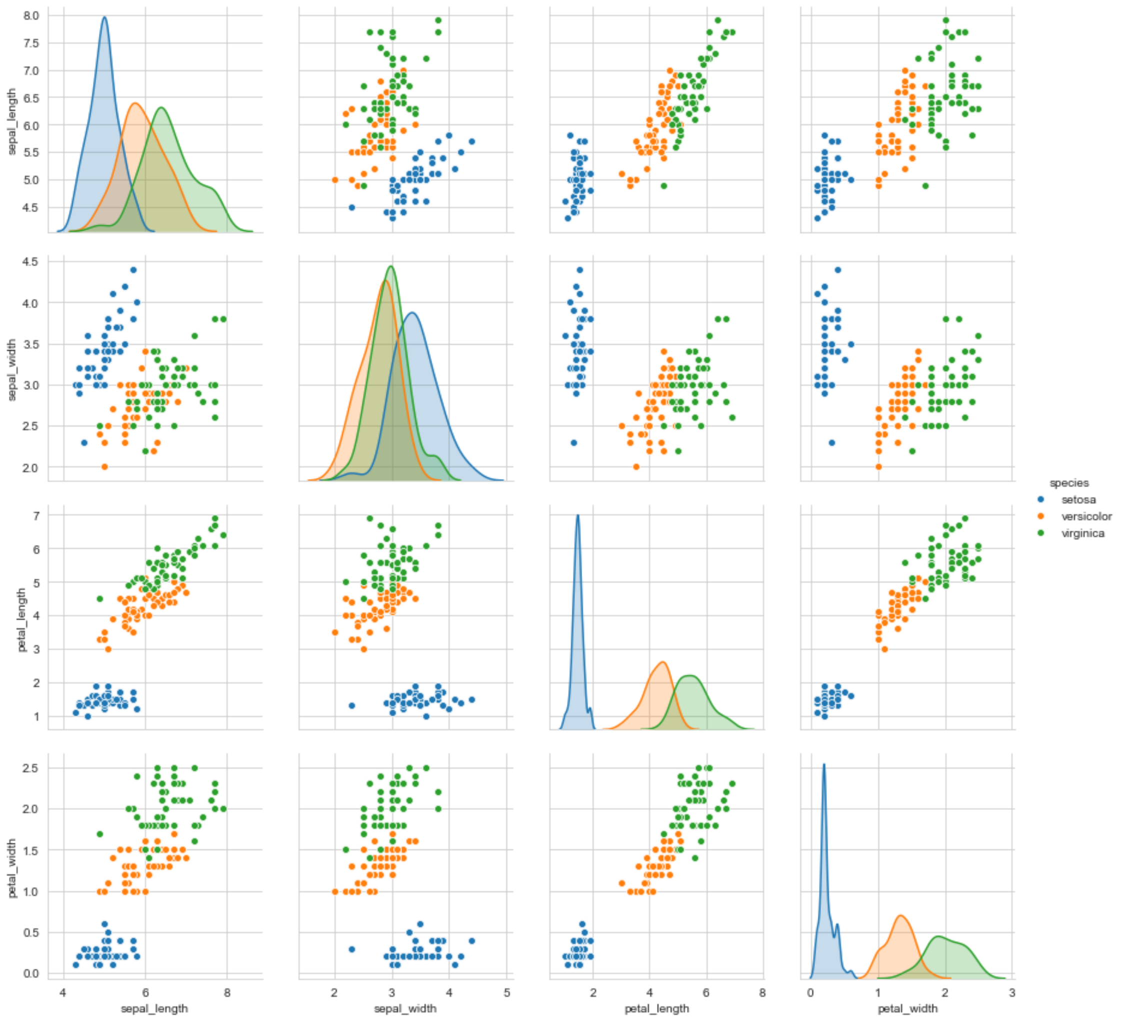
```
In [6]: # 2-D Scatter plot with color-coding for each flower type/class.
        # Here 'sns' corresponds to seaborn.

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

        # Notice that the blue points can be easily seperated
        # from red and green by drawing a line.
        # But red and green data points cannot be easily seperated.
        # Can we draw multiple 2-D scatter plots for each combination of features?
        # How many cobinations exist? 4C2 = 6.
```



3 Observations/Conclusion:

- Using sepal_length and sepal_width features, we can distiguish between setosa flowers from others
- Seperating versicolor from viriginica is much harder as they have considerable overlap
- we can draw a linear line between blue points and red, green points

```
In [7]: plt.close();
        sns.set_style("whitegrid");
        sns.pairplot(iris, hue="species", size=3);
        plt.show();
```



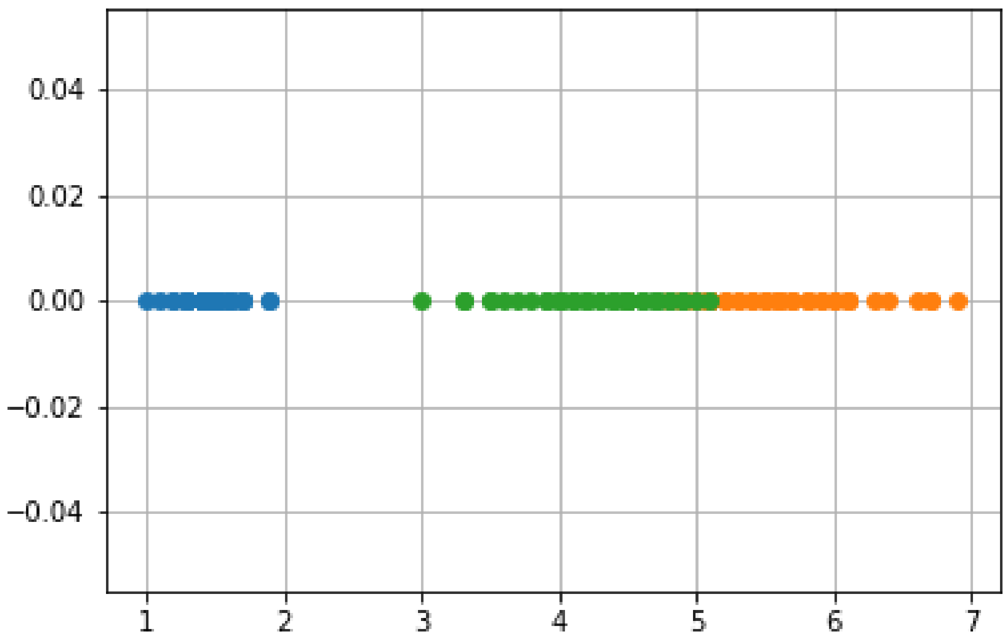
4 Probability Density Function(PDF)

5 & Kernal Density Estimation(KDE)

1-D Array

```
In [3]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np

        iris = pd.read_csv('iris.csv')

        iris_setosa= iris.loc[iris["species"]=="setosa"];
        iris_virginica= iris.loc[iris["species"]=="virginica"];
```

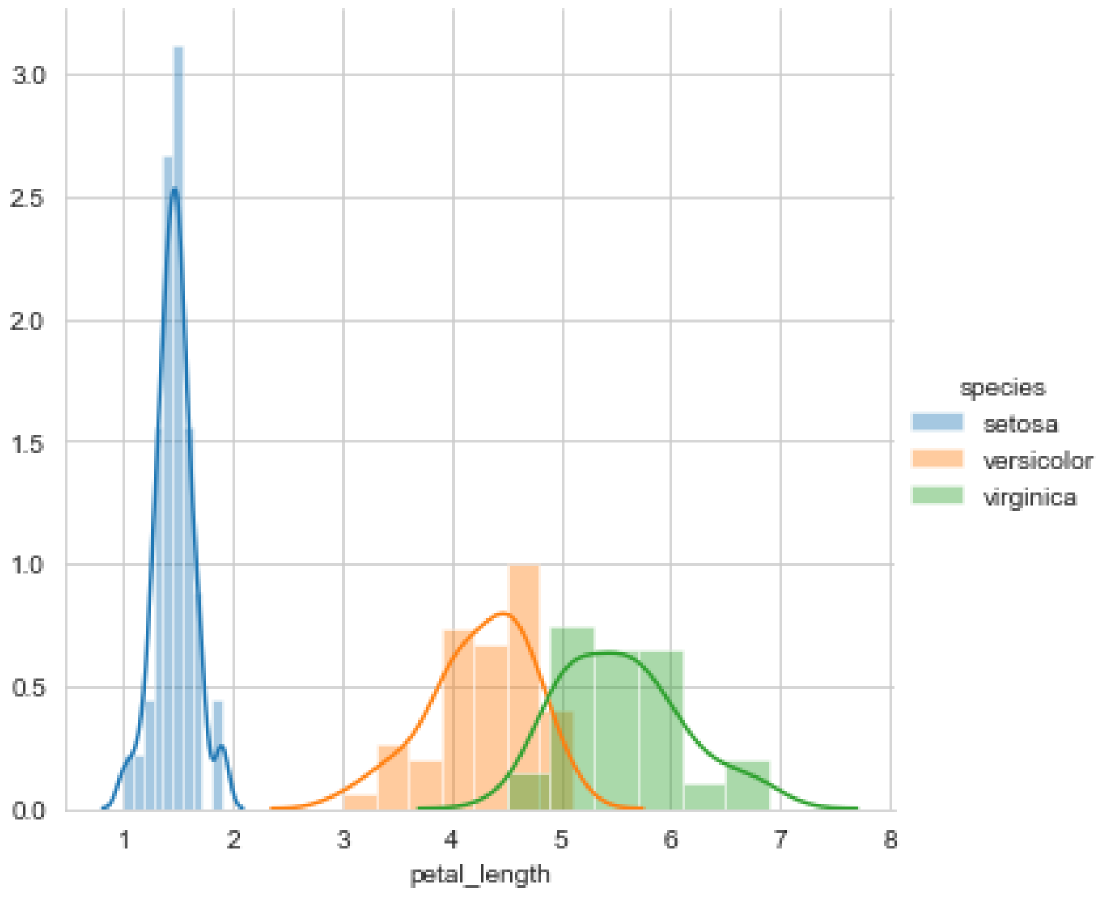
```
iris_versicolor= iris.loc[iris["species"]=="versicolor"];

plt.plot(iris_setosa["petal_length"], np.zeros_like(iris_setosa['petal_length']), 'o')
plt.plot(iris_virginica["petal_length"], np.zeros_like(iris_virginica['petal_length'])
plt.plot(iris_versicolor["petal_length"], np.zeros_like(iris_versicolor['petal_length'

plt.grid()
plt.show()

# X- Axis is Petal Length
```



6 Histogram

- a diagram consisting of rectangles whose area is proportional to the frequency of a variable and whose width is equal to the class interval

```
In [8]: sns.FacetGrid(iris, hue="species", size=5)\
            .map(sns.distplot, "petal_length")\
            .add_legend();
        plt.show();
```

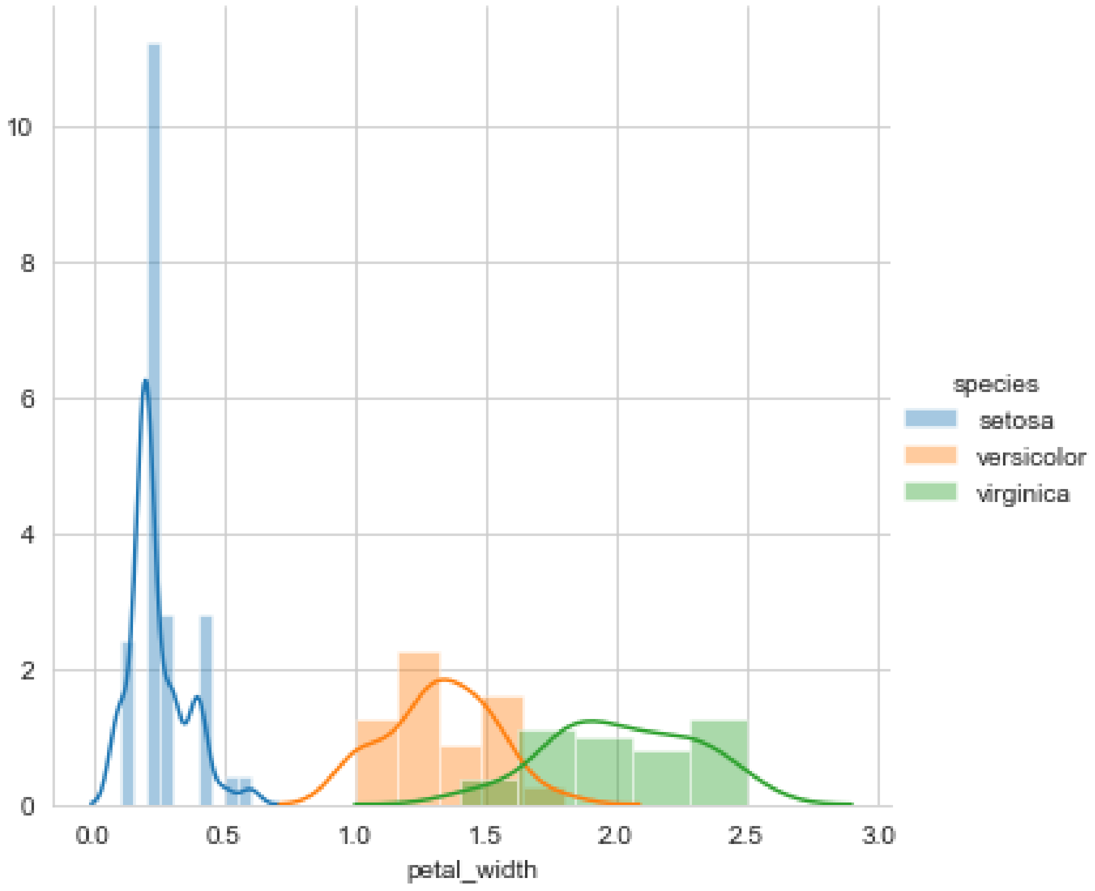


- The blue, orange,green lines called PDF (Probability Density Function)

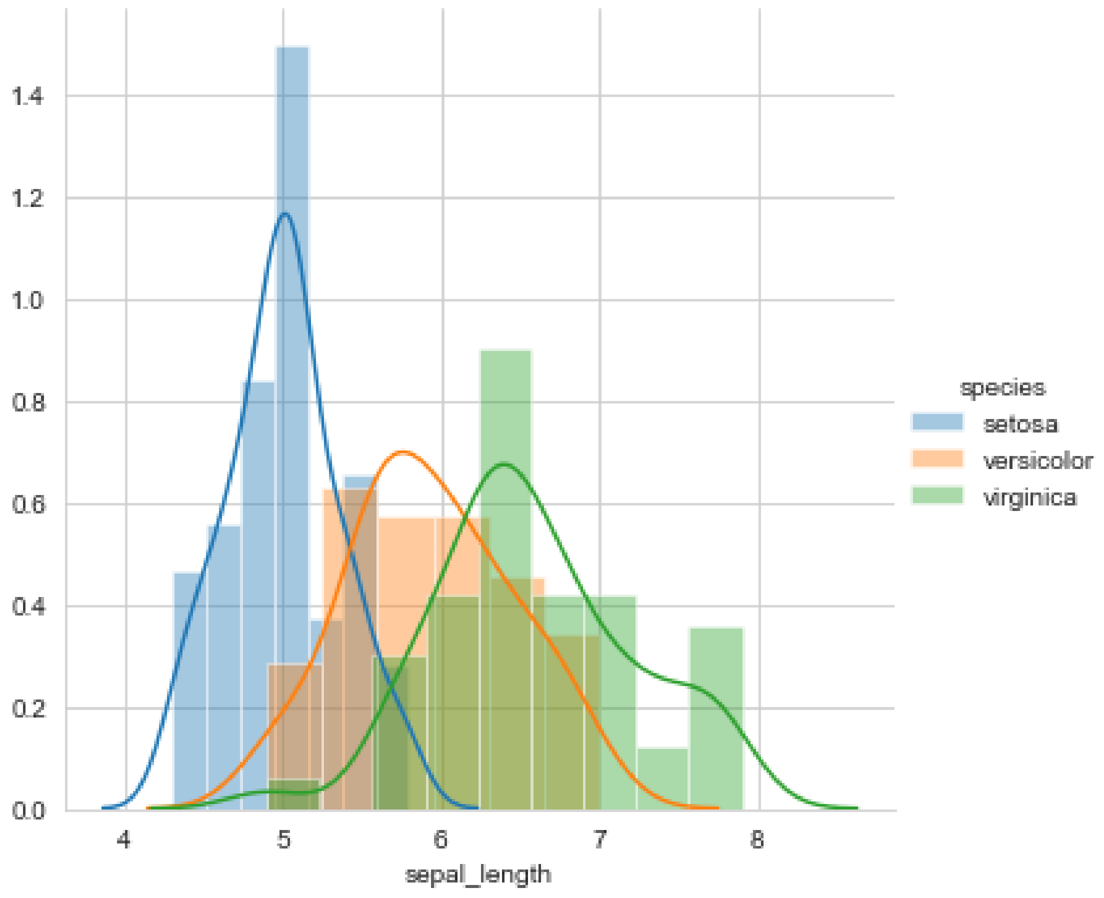
```
In [ ]: """
        we can calculate with if else not exactly but approximately

            if PL<=2
            then setosa

            if PL>=2 & PL<=4.5
            then versicolor # we can not consider 5 coz there is much part of green(verginica)

            final

            if PL<=2
            then setosa
            elif PL>=2 & PL<=4.7 # we can not consider 4.5 coz there is much part of Orange(ve
            then versicolor
            else
```

```
    virginica
"""
```

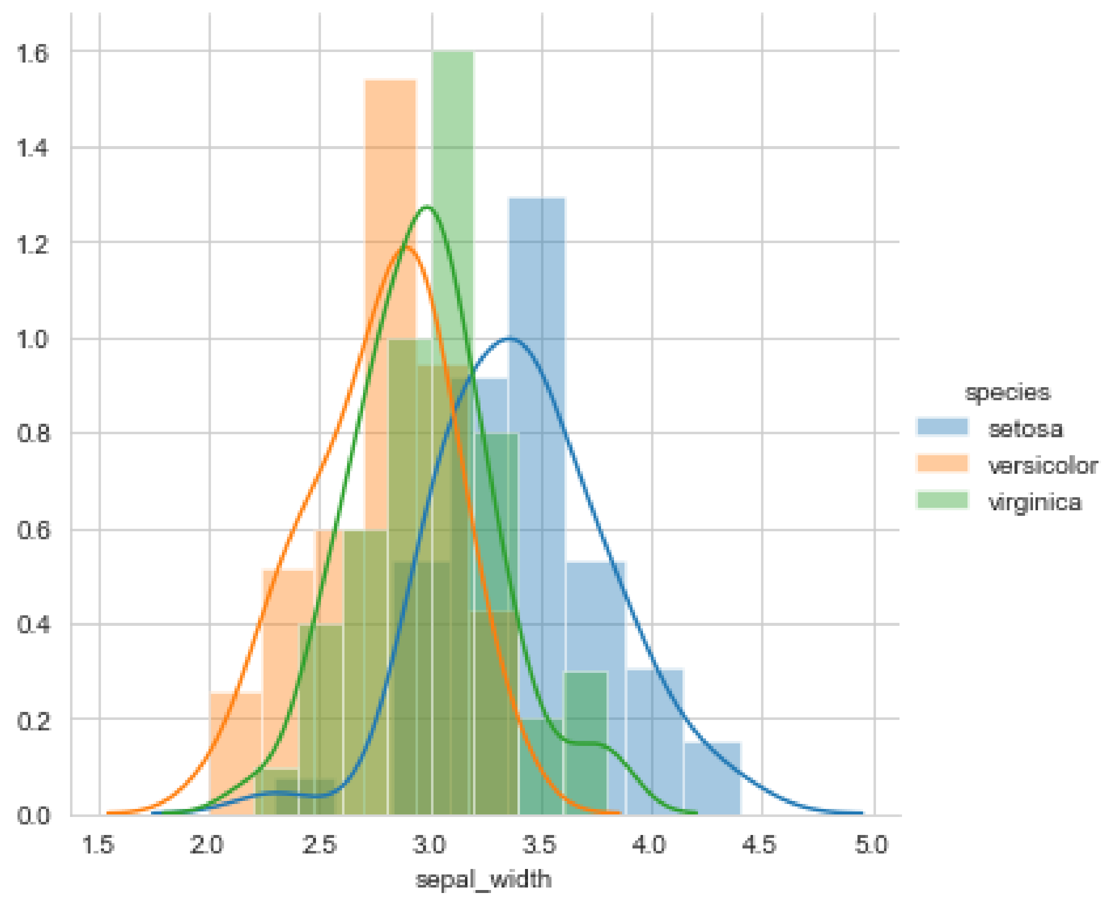
```
In [9]: sns.FacetGrid(iris, hue="species", size=5)\
            .map(sns.distplot, "petal_width")\
            .add_legend();
        plt.show();
```



```
In [10]: sns.FacetGrid(iris, hue="species", size=5)\
             .map(sns.distplot, "sepal_length")\
             .add_legend();
         plt.show();
```



```
In [11]: sns.FacetGrid(iris, hue="species", size=5)\
             .map(sns.distplot, "sepal_width")\
             .add_legend();
         plt.show();
```



- Univariate Analysis Using PDF, Univariate = single varible
- So the point is, if I want to choose 1 diagram I will choose petal_length because all the three colors are well seperated from each other.
- If I want to choose two pair then I will choose PL & PW

PL > PW >> SL >> SW

7 Cumulative Distribution Function

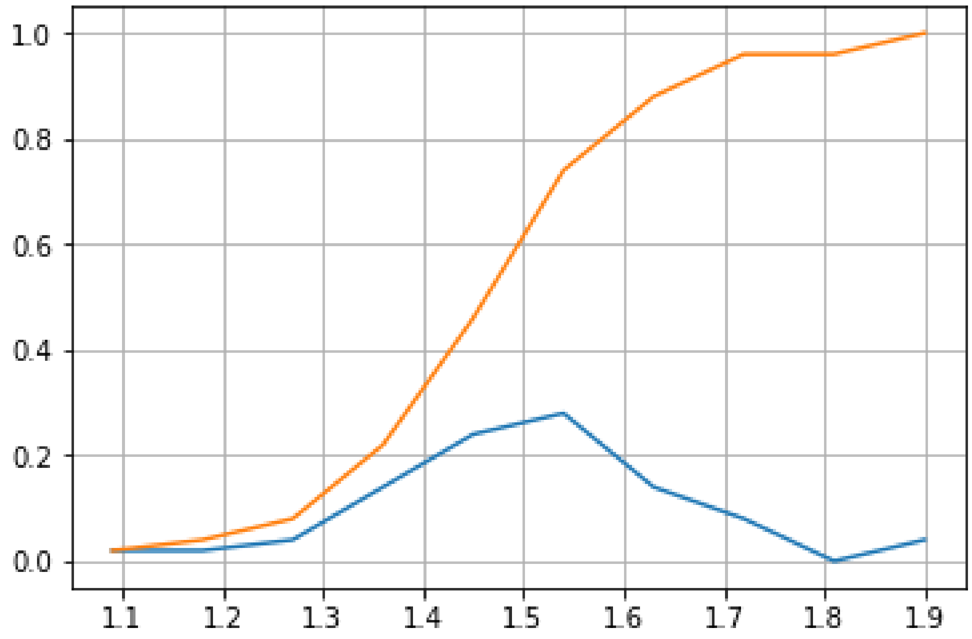
```
In [5]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np



        iris = pd.read_csv('iris.csv')
```

```
iris_setosa= iris.loc[iris["species"]=="setosa"];
iris_virginica= iris.loc[iris["species"]=="virginica"];
iris_versicolor= iris.loc[iris["species"]=="versicolor"];
```

```
counts, bin_edges = np.histogram(iris_setosa['petal_length'], bins=10,
                                 density=True)

pdf=counts/(sum(counts))
print(pdf);
print(bin_edges);
```

```
cdf=np.cumsum(pdf)  # cumsum = cumulative sum
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)

plt.grid();
plt.show();
```

```
[0.02 0.02 0.04 0.14 0.24 0.28 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
```



- Blue Line is PDF

- Orange line is CDF

How CDF biuld?
All flowers = 150 from them setosa is 50 so divide all numbers from 1,2,3----50 by 50 you will get the result and from those points cdf made the orange line

8 What is bins and Bin_edges?

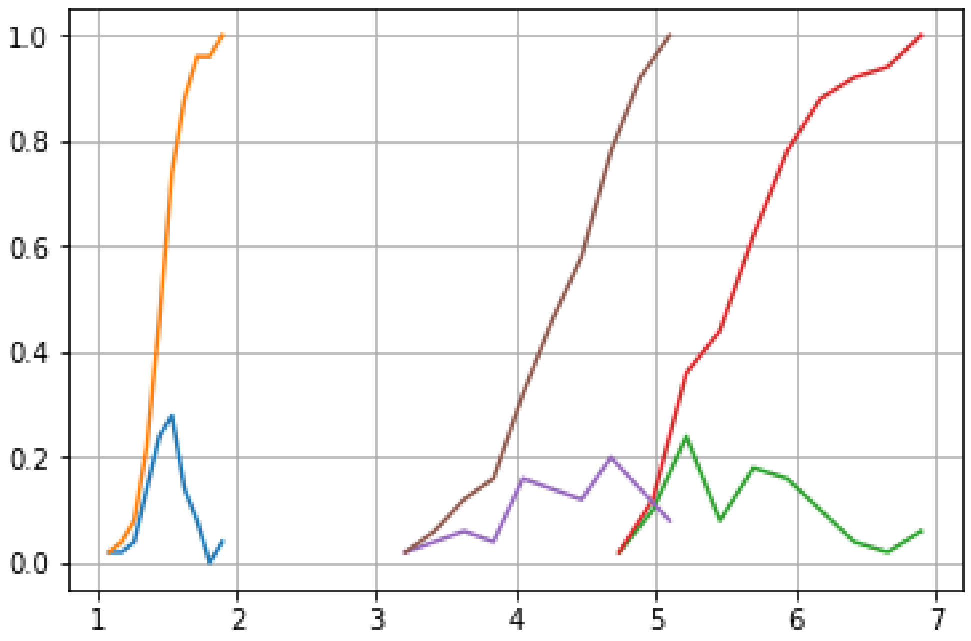
- Bins are the number of intervals you want to divide all of your data into, such that it can be displayed as bars on a histogram.
- A simple method to work our how many bins are suitable is to take the square root of the total number of values in your distribution.

Here bins = range(5) means 0 to 1 interval 1 to 2 interval 2 to 3 interval 3 to 4 interval

```
In [6]: # Plots of CDF of petal_length for various types of flowers.

        # Misclassification error if you use petal_length only.


        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np
        iris = pd.read_csv('iris.csv')

        iris_setosa= iris.loc[iris["species"]=="setosa"];
        iris_virginica= iris.loc[iris["species"]=="virginica"];
        iris_versicolor= iris.loc[iris["species"]=="versicolor"];
        counts, bin_edges = np.histogram(iris_setosa['petal_length'], bins=10,
                                         density=True)
        pdf=counts/(sum(counts))
        print(pdf);
        print(bin_edges);
        cdf=np.cumsum(pdf)  # cumsum = cumulative sum
        plt.plot(bin_edges[1:],pdf)
        plt.plot(bin_edges[1:],cdf)


        counts, bin_edges = np.histogram(iris_virginica['petal_length'], bins=10,
                                         density=True)
        pdf=counts/(sum(counts))
        print(pdf);
        print(bin_edges);
        cdf=np.cumsum(pdf)  # cumsum = cumulative sum
        plt.plot(bin_edges[1:],pdf)
        plt.plot(bin_edges[1:],cdf)
```

```
counts, bin_edges = np.histogram(iris_versicolor['petal_length'], bins=10,
                                 density=True)
pdf=counts/(sum(counts))
print(pdf);
print(bin_edges);
cdf=np.cumsum(pdf)  # cumsum = cumulative sum
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)

plt.grid();
plt.show();
```

```
[0.02 0.02 0.04 0.14 0.24 0.28 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
[0.02 0.1  0.24 0.08 0.18 0.16 0.1  0.04 0.02 0.06]
[4.5  4.74 4.98 5.22 5.46 5.7  5.94 6.18 6.42 6.66 6.9 ]
[0.02 0.04 0.06 0.04 0.16 0.14 0.12 0.2  0.14 0.08]
[3.   3.21 3.42 3.63 3.84 4.05 4.26 4.47 4.68 4.89 5.1 ]
```



PDF:- Setosa = blue, versicolor = purple, virginica = green
Calculation(Prediction)

- if PL<2 then setosa
- if PL>2 && PL>=5 then versicolor # 95% correct
- if PL>2 && PL>5 then virginca # 10% mistake

so we will go with 95% correct

9 Mean, Variance and Std-Deviation

```
In [17]: import pandas as pd
         import seaborn as sns
         import matplotlib.pyplot as plt
         import numpy as np
         iris = pd.read_csv('iris.csv')

         iris_setosa= iris.loc[iris["species"]=="setosa"];
         iris_virginica= iris.loc[iris["species"]=="virginica"];
         iris_versicolor= iris.loc[iris["species"]=="versicolor"];

         print("Means:")
         print(np.mean(iris_setosa['petal_length']))
         #Mean with an outlier
         print(np.mean(np.append(iris_setosa['petal_length'],50))); # outlier means one point

         print(np.mean(iris_virginica['petal_length']))
         print(np.mean(iris_versicolor['petal_length']))


         print("\nstd-div")
         print(np.std(iris_setosa['petal_length']))
         print(np.std(iris_virginica['petal_length']))
         print(np.std(iris_versicolor['petal_length']))
```

```
Means:
1.464
2.4156862745098038
5.5520000000000005
4.26

std-div
0.17176728442867112
0.546347874526844
0.4651881339845203
```

- The standard deviation is the square root of the variance

10 Median, Percentile and Quantile

```
In [7]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np

        iris = pd.read_csv('iris.csv')
```

```
iris_setosa= iris.loc[iris["species"]=="setosa"];
iris_virginica= iris.loc[iris["species"]=="virginica"];
iris_versicolor= iris.loc[iris["species"]=="versicolor"];


print("\nMedians: ")
print(np.median(iris_setosa['petal_length']))
#Mean with an outlier
print(np.mean(np.append(iris_setosa['petal_length'],50))); # outlier means one point v

print(np.median(iris_virginica['petal_length']))
print(np.median(iris_versicolor['petal_length']))


print("\nQuantiles: ")
print(np.percentile(iris_setosa['petal_length'], np.arange(0,100,25)))
print(np.percentile(iris_virginica['petal_length'], np.arange(0,100,25)))
print(np.percentile(iris_versicolor['petal_length'], np.arange(0,100,25)))

print("\n 90th Percentiles: ")
print(np.percentile(iris_setosa['petal_length'], 90))
print(np.percentile(iris_virginica['petal_length'], 90))
print(np.percentile(iris_versicolor['petal_length'], 90))

# MAD is Median Absolute Deviation

from statsmodels import robust
print("\nMedian Absolute Deviation")
print(robust.mad(iris_setosa['petal_length']))
print(robust.mad(iris_virginica['petal_length']))
print(robust.mad(iris_versicolor['petal_length']))
```

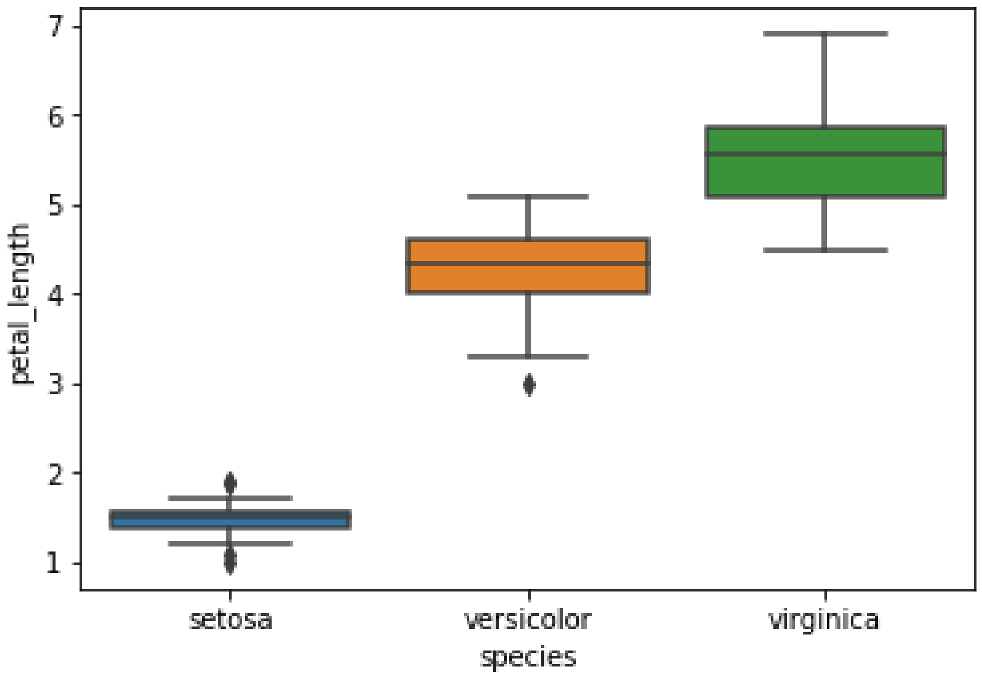
```
Medians:
1.5
2.4156862745098038
5.55
4.35

Quantiles:
[1.    1.4   1.5   1.575]
[4.5   5.1   5.55  5.875]
[3.   4.   4.35 4.6 ]

 90th Percentiles:
1.7
6.3100000000000005
```

```
4.8

Median Absolute Deviation
0.14826022185056031
0.6671709983275211
0.5189107764769602
```

11 Box-Plot with Whiskers

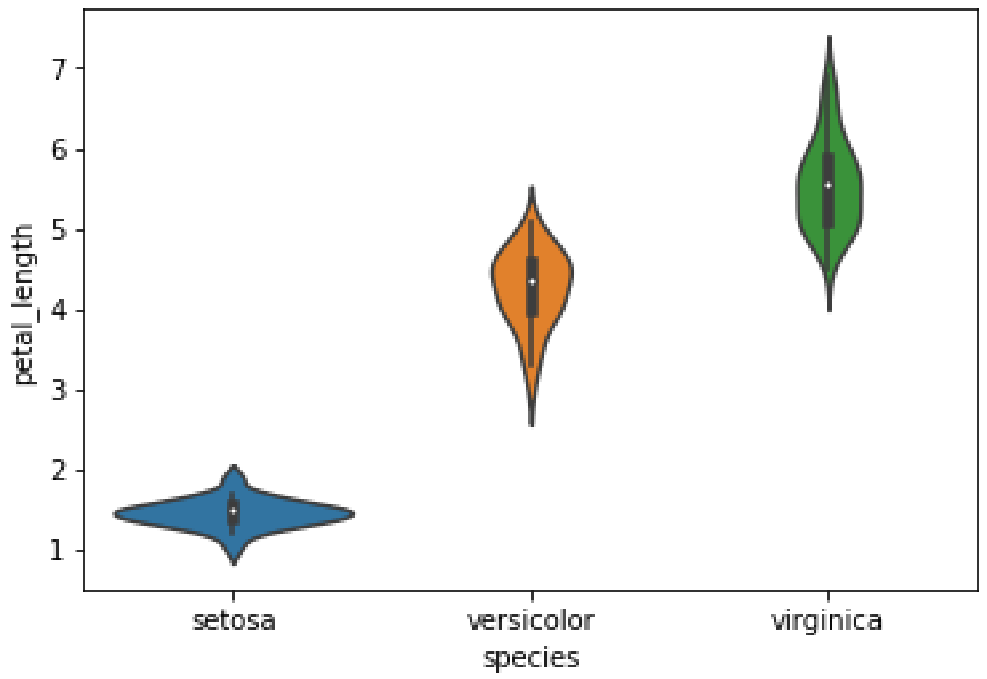
Whiskers means Long mouth hairs of cat

```
In [8]: import seaborn as sns

        sns.boxplot(x='species', y='petal_length', data=iris)
        plt.show()
```



```
In [ ]: """
        if PL>2 & PL<=5 then versicolor
        if PL>=5 then virginica # so this is not correct PL of verginica is 25% present in ver
        """
```

```
In [11]: sns.violinplot(x='species', y='petal_length', data=iris, size=8)
         plt.show()
```



12 Summarising plots in English

- Explain your findings/conclusions in plain english
- Never forget your objective(the problem you are solving). Perform all of your objective along with objectives

13 Univariate Bivariate and Multivariate

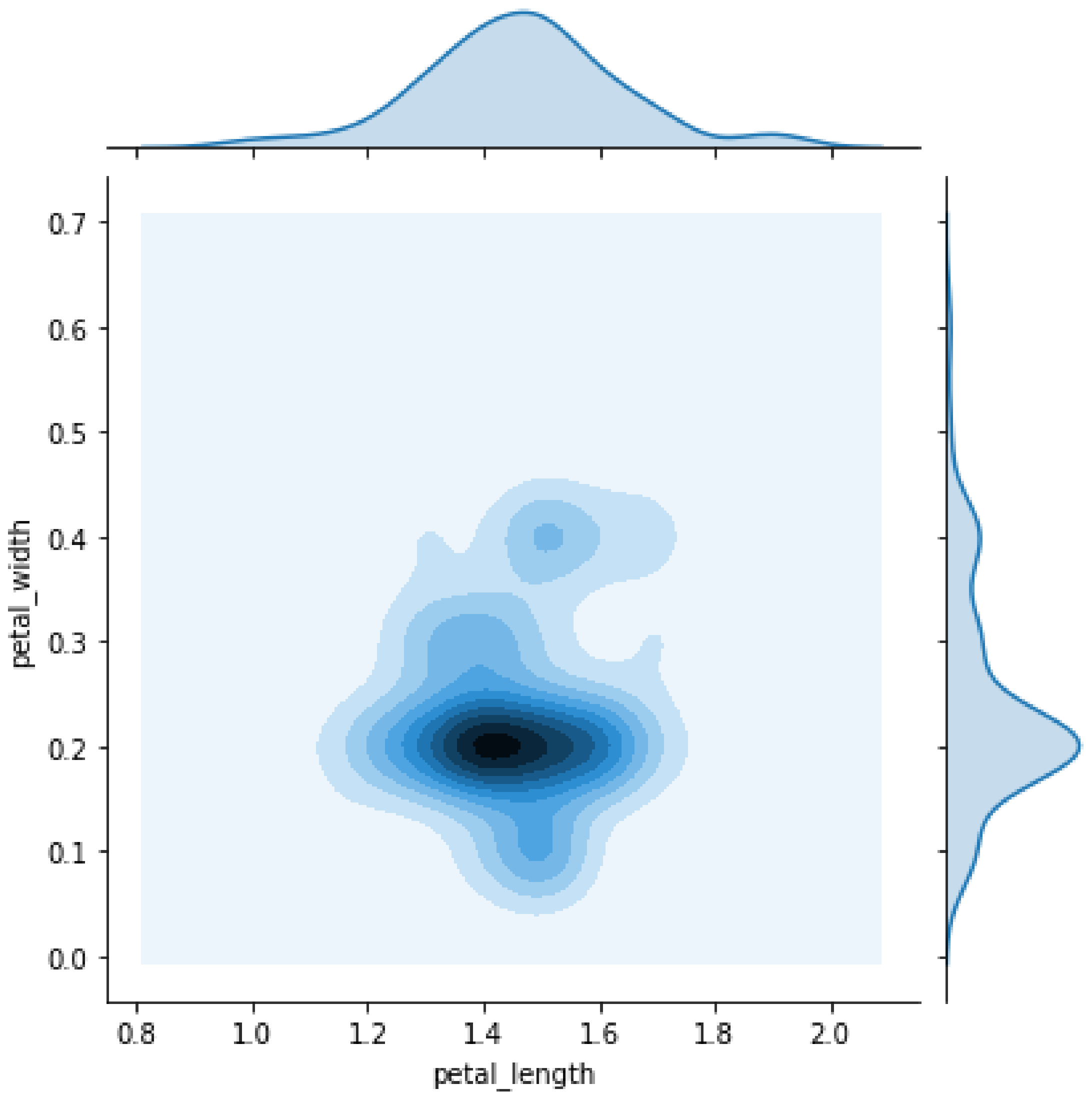
Def: Univariate= You are looking at one variable e.g. Petal_Length and you are plotting everything like PDF, CDF, violin,boxplot etc

Bivariate = Pair plot or scatter plot of two variable like PL,PW

Multivariate = When you are looking at more than two variables

14 Multivariate probability density, Contour plot

```
In [16]: sns.jointplot(x='petal_length', y='petal_width', data=iris_setosa, kind='kde') #kde=k
         plt.show();
```



The lines which are dark blue to light blue called contour

```
In [ ]: #can change integer value into string
        #haberman['Surv_status']=haberman['Surv_status'].map({1:'yes',2:'no'})
```

```
In [ ]: #Difference Between observation and conclusion(inference)
```

Observations are not final; therefore, it's okay to make mistakes, but conclusions are final and so errors should be avoided.

You could also try out 3D plots using libraries such as plotly

```
In [ ]:
```