VIT-AP UNIVERSITY, ANDHRA PRADESH

Lab Exercise – 11: Naïve Bayes Classifier & Accuracy Calculation

Academic year: 2022-2023 **Branch/ Class:** B.Tech

Semester: Fall **Date:27-11-22**

Faculty Name: Dr Aravapalli Rama Satish **School:** SCOPE

Student name: MAJJIGA JASWANTH **Reg. no.:20BCD7171**

---

1. Develop Naïve Bayesian classifier for the data set “tennis.csv”. Consider the following test tuple and predict the whether a player can play tennis or not?

   Test Tuple: (Sunny, cool, high, true)

Code:

```
import pandas as pd
class_value={}
coloumn={}
data=[]
sample_data={}
result={}
global diff_values
diff_values=['empty']

def return_different_class_value(value):
    found=0
    for x in class_value:
        if(x==value):
            found=1
    if(found==0):
        class_value[value]=1
    else:
        class_value[value]+=1

def find_diff_value(values):
    already=0
    for index in diff_values:
        if(index==values):
            already=1
    if(already==0):
        diff_values.append(values)

def get_sample_data(prediction_value):
    for key in coloumn:
        attribute_no=1
        if(key!=prediction_value):
```

```
            for i in range(0,end):
                find_diff_value(data[i][coloumn[key]])
            del diff_values[0]
            for fields in diff_values:
                print('%d. %s'%(attribute_no,fields))
                attribute_no+=1
            what_data=int(input(' >> Select the sample data, for coloum
n `` %s ``,(Enter no) : '%(coloumn[key])))
            sample_data[coloumn[key]]=diff_values[what_data-1]
            del diff_values[:]
            diff_values.append('empty')
            print("\n")


def choose_value():
    global to_find
    to_find=int(input(' >> Choose the coloumn to predict, (Only Enter,
the number besides your choice):'))
    print(' \n >> Predicting Coloumn : %s  '%(coloumn[to_find]))
    global end
    end=len(data)
    for i in range(0,end):
        return_different_class_value(data[i][coloumn[to_find]])
    print("\n INSERT SAMPLE DATA ")
    get_sample_data(to_find)

def get_index(value):
    for c in coloumn:
        if(value==coloumn[c]):
            return c

def find_probability(sample,classes):
    index=get_index(sample)
    k=0
    for i in range(0,end):
        if(sample_data[sample]==data[i][coloumn[index]] and classes==da
ta[i][coloumn[to_find]]):
            k+=1
    return k

def find_max(d):
     v=list(d.values())
     k=list(d.keys())
     return k[v.index(max(v))]

def get_laplacian_estimation(flag):
    calculation=[]
    for values in prob_dict:
```

```
        for classes in class_value:
            if(flag==True):
                lap_value=prob_dict[values][classes]+1
                lap_div=class_value[classes]+1
            else:
                lap_value=prob_dict[values][classes]
                lap_div=class_value[classes]
            probability=(lap_value/lap_div)
            prob_dict[values][classes]=probability
    for classes in class_value:
        del calculation[:]
        total=0
        for values in prob_dict:
            calculation.append(prob_dict[values][classes])
        for i in range(0,len(calculation)):
            if(i==1):
                total=calculation[i]*calculation[i-1]
            if(i>1):
                total=calculation[i]*total
        if(flag==True):
            lap_total=end+1
            lap_class=class_value[classes]+1
        else:
            lap_total=end
            lap_class=class_value[classes]
        total=total*(lap_class/lap_total)
        result[classes]=total
    answer=find_max(result)
    return answer

def print_dict_as_table(dict):
    df = pd.DataFrame(dict).T
    df.fillna(0, inplace=True)
    print(df)

def main():
    import csv
    with open('tennis.csv') as csvfile:
        reader = csv.DictReader(csvfile)
        sr_no=1
        for row in reader:
            data.append(row)
            sr_no+=1
    print('\n  SAMPLE DATA GIVEN =\n')
    print(pd.DataFrame(data,index=list(range(1,len(data)+1))))
    print('\n  SELECT A COLOUMN TO PREDICT USING NAIVE BIAS  \n')
    no_2=1
    for key in data[0]:
```

```
        print('%d. %s'%(no_2,key))
        coloumn[no_2]=key
        no_2+=1
    choose_value()
    global prob_dict
    laplacian=False
    prob_dict={}
    for samples in sample_data:
        temp_dict={}
        for classes in class_value:
            prob=find_probability(samples,classes)
            temp_dict[classes]=prob
        prob_dict[samples]=temp_dict
    print('\n SAMPLE DATA \n')
    for s in sample_data:
        print(' >> %s :  %s'%(s,sample_data[s]))
    print("\n DIFFERENT CLASS VALUES IN PREDICTION COLOUMN =\n")
    for c in class_value:
        print(' >> %s :  %s'%(c,class_value[c]))
    print('\n VALUES MATCHED FOR SAMPLE DATA \n')
    print_dict_as_table(prob_dict)
    for prob in prob_dict:
        for classes in class_value:
            if(prob_dict[prob][classes]== 0):
                laplacian=True
    print("\n  HYPOTHESIS FOR EACH SAMPLE DATA =\n")
    if(laplacian==True):
        print('>> Applying Laplacian Smoothing.(Laplacian Smoothing = T
rue)\n')
        lap=True
        predicted_value=get_laplacian_estimation(lap)
    else:
        print(' >> Laplacian Smoothing not found.')
        lap=False
        predicted_value=get_laplacian_estimation(lap)
    print_dict_as_table(prob_dict)
    return predicted_value

if __name__=="__main__":
    answer=main()
    print('\n FINAL PROBABILITY FOR EACH PREDICTION \n')
    for i in result:
        print(' >> %s  : %s '%(i,result[i]))
    print('\n >>Prediction using naive bias is, %s = %s'%(coloumn[to_fi
nd],answer))
```

Output:

```
SAMPLE DATA GIVEN =

    outlook  temp humidity  windy play
1     sunny   hot     high  false   no
2     sunny   hot     high   true   no
3  overcast   hot     high  false  yes
4     rainy  mild     high  false  yes
5     rainy  cool   normal  false  yes
6     rainy  cool   normal   true   no
7  overcast  cool   normal   true  yes
8     sunny  mild     high  false   no
9     sunny  cool   normal  false  yes
10    rainy  mild   normal  false  yes
11    sunny  mild   normal   true  yes
12 overcast  mild     high   true  yes
13 overcast   hot   normal  false  yes
14    rainy  mild     high   true   no

SELECT A COLOUMN TO PREDICT USING NAIVE BIAS

1. outlook
2. temp
3. humidity
4. windy
5. play
 >> Choose the coloumn to predict, (Only Enter, the number besides your choice):1

 >> Predicting Coloumn : outlook

 INSERT SAMPLE DATA
1. hot
2. mild
3. cool
 >> Select the sample data, for coloumn `` temp ``,(Enter no) : 3


1. high
2. normal
 >> Select the sample data, for coloumn `` humidity ``,(Enter no) : 1


1. false
2. true
 >> Select the sample data, for coloumn `` windy ``,(Enter no) : 2


1. no
2. yes
 >> Select the sample data, for coloumn `` play ``,(Enter no) : 2
```

```
SAMPLE DATA

>> temp :  cool
>> humidity :  high
>> windy :  true
>> play :  yes

DIFFERENT CLASS VALUES IN PREDICTION COLOUMN =

>> sunny :  5
>> overcast :  4
>> rainy :  5

VALUES MATCHED FOR SAMPLE DATA

          sunny  overcast  rainy
temp          1         1      2
humidity      3         2      2
windy         2         2      2
play          2         4      3

  HYPOTHESIS FOR EACH SAMPLE DATA =

 >> Laplacian Smoothing not found.
          sunny  overcast  rainy
temp        0.2      0.25    0.4
humidity    0.6      0.50    0.4
windy       0.4      0.50    0.4
play        0.4      1.00    0.6
```

```
FINAL PROBABILITY FOR EACH PREDICTION

>> sunny  : 0.006857142857142858
>> overcast  : 0.017857142857142856
>> rainy  : 0.013714285714285719

>>Prediction using naive bias is, outlook = overcast
```

2. Construct Decision Tree Classifier for diabetes.csv file using predefined (like mlxtend) packages.

   **Calculate the accuracy of a classifier using K-Fold cross validation and Bootstrap method.

   Code:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

data =  pd.read_csv("diabetes.csv")
data.head()
```

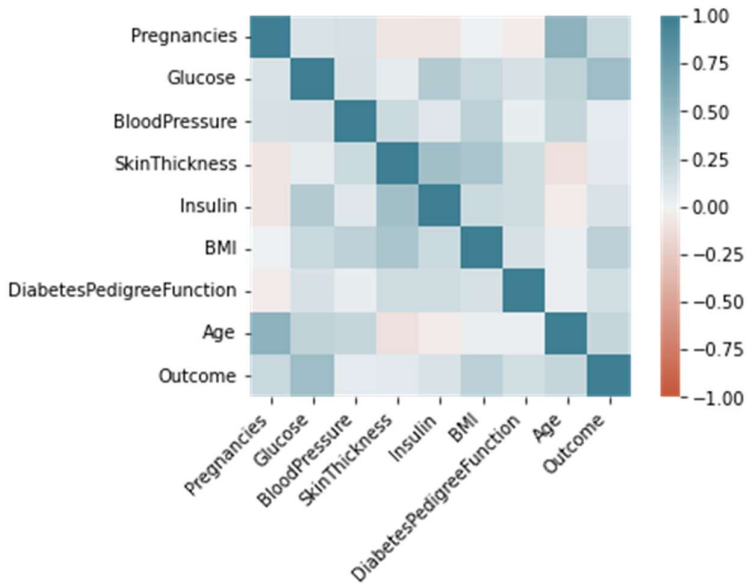
Output:

	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

Code:

```
import seaborn as sns
corr = data.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```

Output:



Code:

```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0
.3, random_state=1)
```

Code:

```
classifier = DecisionTreeClassifier()
classifier = classifier.fit(X_train, Y_train)
```

Code :

```
y_pred = classifier.predict(X_test)
print(y_pred)
```

Output :

```
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0 1 0 1 1 0 0
 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0
 1 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0
 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 0 0
 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0]
```

Code :

```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test, y_pred)
print(confusion_matrix(Y_test, y_pred))
```

output :

```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test, y_pred)
print(confusion_matrix(Y_test, y_pred))
```

```
[[111  35]
 [ 44  41]]
```

Code :

```
print("Accuracy:", metrics.accuracy_score(Y_test,y_pred))
```

output:

```
print("Accuracy:", metrics.accuracy_score(Y_test,y_pred))
```

```
Accuracy: 0.658008658008658
```

,

```
pip install -U scikit-learn

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (1.0.2)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.21.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.7.3)

[13] pip install six

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (1.15.0)
```

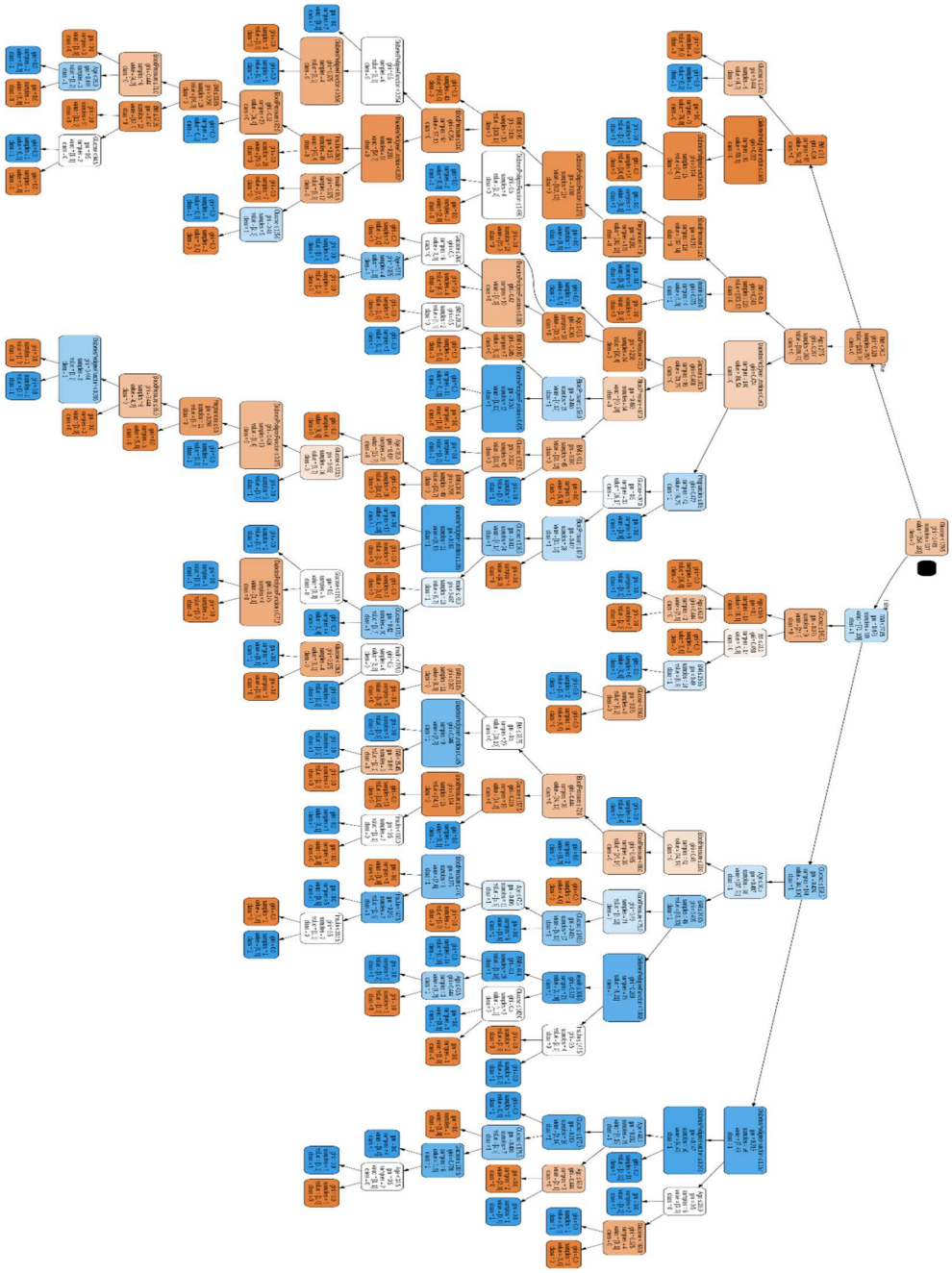
```
[15] pip install --upgrade scikit-learn==0.20.3

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn==0.20.3 in /usr/local/lib/python3.7/dist-packages (0.20.3)
Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.7.3)
Requirement already satisfied: numpy>=1.8.2 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.21.6)

[16] pip install sklearn

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting sklearn
  Downloading sklearn-0.0.post1.tar.gz (3.6 kB)
Building wheels for collected packages: sklearn
  Building wheel for sklearn (setup.py) ... done
  Created wheel for sklearn: filename=sklearn-0.0.post1-py3-none-any.whl size=2344 sha256=a252114ba252c04797815e046f21784e78175e824eae2
  Stored in directory: /root/.cache/pip/wheels/42/56/cc/4a8bf86613aafd5b7f1b310477667c1fca5c51c3ae4124a003
Successfully built sklearn
Installing collected packages: sklearn
Successfully installed sklearn-0.0.post1
```

```
!pip install --upgrade scikit-learn==0.20.3

g in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
ting scikit-learn==0.20.3
loading scikit_learn-0.20.3-cp37-cp37m-manylinux1_x86_64.whl (5.4 MB)
                                        | 5.4 MB 9.9 MB/s
ement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.7.3)
ement already satisfied: numpy>=1.8.2 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.21.6)
ling collected packages: scikit-learn
mpting uninstall: scikit-learn
und existing installation: scikit-learn 1.0.2
installing scikit-learn-1.0.2:
Successfully uninstalled scikit-learn-1.0.2
 pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the so
brick 1.5 requires scikit-learn>=1.0.0, but you have scikit-learn 0.20.3 which is incompatible.
nced-learn 0.8.1 requires scikit-learn>=0.24, but you have scikit-learn 0.20.3 which is incompatible.
sfully installed scikit-learn-0.20.3
G: The following packages were previously imported in this runtime:
earn]
st restart the runtime in order to use newly installed versions.

RT RUNTIME
```

Code:

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(classifier, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols,c
lass_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Output:



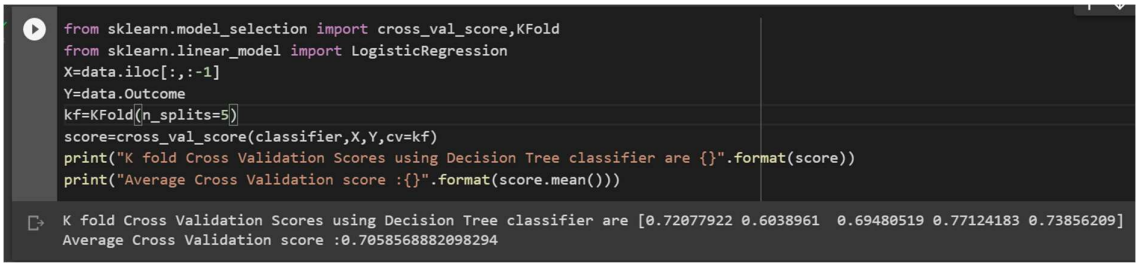
K-fold cross validation and bootstrap

Code:

```
from sklearn.model_selection import cross_val_score,KFold

from sklearn.linear_model import LogisticRegression
X=data.iloc[:,:-1]
Y=data.Outcome
kf=KFold(n_splits=5)
score=cross_val_score(classifier,X,Y,cv=kf)
```

```
print("K fold Cross Validation Scores using Decision Tree classifier ar
e {}".format(score))
print("Average Cross Validation score :{}".format(score.mean()))
```

output:



Code:

```
import numpy
from pandas import read_csv
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
n_iterations=1000
n_size=int(len(data)*0.50)
values=data.Outcome
stats=list()
for i in range(n_iterations):
  train=resample(values,n_samples=n_size)
  test=numpy.array([x for x in values if x.tolist() not in train.tolist
()])
  model=DecisionTreeClassifier()
  model.fit([train[:,:-1],train[:,-1]])
  predictions=model.predict(test[:,:-1])
  score=accuracy_score(test[:,-1],predictions)
```

output:

```
0.655982905982906
0.7130801687763713
0.6631130063965884
0.6802575107296137
0.7021276595744681
0.6919831223628692
0.6886993603411514
0.6708860759493671
0.6702355460385439
0.6865671641791045
```