**Ex.no:6 DHAMODARAN.B**

**Date:08.09.2020** **1832016**

**NAIVE BAYES CLASSIFIER**

**Problem Statement:**

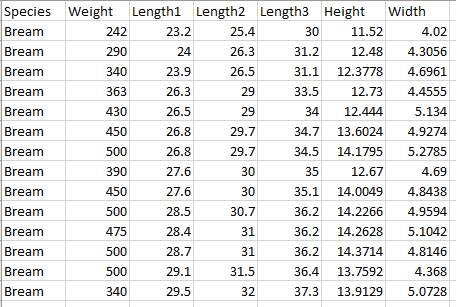
Fishes are one of the greatest sources of protein and the total turnover of the fish industry is about 45,000 crore Rupees. Thus, ensuring the quality of fish is necessary. Since the fishes are priced on their weight and how rare the species is, detecting the species of the fish is necessary to determine the price. Our motto is to predict the species of the fish with the physical measurements of the fish. By predicting the species of the fish, we can determine the price of the fish. This can be achieved by using Naive Bayes Classifier.

**Problem Description:**

The dataset had data on the body measurements of the fish such as width, height and breadth, weight etc... It is pre-known that the different species vary in their physical characteristics and body measurements too. The dependent variable or the target variable that we want to predict is ‘species’ and the independent variables are the ‘weight’, ‘length1’, ’length2’, ’lenght3’, ’height’ and ‘width’. All the independent features were continuous in nature. The dataset had information on 7 species but we have taken only two species i.e. ‘Bream’ and ‘Perch’. The information on two species ‘Bream’ and ‘Perch’ had about 91 data instances.

The data are scaled using the standard scaling technique and later it was split into train and test in the ratio 85:15. This ratio of splitting had been done since the size of the dataset is quite less for training. Scaled data helps in making the computation faster. Thus, successfully preprocessed data can now be used for fitting of model. The Naive Bayes classifier model is implemented using the scikit’s learn’ s GassianNB function and later it is implemented from the scratch. Both the models built from the scratch and the scikit-learns implementation are compared for their performance under different metrics.

**Sample Dataset:**



**Code:**

**Naïve Bayes Classifier Scikit Learn implementation:**

import pandas as pd

import numpy as np

dataset=pd.read\_csv('Fish.csv') print("The Different Species are :",list(dataset['Species'].unique()))

print("The data for the species Bream and Perch are :")

dataframe=pd.DataFrame(dataset[dataset['Species'].isin(['Bream','Perch'])])

dataframe.index=range(len(dataframe))

from sklearn.preprocessing import StandardScaler

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

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

scaler=StandardScaler()

dataframe.iloc[:,1:]=scaler.fit\_transform(dataframe.iloc[:,1:])

xtrain,xtest,ytrain,ytest=train\_test\_split(dataframe.iloc[:,1:].values,dataframe['Species'].values,test\_size=0.15)

from sklearn.naive\_bayes import GaussianNB

NB\_model=GaussianNB()

NB\_model.fit(xtrain,ytrain)

ypred=NB\_model.predict(xtest)

print("Predictions \n",ypred)

print("Confusion Matrix \n",confusion\_matrix(ytest,ypred))

print(classification\_report(ytest,ypred))

probs=NB\_model.predict\_proba(xtest)

probs=probs[:,1]

from sklearn.metrics import roc\_curve

from sklearn.metrics import roc\_auc\_score

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelBinarizer

binarizer=LabelBinarizer()

ytest\_b=binarizer.fit\_transform(ytest)

fpr,tpr,\_=roc\_curve(ytest\_b,probs)

random\_probs = [0 for \_ in range(len(ytest))]

p\_fpr,p\_tpr,\_ = roc\_curve(ytest\_b,random\_probs)

auc\_score=roc\_auc\_score(ytest,probs)

print("AUC SCORE : " ,auc\_score)

plt.plot(p\_fpr, p\_tpr, linestyle='--')

plt.plot(fpr, tpr, marker='.', label='Naive Bayes (area=%0.2f)'% auc\_score)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title("ROC-AUC CURVE(Scikit Learn Implementation)")

plt.legend()

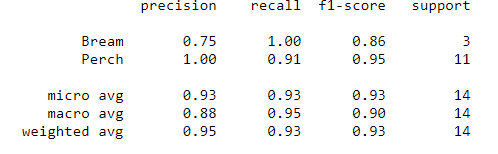
plt.show()

**Output:**



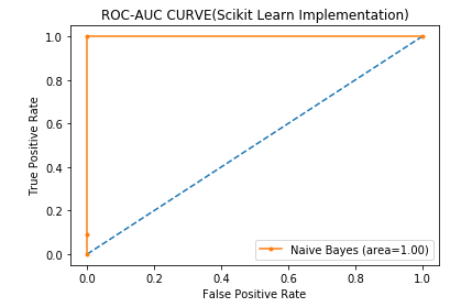


CLASSIFICATION REPORT









**Inference**:

The Naive Bayes classifier implemented with Scikit learn has produced an accuracy of 92% which implements that the model correctly classifies about 92% of the instances and the rest 8% were misclassified. Precision is the ratio of true positive to the total no. of instances classified as positive. Only the 75% of the Bream species which are classified as Bream are actually in nature while the rest 25% are not Bream but classified as Bream. But in the case of Perch, all the instances classified as Perch are actually in Perch in nature. The AUC score of the model is 1.00 which implies that the model is very well capable of the distinguishing the species.

**Implementation From Scratch:**

from math import pi

class Naive\_Bayes:

def \_\_init\_\_(self):

mean=None

labels=None

stdev=None

prior\_probs=None

def group\_by\_class(self,xdata,ydata):

self.labels=np.unique(ydata).flatten()

self.mean=np.empty(shape=(len(self.labels),xdata.shape[1]))

self.stdev=np.empty(shape=(len(self.labels),xdata.shape[1]))

for i in range(len(self.labels)):

data=np.take(xdata,np.where(ydata==self.labels[i]),axis=0)

self.mean[i]=data.mean(axis=1)

self.stdev[i]=data.std(axis=1)

return self.mean,self.stdev

def prior\_probability(self,xdata,ydata):

self.prior\_probs=np.empty(shape=len(self.labels))

for i in range(len(self.labels)):

data=np.take(xdata,np.where(ydata==self.labels[i]),axis=0).reshape(-1,xdata.shape[1])

self.prior\_probs[i]=len(data)/len(ydata)

return self.prior\_probs

def train(self,xdata,ydata):

self.group\_by\_class(xdata,ydata)

self.prior\_probability(xdata,ydata)

def estimate\_likelihood(self,data):

likelihood=np.empty(shape=(self.mean.shape))

for i in range(len(self.labels)):

exponent=np.exp(-((data-self.mean[i])\*\*2/(2\*self.stdev[i]\*\*2)))

#print("{} {}".format((1 / (np.sqrt(2 \* pi) \* self.stdev[i])) \* exponent, self.labels[i]))

likelihood[i] = (1 / (np.sqrt(2 \* pi) \* self.stdev[i])) \* exponent

return likelihood

def predict(self,data):

likelihood=self.estimate\_likelihood(data)

predicted\_probs=np.empty(shape=(len(self.labels)))

for i in range(len(self.labels)):

predicted\_probs[i]=np.prod(likelihood[i])\*self.prior\_probs[i]

return predicted\_probs

def predict\_class(self,data):

predicted\_class=np.empty(shape=len(data.reshape(-1,6)),dtype=object)

for i in range(len(data.reshape(-1,6))):

predicted\_probs=self.predict(data[i])

if predicted\_probs[0]> predicted\_probs[1]:

predicted\_class[i]=self.labels[0]

elif predicted\_probs[1]> predicted\_probs[0]:

predicted\_class[i]=self.labels[1]

return predicted\_class

def predict\_probability(self,xdata):

xdata=xdata.reshape(-1,6)

predicted\_probability=np.empty(shape=(len(xdata),len(self.labels)))

for i in range(len(xdata)):

predicted\_probability[i]=self.predict(xdata[i])

return predicted\_probability

xdata=xtrain;ydata=ytrain;

nb=Naive\_Bayes()

nb.train(xtrain,ytrain)

predicted\_class=nb.predict\_class(xtest)

print(predicted\_class)

print("Confusion Matrix : \n",confusion\_matrix(ytest,predicted\_class))

print(classification\_report(ytest,predicted\_class))

binarizer=LabelBinarizer()

ytest\_b=binarizer.fit\_transform(ytest)

fpr,tpr,\_=roc\_curve(ytest\_b,probs)

random\_probs = [0 for \_ in range(len(ytest))]

p\_fpr,p\_tpr,\_ = roc\_curve(ytest\_b,random\_probs)

auc\_score=roc\_auc\_score(ytest,probs)

print(auc\_score)

plt.plot(p\_fpr, p\_tpr, linestyle='--')

plt.plot(fpr, tpr, marker='.', label='Naive Bayes (area=%0.2f)'% auc\_score)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title("ROC-AUC CURVE(From Scratch Implementation)")

plt.legend()

plt.show()

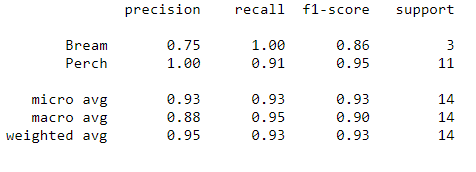
**Output:**

PREDICTIONS



****

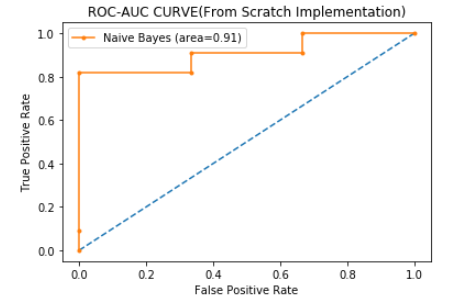
CLASSIFICATION REPORT





AUC SCORE

****

****

**Inference:**

The Naive Bayes classifier implemented from Scratch has produced an accuracy of 92% which implements that the model correctly classifies about 92% of the instances and the rest 8% were misclassified. Precision is the ratio of true positive to the total no. of instances classified as positive. Only the 75% of the Bream species which are classified as Bream are actually in nature while the rest 25% are not Bream but classified as Bream. But in the case of Perch, all the instances classified as Perch are actually in Perch in nature. The AUC score of the model is 0.91 which implies that the model is very well capable of the distinguishing the species.

**Model Comparison and Conclusion:**

When we look at both the models, the confusion matrix, accuracy and the classification report are identical for both the models. With the help of these confusion matrix and the classification report it is not possible for us to determine the best model. Considering the AUC score, the scikit learn implementation has a score of 1.0 which implements that the model is able to well distinguish the classes. From the scratch implementation of Naive Bayes has an AUC score of 0.91 which is less compared to AUC score of scikit learn implementation. So, it can be concluded that the scikit learn implementation of Naive Bayes is better at predicting the probabilities for each class and comparatively it is also better at distinguishing between the classes.