Assignment -6 -Data Analytics 3 - Naive Bayes Classification

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

1. Implement Simple Naive Bayes classification algorithm using Python/R on iris.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

In [2]:
```
# Description

# It includes three iris species with 50 samples each as well as some properties about
#One flower species is linearly separable from the other two, but the other two are not

# The columns in this dataset are:

# Id
# SepalLengthCm
# SepalWidthCm
# PetalLengthCm
# PetalWidthCm
# Species
```

In [3]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

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

Out[3]:

	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

In [4]:
```
iris['Species'].unique()
```

Out[4]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [5]:

```
iris.describe(include='all')
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Iris-versicolor
freq	NaN	NaN	NaN	NaN	NaN	50
mean	75.500000	5.843333	3.054000	3.758667	1.198667	NaN
std	43.445368	0.828066	0.433594	1.764420	0.763161	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

In [6]:

```
iris.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
```

In [7]:

```
#Removing unecessary columns
iris.drop(columns="Id",inplace=True)
```

In [8]:

```
#Checking for null values
iris.isnull().sum()
```

Out[8]:

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Correlations

In [9]:

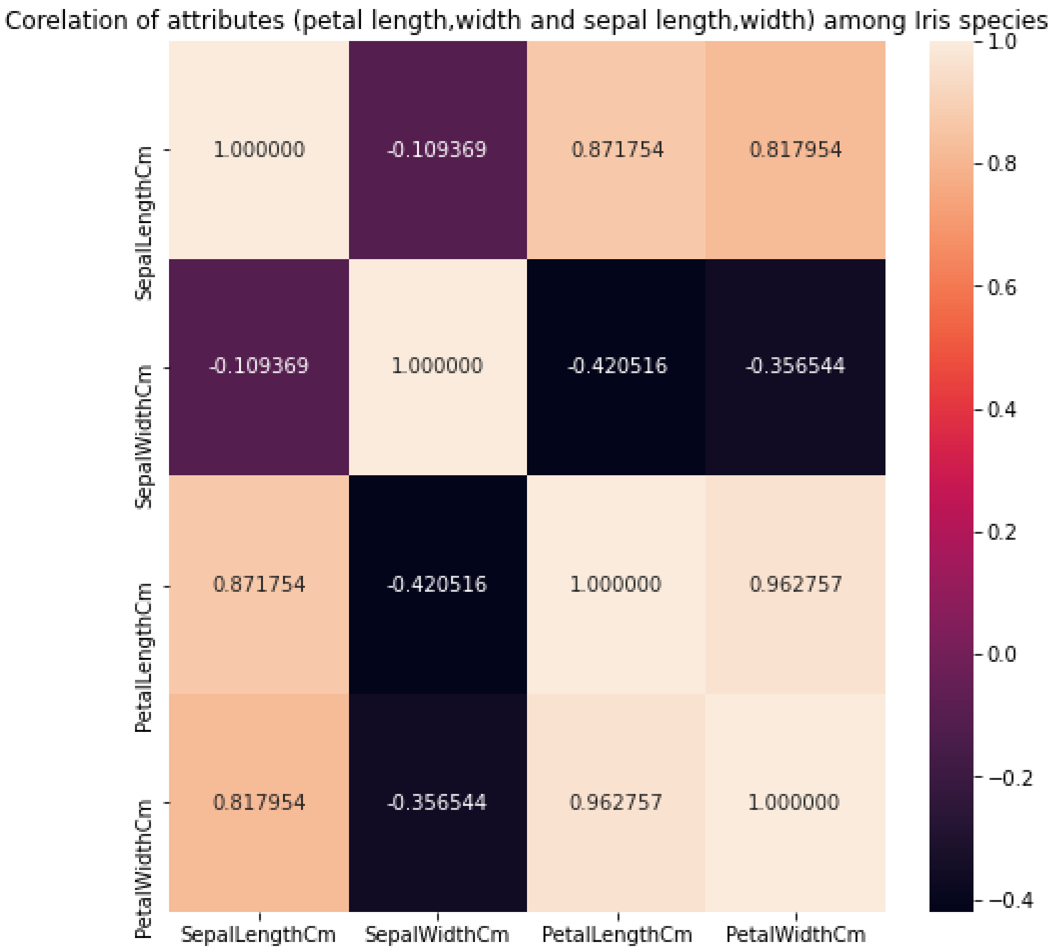
```
iris.corr()
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [10]:

```
plt.subplots(figsize = (8,8))
sns.heatmap(iris.corr(),annot=True,fmt="f").set_title("Corelation of attributes (petal
plt.show()
```



Splitting The Data into Training And Testing Dataset

In [11]:

```
X=iris.iloc[:,0:4].values
y=iris.iloc[:,4].values

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

Naive Bayes Classifiers

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text classification that includes a high-dimensional training dataset.Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.It is a **probabilistic classifier**, which means it predicts on the basis of the probability of an object.Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Bayes' Theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{(P(B|A)P(A))}{P(B)}$$

Where,

- **P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.
- **P(B|A) is Likelihood probability**: Probability of the evidence given that the probability of a hypothesis is true.
- **P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.
- **P(B) is Marginal Probability**: Probability of Evidence.

In [12]:

```
#Metrics
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score

#Model Select
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

In [13]:

```
#Train and Test split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
```

In [14]:

```
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)
```

Confusion Matrix

In [20]:
```
cm = confusion_matrix(y_test, Y_pred)
print('Confusion matrix for Naive Bayes\n',cm)
```

```
Confusion matrix for Naive Bayes
 [[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
```

In [21]:
```
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)
FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)

print("['SetosaTP','VersicolorTP','VirginicaTP']",TP)
print("['SetosaFP','VersicolorFP','VirginicaFP']",FP)
print("['SetosaFN','VersicolorFN','VirginicaFN']",FN)
print("['SetosaTN','VersicolorTN','VirginicaTN']",TN)
```

```
['SetosaTP','VersicolorTP','VirginicaTP'] [16. 18. 11.]
['SetosaFP','VersicolorFP','VirginicaFP'] [0. 0. 0.]
['SetosaFN','VersicolorFN','VirginicaFN'] [0. 0. 0.]
['SetosaTN','VersicolorTN','VirginicaTN'] [29. 27. 34.]
```

Accuracy

In [22]:
```
accuracy = accuracy_score(y_test,Y_pred)
print('accuracy_Naive Bayes: %.3f' %accuracy)
```

```
accuracy_Naive Bayes: 1.000
```

In [23]:
```
precision =precision_score(y_test, Y_pred,average='micro')
print('precision_Naive Bayes: %.3f' %precision)
```

```
precision_Naive Bayes: 1.000
```

Recall

In [24]:
```
recall =  recall_score(y_test, Y_pred,average='micro')
print('recall_Naive Bayes: %.3f' %recall)
```

```
recall_Naive Bayes: 1.000
```