EXPERIMENT NO-5

5) Execute the Naïve Bayes algorithm with suitable data set and do proper analysis on the result. Also implement Naïve Bayes algorithm using python.

AJINKYA NAM RA EN21CS301062 6A- CSE

```
import pandas as pd import numpy as np
iris=pd . read_csv( " / content/ iris_2d .
csv" )
```

Generate a slider using jupyter widgets	Close
Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro	

```
X iris = iris [ [ " petal length" , " petalwidth" ] ] . values
y_iris = iris [ ["class" ] ] . values

from sklearn . model_selection import train_test_split X_train,X_test,y_train,y_test
= train_test_split(X_iris,y_iris,test_size=.2,random_state=10)

import numpy as np from sklearn .
naive_bayes import GaussianNB gnb =
GaussianNB() gnb. fit (X_train, y_train) gnb .
class_count_ np.array( [40. , 37 . , 43 . ] ) gnb.
classes_
ray(['Iris-setosa ' , 'Iris-versicolor' , ' Iris-virginica ' ] , dtype='<U15' )

/usr/10ca1/1ib/python3.1Ø/dist-packages/sk1earn/uti1s/va1idation. py:1143: DataConversionWarning: A column-vector y was passed when a
  Id y = column_or_Id(y, warn-True)
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype='<U15')

y_predict=gnb . predict (X_test)

from sklearn import metrics metrics .

accuracy_ score (y_test, y_predict)
```

EXPERIMENT NO-6

6) Identify a data set for executing the Decision Tree algorithm to implement using python and analyse the same with cross validation and percentage split

AJINKYA NAM RA EN21CS301062 6A- CSE

```
import pandas as pd
import numpy as np
iris=pd.read_csv("/content/iris_2d.csv")
iris.head()
iris.columns
```

```
Index(['petallength', 'petalwidth', 'class'], dtype='object')
```

```
X_iris = iris[["petallength","petalwidth"]].values
X_iris
X_iris.shape
y_iris = iris[["class"]].values
y_iris
```

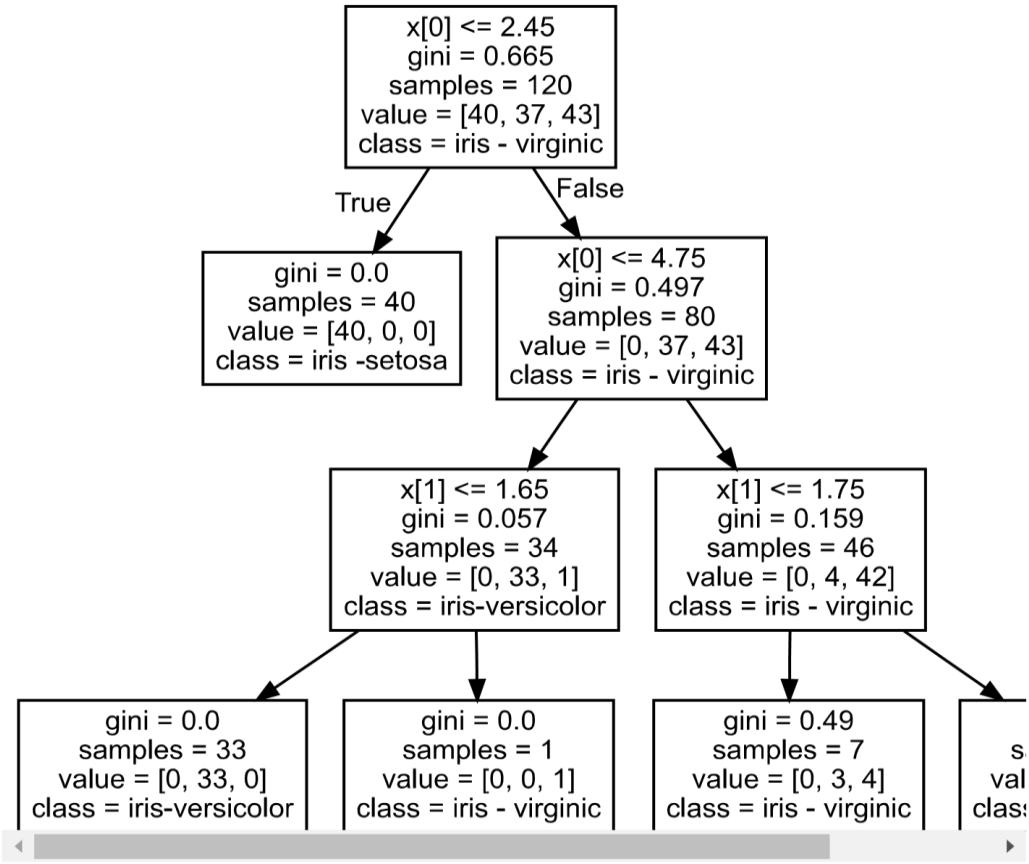
```
       ['Iris-versicolor'],
       ['Iris-versicolor'],
       ['Iris-versicolor'],
       ['Iris-versicolor'],
       ['Iris-versicolor'],
       ['Iris-versicolor'],
       ['Iris-versicolor'],
       ['Iris-versicolor'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica'],
       ['Iris-virginica']], dtype=object)
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_iris,y_iris,test_size=.2,random_state=10)
from sklearn.tree import DecisionTreeClassifier
```

```
tree_clf = DecisionTreeClassifier(max_depth=3 ,random_state = 42)
tree_clf.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=42)
```

```
from sklearn.tree import export_graphviz
export_graphviz(tree_clf,out_file = "iris_tree.dot" , class_names=np.array(['iris -setosa','iris-versicolor','iris - virginic']))
from graphviz import Source
Source.from_file("iris_tree.dot")
```




```
y_predct=tree_clf.predict(X_test)
y_predct
```

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

```
from sklearn import metrics
metrics.accuracy_score(y_test,y_predct)
0.9333333333333333
```

```
0.9333333333333333
```

```
tree_clf_entropy = DecisionTreeClassifier(max_depth=3,random_state=42,criterion='entropy')
tree_clf_entropy.fit(X_train,y_train)
from sklearn.tree import export_graphviz
export_graphviz(tree_clf_entropy,out_file="iris_tree_etropy.dot",class_names=np.array(['iris -setosa','iris-versicolor','iris - virginic']))
from graphviz import Source
Source.from_file("iris_tree_etropy.dot")
```