**DSBDAL Assignment no. 6**

**Problem Statement:**

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

Naive Bayes is a statistical classification technique based on the Bayes Theorem and one of the simplest Supervised Learning algorithms. The Naive Bayes classifier is a quick, accurate, and trustworthy method, especially on large datasets.

The simple formula of Bayes theorem is:

**P(A|B) =**

Where P(A) and P(B) are two independent events and (B) is not equal to zero.

P(A | B): is the conditional probability of event A occurring given that B is true.

P( B | A): is the conditional probability of event B occurring given that A is true.

P(A) and P(B):  are the probabilities of A and B occurring independently of one another (the marginal probability).

**What is Naive Bayes Classification?**

The Naive Bayes classification algorithm is a probabilistic classifier, and it belongs to [Supervised Learning](https://hands-on.cloud/overview-of-supervised-machine-learning-algorithms/). It is based on probability models that incorporate strong independence assumptions. The independence assumptions often do not have an impact on reality. Therefore they are considered naive.

Another assumption made by the Naive Bayes classifier is that all the predictors have an equal effect on the outcome. The Naive Bayes classification has the following different types:

* The [**Multinomial Naive Bayes**](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Multinomial_na%C3%AFve_Bayes) method is a common Bayesian learning approach in natural language processing. Using the Bayes theorem, the program estimates the tag of a text, such as an email or a newspaper piece. It assesses the likelihood of each tag for a given sample and returns the tag with the highest possibility.
* The [**Bernoulli Naive Bayes**](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Bernoulli_na%C3%AFve_Bayes) is a part of the family of Naive Bayes. It only takes binary values. There may be multiple features, but each is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors.
* The [**Gaussian Naive Bayes**](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Gaussian_na%C3%AFve_Bayes) is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. To build a simple model using Gaussian Naive Bayes, we assume the data is characterized by a Gaussian distribution with no covariance (independent dimensions) between the parameters. This model may be fit simply by calculating the mean and standard deviation of the points within each label.

Naive Bayes classifier makes two fundamental assumptions on the observations.

* The target classes are **independent** to each other. Consider a rainy day with strong winds and high humidity. These two features, wind and humidity, would be treated as independent by a Naive classifier. That is to say, each feature would impose its own probabilities on the outcome, such as rain in this case.
* Prior probabilities for the target classes are **equal**. That is, before calculating the posterior probability of each class, the classifier will assign each target class the same prior probability.

Inabove problem statement we use Iris flower dataset to implement Simple Naïve Bayes classification algorithm. Use Sepal Length, Sepal Width, Petal length and Petal Width as input And Class is as Output.

**CODING:**

Read Dataset

import pandas as pd

df=pd.read\_csv("https://gist.githubusercontent.com/netj/8836201/raw/6f9306ad21398ea43cba4f7d537619d0e07d5ae3/iris.csv")

df

**Read data types of feature and convert all features in one datatypes like Float/int**

df.dtypes

**Here class is object type convert it in to 0,1 format by assigning it as category**

df['variety']=df['variety'].astype('category')

df.dtypes

df['variety']=df['variety'].cat.codes

df

df.isnull().sum()

(df <= 0).sum()

# check for outliers

# co-relation matrix

def DetectOutlier(df,var):

  Q1 = df[var].quantile(0.25)

  Q3 = df[var].quantile(0.75)

  IQR = Q3 - Q1

  high, low = Q3+1.5\*IQR, Q1-1.5\*IQR

  print("Highest allowed in variable:", var, high)

  print("lowest allowed in variable:", var, low)

  count = df[(df[var] > high) | (df[var] < low)][var].count()

  print('Total outliers in:',var,':',count)

DetectOutlier(df,'sepal.length')

DetectOutlier(df,'sepal.width')

DetectOutlier(df,'petal.length')

DetectOutlier(df,'petal.width')

import seaborn as sns

sns.boxplot(df['sepal.width'])

def OutlierRemoval(df,var):

  Q1 = df[var].quantile(0.25)

  Q3 = df[var].quantile(0.75)

  IQR = Q3 - Q1

  high, low = Q3+1.5\*IQR, Q1-1.5\*IQR

  print("Highest allowed in variable:", var, high)

  print("lowest allowed in variable:", var, low)

  count = df[(df[var] > high) | (df[var] < low)][var].count()

  print('Total outliers in:',var,':',count)

  df = df[((df[var] >= low) & (df[var] <= high))]

  return df

print(df.shape)

df = OutlierRemoval(df,'sepal.width')

print(df.shape)

import seaborn as sns

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

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

# split the data into inputs and outputs

X = df.iloc[:, [0,2,3]].values

y = df.iloc[:, 4].values

# training and testing data

from sklearn.model\_selection import train\_test\_split

# assign test data size 25%

X\_train, X\_test, y\_train, y\_test =train\_test\_split(X,y,test\_size= 0.25, random\_state=0)

# importing standard scaler

from sklearn.preprocessing import StandardScaler

# scalling the input data

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train)

X\_test = sc\_X.fit\_transform(X\_test)

# importing standard scaler

from sklearn.preprocessing import StandardScaler

# scalling the input data

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train)

X\_test = sc\_X.fit\_transform(X\_test)

# importing accuracy score

from sklearn.metrics import accuracy\_score

# printing the accuracy of the model

print(accuracy\_score(y\_pred, y\_test))

# import Gaussian Naive Bayes classifier

from sklearn.naive\_bayes import GaussianNB

# create a Gaussian Classifier

classifer1 = GaussianNB()

# training the model

classifer1.fit(X\_train, y\_train)

# testing the model

y\_pred1 = classifer1.predict(X\_test)

# importing accuracy score

from sklearn.metrics import accuracy\_score

# printing the accuracy of the model

print(accuracy\_score(y\_test,y\_pred1))

# importing the required modules

import seaborn as sns

from sklearn.metrics import confusion\_matrix

# passing actual and predicted values

cm = confusion\_matrix(y\_test, y\_pred)

# true write data values in each cell of the matrix

sns.heatmap(cm, annot=True)

plt.savefig('confusion.png')

# importing classification report

from sklearn.metrics import classification\_report

# printing the report

print(classification\_report(y\_test, y\_pred))

o/p: precision recall f1-score support

0 0.88 1.00 0.94 15

1 1.00 0.46 0.63 13

2 0.64 1.00 0.78 9

accuracy 0.81 37

macro avg 0.84 0.82 0.78 37

weighted avg 0.87 0.81 0.79 37