EXP NO:1 **STUDY ON PYTHON PACKAGES**

25/01/25

AIM:

To study about python packages required to work with Data Analytics (Numpy, Pandas, Statsmodels, Scipy)

NUMPY:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

```
import numpy as np
arr = np.array([5, 10, 15])
print('First array is:')
print(arr)
print('\nApplying power fn:')
print(np.power(arr, 2))
print('\nSecond array is:')
arr1 = np.array([1, 2, 3])
print(arr1)
print('\nApplying power function again:')
print(np.power(arr, arr1))
```

OUPUT:

```
First array is:
[ 5 10 15]

Applying power fn:
[ 25 100 225]

Second array is:
[1 2 3]

Applying power function again:
[   5  100 3375]
```

PANDAS:

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.

- import pandas as pd

```
lst=['sai', 'saber', 'ashwin', 'madhav']

df = pd.DataFrame(lst)

print(df)
```

- import pandas as pd

```
technologies = {

'Courses':["WT","CN","CE","PR",None],

 'Fee' :[20000,25000,22000,None,30000],
```

'Duration':['30days','40days','35days','None','50days'],

'Discount':[1000,2300,1200,2000,None]}

index_labels=['r1','r2','r3','r4','r5']

df = pd.DataFrame(technologies,index=index_labels)

print(df)

OUTPUT:

```
         0
0     sai
1   saber
2  ashwin
3  madhav
```

```
   Courses      Fee Duration  Discount
r1      WT  20000.0   30days    1000.0
r2      CN  25000.0   40days    2300.0
r3      CE  22000.0   35days    1200.0
r4      PR      NaN     None    2000.0
r5    None  30000.0   50days       NaN
```

STATSMODELS:

Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license.

SCIPY:

SciPy is a python library that is useful in solving many mathematical equations and algorithms. It is designed on the top of Numpy library that gives more extension of finding scientific mathematical formulae like Matrix Rank, Inverse, polynomial equations, LU Decomposition, etc. Using its high level functions will significantly reduce the complexity of the code and helps in better analyzing the data. SciPy is an interactive Python session used as a data-processing library that is made to compete with its rivalries such as MATLAB, Octave, R-Lab,etc. It has many user-friendly, efficient and easy-to-use functions that helps to solve problems like numerical integration, interpolation, optimization, linear algebra and statistics.

- from scipy import linalg

```
import numpy as np
two_d_array = np.array([ [4,5], [3,2] ])
linalg.det( two_d_array )
```

- from scipy import poly1d

```
poly=poly1d([1,3,9]) #creating the polynomial
print("The polynomial is:\n",poly)
print("The value of the polynomial for the value of x=3 is:\n",poly(3))
print("The product of the polynomial with itself is:\n",poly*poly)
print("The integration of the polynomial is:\n",poly.integ(k=5))
print("The differentiation of the polynomial is:\n",poly.deriv())
```

OUTPUT:

```
Out[12]:  -7.0
```

```
The polynomial is:
    2
1 x + 3 x + 9
The value of the polynomial for the value of x=3 is:
 27
The product of the polynomial with itself is:
    4     3      2
1 x + 6 x + 27 x + 54 x + 81
The integration of the polynomial is:
         3       2
0.3333 x + 1.5 x + 9 x + 5
The differentiation of the polynomial is:

2 x + 3
```

EXP NO:2 **DESCRIPTIVE ANALYSIS**

08/02/23

AIM:

To explore the various commands for performing descriptive data analysis on various dataset.

CODE:

- import pandas as pd

```
df = pd.read_csv("Iris.csv")
df.head()
```

Out[14]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

- df.info()

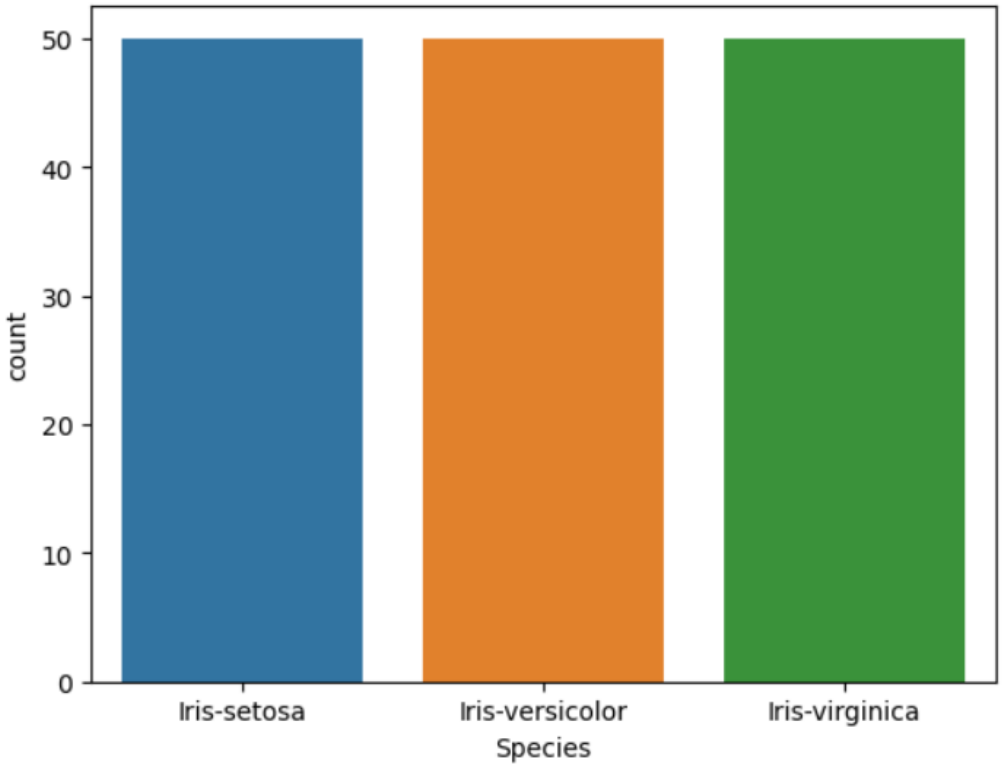
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

- import seaborn as sns

import matplotlib.pyplot as plt

sns.countplot(x='Species', data=df,)

plt.show()



- df.isnull().sum()/len(df)*100

```
Out[5]: Id              0.0
        SepalLengthCm   0.0
        SepalWidthCm    0.0
        PetalLengthCm   0.0
        PetalWidthCm    0.0
        Species         0.0
        dtype: float64
```

K-squared test

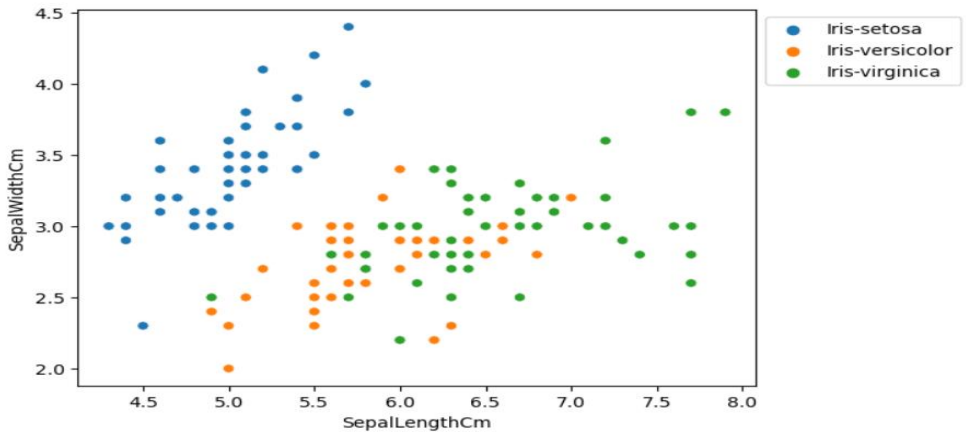
- from scipy import stats

```
print(["PetalWidthCm"])
a, b = stats.normaltest(df[["PetalWidthCm"]])
print(a, b)
alpha = 0.05
if b < alpha:
 print("Null hypotheses can be rejected")
else:
 print("Null hypotheses cannot be rejected")
```

```
['PetalWidthCm']
[136.77701788] [1.99181015e-30]
Null hypotheses can be rejected
```

- import seaborn as sns

```
sns.scatterplot(x='SepalLengthCm',y='SepalWidthCm',hue='Species',
data=df, )
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```



- import pandas as pd

df = pd.read_csv("heartdisease.csv")

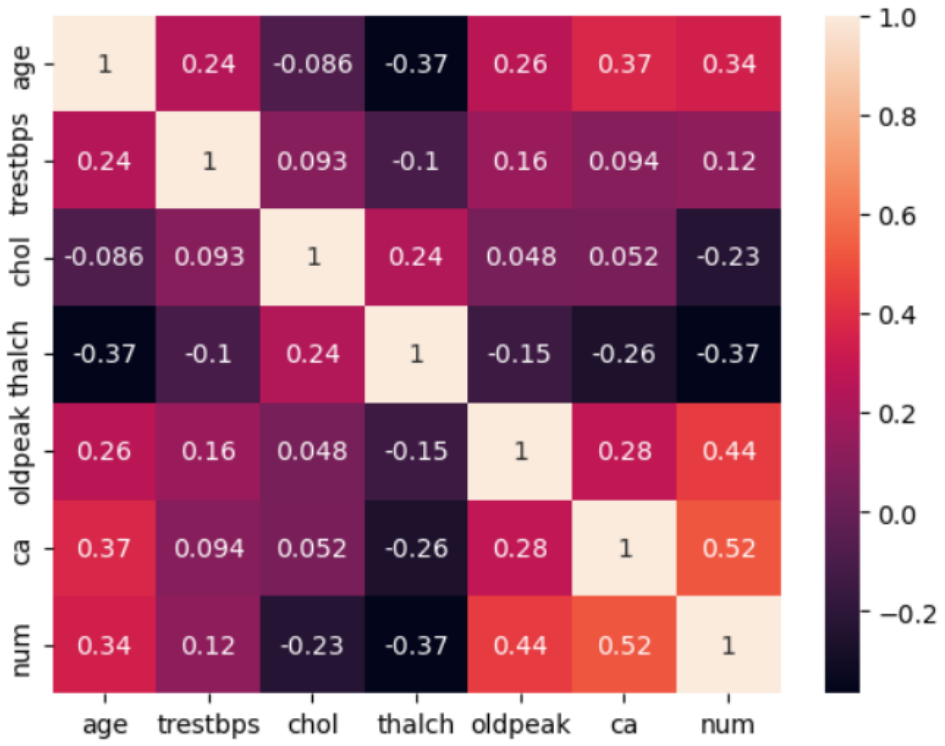
df.head()

Out[1]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	2.0	reversable defect	1
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0.0	normal	0
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0.0	normal	0

- sns.heatmap(df.corr(method='pearson').drop(['id'], axis=1).drop(['id'], axis=0), annot = True);

plt.show()



- sum_data = df["oldpeak"].sum()

mean_data = df["oldpeak"].mean()

median_data = df["oldpeak"].median()

mode_data = df["oldpeak"].mode()

print("Sum:",sum_data, "\nMean:", mean_data, "\nMedian:",median_data, "\nMode:" ,mode_data)

```
Sum: 754.0
Mean: 0.8787878787878787
Median: 0.5
Mode: 0    0.0
Name: oldpeak, dtype: float64
```

- df.skew()

```
Out[3]: id          0.000000
        age        -0.195994
        trestbps    0.213334
        chol       -0.613836
        fbs         1.795987
        thalch     -0.211119
        exang       0.453582
        oldpeak     1.041427
        ca          1.165978
        num         0.968880
        dtype: float64
```

- df.kurt()

```
Out[4]: id          -1.200000
        age         -0.382930
        trestbps     2.958664
        chol         0.062273
        fbs          1.228523
        thalch      -0.479725
        exang       -1.798427
        oldpeak      1.127069
        ca           0.199498
        num         -0.104325
        dtype: float64
```

RESULT:

Thus, the various commands for performing descriptive data analysis on the Iris Dataset have been executed and verified.

EXP NO:3 **UNI, BI AND MULTI-VARIATE ANALYSIS**

08/02/23

AIM:

To perform uni, bi and multi-variate analysis on the Diabetes dataset.

CODE:

- import pandas as pd

```
df = pd.read_csv("diabetes1.csv")
df.head()
```

Out[1]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

- df.shape

```
Out[2]: (768, 9)
```

- df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

- df.describe()

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

- df.isnull().sum()

```
Out[5]: Pregnancies                 0
        Glucose                     0
        BloodPressure               0
        SkinThickness               0
        Insulin                     0
        BMI                         0
        DiabetesPedigreeFunction    0
        Age                         0
        Outcome                     0
        dtype: int64
```

- df.value_counts("Outcome")

```
Out[7]: Outcome
        0    500
        1    268
        dtype: int64
```

- sum_data = df["SkinThickness"].sum()

mean_data = df["SkinThickness"].mean()

median_data = df["SkinThickness"].median()

mode_data = df["SkinThickness"].mode()

print("Sum:",sum_data, "\nMean:", mean_data, "\nMedian:",median_data, "\nMode:" ,mode_data)

```
Sum: 15772
Mean: 20.536458333333332
Median: 23.0
Mode: 0    0
Name: SkinThickness, dtype: int64
```

- df.skew()

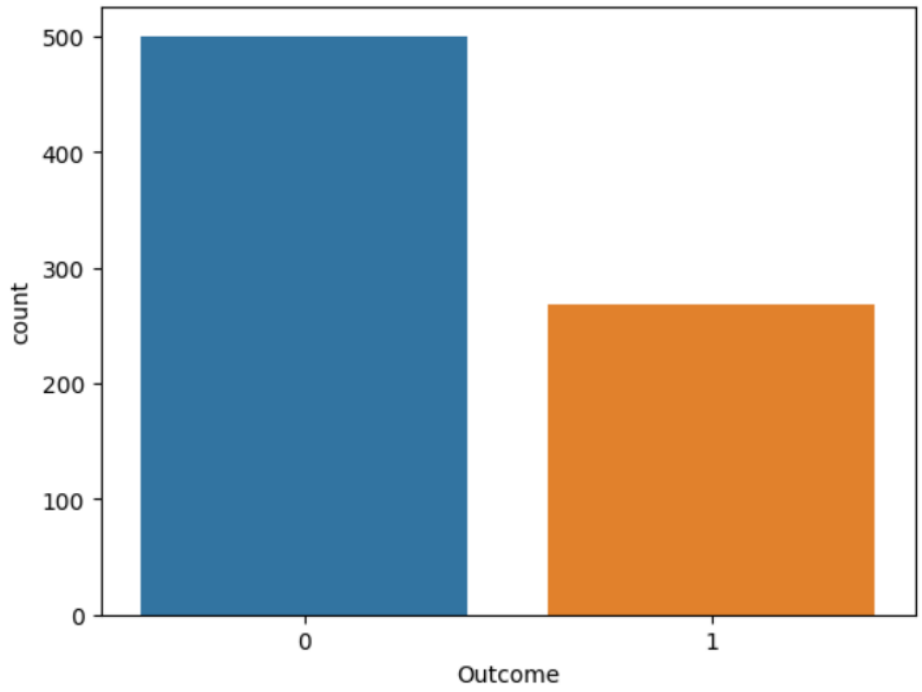
```
Out[26]: Pregnancies                0.901674
         Glucose                    0.173754
         BloodPressure             -1.843608
         SkinThickness              0.109372
         Insulin                    2.272251
         BMI                       -0.428982
         DiabetesPedigreeFunction   1.919911
         Age                        1.129597
         Outcome                    0.635017
         dtype: float64
```

- df.kurt()

```
Out[28]: Pregnancies                0.159220
         Glucose                    0.640780
         BloodPressure              5.180157
         SkinThickness             -0.520072
         Insulin                    7.214260
         BMI                        3.290443
         DiabetesPedigreeFunction   5.594954
         Age                        0.643159
         Outcome                   -1.600930
         dtype: float64
```

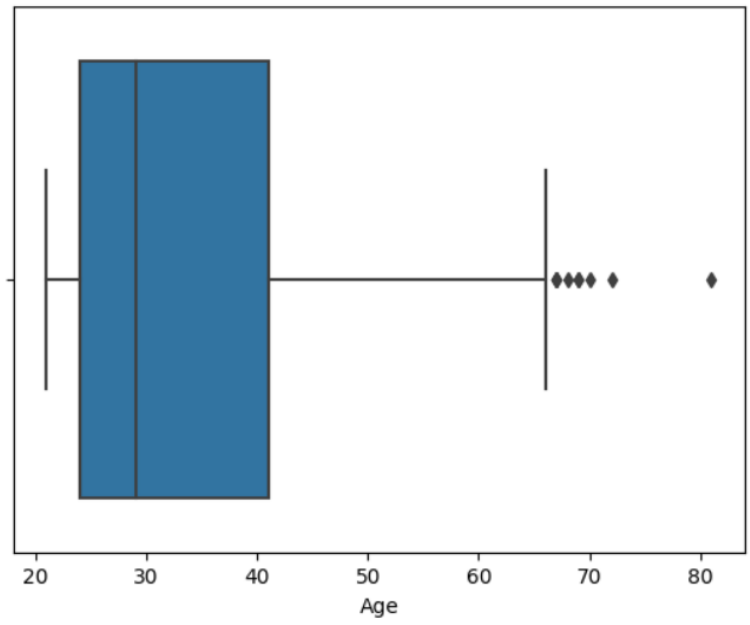
- import seaborn as sns

```
import matplotlib.pyplot as plt
sns.countplot(x='Outcome', data=df, )
plt.show()
```



- sns.boxplot(x='Age', data=df)

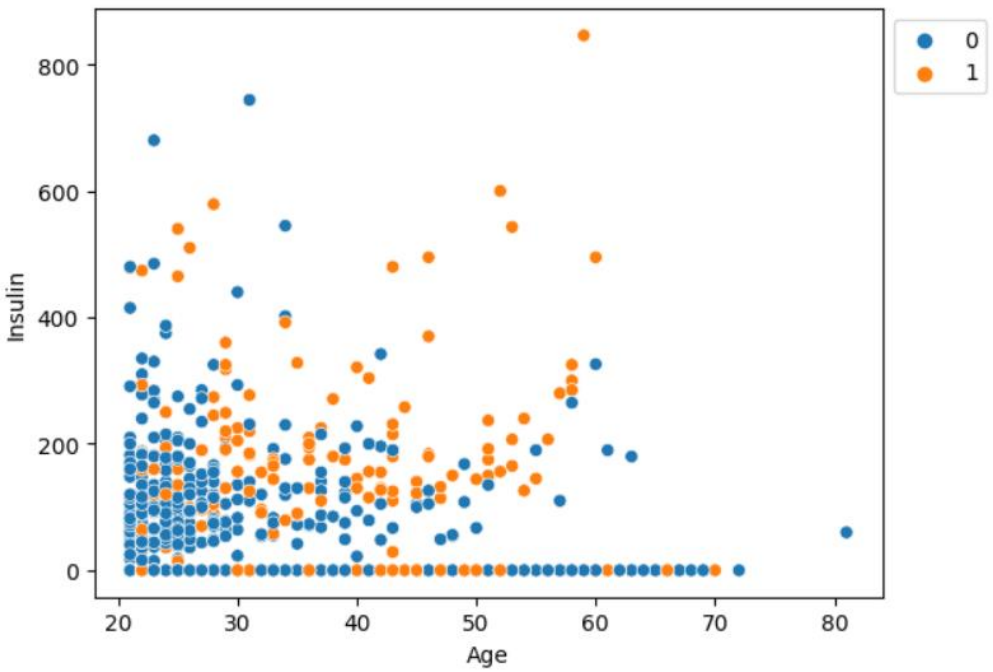
```
Out[12]: <AxesSubplot: xlabel='Age'>
```



- sns.scatterplot(x='Age', y='Insulin', hue='Outcome', data=df,)

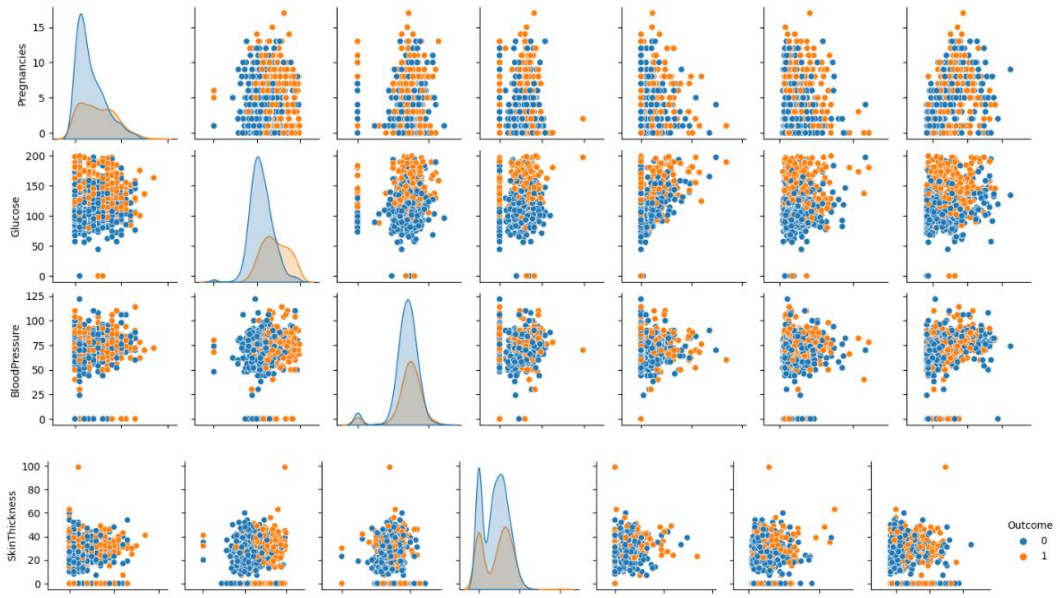
plt.legend(bbox_to_anchor=(1, 1), loc=2)

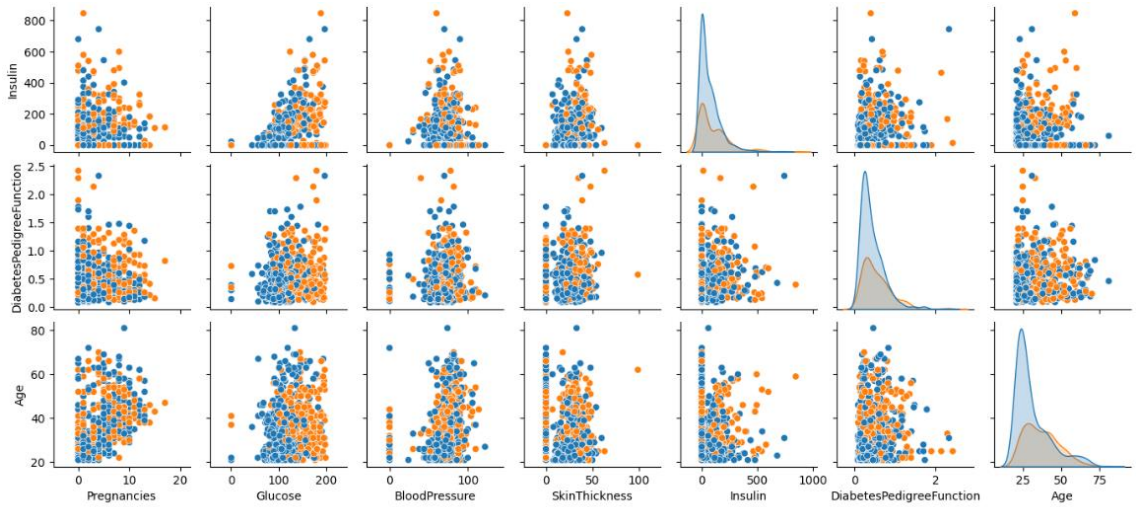
plt.show()



- sns.pairplot(df.drop(['BMI'], axis = 1), hue='Outcome', height=2)

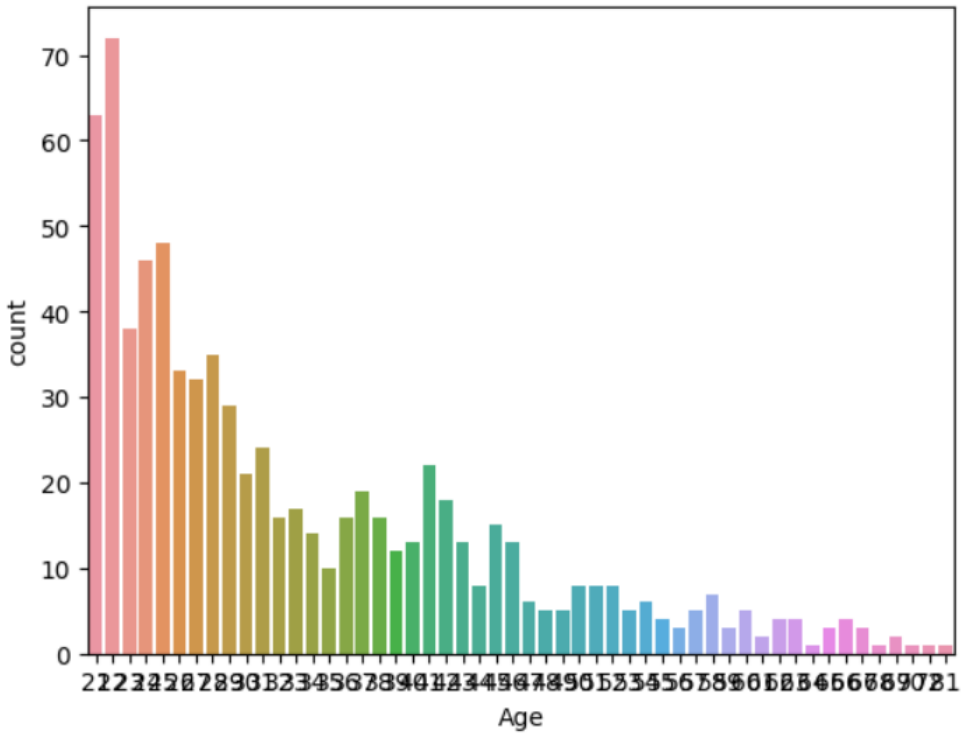
Out[15]: <seaborn.axisgrid.PairGrid at 0x265e50e3190>





- sns.countplot(x="Age", data=df,)

plt.show()



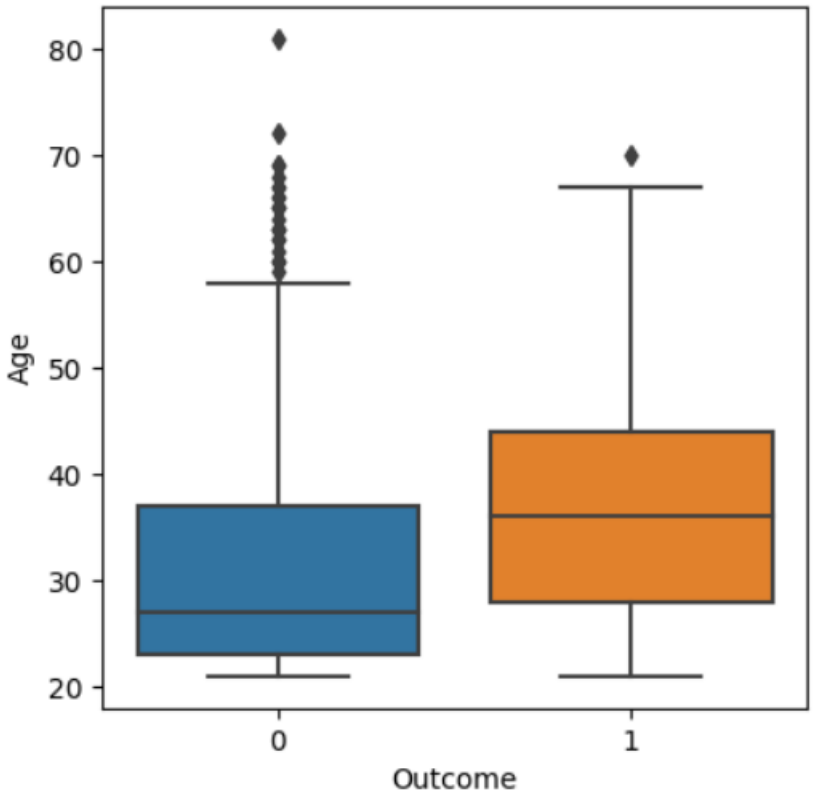
- def graph(y):

sns.boxplot(x="Outcome", y=y, data=df)

plt.figure(figsize=(10,10))

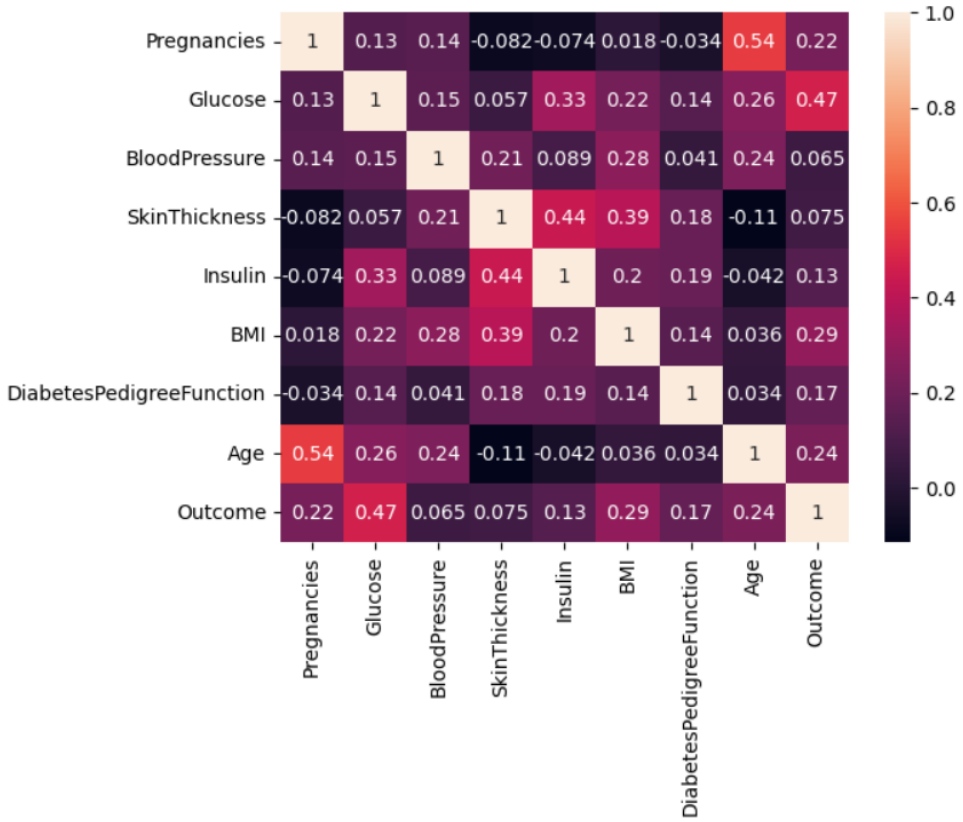
plt.subplot(221)

graph("Age")



- sns.heatmap(df.corr(method='pearson'), annot = True);

plt.show()



RESULT:

Thus, uni, bi and multi-variate analysis on the Diabetes dataset has been performed and verified.

EXP NO:3 **REGRESSIONS**

22/02/23

AIM:

To perform various types of Regression on the iris and diabetes dataset.

CODE:

1. LINEAR REGRESSION:

- import pandas as pd

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
```

- df = pd.read_csv('iris1.csv')

```
df.head()
```

Out[24]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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

- df['variety'].value_counts()

```
Out[25]: Setosa        50
         Versicolor    50
         Virginica     50
         Name: variety, dtype: int64
```

- def getPredictions(flowerValues, xLabel, yLabel):

```
X = np.array(flowerValues[xLabel]).reshape(-1,1)
y = np.array(flowerValues[yLabel]).reshape(-1,1)
reg = LinearRegression().fit(X, y)
y_pred = reg.predict(X)
return y_pred
```

- setosa = df[df['variety'] == 'Setosa']

```
setosa = setosa.loc[:, setosa.columns != 'variety']
setosa.columns = ['sepalLength', 'sepalWidth', 'petalLength', 'petalWidth']
setosa.head()
```

Out[27]:

	sepalLength	sepalWidth	petalLength	petalWidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

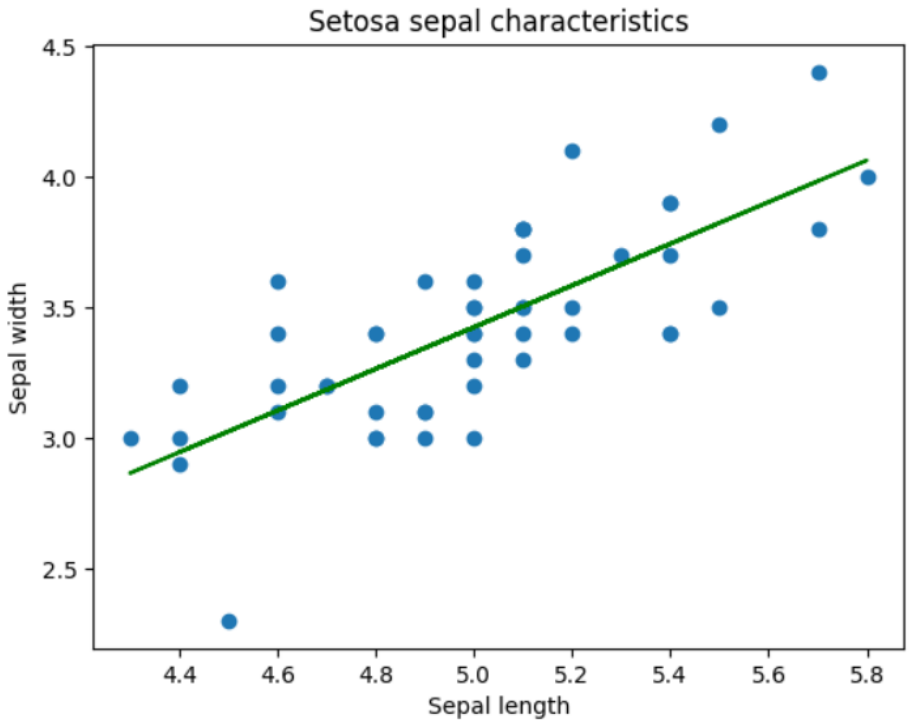
- y_pred = getPredictions(setosa, 'sepalLength', 'sepalWidth')

```
plt.scatter(setosa['sepalLength'], setosa['sepalWidth'])
plt.plot(setosa['sepalLength'], y_pred, color='g')
plt.xlabel('Sepal length')
```

plt.ylabel('Sepal width')

plt.title('Setosa sepal characteristics')

```
Out[28]: Text(0.5, 1.0, 'Setosa sepal characteristics')
```



- y_pred = getPredictions(versicolor, 'petalLength', 'petalWidth')

plt.scatter(versicolor['petalLength'], versicolor['petalWidth'])

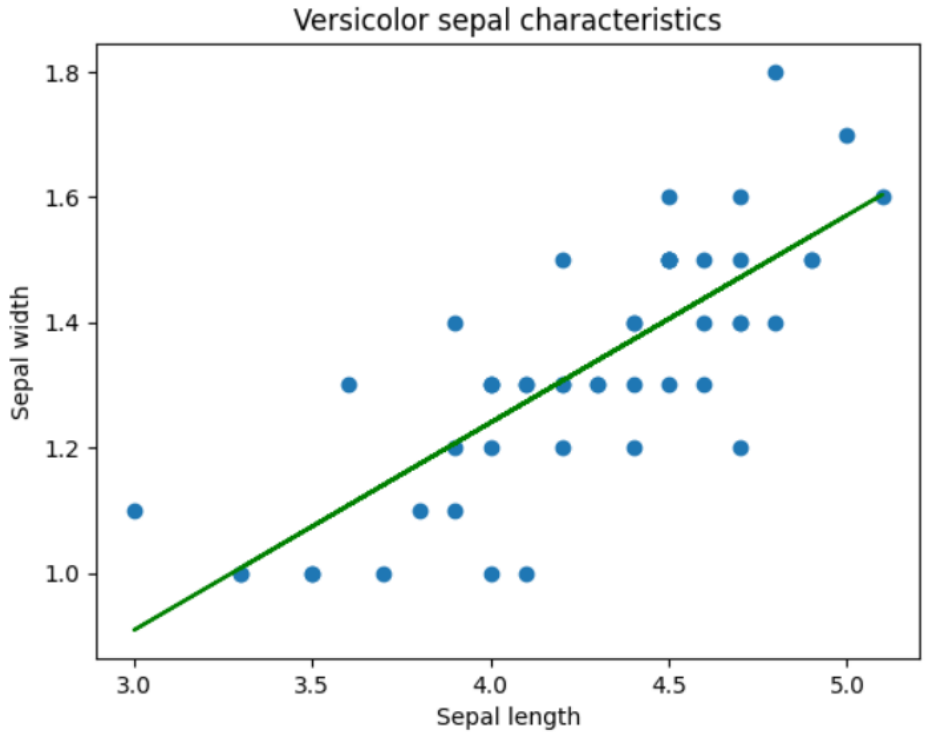
plt.plot(versicolor['petalLength'], y_pred, color='g')

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

plt.title('Versicolor sepal characteristics')

```
Out[30]: Text(0.5, 1.0, 'Versicolor sepal characteristics')
```



2. LOGISTIC REGRESSION:

- import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression

- col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']

df = pd.read_csv("diabetes1.csv", names=col_names, header=0)

df.head()

Out[35]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

#split dataset in features and target variable

- feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']

X = df[feature_cols] # Features

y = df.label # Target variable

split X and y into training and testing sets

- from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)

import the class

- from sklearn.linear_model import LogisticRegression

instantiate the model (using the default parameters)

- logreg = LogisticRegression(random_state=16)

fit the model with data

- logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

- logreg.score(X_test, y_test)

```
Out[39]: 0.8177083333333334
```

import the metrics class

- from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

cnf_matrix

```
Out[40]: array([[115,  10],
                [ 25,  42]], dtype=int64)
```

import required modules

- import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

class_names=[0,1] # name of classes

fig, ax = plt.subplots()

tick_marks = np.arange(len(class_names))

plt.xticks(tick_marks, class_names)

plt.yticks(tick_marks, class_names)

create heatmap

- sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set_label_position("top")

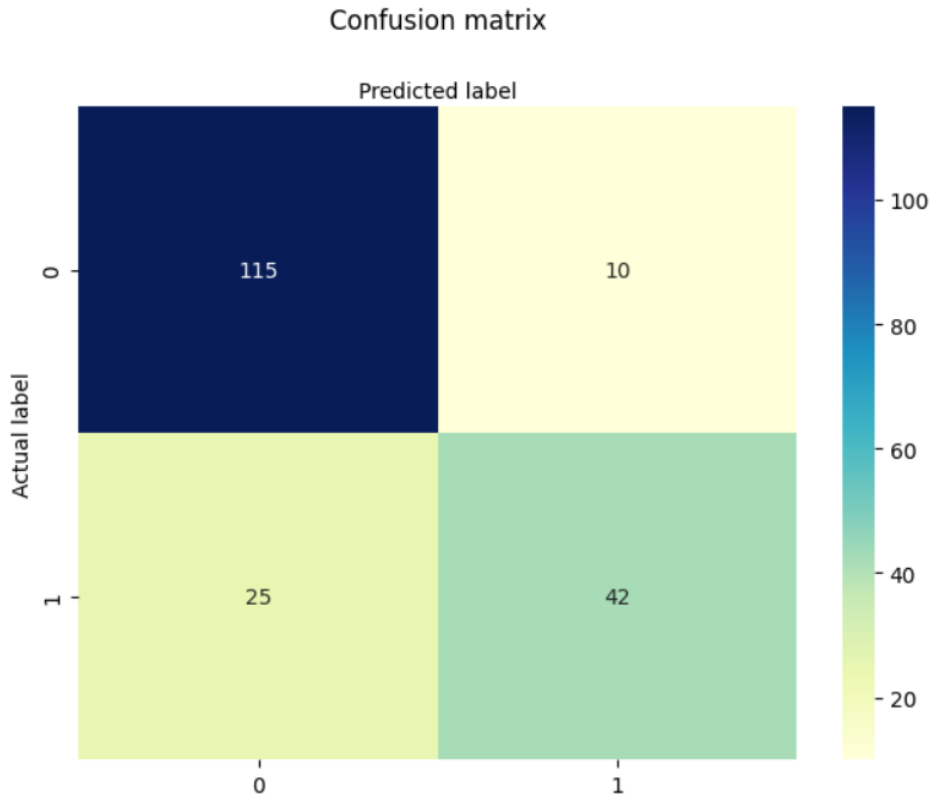
plt.tight_layout()

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

```
Out[41]: Text(0.5, 427.9555555555555, 'Predicted label')
```



- from sklearn.metrics import classification_report

target_names = ['without diabetes', 'with diabetes']

print(classification_report(y_test, y_pred, target_names=target_names))

```
                  precision    recall  f1-score   support

 without diabetes       0.82      0.92      0.87       125
    with diabetes       0.81      0.63      0.71        67

         accuracy                           0.82       192
        macro avg       0.81      0.77      0.79       192
     weighted avg       0.82      0.82      0.81       192
```

3. MULTIPLE REGRESSION:

- import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

import seaborn as sns

- df = pd.read_csv('iris1.csv')

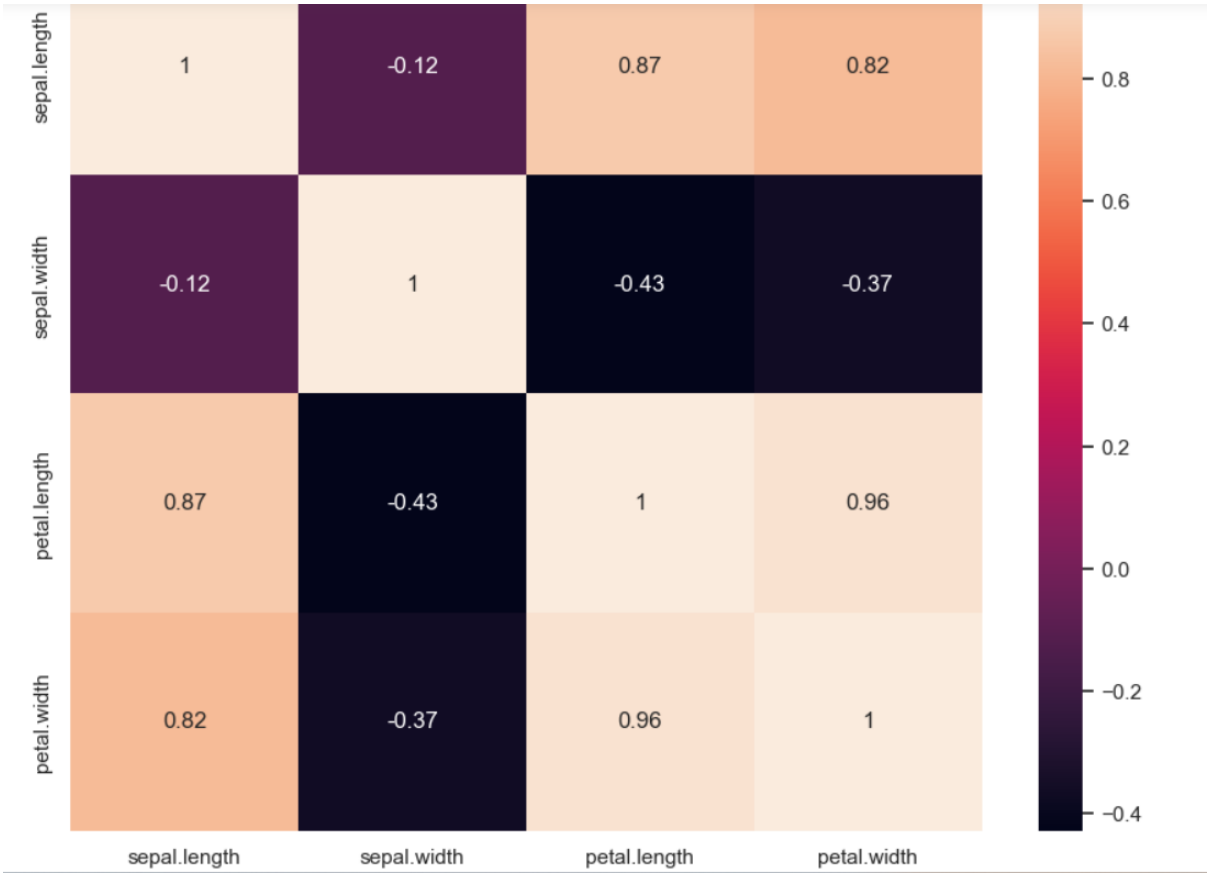
df.head()

Out[11]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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

- sns.set(rc={'figure.figsize':(10.7,8.27)})

sns.heatmap(df.corr(),annot=True)



➢ independant_vars = ['petal.width', 'sepal.width', 'petal.length']

feature =np.array(df[independant_vars]).reshape(-1,len(independant_vars))

res = np.array(df['petal.width']).reshape(-1,1)

X_train,X_test,y_train,y_test=train_test_split(feature,res,test_size=0.3)

#model and training

➢ model=LinearRegression()

fittedModel=model.fit(X_train,y_train)

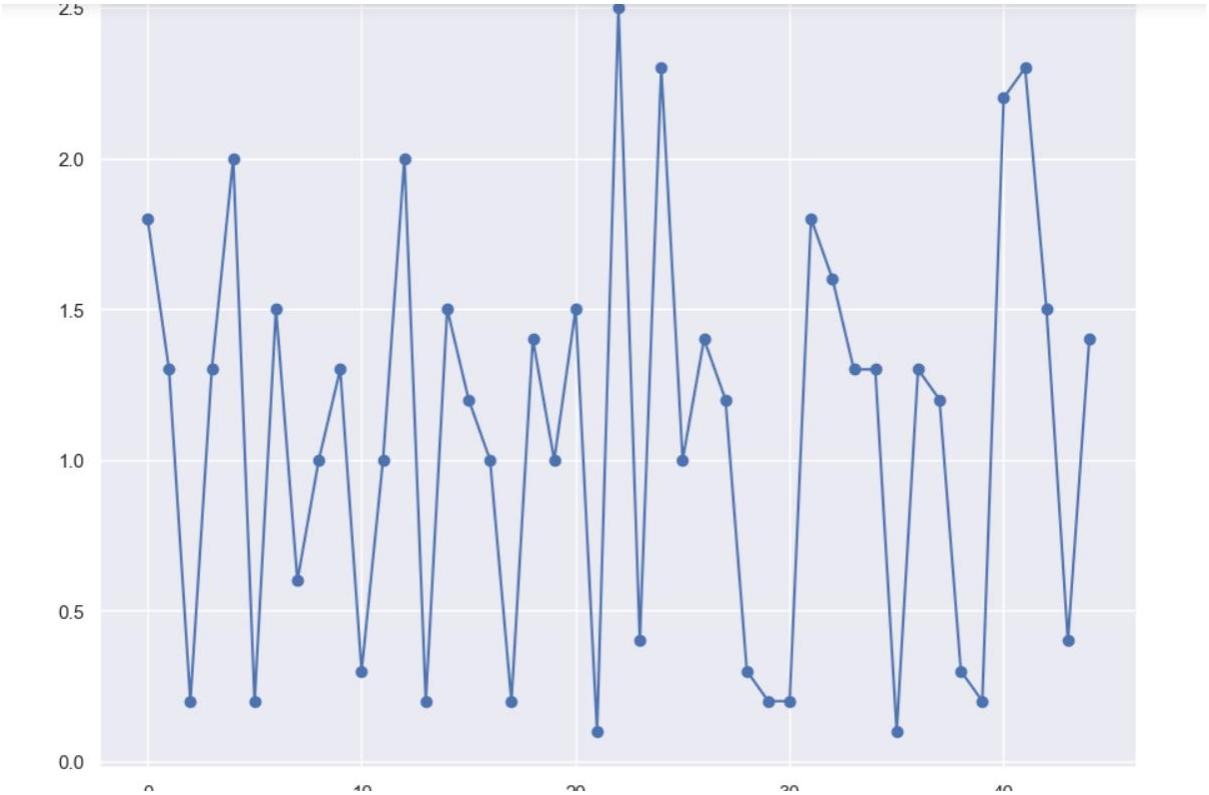
predictions=fittedModel.predict(X_test)

fittedModel.score(X_test,y_test)

```
Out[14]: 1.0
```

- plt.scatter(np.arange(len(y_test)), predictions)

plt.plot(np.arange(len(y_test)), y_test)



RESULT:

Thus, various types of regression using iris and diabetes dataset has been implemented and verified successfully.

EXP NO:5 **BAYESIAN CLASSIFIER AND SVM**

01/01/23

AIM:

To perform Bayesian classifier and Support Vector Machine(SVM) on both iris and diabetes dataset.

CODE:

1. BAYESIAN CLASSIFIER:

- import numpy as np

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
```

- df = pd.read_csv("iris1.csv")

```
df.head()
```

Out[2]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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

- X = df.iloc[:,:4].values

y = df['variety'].values

- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

- classifier = GaussianNB()

classifier.fit(X_train, y_train)

```
Out[13]: GaussianNB()
```

- y_pred = classifier.predict(X_test)

y_pred

```
Out[14]: array(['Versicolor', 'Setosa', 'Versicolor', 'Setosa', 'Versicolor',
                'Virginica', 'Setosa', 'Versicolor', 'Versicolor', 'Virginica',
                'Virginica', 'Setosa', 'Versicolor', 'Virginica', 'Setosa',
                'Virginica', 'Virginica', 'Virginica', 'Setosa', 'Virginica',
                'Setosa', 'Setosa', 'Setosa', 'Virginica', 'Setosa', 'Setosa',
                'Virginica', 'Versicolor', 'Virginica', 'Setosa'], dtype='<U10')
```

- print ("Accuracy : ", accuracy_score(y_test, y_pred))

```
Accuracy :  1.0
```

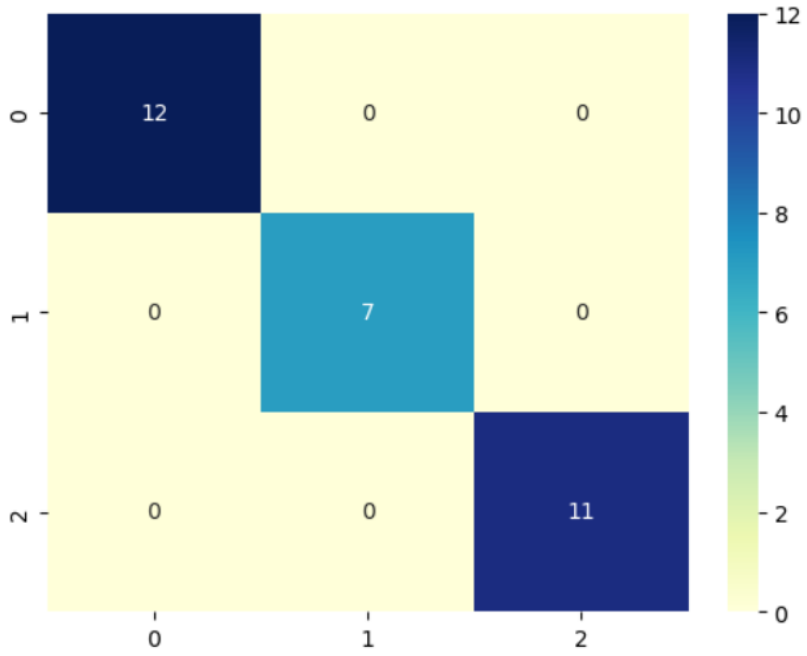
➢ cm = confusion_matrix(y_test, y_pred)

cm

```
Out[16]: array([[12,  0,  0],
                [ 0,  7,  0],
                [ 0,  0, 11]], dtype=int64)
```

➢ sns.heatmap(cm, annot=True, cmap="YlGnBu")

```
Out[17]: <AxesSubplot:>
```



➢ df = pd.read_csv("diabetes1.csv")

df.head()

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

- X = df.iloc[:,:8].values

y = df['Outcome'].values

- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

- classifier = GaussianNB()

classifier.fit(X_train, y_train)

- y_pred = classifier.predict(X_test)

y_pred

```
Out[24]: array([0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
                1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
                0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
                0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1],
               dtype=int64)
```

- print ("Accuracy : ", accuracy_score(y_test, y_pred))

```
Accuracy :  0.7922077922077922
```

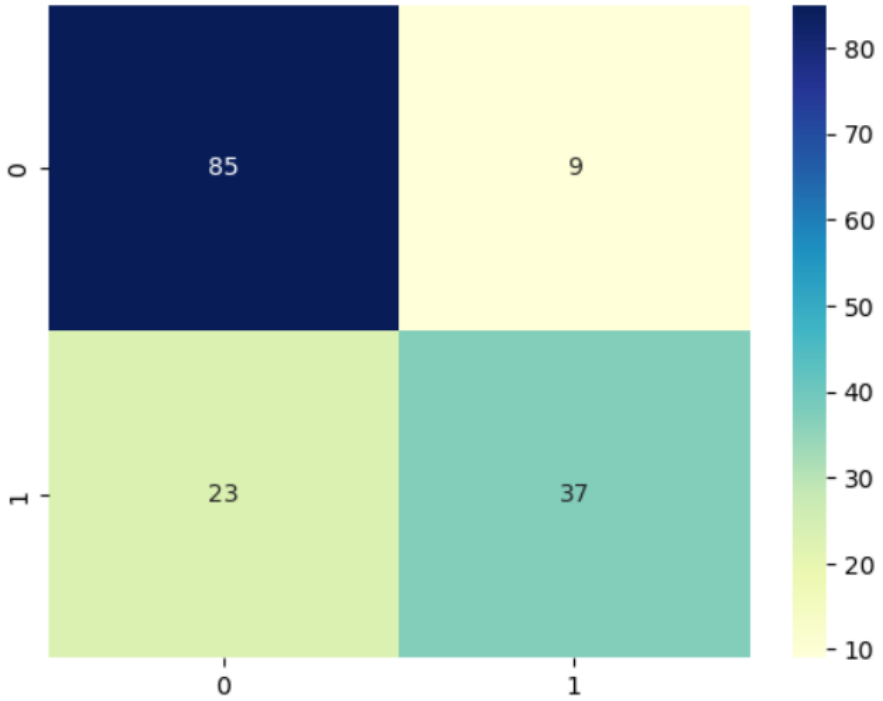
- cm = confusion_matrix(y_test, y_pred)

cm

```
Out[26]: array([[85,  9],
                [23, 37]], dtype=int64)
```

- sns.heatmap(cm, annot=True, cmap="YlGnBu")

Out[27]: <AxesSubplot:>



2. SUPPORT VECTOR MACHINE(SVM):

- df = pd.read_csv("iris1.csv")

df.head()

Out[29]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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

- X = df.iloc[:,:4].values

y = df['variety'].values

- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

- accuracy_lin = linear.score(X_test, y_test)

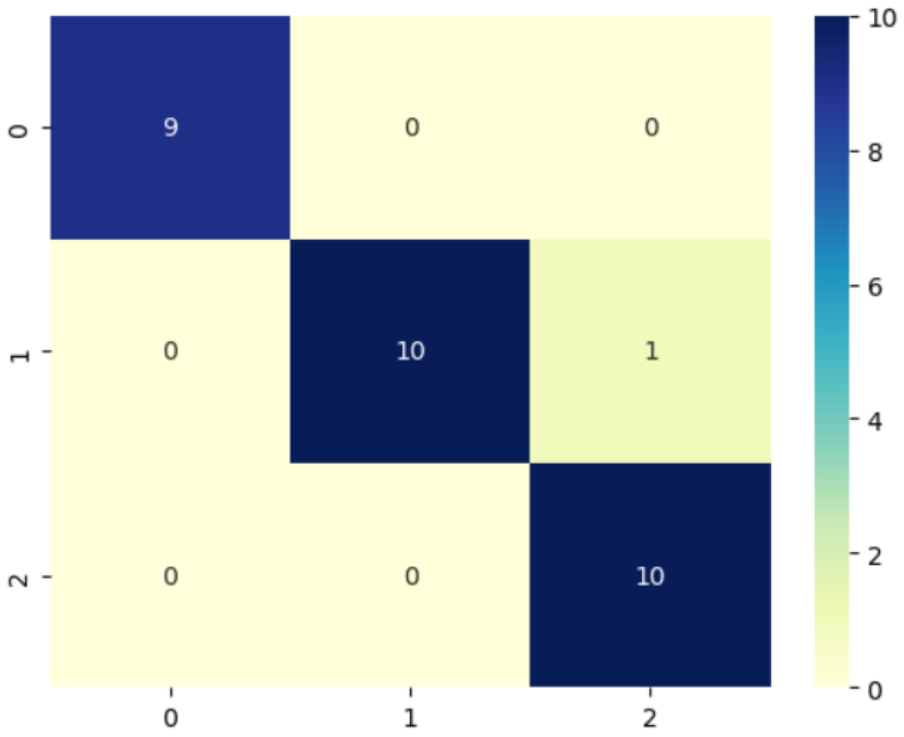
```
accuracy_poly = poly.score(X_test, y_test)
accuracy_rbf = rbf.score(X_test, y_test)
accuracy_sig = sig.score(X_test, y_test)
print("Accuracy Linear Kernel:", accuracy_lin)
print("Accuracy Polynomial Kernel:", accuracy_poly)
print("Accuracy Radial Basis Kernel:", accuracy_rbf)
print("Accuracy Sigmoid Kernel:", accuracy_sig)
```

```
Accuracy Linear Kernel: 0.9666666666666667
Accuracy Polynomial Kernel: 0.8666666666666667
Accuracy Radial Basis Kernel: 1.0
Accuracy Sigmoid Kernel: 0.8
```

- cf_matrix_linear = confusion_matrix(y_test, linear.predict(X_test))

```
cf_matrix_poly = confusion_matrix(y_test, linear.predict(X_test))
cf_matrix_rbf = confusion_matrix(y_test, linear.predict(X_test))
cf_matrix_sig = confusion_matrix(y_test, linear.predict(X_test))
sns.heatmap(cf_matrix_linear, annot=True, cmap="YlGnBu")
```

Out[35]: <AxesSubplot:>



➢ df = pd.read_csv("diabetes1.csv")

df.head()

Out[38]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

➢ X = df.iloc[:,:8].values

y = df['Outcome'].values

➢ X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

- linear=svm.SVC(kernel='linear',C=1,decision_function_shape='ovo').fit(X_train, y_train)

rbf=svm.SVC(kernel='rbf',gamma=1,C=1,decision_function_shape='ovo').fit(X_train, y_train)

poly=svm.SVC(kernel='poly',degree=3,C=1,decision_function_shape='ovo').fit(X_train, y_train)

sig=svm.SVC(kernel='sigmoid',C=1,decision_function_shape='ovo').fit(X_train, y_train)

- accuracy_lin = linear.score(X_test, y_test)

accuracy_poly = poly.score(X_test, y_test)

accuracy_rbf = rbf.score(X_test, y_test)

accuracy_sig = sig.score(X_test, y_test)

print("Accuracy Linear Kernel:", accuracy_lin)

print("Accuracy Polynomial Kernel:", accuracy_poly)

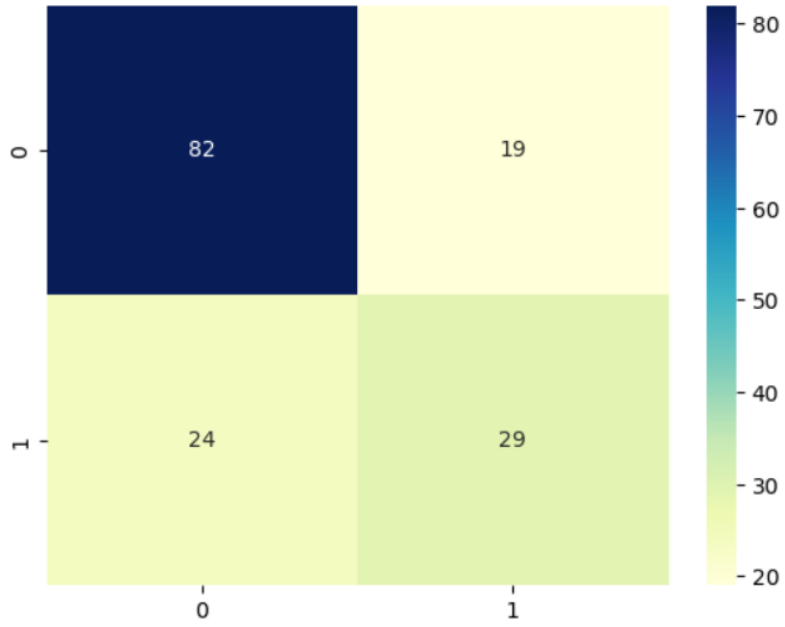
print("Accuracy Radial Basis Kernel:", accuracy_rbf)

print("Accuracy Sigmoid Kernel:", accuracy_sig)

```
Accuracy Linear Kernel: 0.7207792207792207
Accuracy Polynomial Kernel: 0.6948051948051948
Accuracy Radial Basis Kernel: 0.7402597402597403
Accuracy Sigmoid Kernel: 0.7142857142857143
```

- sns.heatmap(confusion_matrix(y_test,linear.predict(X_test)),annot=True, cmap="YlGnBu")

Out[43]: <AxesSubplot:>



RESULT:

Thus, Bayesian Classifier and Support Vector Machine has been performed on both iris and diabetes dataset and has been verified.

EXP NO:6 **VISUALIZATION**

08/03/23

AIM:

To perform Visualization on both iris and diabetes dataset using the libraries like matplotlib and seaborn.

CODE:

- import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn as sns

- df = pd.read_csv("iris1.csv")

df.head()

Out[2]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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

- setosa = df[df['variety'] == 'Setosa']

versicolor = df[df['variety'] == 'Versicolor']

virginica = df[df['variety'] == 'Virginica']

df['variety'].value_counts()

```
Out[3]: Setosa        50
        Versicolor    50
        Virginica     50
        Name: variety, dtype: int64
```

- plt.plot(setosa['sepal.length'], ls=':')

plt.plot(versicolor['sepal.length'], ls='--')

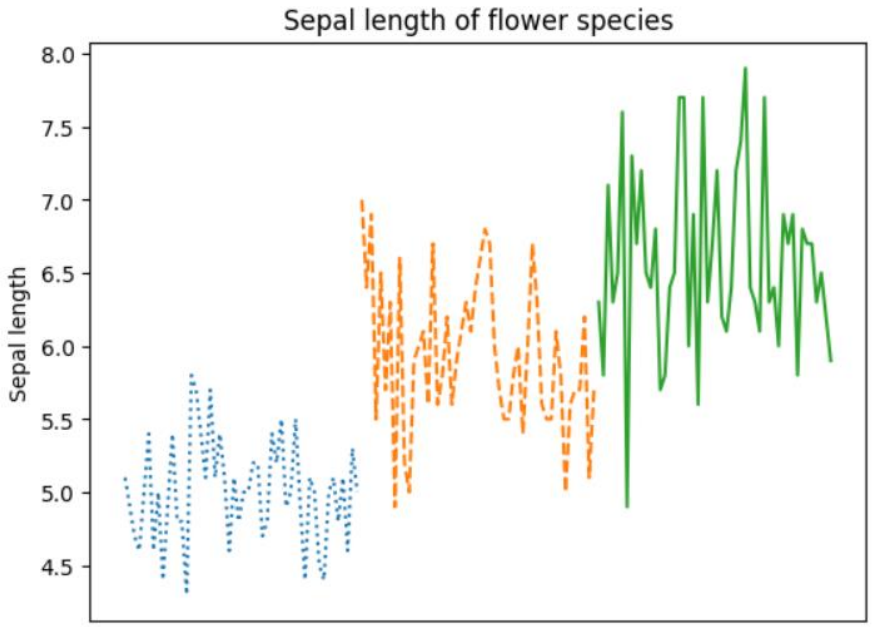
plt.plot(virginica['sepal.length'])

plt.ylabel("Sepal length")

plt.title("Sepal length of flower species")

plt.xticks([])

```
Out[4]: ([], [])
```



- plt.scatter(np.arange(len(setosa.index)),(setosa['petal.length']),marker='1', c='red', label='Setosa')

plt.scatter(np.arange(len(versicolor.index)),(versicolor['petal.length']),marker='+', c='purple', label='Versicolor')

plt.scatter(np.arange(len(virginica.index)),(virginica['petal.length']), marker='*', c='g', label='Virginica')

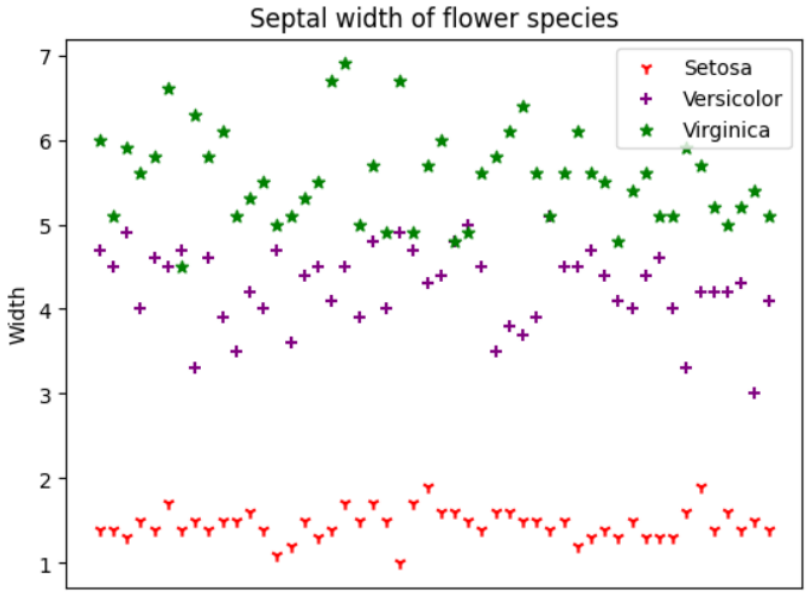
plt.ylabel("Width")

plt.title("Septal width of flower species")

plt.legend()

plt.xticks([])

```
Out[5]: ([], [])
```



- plt.bar([1],setosa['petal.length'].mean(), width=0.4, label='setosa', color='olive')

plt.bar([1.7],versicolor['petal.length'].mean(),width=0.4,label='versicolor',color='slategray')

plt.bar([2.5],virginica['petal.length'].mean(),width=0.4,label='virginica',color=

'skyblue')

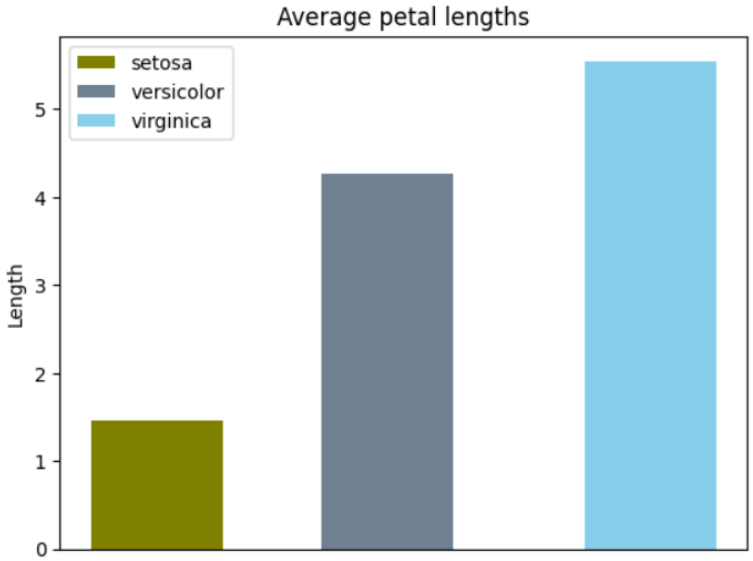
plt.legend()

plt.ylabel("Length")

plt.title("Average petal lengths")

plt.xticks([])

```
Out[6]: ([], [])
```



- without_species = df.drop('variety', axis=1)

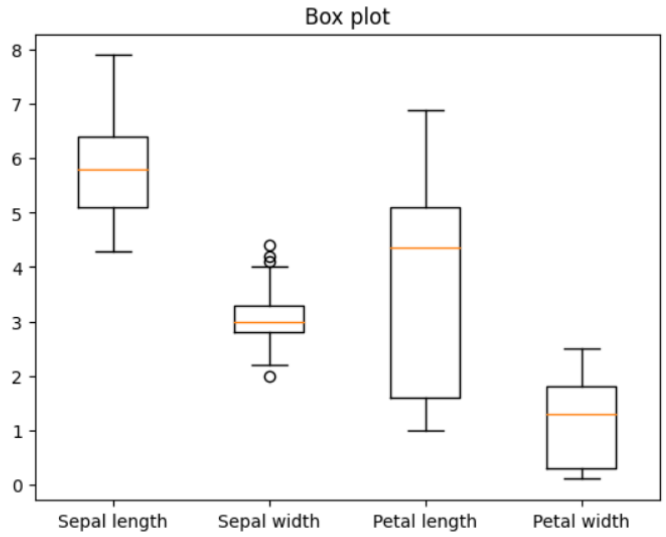
transposed_without_species = without_species.transpose()

plt.boxplot(without_species)

plt.xticks([1,2,3,4], ['Sepal length', 'Sepal width', 'Petal length', 'Petal width'])

plt.title("Box plot")

```
Out[7]: Text(0.5, 1.0, 'Box plot')
```



- plt.hist(setosa['petal.width'], label='Setosa')

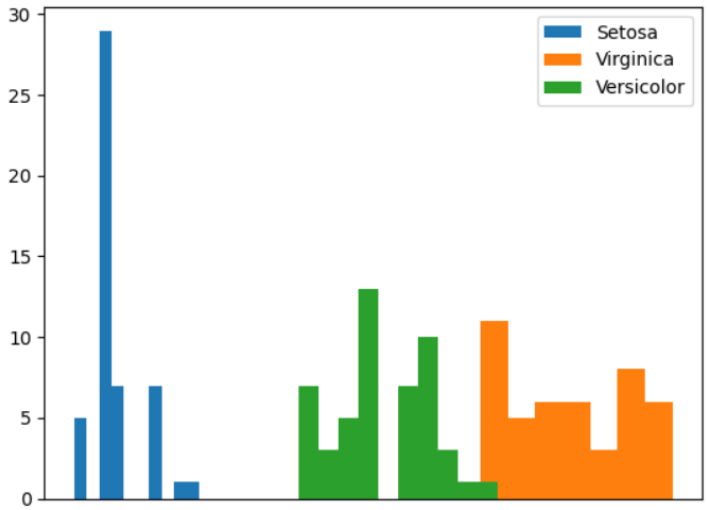
plt.hist(virginica['petal.width'], label='Virginica')

plt.hist(versicolor['petal.width'], label='Versicolor')

plt.legend()

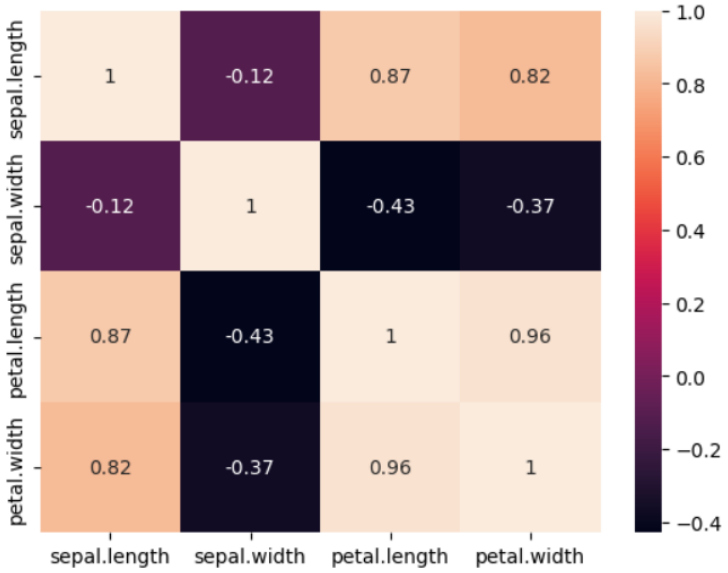
plt.xticks([])

```
Out[8]: ([], [])
```



- sns.heatmap(df.corr(method='pearson'),annot = True);

plt.show()



- df = pd.read_csv("diabetes1.csv")

df.head()

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

- one = df[df['Outcome'] == 1]

zero = df[df['Outcome'] == 0]

df['Outcome'].value_counts()

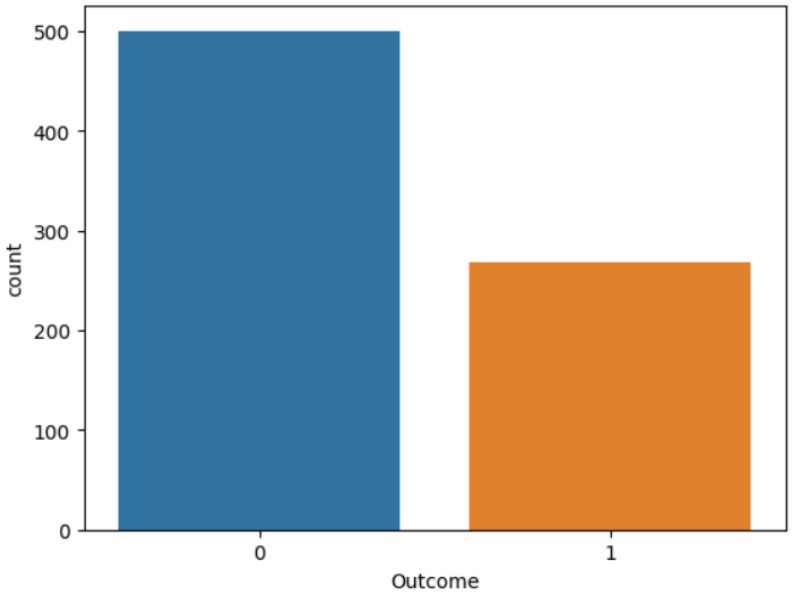
```
Out[14]: 0    500
         1    268
         Name: Outcome, dtype: int64
```

- import seaborn as sns

import matplotlib.pyplot as plt

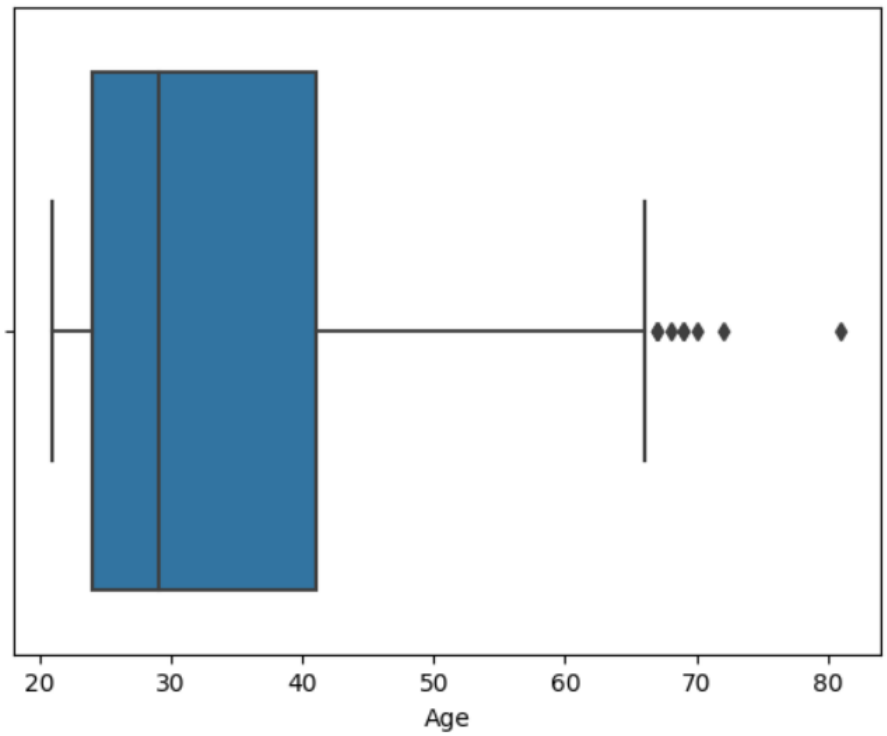
sns.countplot(x='Outcome', data=df,)

plt.show()



- sns.boxplot(x='Age', data=df)

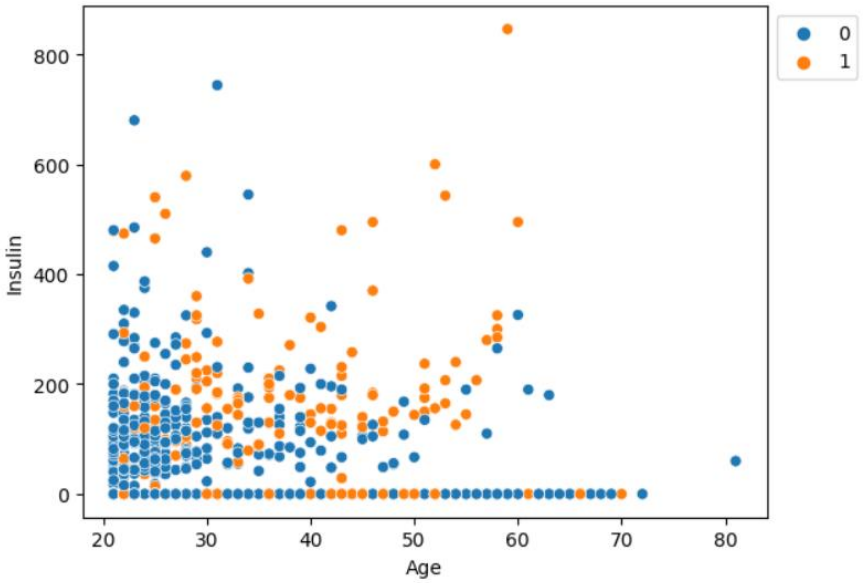
```
Out[12]: <AxesSubplot: xlabel='Age'>
```



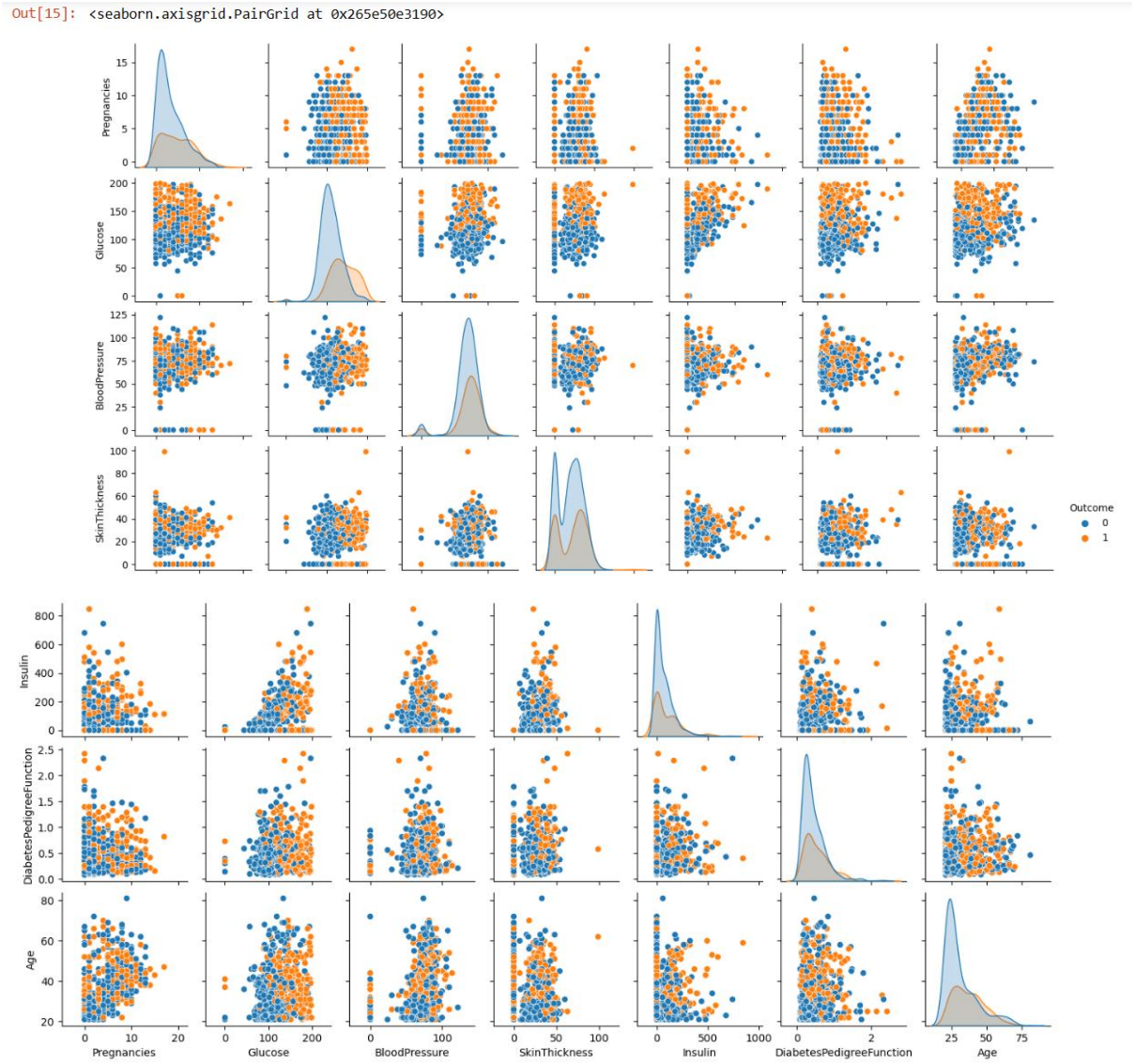
- sns.scatterplot(x='Age', y='Insulin', hue='Outcome', data=df,)

plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()

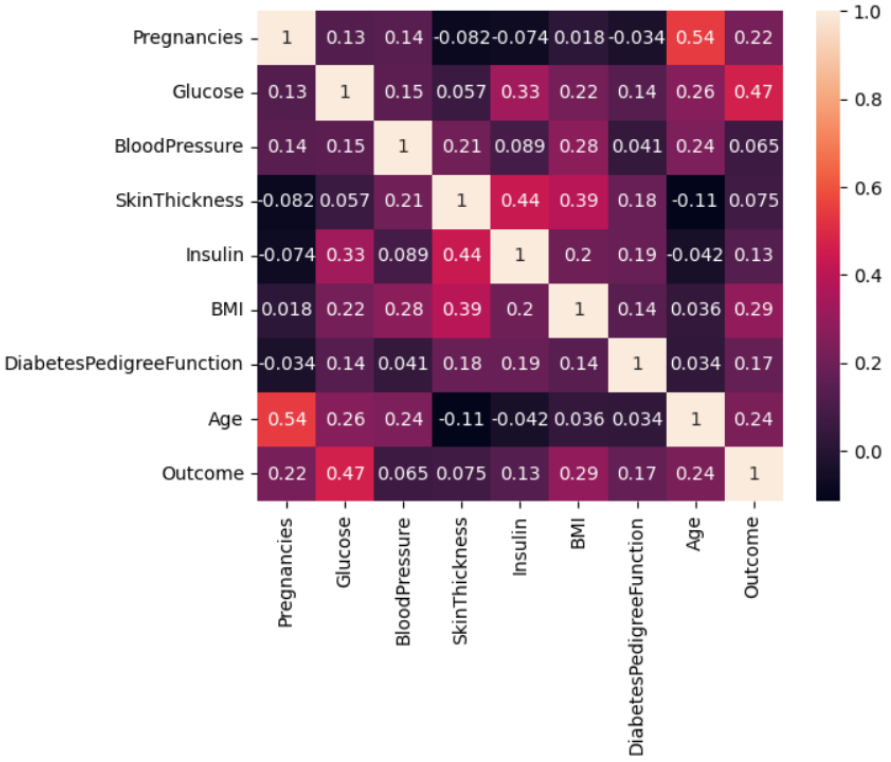


- sns.pairplot(df.drop(['BMI'], axis = 1), hue='Outcome', height=2)



- sns.heatmap(df.corr(method='pearson'), annot = True);

plt.show()



RESULT:

Thus, Visualization has been performed on both iris and diabetes dataset using matplotlib and seaborn libraries.