Dr. D. Y.Patil Institute of Technology, Pimpri

Department of Computer Engineering

Lab Manual of

Computational Programming Laboratory (310302)

For T.E. (Honours* in Artificial Intelligence and Machine Learning)

Academic Year: 2020-2021

Semester- V

Prepared By, Prof. ArchanaChaudhari

List of Assignments

Suggested list of assignments (Use suitable programming language/Tool for implementation)	
Sr. No	**Assignment statement**
1.	Compute Estimators of the main statistical measures like Mean, Variance, Standard Deviation, Covariance, Correlation and Standard error with respect to any example. Display graphically the distribution of samples.
2.	Plot the Normal Distribution for class test result of a particular subject. Identify the Skewness and Kurtosis
3.	Load the dataset: birthwt Risk Factors Associated with Low Infant Birth Weight at https://raw.github.com/neurospin/pystatsml/master/datasets/birthwt.csv 1. Test the association of mother's (bwt) age and birth weight using the correlation test and linear regeression. 2. Test the association of mother's weight (lwt) and birth weight using the correlation testand linear regeression. 3. Produce two scatter plot of: (i) age by birth weight; (ii) mother's weight by birth weight. Elaborate the Conclusion ?
4.	Apply Basic PCA on the iris dataset. The data set is available at: https://raw.github.com/neurospin/pystatsml/master/datasets/iris.csv • Describe the data set. Should the dataset been standardized? • Describe the structure of correlations among variables. • Compute a PCA with the maximum number of components . • Compute the cumulative explained variance ratio. Determine the number of components*K*by your computed values. • Print the*K*principal components directions and correlations of the*K*principal compo-nents with the original variables. Interpret the contribution of the original variables into the PC. • Plot the samples projected into the*K*first PCs. • Color samples by their species.
5.	Perform clustering of the iris dataset based on all variables using Gaussian mixture models. Use PCA to visualize clusters.

Assignment 1

Title

Main Statistical Measures

Problem Statement:

Compute Estimators of the main statistical measures like Mean, Variance, Standard Deviation, Covariance, Correlation and Standard error with respect to any example. Display graphically the distribution of samples.

Learning Objectives:

To understand modern computational methods used in statistics.

Learning Outcome

Identify the suitable method of statistics on the given data to solve the problem of any heuristic approach of prediction.

Prerequisite:

- Basics of Statistics
- Any programming Language (Ex. Python)

Theory:

i) Mean

The most commonly used measure of central tendency of a set of observations is the mean of the observations.The mean of a set of observations is their average. It is equal to the sum of all observations divided by the number of observations in the set.

Let us denote the observations by $X_1, X_2, \ldots, X_n$. That is, the first observation is denoted by X_1, the second by X_2 , and so on to the nth observation, X_n. The sample mean is denoted by x

Mean of a sample:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Where $\sum$ is summation notation. The summation extends over all data points.

When our observation set constitutes an entire population, instead of denoting the mean by $\bar{X}$ we use the symbol μ (the Greek letter mu). For a population, we use N as the number of elements instead of n. The population mean is defined as follows.

Mean of a population:

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

Population vs. Sample

A population refers to the summation of all the elements of interest to the researcher.

- Examples: the no. of people in a country, the number of hedge funds in the U.S., or even the total no. of CFA candidates in a given year.

A sample is just a set of elements that represent the population as a whole. By analyzing sample data, we are able to make conclusions about the entire population.

- For example, if we sample the returns of 30 hedge funds spread across the U.S., we can use the results to make reasonable conclusions about the market as a whole (well over 10,000 hedge funds).

ii) Variance

The variance of a set of observations is the average squared deviation of the data points from their mean.

When our data constitute a sample, the variance is denoted by S^2 and the averaging is done by dividing the sum of the squared deviations from the mean by n – 1. When our observations constitute an entire population, the variance is denoted by σ^2, and the averaging is done by dividing by N. (And σis the Greek letter sigma; we call the variance *sigma squared*. The capital sigma is known to you as the symbol we use for summation, ∑.)

Sample variance:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Population variance:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$$

where μ is the population mean.

iii) Standard Deviation

The standard deviation of a set of observations is the (positive) square root of the variance of the set.

The standard deviation of a sample is the square root of the sample variance, and the standard deviation of a population is the square root of the variance of the population

Sample standard deviation:

$$s = \sqrt{s^2} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Population standard deviation:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

iv) Covariance

Covariance is a measure of how closely two assets move together. In covariance, we focus on the relationship between the deviations of some two variables rather than the deviation from the mean of one variable.

If the means of random variables x and y are known, then the covariance between the two random variables can be determined as follows:

$$\hat{\sigma}_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

If we do not know the means, then the equation changes to:

0.

$$\hat{\sigma}_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu}_x)(y_i - \hat{\mu}_y)$$

v) Correlation

Correlation is a concept that is closely related to covariance in the following way:

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

Correlation ranges between +1 and -1 and is, therefore, much easier to interpret than covariance. Two variables are perfectly correlated if their correlation is equal to +1, uncorrelated if their correlation is equal to 0, and move in perfectly opposite directions if their correlation is equal to -1.

vi) Standard Error

The mean square error MSE is an unbiased estimator of the variance of the population errors ε, which we denote by σ^2.

The mean square error is

$$\text{MSE} = \frac{\text{SSE}}{n-(k+1)} = \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{n-(k+1)}$$

The square root of MSE is usually denoted by *s* and isreferred to as the *standard error of estimate*.

The standard error of estimate is

$$s = \sqrt{MSE}$$

vii) **Python Numpy Package**

Numpy stands for numerical python. Fundamental package for numerical computations in Python. Supports N-dimensional array objects that can be used for processing multidimensional data. Supports different data-types. Using Numpy we can perform

- Mathematical and logical operations on arrays
- Fourier transforms
- Linear algebra operations
- Random number generation

Ordered collection of elements of basic data types of given length

Syntax: numpy.array(object)

```
import numpy as np

In [2]: x=np.array([2,3,4,5])

In [3]: print(type(x))
<class 'numpy.ndarray'>


In [4]: print(x)
[2 3 4 5]
```

viii) Python Pandas Package:

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool built on top of the Python programming language.

Besides creating a DataFrame by reading a file, you can also create one via a Pandas Series. Series are one dimensional labeled Pandas arrays that can contain any kind of data, even NaNs (Not A Number), which are used to specify missing data. Let's create a small DataFrame, consisting of the grades of a high schooler:

```
classes = pd.Series(["Mathematics", "Chemistry", "Physics", "History", "Geography",
"German"])
grades  =pd.Series([90,54,77,22,25])
pd.DataFrame({"Classes": classes, "Grades": grades})
```

The result look like:

	Classes	Grades
0	Mathematics	90
1	Chemistry	54
2	Physics	77
3	History	22
4	Geography	25
5	German	NaN

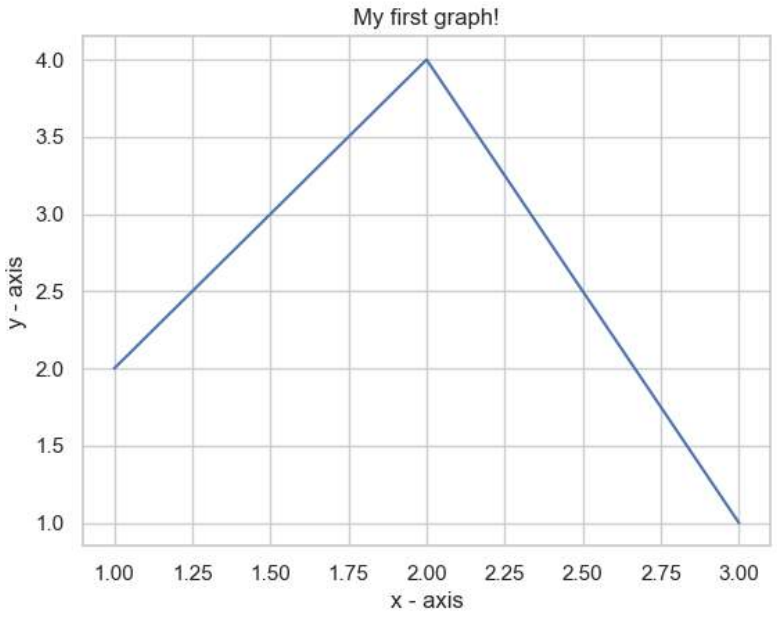
ix) Python Matplotlib Package:

Matplotlib is arguably the most popular graphing and data visualization library for Python.

```
# importing the required module
importmatplotlib.pyplot as plt

# x axis values
x =[1,2,3]
# corresponding y axis values
y =[2,4,1]

# plotting the points
plt.plot(x, y)

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('My first graph!')

# function to show the plot
plt.show()
```

Following steps were followed:

- Define the x-axis and corresponding y-axis values as lists.
- Plot them on canvas using **.plot()** function.
- Give a name to x-axis and y-axis using **.xlabel()** and **.ylabel()** functions.
- Give a title to your plot using **.title()** function.
- Finally, to view your plot, we use **.show()** function.



Actual Code:

a) Import libraries

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

b) Using Numpy generate the random number

```
In [2]: np.random.seed(5)
        x = np.random.randint(10,50,10)
        y = np.random.randint(20,60,10)
```

```
In [3]: x.sort()
        y.sort()
```

c) Mean

```
In [4]: def calc_mean(dataset):

            '''
                Def     : Mean is defined as the arithmetic average of
                          a population.

                Formula : (sum of obesrvations)/(No. of observations)
            '''
            return dataset.sum()/len(dataset)
```

d) Variance

```
In [5]: def calc_variance(dataset,mean):

            '''
                Def     : Variance is the degree of variation/spread
                          in the dataset.

                Formula : 1) Σ((X - X_mean)^2) / n
            '''
            squared_diff = np.square(dataset-mean)
            return calc_mean(squared_diff)
```

e) Standard Deviation

```
In [6]: def calc_SD(variance):

            '''
                Def     : 1) Standard deviation is the amount of deviation
                             of points around the mean.
                          2) Variation but in terms of the actual dataset.

                Formula : √(variance)
            '''

            return np.sqrt(variance)
```

f) Covariance

```
In [7]: def calc_covariance(dataset1,dataset2):

            '''
                Def     : Covariance measures the relationship trend
                          between two sets of data.

                Formula : 1) Σ((X - X_mean)*(Y - Y_mean)) / n
            '''
            mean1 = calc_mean(dataset1)
            mean2 = calc_mean(dataset2)
            return np.sum(np.multiply(dataset1-mean1,dataset2-mean2))/len(dataset1)
```

g) Correlation

```
In [8]: def calc_correlation(dataset1,dataset2):

            '''
                Def     : Covariance measures the relationship trend
                          between two sets of data.

                Formula : 1) Σ((X - X_mean)*(Y - Y_mean)) / √(Σ(X - X_mean)^2*Σ(Y - Y_mean)^2)
            '''
            mean1 = calc_mean(dataset1)
            mean2 = calc_mean(dataset2)

            num =   np.sum(np.multiply(dataset1-mean1,dataset2-mean2))
            de = np.multiply(np.sum(np.square(dataset1-mean1)),np.sum(np.square(dataset2-mean2)))
            return num/np.sqrt(de)
```

h) Standard error

```
In [9]: def calc_SE(dataset,sd):

            '''
                Def     : The standard error is a statistical term that
                          measures the accuracy with which a sample
                          distribution represents a population by using
                          standard deviation.

                Formula :  Standard_deviation / √(n)
            '''

            return sd/np.sqrt(len(dataset))
```

i) Output

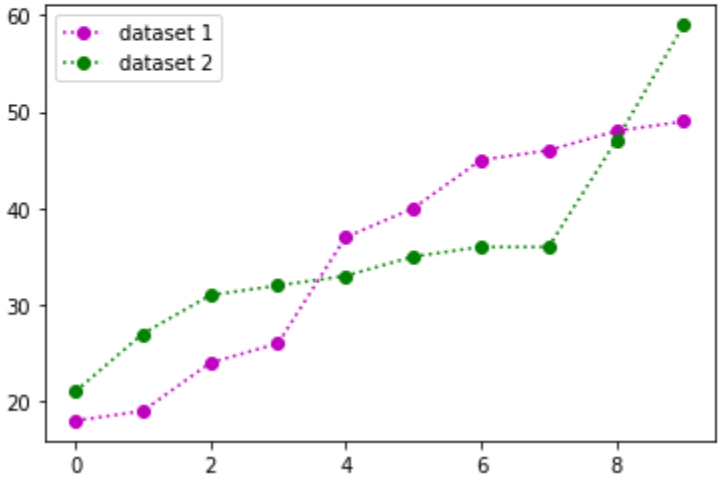
```
In [10]: mean = calc_mean(x)
         mean2 = calc_mean(y)
         variance = calc_variance(x,mean)
         S_D = calc_SD(variance)
         covariance = calc_covariance(x,y)
         correlation = calc_correlation(x,y)
         S_E = calc_SE(x,S_D)
```

```
In [11]: print(mean,mean2,variance,S_D,covariance,correlation,S_E)

35.2 35.7 136.16 11.668761716651858 94.46000000000001 0.8070540019015658 3.6899864498396195
```

j) Graphical Display

```
In [12]: plt.plot(x,"mo:",label="dataset 1")
         plt.plot(y,"go:",label = "dataset 2")

         plt.legend(loc="upper left")
```

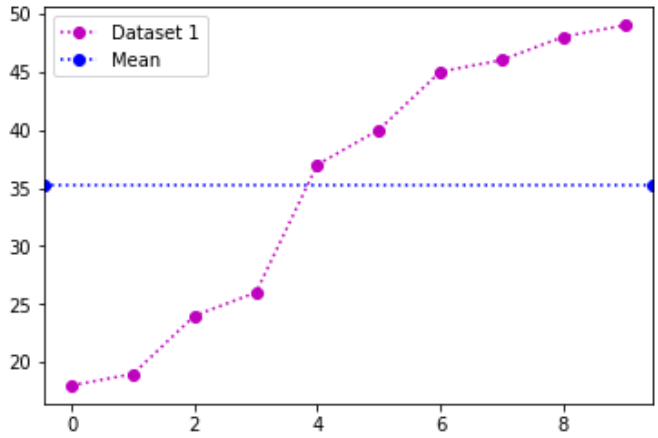
```
Out[12]: <matplotlib.legend.Legend at 0x94c8fe8>
```



Mean

In [13]:
```
plt.plot(x,"mo:",label="Dataset 1")
plt.axhline(mean,color='b',marker= 'o', linestyle=':',label="Mean")
plt.legend(loc="upper left")
```

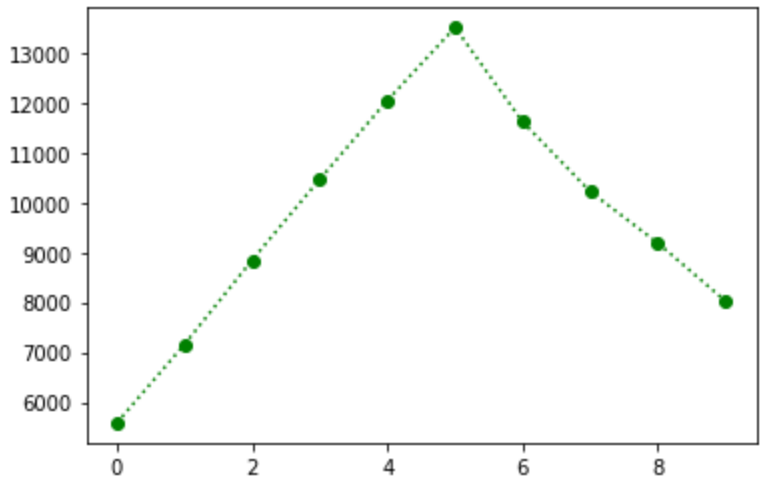
Out[13]: <matplotlib.legend.Legend at 0xb55b0b8>



Corelation

In [15]:
```
corr = np.correlate(x, y, "same")
plt.plot(list(corr),"go:",label = "Correlation")
```

Out[15]: [<matplotlib.lines.Line2D at 0xb5d8c70>]



Conclusion/Analysis:

Hence we are able to study basic concepts of statistics and display the distribution of samples graphically

Program codes with sample output

Assignment Question?

(1) A sample of 15 data is as follows: 17, 18, 17, 17, 13, 18, 5, 5, 6, 7, 8, 9, 20, 17, 3. Calculate the mean of the data

(2) Calculate variance for the data set of 6 scores to walk through the steps.

Dataset					
46	69	32	60	52	41

(3) Calculate the standard deviation for the data 15, 22, 27, 11, 9, 21, 14, 9.

(4) Calculate Covariance for the following datset:

X	2.1	2.5	3.6	4.0
Y	8	10	12	14

(5) Calculate correlation coefficient for the following data

X	2	4	5	6	8	11
Y	18	12	10	8	7	5

(6) Find the Standard error for the following data

Name	Height to nearest 0.5 cm
Waldo	150.5
Finn	170.0
Henry	160.0
Alfie	161.0
Shane	170.5

Assignment 2

Title

Normal Distribution

Problem Statement:

Plot the Normal Distribution for class test result of a particular subject. Identify the Skewness and Kurtosis

Learning Objectives:

To get detailed approach of simulation, estimation and visualization of statistical data

Learning Outcome

Apply appropriate statistical concepts and skills to solve problems in both familiar and unfamiliar situations including those in real-life contexts.

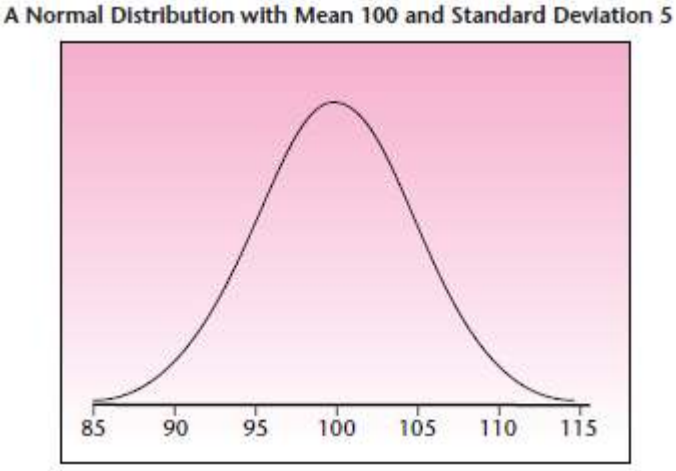
Prerequisite:

- Concept of data distribution

Theory:

x) Normal Distribution

The normal distribution is an important continuous distribution because a good number of random variables occurring in practice can be approximated to it. If a random variable is affected by many independent causes, and the effect of each cause is not overwhelmingly large compared to other effects, then the random variable will closely follow a normal distribution. A normal distribution, sometimes called the bell curve, is a distribution that occurs naturally in many situations.



For a normal distribution with mean μ and standard deviation σ, the probability density function f(x) is given by the complicated formula

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- **Properties of Normal Distribution**

  If several independent random variables are normally distributed, then their sum will also be normally distributed. The mean of the sum will be the sum of all the individual means, and by virtue of the independence, the variance of the sum will be the sum of all the individual variances.

  We can write this in algebraic form as

If $X_1, X_2, \ldots, X_n$ are independent random variables that are normally distributed, then their sum S will also be normally distributed with

$$E(S) = E(X_1) + E(X_2) + \cdots + E(X_n)$$

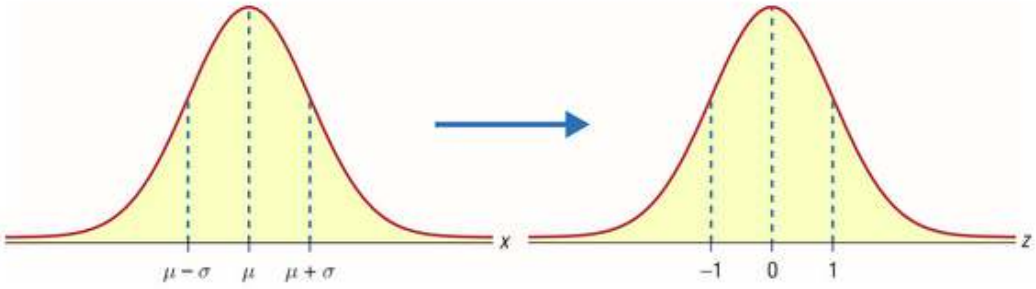
and

$$V(S) = V(X_1) + V(X_2) + \cdots + V(X_n)$$

- **The Standard Normal Distribution**

  Standard Normal Distribution is a special case of Normal Distribution when $\boldsymbol{\mu = 0}$ and $\boldsymbol{\sigma = 1}$. For any Normal distribution, we can convert it into Standard Normal distribution using the formula:

$$z = \frac{x - \mu}{\sigma}$$



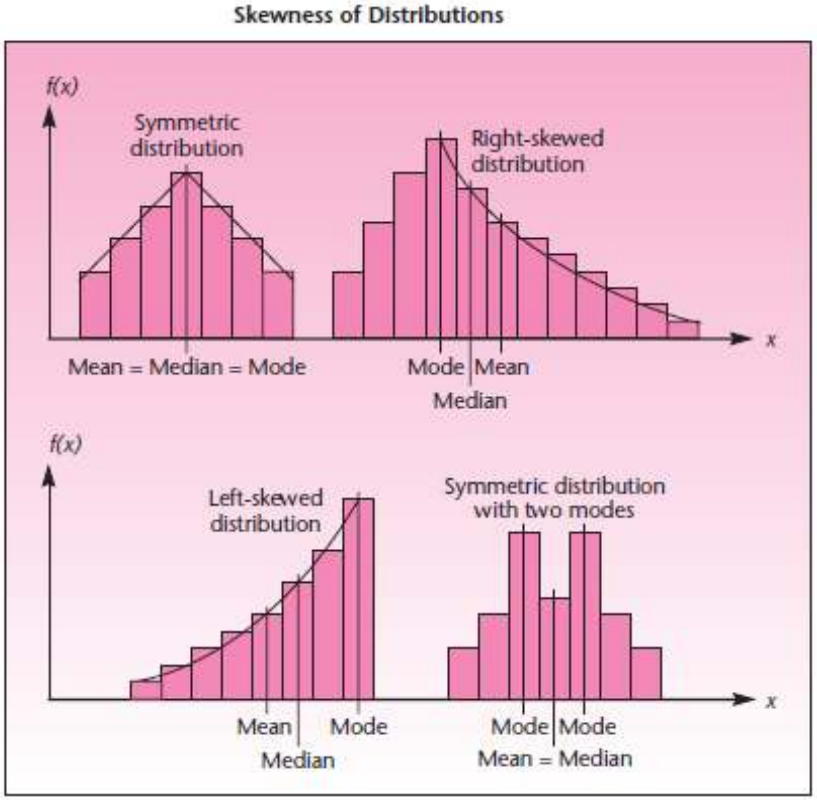
xi) Skewness

In addition to measures of location, such as the mean or median, and measures of variation, such as the variance or standard deviation, two more attributes of a frequency
distribution of a data set may be of interest to us. These are skewness and kurtosis.

Skewness is a measure of the degree of asymmetry of a frequency distribution.

When the distribution stretches to the right more than it does to the left, we say that the

distribution is right skewed. Similarly, a left-skewed distribution is one that stretches asymmetrically to the left.



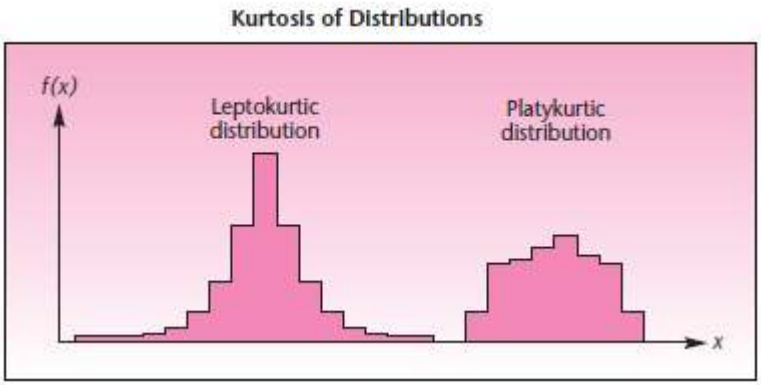
A symmetric distribution with a single mode has mode = mean = median. Generally, for a right-skewed distribution, the mean is to the right of the median, which in turn lies to the right of the mode (assuming a single mode). Mode > Median > Mean. The opposite is true for left-skewed distributions.

xii) **Kurtosis**

Kurtosis is a measure of the peakedness of a distribution. The larger the kurtosis, the more peaked will be the distribution. The kurtosis is calculated and reported either as an absolute or a relative value. Absolute kurtosis is always a positive number.

$$\text{Relative kurtosis} = \text{Absolute kurtosis} - 3$$

This value of 3 is taken as the datum to calculate the relative kurtosis. The relative kurtosis can be negative. We will always work with relative kurtosis.

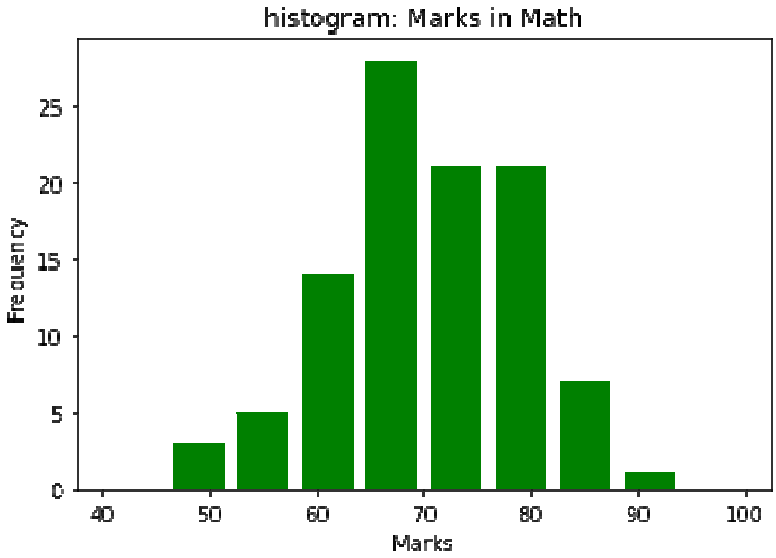


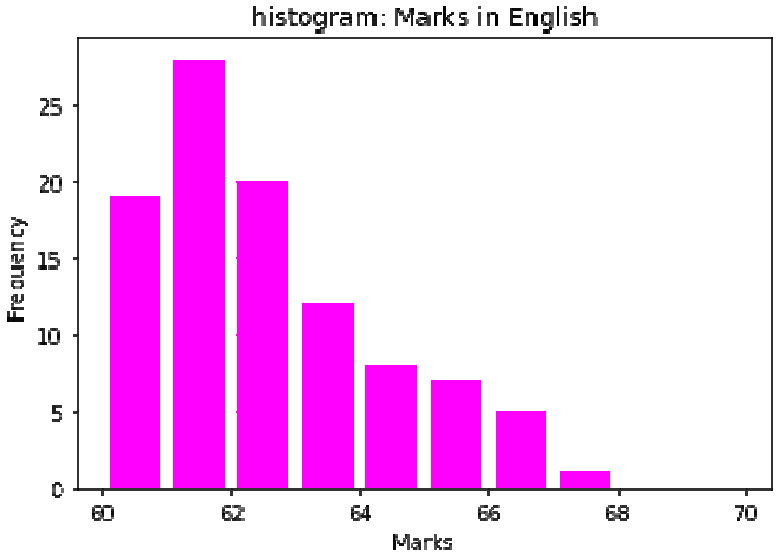
A negative kurtosis implies a flatter distribution than the normal distribution, and it is called platykurtic. A positive kurtosis implies a more peaked distribution than the normal distribution, and it is called leptokurtic.

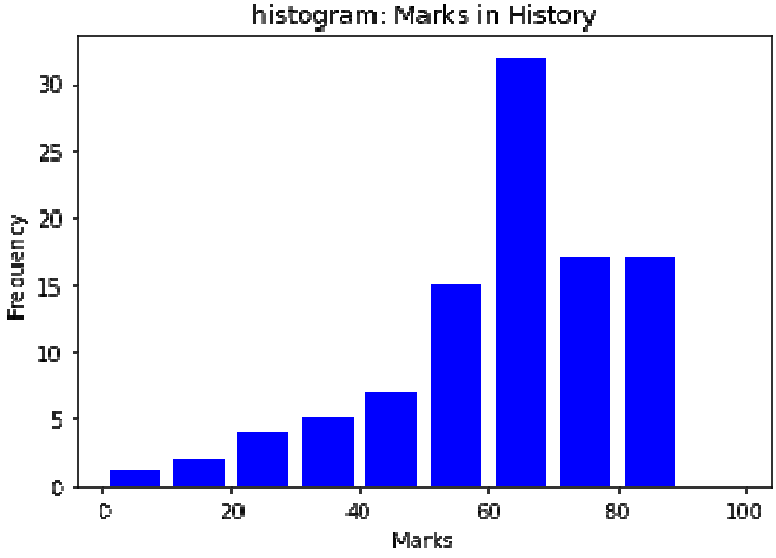
xiii) Plot the Normal Distribution

- Histogram
  A Histogram visualizes the distribution of data over a continuous interval
  Each bar in a histogram represents the tabulated frequency at each interval/bin
  In simple words, height represents the frequency for the respective bin (interval)

```
# setting the ranges and no. of intervals
range = (0, 100)
bins = 10

# plotting a histogram
plt.hist(df['History_Marks'], bins, range, color = 'blue',
histtype = 'bar', rwidth = 0.8)

# x-axis label
plt.xlabel('Marks')
# frequency label
plt.ylabel('Frequency')
# plot title
plt.title('histogram: Marks in History')

# function to show the plot
plt.show()
```


histogram: Marks in Math
Frequency
Marks
0
5
10
15
20
25
40
50
60
70
80
90
100

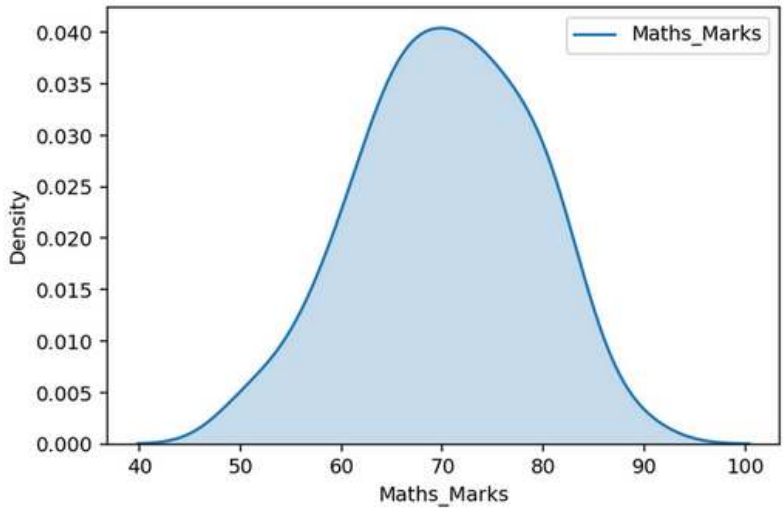

histogram: Marks in English
Frequency
Marks
0
5
10
15
20
25
60
62
64
66
68
70

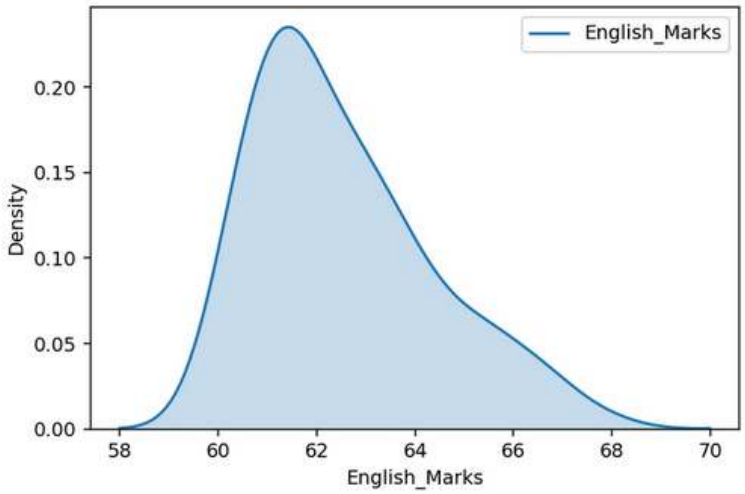

histogram: Marks in History
Frequency
Marks
0
5
10
15
20
25
30
0
20
40
60
80
100

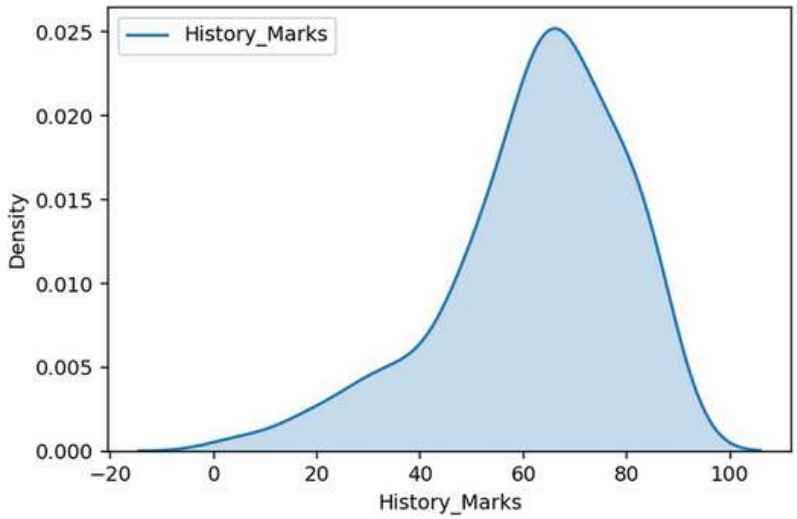
- KDE Plots

  Histogram results can vary wildly if you set different numbers of bins or simply change the start and end values of a bin. To overcome this, we can make use of the density function.

  A density plot is a smoothed, continuous version of a histogram estimated from the data. The most common form of estimation is known as kernel density estimation (KDE). In this method, a continuous curve (the kernel) is drawn at every individual data point and all of these curves are then added together to make a single smooth density estimation.

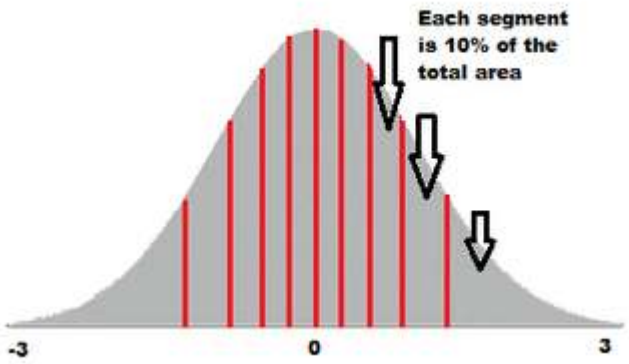




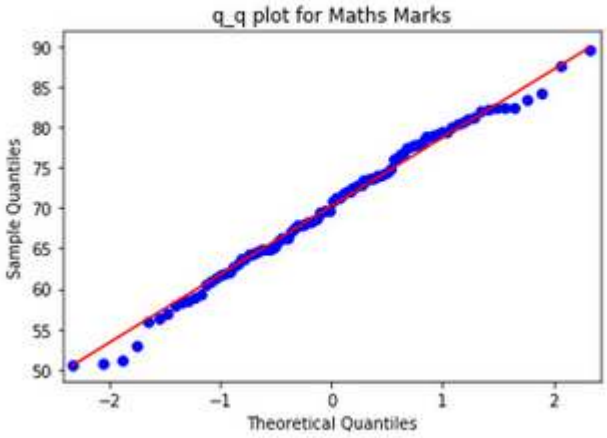


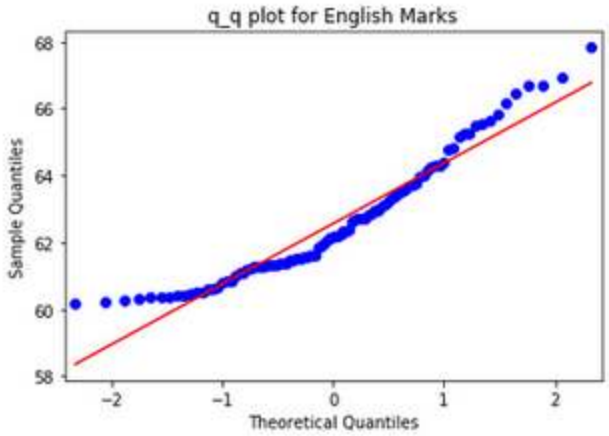
- Q_Q Plot
  Quantiles are cut points dividing the range of a probability distribution into continuous intervals with equal probabilities or dividing the observations in a sample in the same way.

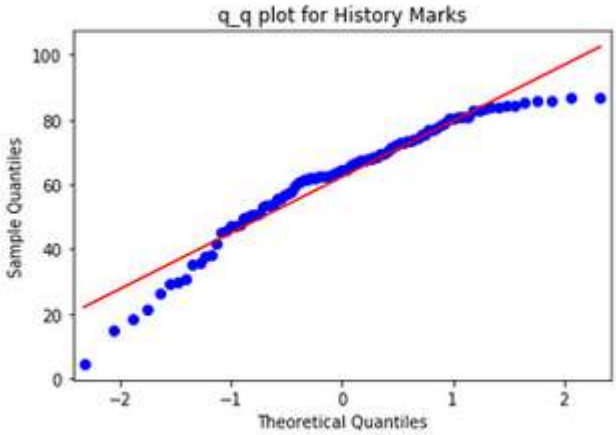
  - 2 quantile is known as the Median
  - 4 quantile is known as the Quartile
  - 10 quantile is known as the Decile
  - 100 quantile is known as the Percentile



10 quantile will divide the Normal Distribution into 10 parts each having 10 % of the data points. The **Q-Q plot** or quantile-quantile plot is a scatter plot created by plotting two sets of quantiles against one another.
Here, we will plot theoretical normal distribution quantiles and compare them against observed data quantiles:







Python Code to Understand Normal Distribution, Skewness and Kurtosis:

k) Import libraries

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import statsmodels.api as sm
        import warnings
        warnings.filterwarnings("ignore")

        # show plots inline
        %matplotlib inline
```

l) Read file using pandas

```
In [2]: df = pd.read_csv('Marks.csv')
        df.head()
```

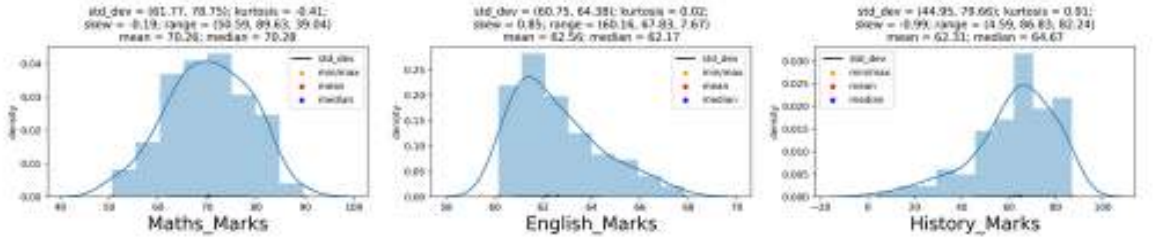
Out[2]:

	Maths_Marks	English_Marks	History_Marks
0	67.970442	60.990250	72.200954
1	58.438226	64.157607	62.687182
2	82.354172	62.998874	50.171015
3	72.909983	64.255032	62.686110
4	81.140560	64.819401	57.546319

m) Plot

```
In [3]: UVA_numeric(data):
var_group = data.columns
size = len(var_group)
plt.figure(figsize = (7*size,3), dpi = 400)

#looping for each variable
for j,i in enumerate(var_group):


    # calculating descriptives of variable
    mini = data[i].min()
    maxi = data[i].max()
    ran = data[i].max()-data[i].min()
    mean = data[i].mean()
    median = data[i].median()
    st_dev = data[i].std()
    skew = data[i].skew()
    kurt = data[i].kurtosis()

    # calculating points of standard deviation
    points = mean-st_dev, mean+st_dev

    #Plotting the variable with every information
    plt.subplot(1,size,j+1)
    sns.distplot(data[i],hist=True, kde=True)

    sns.lineplot(points, [0,0], color = 'black', label = "std_dev")
    sns.scatterplot([mini,maxi], [0,0], color = 'orange', label = "min/max")
    sns.scatterplot([mean], [0], color = 'red', label = "mean")
    sns.scatterplot([median], [0], color = 'blue', label = "median")
    plt.xlabel('{}'.format(i), fontsize = 20)
    plt.ylabel('density')
    plt.title('std_dev = {}; kurtosis = {};\nskew = {}; range = {}\nmean = {}; median = {}'.format((round(points[0
                                                                                                    round(kurt,2),
                                                                                                    round(skew,2),
                                                                                                    (round(mini,2),
                                                                                                    round(mean,2),
                                                                                                    round(median,2)
```

```
In [4]: UVA_numeric(df)
```



For Mathematics Marks, values follow the straight line indicating that they come from a Normal Distribution. On the other side for English Marks, larger values are larger as expected from a Normal Distribution and smaller values are not as small as expected from a Normal Distribution which is also the case in a right-skewed distribution.

While for History Marks, larger values are not as large as expected from a Normal Distribution and smaller values are smaller as expected from a Normal Distribution which happens to be the case in a left-skewed distribution.

Conclusion/Analysis:

Hence plotted the Normal Distribution for class test result of a particular subjects.

Program codes with sample output

Assignment Question?

(1) What is the probability that a standard normal random variable will be between the values -2 and 1?
(2) Is it likely that a standard normal random variable will have a value less than -4?
(3) What is Kurtosis?
(4) What is Leptokurtic Distribution and Platykurtic Distribution?

Assignment 3

Title

Statistical Modeling

Problem Statement:

Load the dataset: birthwt Risk Factors Associated with Low Infant Birth Weight at https://raw.github.com/neurospin/pystatsml/master/datasets/birthwt.csv

1. Test the association of mother's (bwt) age and birth weight using the correlation test and linear regression.
2. Test the association of mother's weight (lwt) and birth weight using the correlation test and linear regression.
3. Produce two scatter plot of: (i) age by birth weight; (ii) mother's weight by birth weight. Elaborate the Conclusion

Learning Objectives:

- To understand the role of computation as a tool of discovery in data analysis.
- Compute and interpret a correlation coefficient
- Compute and interpret coefficients in a linear regression analysis

Learning Outcome

Design and analyze real world engineering problems by applying various statistical modeling techniques.

Prerequisite:

- Concept of data distribution

Theory:

A correlation or simple linear regression analysis can determine if two numeric variables are significantly linearly related. For two related variables, the correlation measures the association between the two variables. In contrast, a linear regression is used for the prediction of the value of one variable from another.

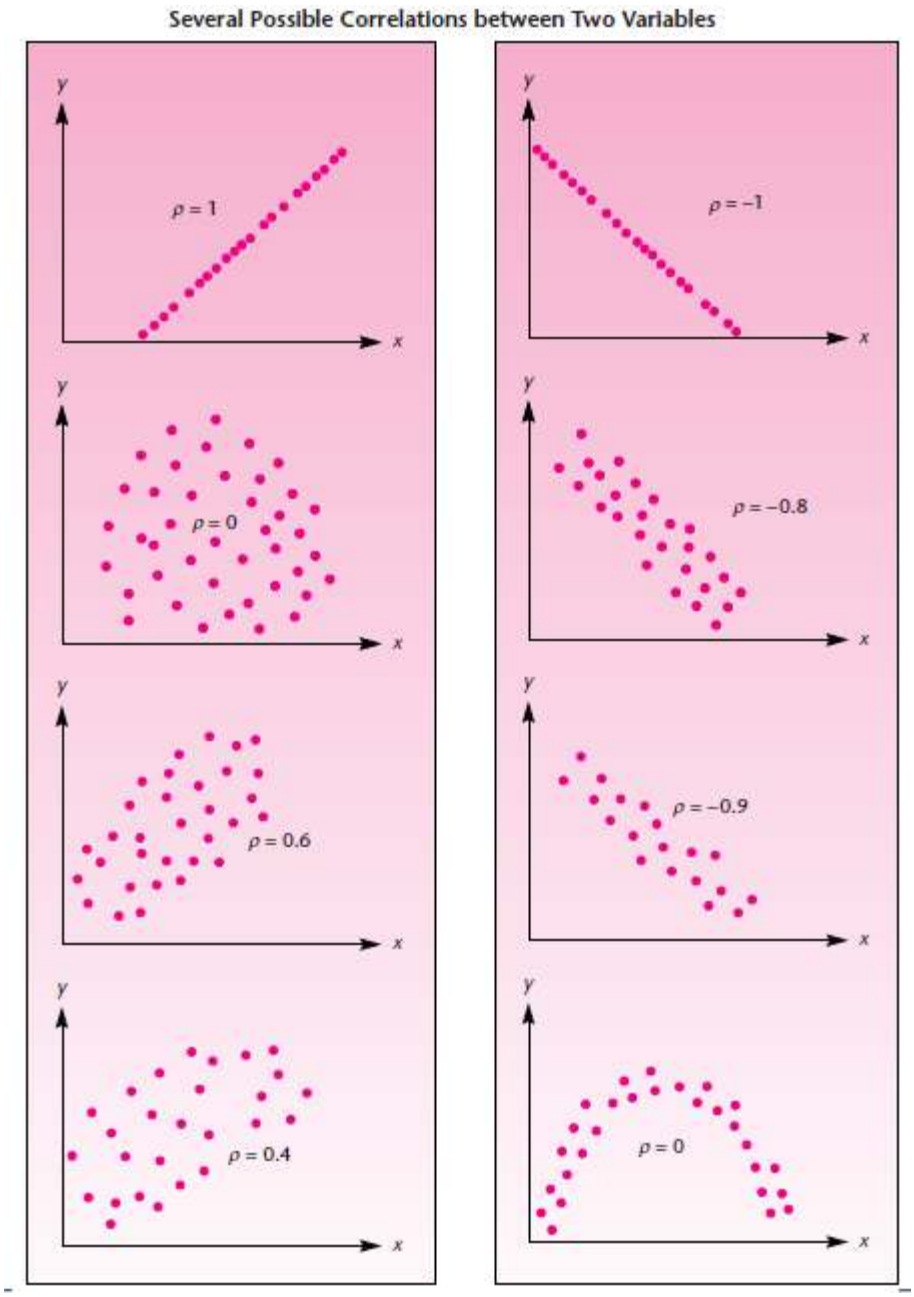
xiv) Correlation

The correlation coefficient between two variables answers the question: "Are the two variables related? That is, if one variable changes, does the other also change?" The correlation between two variables is a measure of the linear relationship between them. The correlation gives an indication of how well the two variables move together in a straight-line fashion. The correlation between X and Y is the same as the correlation between Y and X. We now define correlation more formally.

The correlation between two random variables X and Y is a measure of the degree of linear association between the two variables.

Two variables are highly correlated if they move well together. Correlation is indicated by the correlation coefficient.

The population correlation coefficient is denoted by ρ. The coefficient ρ can take on any value from -1, through 0, to 1.



The possible values of ρ and their interpretations are given below.

- When ρ is equal to zero, there is no correlation. That is, there is no linear relationship between the two random variables.
- When ρ = 1, there is a perfect, positive, linear relationship between the two variables. That is, whenever one of the variables, X or Y, increases, the other variable also increases; and whenever one of the variables decreases, the other one must also decrease.
- When ρ = -1, there is a perfect negative linear relationship between X and Y. When X or Y increases, the other variable decreases; and when one decreases, the other one must increase.

- When the value of ρ is between 0 and 1 in absolute value, it reflects the relative strength of the linear relationship between the two variables. For example, a correlation of 0.90 implies a relatively strong positive relationship between the two variables. A correlation of -0.70 implies a weaker, negative (as indicated by the minus sign), linear relationship. A correlation ρ = 0.30 implies a relatively weak (positive) linear relationship between X and Y.

In correlation analysis, we will assume that both X and Y are normally distributed random variables with means μ_X and μ_Y and standard deviations$\sigma_X$andσ_X, respectively. We define the covariance of X andY as follows:

The covariance of two random variables X and Y is

$$\text{Cov}(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$$

where μ_X is the (population) mean of X and μ_Y is the (population) mean of Y.

The population correlation coefficient or sample correlation, can take any value from -1 to +1.

The population correlation coefficient is

$$\rho = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

Like all population parameters, the value of ρ is not known to us, and we need to estimate it from our random sample of (X,Y) observation pairs. It turns out that a sample estimator of Cov(X,Y) is $SS_{XY}/(n-1)$; an estimator of σ_X is $\sqrt{SS_X/(n-1)}$and an estimator of σ_Y is $\sqrt{SS_Y/(n-1)}$. Substituting these estimators for their population counterparts in above equestion, and noting that the term n-1 cancels, we get the sample correlation coefficient, denoted by r. This estimate of ρ, also referred to as the Pearson product-moment correlation coefficient,

The sample correlation coefficient is

$$r = \frac{SS_{XY}}{\sqrt{SS_x SS_Y}}$$

xv) Linear Regression

In Recall from algebra that the equation of a straight line is $Y = A + BX$, where A is they intercept and B is the slope of the line. In simple linear regression, we model the relationship between two variables X andY as a straight line. Therefore, our model must contain two parameters: an intercept parameter and a slope parameter. The usual notation for the population intercept isβ_0, and the notation for the population slope is β_1 . If we include the error term ε, the population regression model is

The population simple linear regression model is

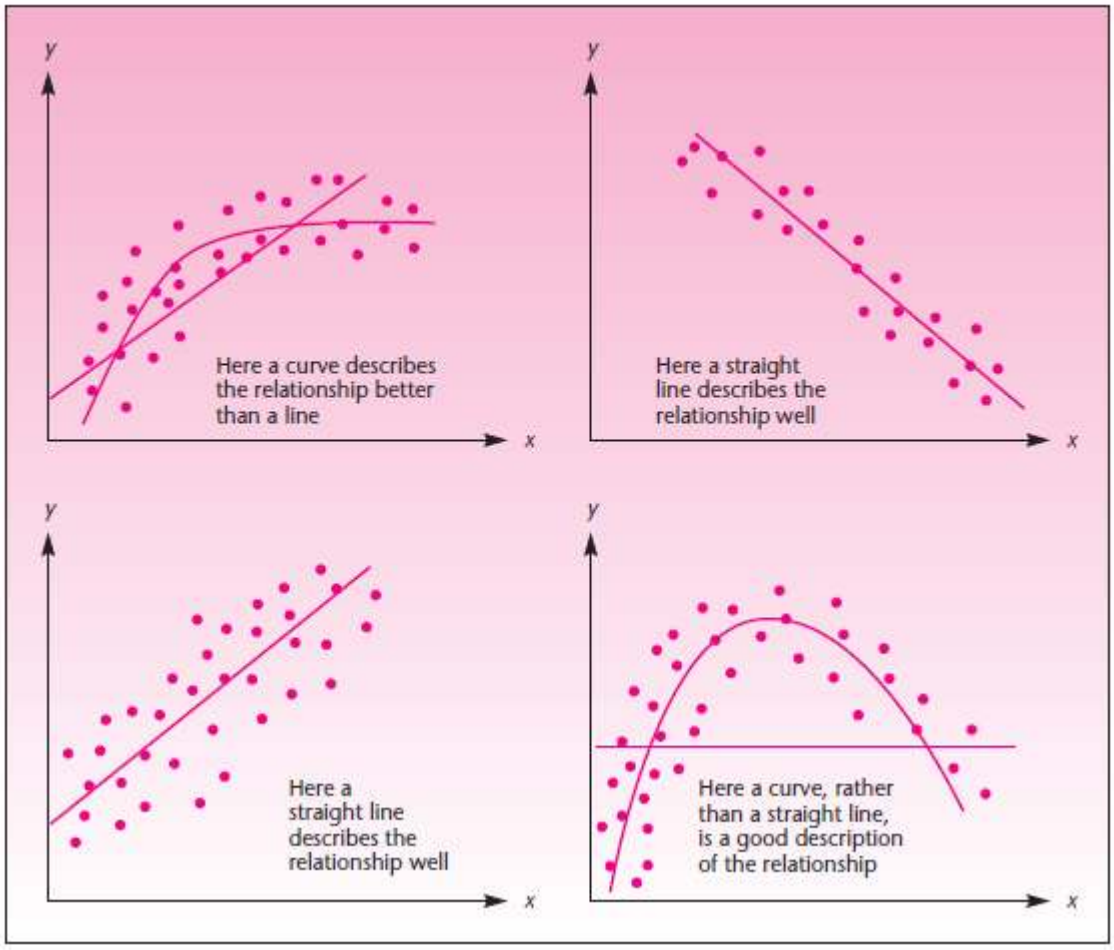
$$Y = \beta_0 + \beta_1 X + \epsilon$$

where Y is the dependent variable, the variable we wish to explain or predict; X is the independent variable, also called the *predictor* variable; and ϵ is the error term, the only random component in the model and thus the only source of randomness in Y.

The model parameters are as follows:
$\beta_0$is the Y intercept of the straight line given by $Y = \beta_0 + \beta_1 X$ (the line does not contain the error term).
$\beta_1$is the slope of the line $Y = \beta_0 + \beta_1 X$

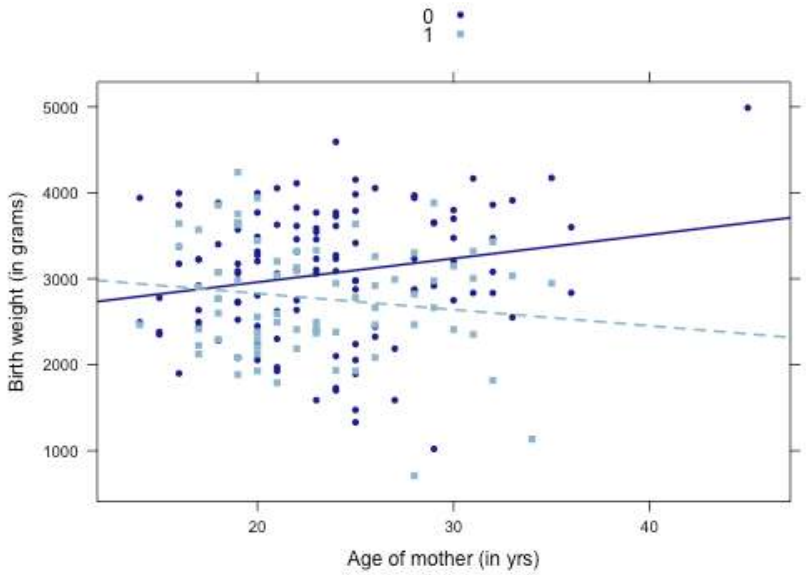
The simple linear regression model applies only if the true relationship between the two variables X and Y is a straight-line relationship. If the relationship is curved (curvilinear)

Some Possible Relationships between *X* and *Y*



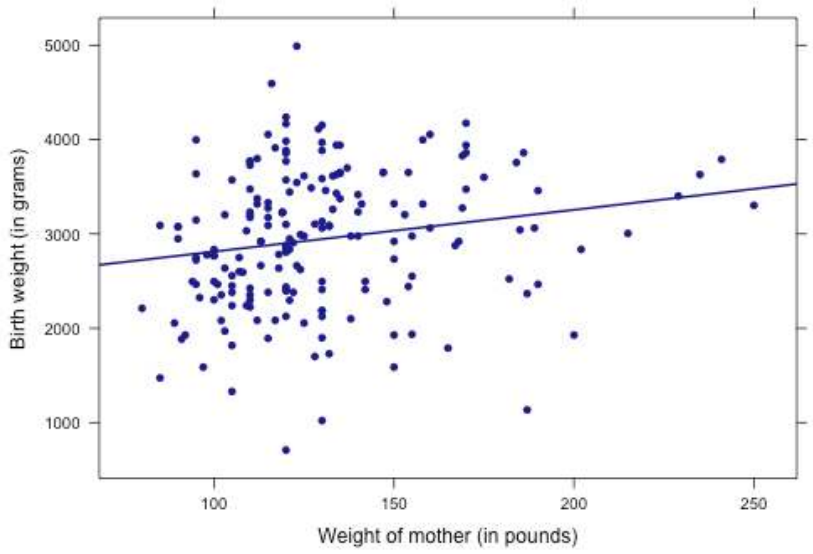
Python Code to Understand Correlation and regression:

```
xyplot(birthweight ~ age, groups = smoke, xlab = "Age of mother (in yrs)", ylab = "Birth weight (in grams)",
    data = LBW, auto.key = TRUE, type = c("p", "r"))
```



There is a relatively weak linear relationship between the birth weight and the age of mother. This can be seen in the scatterplot by the relatively flat slope of the regression line. Also, the correlation between \(birthweight \) and \(age \) is 0.09, indicating a relatively weak association between the two variables. Finally, there is some concern about the rightmost point in the plot, which represents a large baby born to an older mother. This may be a point of high leverage, given its distance from the mean age of the mothers and its departure from the general pattern of the data.

```
xyplot(birthweight ~ momweight, xlab = "Weight of mother (in pounds)", ylab = "Birth weight (in grams)",
    data = LBW, type = c("p", "r"))
```



The coefficients suggest that each pound of a mother's weight is associated with an additional 4 grams in the expected birth weight of her child.

Conclusion/Analysis:

Hence correlation and linear regression for the given dataset birthwt Risk Factors Associated with Low Infant Birth Weight calculated and produced the scatter plot

Program codes with sample output

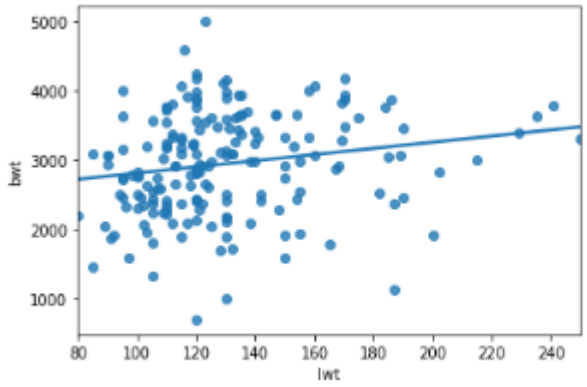
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline

url = 'https://github.com/duchesnay/pystatsml/raw/master/datasets/birthwt.csv'
df = pd.read_csv(url)

print(df.head())
```

```
   low  age  lwt  race  smoke  ptl  ht  ui  ftv   bwt
0    0   19  182     2      0    0   0   1    0  2523
1    0   33  155     3      0    0   0   0    3  2551
2    0   20  105     1      1    0   0   0    1  2557
3    0   21  108     1      1    0   0   1    2  2594
4    0   18  107     1      1    0   0   1    0  2600
```

```
# association of mother's (bwt) age and birth weight using the correlation test
# and linear regression

sb.regplot(x = 'age', y = 'bwt', ci = None, data = df)
```

```
# association of mother's weight (lwt) and birth weight using the correlation test
# and linear regression

sb.regplot(x = 'lwt', y = 'bwt', ci = None, data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fabc4a0d090>
```



Assignment Question?

(1) If two variables are correlated are they causally related?
(2) How do I test the assumptions underlying linear regression?

Assignment 4

Title

Statistical Processing

Problem Statement:

Load Apply Basic PCA on the iris dataset. The data set is available at: https://raw.github.com/neurospin/pystatsml/master/datasets/iris.csv

- Describe the data set. Should the dataset been standardized?
- Describe the structure of correlations among variables.
- Compute a PCA with the maximum number of components.
- Compute the cumulative explained variance ratio. Determine the number of components K by your computed values.
- Print the K principal components directions and correlations of the$K$principal components with the original variables. Interpret the contribution of the original variables into the PC.
- Plot the samples projected into the$K$first PCs.
- Color samples by their species.

Learning Objectives:

To be able to appropriately apply computational methodologies to real world statistical problems.

Learning Outcome

Model and solve computing problem using correlation, and resampling using appropriate statistics algorithms.

Prerequisite:

- Concept of data pre-processing

Theory:

PCA finds the principal components of data.

It is often useful to measure data in terms of its principal components rather than on a normal x-y axis. So what are principal components then? They're the underlying structure in the data. They are the directions where there is the most variance, the directions where the data is most spread out.

Principal Component Analysis (PCA) is an unsupervised dimensionality reduction algorithm. It identifies the hyperplane that lies closest to the data, and then it projects the data onto it preserving the variance.

PCA finds a new set of dimensions (or a set of basis of views) such that all the dimensions are orthogonal (and hence linearly independent) and ranked according to the variance of data along them. It means more important principle axis occurs first. (more important = more variance/more spread out data)

How does PCA work -

- Calculate the covariance matrix X of data points.
- Calculate eigen vectors and corresponding eigen values.
- Sort the eigen vectors according to their eigen values in decreasing order.
- Choose first k eigen vectors and that will be the new k dimensions.
- Transform the original n dimensional data points into k dimensions.

Implementing PCA on a 2-D Dataset

- Step 1: Normalize the data
  First step is to normalize the data that we have so that PCA works properly. This is done by subtracting the respective means from the numbers in the respective column. So if we have two dimensions X and Y, all X become $\boldsymbol{x}$- and all Y become $\boldsymbol{y}$-. This produces a dataset whose mean is zero.
- Step 2: Calculate the covariance matrix
  Since the dataset we took is 2-dimensional, this will result in a 2x2 Covariance matrix.

$$Matrix(Covariance) = \begin{bmatrix} Var[X_1] & Cov[X_1, X_2] \\ Cov[X_2, X_1] & Var[X_2] \end{bmatrix}$$

  Var[X1] = Cov[X1,X1] and Var[X2] = Cov[X2,X2].
- Step 3: Calculate the eigenvalues and eigenvectors
  Next step is to calculate the eigenvalues and eigenvectors for the covariance matrix. The same is possible because it is a square matrix. ƛ is an eigenvalue for a matrix A if it is a solution of the characteristic equation:
  det(ƛI - A) = 0
  Where, I is the identity matrix of the same dimension as A which is a required condition for the matrix subtraction as well in this case and 'det' is the determinant of the matrix. For each eigenvalue ƛ, a corresponding eigen-vector v, can be found by solving:
  (ƛI - A)v = 0
- Step 4: Choosing components and forming a feature vector
  We order the eigenvalues from largest to smallest so that it gives us the components in order or significance. Here comes the dimensionality reduction part. If we have a dataset with n variables, then we have the corresponding n eigenvalues and eigenvectors. It turns out that the eigenvector corresponding to the highest eigenvalue is the principal component of the dataset and it is our call as to how many eigenvalues we choose to proceed our analysis with. To reduce the dimensions, we choose the first p eigenvalues and ignore the rest. We do lose out some information in the process, but if the eigenvalues are small, we do not lose much.

Python Code to Understand Correlation and regression:

Download the Dataset from link:
https://raw.github.com/neurospin/pystatsml/master/datasets/iris.csv

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

%matplotlib inline
```

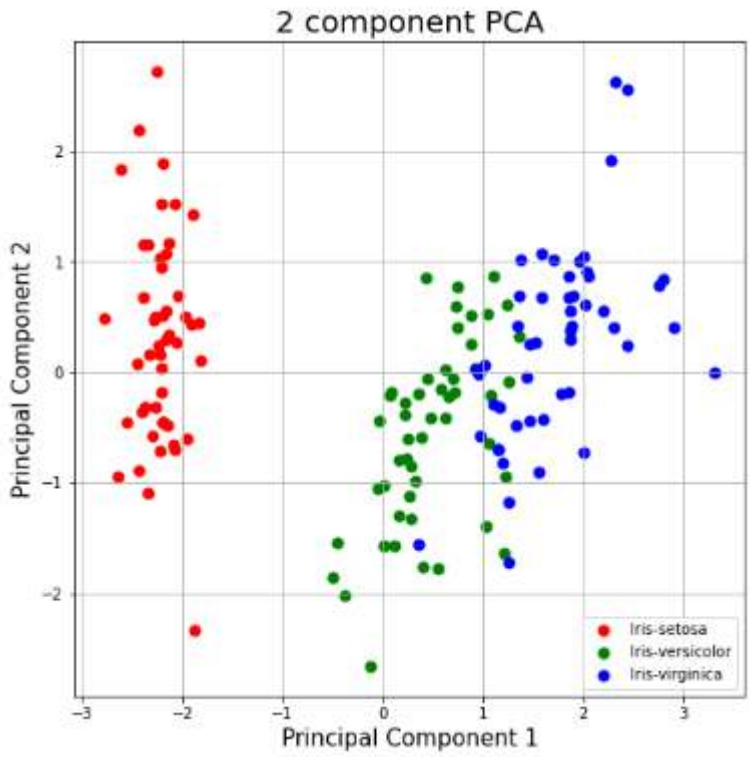
```
[ ] url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

    # load dataset into Pandas DataFrame
    df = pd.read_csv(url, names=['sepal length','sepal width','petal length','petal width','target'])
```

```
[ ] features = ['sepal length', 'sepal width', 'petal length', 'petal width']

    # Separating out the features
    x = df.loc[:, features].values

    # Separating out the target
    y = df.loc[:,['target']].values

    # Standardizing the features
    x = StandardScaler().fit_transform(x)
```

```
[ ] pca = PCA(n_components=2)
    principalComponents = pca.fit_transform(x)
    principalDf = pd.DataFrame(data = principalComponents
                 , columns = ['principal component 1', 'principal component 2'])
```

```
[ ] finalDf = pd.concat([principalDf, df[['target']]], axis = 1)
```

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



Conclusion/Analysis:

Hence we conclude that, Principal Component Analysis (PCA) is useful for visualizing high-dimensional datasets, as it can compress it down to 2 dimensions.

Program codes with sample output

Test cases

(Ex. Without PCA & With PCA)

Test data set

Comparative/complexity analysis

Assignment Question?

(1) Why do we need dimensionality reduction? What are its drawbacks?
(2) How can you evaluate the performance of a dimensionality reduction algorithm on your dataset?
(3) Is it important to standardize before applying PCA?
(4) Is rotation necessary in PCA? If yes, Why? What will happen if you don't rotate the components?

Assignment 5

Title

Gaussian mixture models clustering

Problem Statement:

Perform clustering of the iris dataset based on all variables using Gaussian mixture models. Use PCA to visualize clusters.

Learning Objectives:

To learn the data processing techniques required to get applied on machine learning algorithms.

Learning Outcome

Formulate suitable statistical method required as pre-processing technique for finding the solution of machine learning algorithm.

Prerequisite:

- Concept of clustering

Theory:

The goal of clustering is to find groups that share similar properties. The data in each group should be similar (minimiseintracluster distance), but each cluster should be sufficiently different (maximiseintercluster similarity).

1. Gaussian mixture model (GMM)

   The GMM is a simple but powerful model that performs clustering via density estimation. The features' histogram is modelled as the sum of multiple multivariate Gaussian distributions.

   In one dimension the probability density function of a Gaussian Distribution is given by

$$G(X|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

   Where μ and σ^2 are respectively mean and variance of the distribution.

   For Multivariate Gaussian Distribution, the probability density function is given by

$$G(X|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(X-\mu)^T \boldsymbol{\Sigma}^{-1}(X-\mu)\right)$$

   Here μ is a d dimensional vector denoting the mean of the distribution and ∑ is the d X d covariance matrix.

The parameters cannot be estimated in closed form. This is where Expectation-Maximization algorithm is beneficial.

2. Expectation-Maximization (EM) Algorithm

The Expectation-Maximization (EM) algorithm is an iterative way to find maximum-likelihood estimates for model parameters when the data is incomplete or has some missing data points or has some hidden variables. EM chooses some random values for the missing data points and estimates a new set of data. These new values are then recursively used to estimate a better first data, by filling up missing points, until the values get fixed.

These are the two basic steps of the EM algorithm, namely E Step or Expectation Step or Estimation Step and M Step or Maximization Step.

- Estimation step:
  - Initialize μ_k , $\sum_k$, and π_k by some random values, or by K means clustering results or by hierarchical clustering results.
  - Then for those given parameter values, estimate the value of the latent variables (i.e ϑ_k)
- Maximization Step:
  - Update the value of the parameters(i.e. μ_k , $\sum_k$, and π_k) calculated using ML method.

Algorithm

1. Initialize the mean μ_k,
   the covariance matrix $\sum_k$ and
   the mixing coefficients π_k
   by some random values. (or other values)
2. Compute the γ_k values for all k.
3. Again Estimate all the parameters
   using the current γ_k values.
4. Compute log-likelihood function.
5. Put some convergence criterion
6. If the log-likelihood value converges to some value
   (or if all the parameters converge to some values)
   then **stop**,
   else return to **Step 2**.

Download the Dataset from link:
https://raw.github.com/neurospin/pystatsml/master/datasets/iris.csv

Conclusion/Analysis:

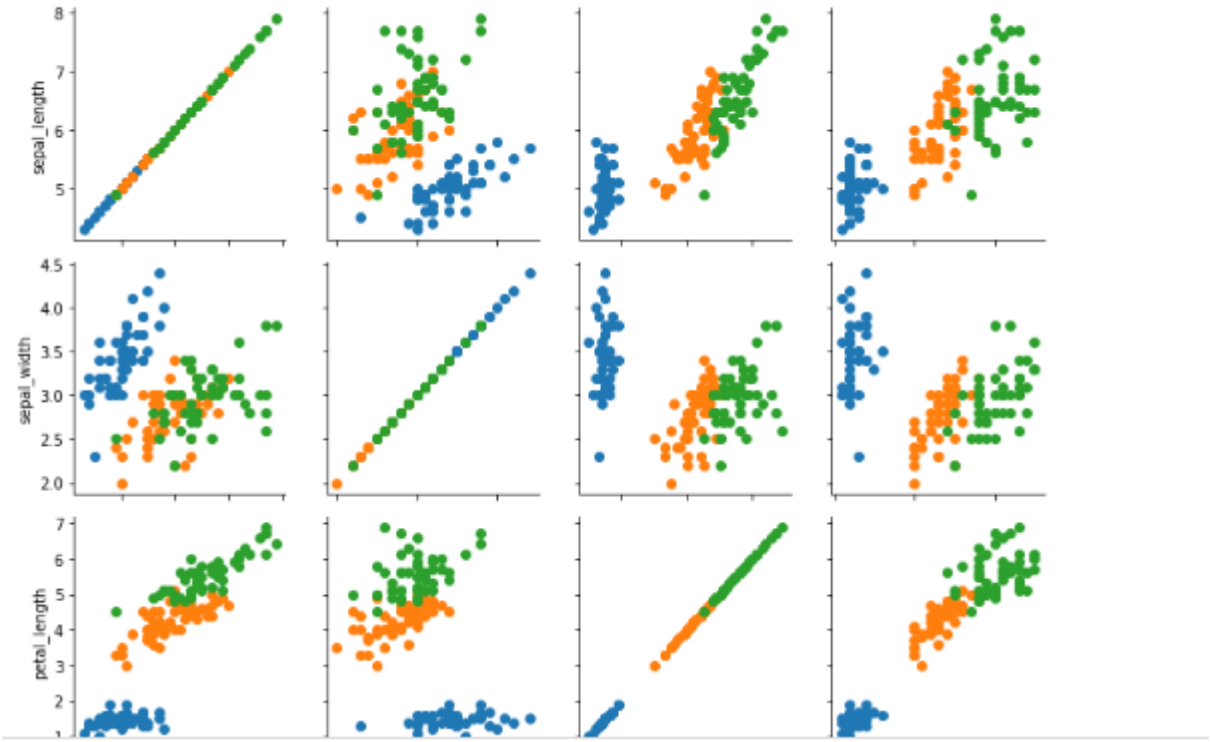
Gaussian Mixture Model (GMM) Clustering handles ellipsoidal distributions, and makes 'soft' assignments to clusters

Program codes with sample output

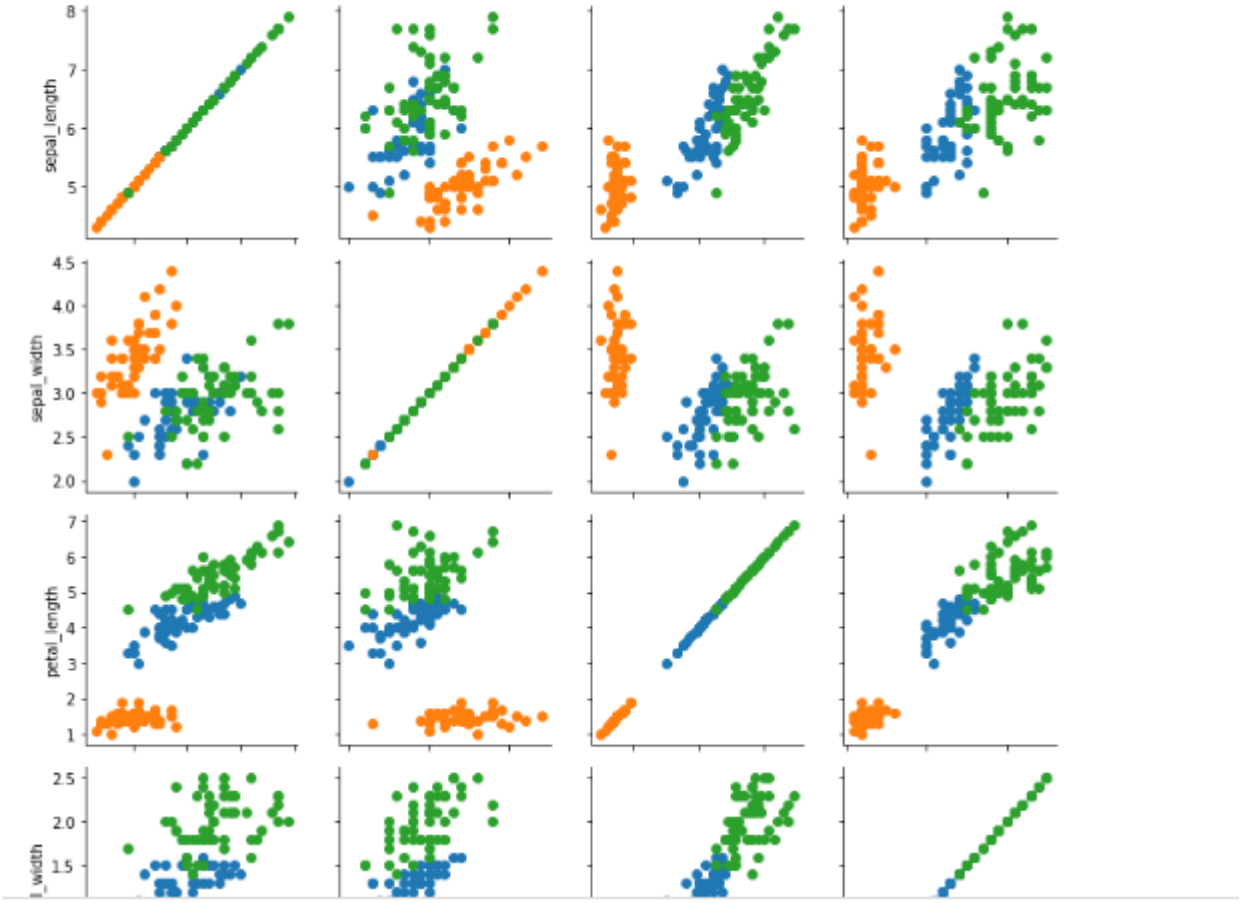
```
[ ] import seaborn as sns
    from sklearn import mixture
    import matplotlib.pyplot as plt

    iris = sns.load_dataset("iris")

    iris.head()
```

	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

```
g = sns.PairGrid(iris, hue="species", vars=['sepal_length','sepal_width','petal_length','petal_width'])
g.map(plt.scatter)
plt.show()
```



```
iris['gmm_pred']=pred_gmm_iris
g = sns.PairGrid(iris, hue="gmm_pred", vars=['sepal_length','sepal_width','petal_length','petal_width'])
g.map(plt.scatter)
plt.show()
```




```
[ ]  iris['gmm_pred'] = pred_gmm_iris

     # TODO: calculate adjusted rand score passing in the original
     # labels and the GMM predicted labels iris['species']
     iris_gmm_score = metrics.adjusted_rand_score(iris['species'],pred_gmm_iris)

     # Print the score
     iris_gmm_score

     0.9038742317748124
```

Test data set
Comparative/complexity analysis
(K-means with GMM)

Assignment Question?

(1) What is use of clustering?
(2) Does the EM algorithm is guaranteed to never decrease the value of its objective function on any iteration?