COURSE OUTCOME 3

PROGRAM - 8

AIM:

Program to implement text classification using Support vector machine.

DATASET:

SMSSpamCollection

```
import nltk
import pandas as pd
```

```
nltk.download_shell()
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> l

Packages:
  [ ] abc................. Australian Broadcasting Commission 2006
  [ ] alpino.............. Alpino Dutch Treebank
  [ ] averaged_perceptron_tagger Averaged Perceptron Tagger
  [ ] averaged_perceptron_tagger_ru Averaged Perceptron Tagger (Russian)
  [ ] basque_grammars..... Grammars for Basque
  [ ] biocreative_ppi..... BioCreAtIvE (Critical Assessment of Information
                           Extraction Systems in Biology)
  [ ] bllip_wsj_no_aux.... BLLIP Parser: WSJ Model
  [ ] book_grammars....... Grammars from NLTK Book
  [ ] brown............... Brown Corpus
  [ ] brown_tei........... Brown Corpus (TEI XML Version)
  [ ] cess_cat............ CESS-CAT Treebank
  [ ] cess_esp............ CESS-ESP Treebank
  [ ] chat80.............. Chat-80 Data Files
  [ ] city_database....... City Database
  [ ] cmudict............. The Carnegie Mellon Pronouncing Dictionary (0.6)
  [ ] comparative_sentences Comparative Sentence Dataset
  [ ] comtrans............ ComTrans Corpus Sample
  [ ] conll2000........... CONLL 2000 Chunking Corpus
  [ ] conll2002........... CONLL 2002 Named Entity Recognition Corpus
Hit Enter to continue: q

---------------------------------------------------------------------------
```

```
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> q
```

```
messages = [line.rstrip() for line in open('/content/SMSSpamCollection')]
print(len(messages))
```

```
5574
```

```
messages[0]
```

```
'ham\tGo until jurong point, crazy.. Available only in bugis n great world
la e buffet    Cine there got amore wat   '
```

```
for mess_no,message in enumerate(messages[:10]):
  print(mess_no,message)
  print('\n')
```

```
0 ham   Go until jurong point, crazy.. Available only in bugis n great worl


1 ham   Ok lar... Joking wif u oni...


2 spam  Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005.


3 ham   U dun say so early hor... U c already then say...


4 ham   Nah I don't think he goes to usf, he lives around here though


5 spam  FreeMsg Hey there darling it's been 3 week's now and no word back! I


6 ham   Even my brother is not like to speak with me. They treat me like ai


7 ham   As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vetta


8 spam  WINNER!! As a valued network customer you have been selected to rec


9 spam  Had your mobile 11 months or more? U R entitled to Update to the la
```

```
messages[0]
```

```
'ham\tGo until jurong point, crazy.. Available only in bugis n great world
la e buffet... Cine there got amore wat...'
```

```
import pandas as pd
```

```
messages=pd.read_csv('/content/SMSSpamCollection',sep='\t',names=['label','messag
messages.head()
```

	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
#classification tasks needs numerical features, So converting strips into vector
#1. function to split words from a sentence into list
#2. remove stopwards
```

```
import string
```

SAMPLE CODE FOR REMOVING PUNCTUATIONS AND STOPWORDS:

```
#removing punctutions
mess = "Sample message! Notice: it has punctuation."
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
nopunc = [c for c in mess if c not in string.punctuation]
nopunc
```

```
['S',
 'a',
 'm',
 'p',
 'l',
       'e',
       ' ',
       'm',
       'e',
       's',
       's',
       'a',
       'g',
       'e',
       ' ',
       'N',
       'o',
       't',
       'i',
       'c',
       'e',
       ' ',
       'i',
       't',
       ' ',
       'h',
       'a',
       's',
       ' ',
       'p',
       'u',
       'n',
       'c',
       't',
       'u',
       'a',
       't',
       'i',
       'o',
       'n']
```

```
nopunc = ''.join(nopunc)
nopunc
```

```
     'Sample message Notice it has punctuation'
```

```
#removing stopwords
#for this, we need to download stopword's corpus from nltk.corpus import stopword
from nltk.corpus import stopwords
```

```
import nltk
nltk.download('stopwords')
```

```
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Package stopwords is already up-to-date!
     True
```

```
nopunc.split()
```

```
     ['Sample', 'message', 'Notice', 'it', 'has', 'punctuation']
```

```
clean_mess = [word for word in nopunc.split() if word.lower() not in stopwords.wo
```

```
clean_mess
```

```
     ['Sample', 'message', 'Notice', 'punctuation']
```

```
#apply above function in our actual dataset
def text_process(mess):
  nopunc=[char for char in mess if char not in string.punctuation]
  nopunc="".join(nopunc)
  return[word for word in nopunc.split() if word.lower() not in stopwords.words("
```

```
  messages.head()
```

	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
#tockenize
messages['message'].head(5).apply(text_process)
```

```
     0    [Go, jurong, point, crazy, Available, bugis, n...
     1                        [Ok, lar, Joking, wif, u, oni]
     2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
     3        [U, dun, say, early, hor, U, c, already, say]
     4    [Nah, dont, think, goes, usf, lives, around, t...
     Name: message, dtype: object
```

```
#converting tokens into vectors so that our machine learning models get understan
```

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
bow_transformer=CountVectorizer(analyzer=text_process).fit(messages['message'])
```

```
print(len(bow_transformer.vocabulary_))
```

```
11425
```

```
mess4=messages['message'][6]
print(mess4)
```

```
Even my brother is not like to speak with me. They treat me like aids patent
```

```
bow4=bow_transformer.transform([mess4])
```

```
print(bow4)
```

```
bow_transformer.get_feature_names()[7800]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Futu
  warnings.warn(msg, category=FutureWarning)
'like'
```

```
#apply this transformation for the whole message column in the dataset
messages_bow=bow_transformer.transform(messages['message'])
print('shape of the Sparse Matrix:',messages['message'])
```

```
shape of the Sparse Matrix: 0       Go until jurong point, crazy.. Available
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                 Will ü b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                           Rofl. Its true to its name
Name: message, Length: 5572, dtype: object
```

```
#check how many nonzero occurences
messages_bow.nnz
```

```
     50548
```

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY

```
#difference between TF and DF
#TF is frequency counter for a term t in document d.
#DF is the count of occurences of term t in the documents set N
from sklearn.feature_extraction.text import TfidfTransformer
```

```
tfidf_transformer=TfidfTransformer().fit(messages_bow)
```

```
tfidf4=tfidf_transformer.transform(bow4)
print(tfidf4)
```

```
       (0, 10629)    0.3352766696931058
       (0, 9971)     0.3268691780062757
       (0, 8761)     0.43700993321905807
       (0, 7800)     0.41453906826037096
       (0, 5193)     0.33843411088434017
       (0, 4590)     0.43700993321905807
       (0, 1802)     0.3352766696931058
```

```
#converting the whole bag of words into tfidf
messages_tfidf = tfidf_transformer.transform(messages_bow)
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
spam_detect_model=MultinomialNB().fit(messages_tfidf,messages['label'])
```

```
all_pred = spam_detect_model.predict(messages_tfidf)
```

```
all_pred
```

```
     array(['ham', 'ham', 'spam', ..., 'ham', 'ham', 'ham'], dtype='<U4')
```

```
from sklearn.model_selection import train_test_split
```

```
msg_train,msg_test,label_train,label_test = train_test_split(messages['message'],
```

```
spam_detect_model=MultinomialNB().fit(messages_tfidf,messages['label'])
```

```
predict = spam_detect_model.predict(messages_tfidf)
```

```
from sklearn.metrics import classification_report
print(classification_report(messages['label'],predict))
```

```
                   precision    recall  f1-score   support

             ham       0.98      1.00      0.99      4825
            spam       1.00      0.85      0.92       747

        accuracy                           0.98      5572
       macro avg       0.99      0.92      0.95      5572
    weighted avg       0.98      0.98      0.98      5572
```

TRAIN TEST SPLIT

```
from sklearn.model_selection import train_test_split
msg_train,msg_test,label_test,label_train = \
train_test_split(messages['message'],messages['label'],test_size=0.2)
```

CREATING A DATA PIPELINE

```
#pipeline
from sklearn.pipeline import Pipeline
pipeline = Pipeline([
            ('bow',CountVectorizer(analyzer=text_process)),       #strings to tok
            ('tfidf',TfidfTransformer()),                         #integer counts
            ('classifier',MultinomialNB()),                       #train on TF-ID
])
```

```
pipeline.fit(msg_test,label_train)
```

```
Pipeline(steps=[('bow',
                 CountVectorizer(analyzer=<function text_process at 0x7f1c62
                ('tfidf', TfidfTransformer()),
                ('classifier', MultinomialNB())])
```

```
predictions = pipeline.predict(msg_test)
```

```
print(classification_report(predictions,label_train))
```

```
              precision    recall  f1-score   support

         ham       1.00      0.95      0.97      1019
        spam       0.64      1.00      0.78        96

    accuracy                           0.95      1115
   macro avg       0.82      0.97      0.88      1115
weighted avg       0.97      0.95      0.96      1115
```

SVM CLASSIFIER

```
from sklearn import model_selection,naive_bayes,svm
from sklearn.metrics import accuracy_score
```

```
#classifer - algorithm - SVM
#fit the training dataset on the classifier
pipeline1 = Pipeline([
                      ('bow',CountVectorizer(analyzer=text_process)),     #string
                      ('tfidf',TfidfTransformer()),                        #intege
                      ('classifier',svm.SVC(C=1.0,kernel='linear',degree=3,gamma=
])
```

```
pipeline1.fit(msg_test,label_train)
```

```
Pipeline(steps=[('bow',
                 CountVectorizer(analyzer=<function text_process at 0x7f1c62
                ('tfidf', TfidfTransformer()),
                ('classifier', SVC(gamma='auto', kernel='linear'))])
```

```
predictions1 = pipeline1.predict(msg_test)
```

```
print(classification_report(predictions1,label_train))
```

```
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00       967
        spam       0.99      1.00      1.00       148

    accuracy                           1.00      1115
   macro avg       1.00      1.00      1.00      1115
weighted avg       1.00      1.00      1.00      1115
```

RESULT:

The program executed successfully and obtained the output.

PROGRAM - 9

AIM:

Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris

#load·iris·data
data·=·load_iris()
```

```
data.data.shape
```

```
     (150, 4)
```

```
print('classes to predict: ',data.target_names)
print('Features: ',data.feature_names)
```

```
     classes to predict:  ['setosa' 'versicolor' 'virginica']
     Features:  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'p
```

```
x = data.data
y = data.target
display(x.shape, y.shape)
```

```
     (150, 4)
     (150,)
```

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

x_train, x_test, y_train, y_test = train_test_split(x,y,random_state = 50, test_s
```

```
#default criterion is GINI
classifier = DecisionTreeClassifier()
classifier.fit(x_train,y_train)
```

```
     DecisionTreeClassifier()
```

```
y_pred = classifier.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
print('Accuracy on train data using Gini: ',accuracy_score(y_train,classifier.pre
print('Accuracy on test data using Gini: ',accuracy_score(y_test,y_pred))
```

```
     Accuracy on train data using Gini:  1.0
     Accuracy on test data using Gini:  0.9473684210526315
```

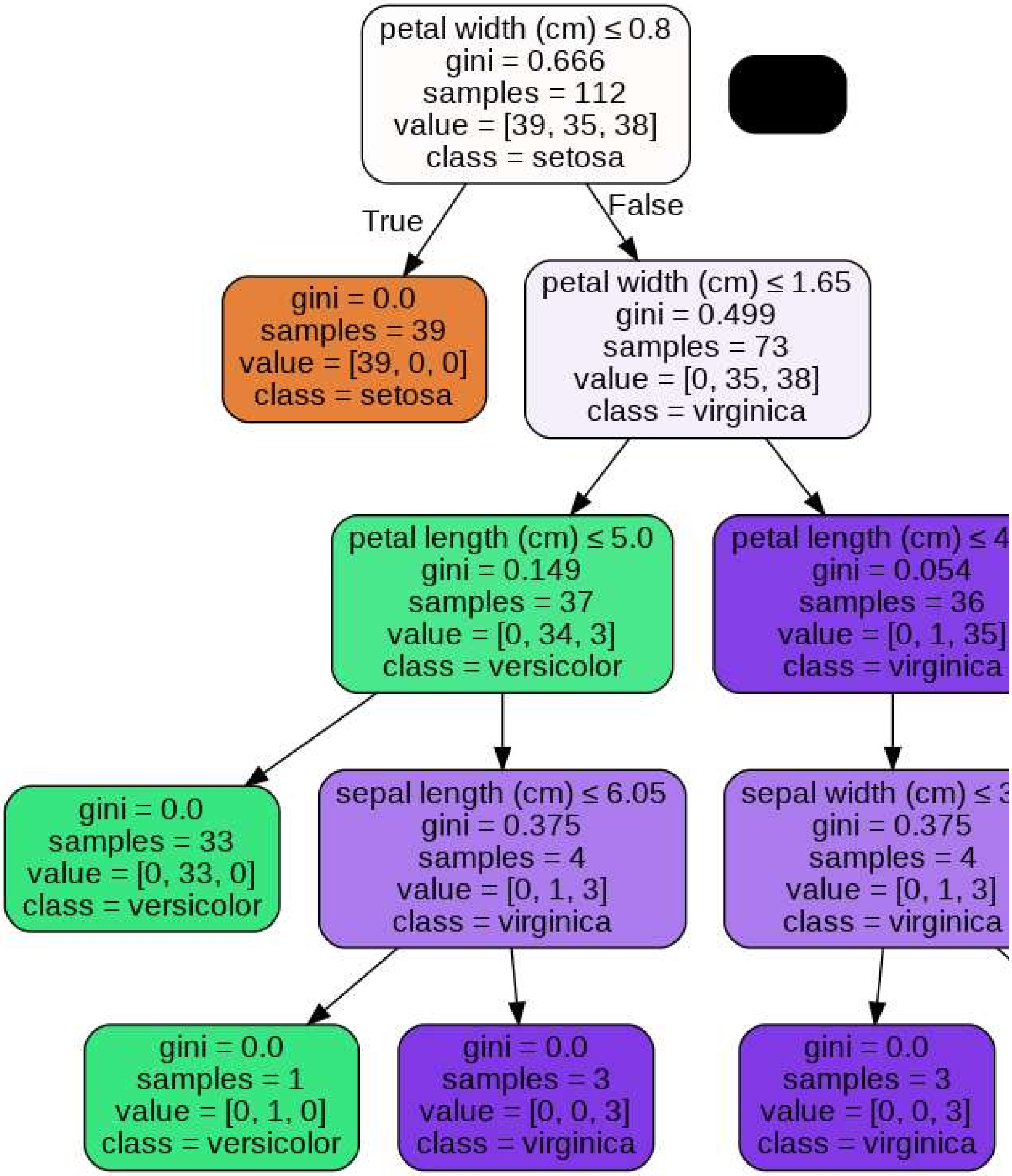
```
#change criterion to entropy
classifier_entropy = DecisionTreeClassifier(criterion='entropy')
classifier_entropy.fit(x_train,y_train)
y_pred_entropy = classifier_entropy.predict(x_test)
print('Accuracy on train data using Gini: ',accuracy_score(y_train,classifier_ent
print('Accuracy on test data using Gini: ',accuracy_score(y_test,y_pred_entropy))
```

```
     Accuracy on train data using Gini:  1.0
     Accuracy on test data using Gini:  0.9473684210526315
```

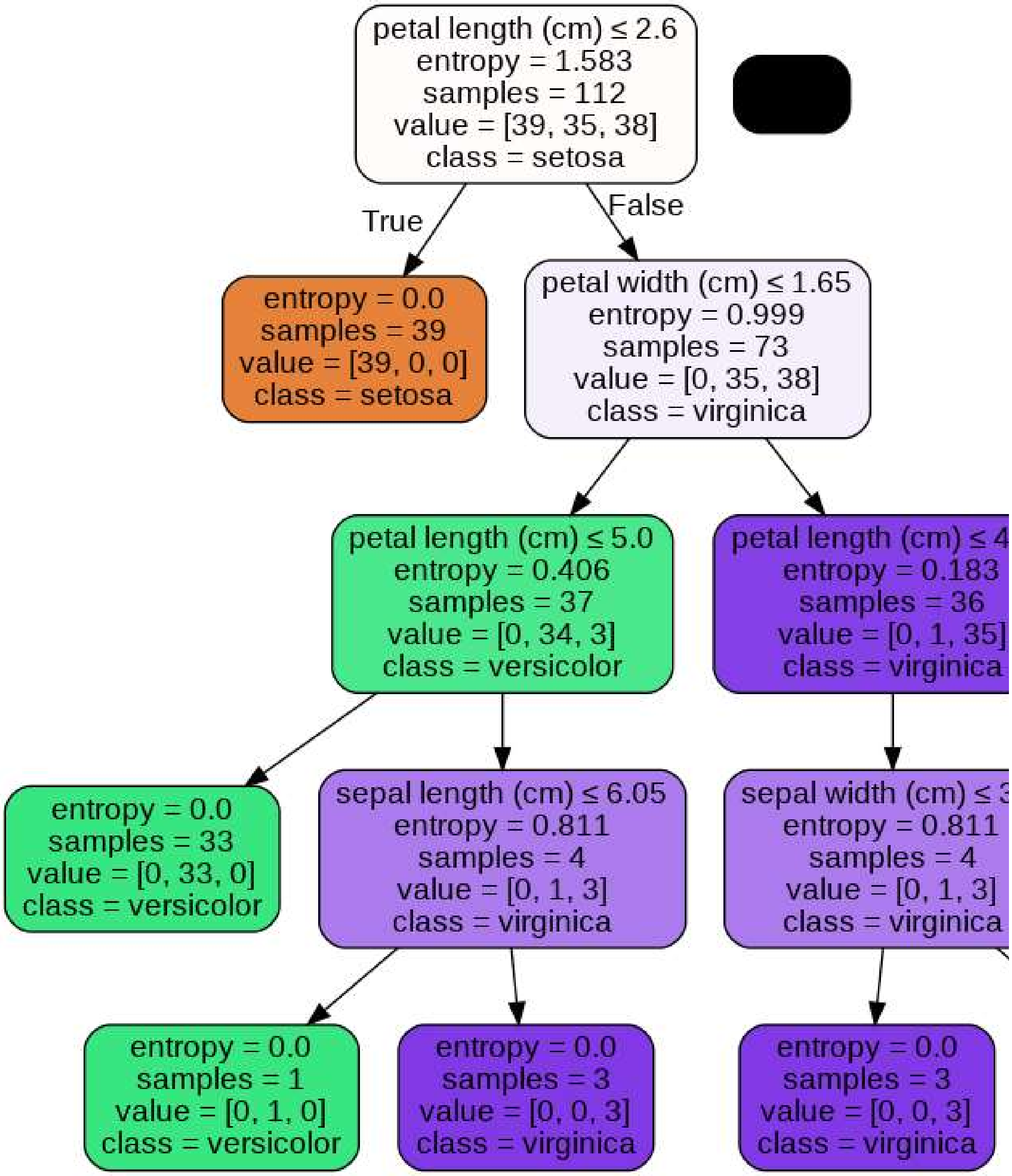
```
#change criterion to entropy with min_samples_split to 50. Default value is 2.
classifier_entropy1 = DecisionTreeClassifier(criterion = 'entropy', min_samples_s

classifier_entropy1.fit(x_train,y_train)
y_pred_entropy1 = classifier_entropy1.predict(x_test)
print('Accuracy on train data using entropy: ',accuracy_score(y_true = y_train,y_
print('Accuracy on test data using entropy: ',accuracy_score(y_true = y_test,y_pr
```

```
     Accuracy on train data using entropy:  0.9642857142857143
     Accuracy on test data using entropy:  0.9473684210526315
```

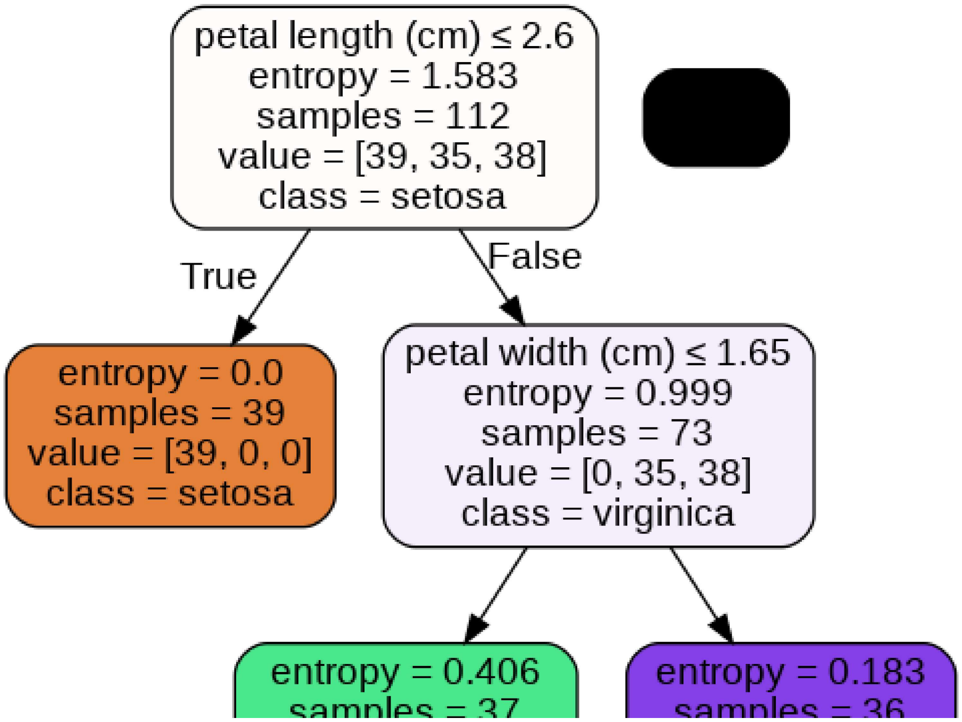
```
#visualize the decision tree

from sklearn.tree import export_graphviz  #for visualization
from six import StringIO   #python 2,3 compatibility package, when the stringIO o
                           #it is initialised by passing a string to the construc
from IPython.display import Image   #Ipython is an interactive shell that is buil
import pydotplus    #python interface to Graphviz's Dot language

dot_data = StringIO()
export_graphviz(classifier, out_file = dot_data, filled = True, rounded = True, s
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



```
dot_data = StringIO()
export_graphviz(classifier_entropy, out_file = dot_data, filled = True, impurity
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



```
dot_data = StringIO()
export_graphviz(classifier_entropy1, out_file = dot_data, filled = True, rounded
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



RESULT:

The program executed successfully and obtained the output.

PROGRAM - 10

AIM:

Program to implement text classification using Support vector machine.

DATASET:

College_Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import·seaborn·as·sns
```

```
df = pd.read_csv('/content/College_Data.csv',index_col=0)
```

```
df.head()
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad
Abilene Christian University	Yes	1660	1232	721	23	52	2885
Adelphi University	Yes	2186	1924	512	16	29	2683
Adrian College	Yes	1428	1097	336	22	50	1036
Agnes Scott College	Yes	417	349	137	60	89	510
Alaska Pacific University	Yes	193	146	55	16	44	249

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 777 entries, Abilene Christian University to York College of Pennsylv
```

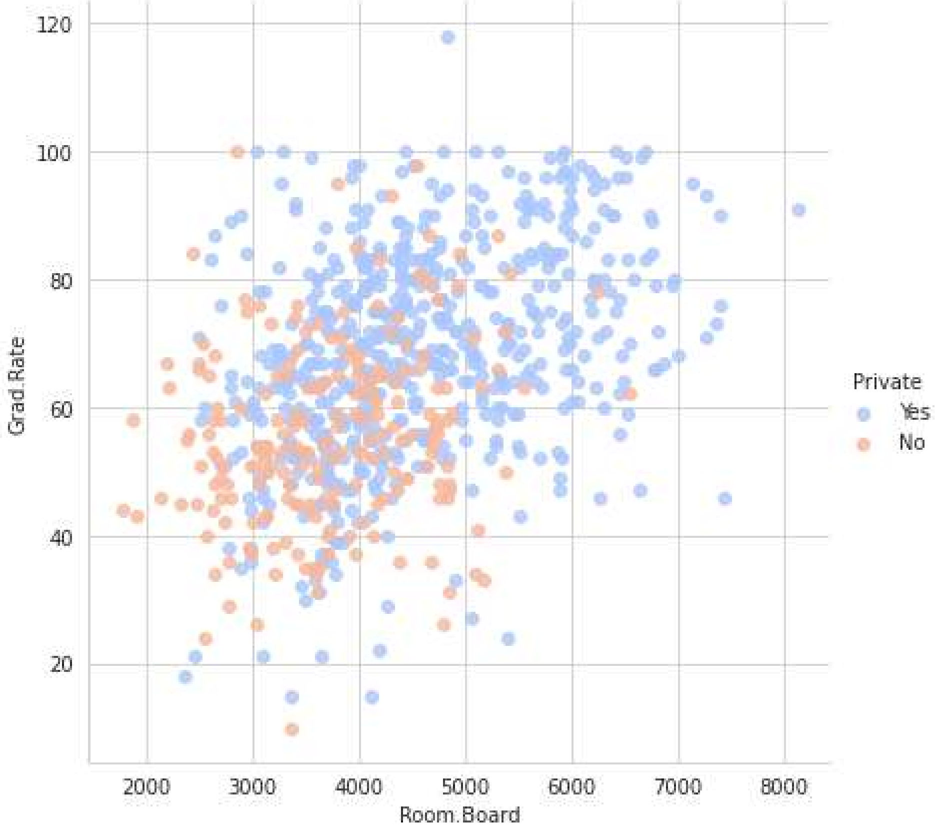
```
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Private      777 non-null    object 
 1   Apps         777 non-null    int64  
 2   Accept       777 non-null    int64  
 3   Enroll       777 non-null    int64  
 4   Top10perc    777 non-null    int64  
 5   Top25perc    777 non-null    int64  
 6   F.Undergrad  777 non-null    int64  
 7   P.Undergrad  777 non-null    int64  
 8   Outstate     777 non-null    int64  
 9   Room.Board   777 non-null    int64  
 10  Books        777 non-null    int64  
 11  Personal     777 non-null    int64  
 12  PhD          777 non-null    int64  
 13  Terminal     777 non-null    int64  
 14  S.F.Ratio    777 non-null    float64
 15  perc.alumni  777 non-null    int64  
 16  Expend       777 non-null    int64  
 17  Grad.Rate    777 non-null    int64  
dtypes: float64(1), int64(16), object(1)
memory usage: 115.3+ KB
```

```
df.describe()
```

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Unc
count	777.000000	777.000000	777.000000	777.000000	777.000000	777
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850
min	81.000000	72.000000	35.000000	1.000000	9.000000	139
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643

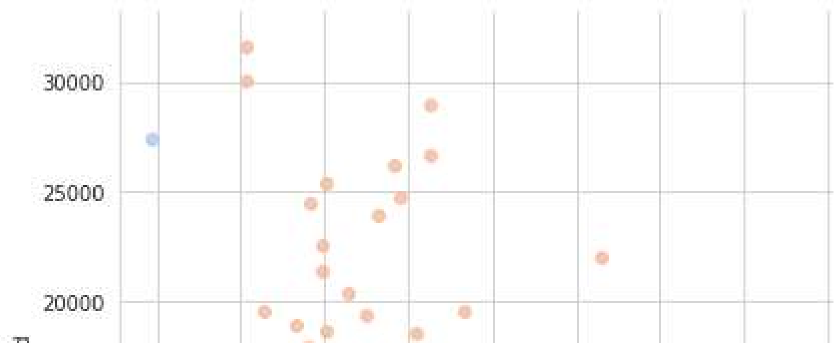
```
sns.set_style('whitegrid')
sns.lmplot('Room.Board','Grad.Rate',data=df, hue='Private',
           palette='coolwarm',size=6,aspect=1,fit_reg=False)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarr
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/regression.py:581: UserWarnir
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7f017f25bed0>
```



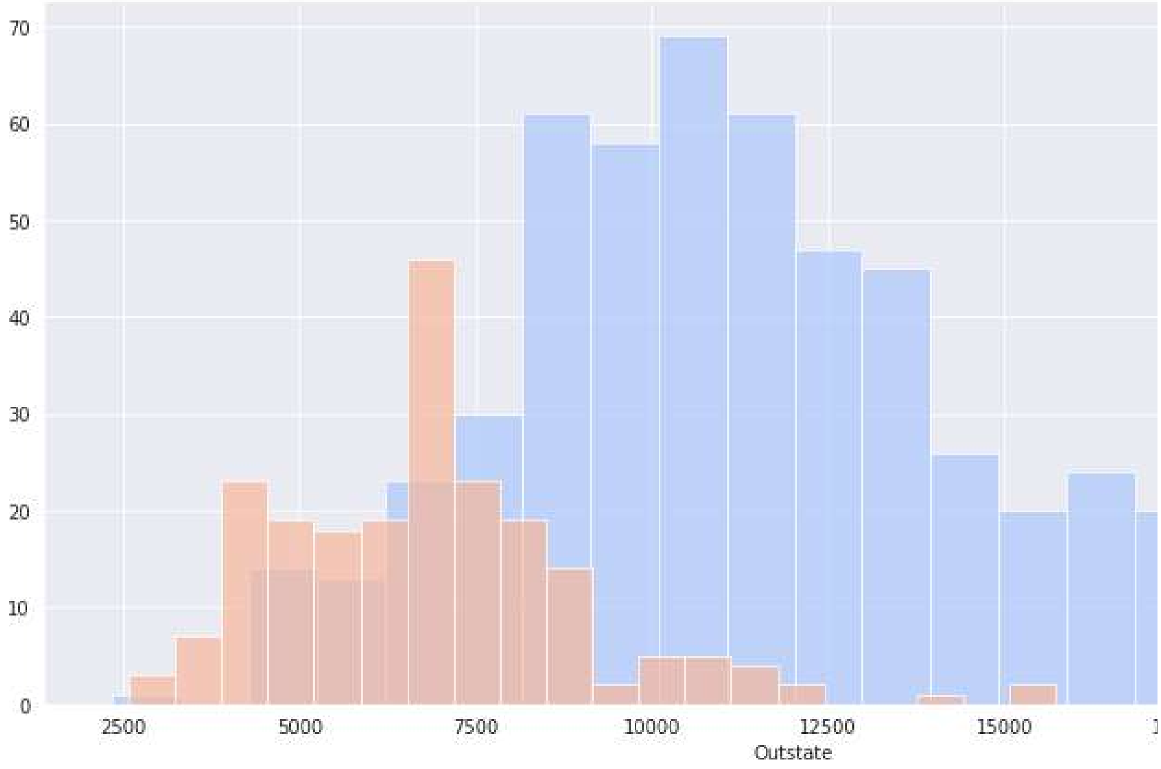
```
sns.set_style('whitegrid')
sns.lmplot('Outstate','F.Undergrad',data=df, hue='Private',
           palette='coolwarm',size=6,aspect=1,fit_reg=False)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarr
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/regression.py:581: UserWarnir
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7f017690dd50>
```



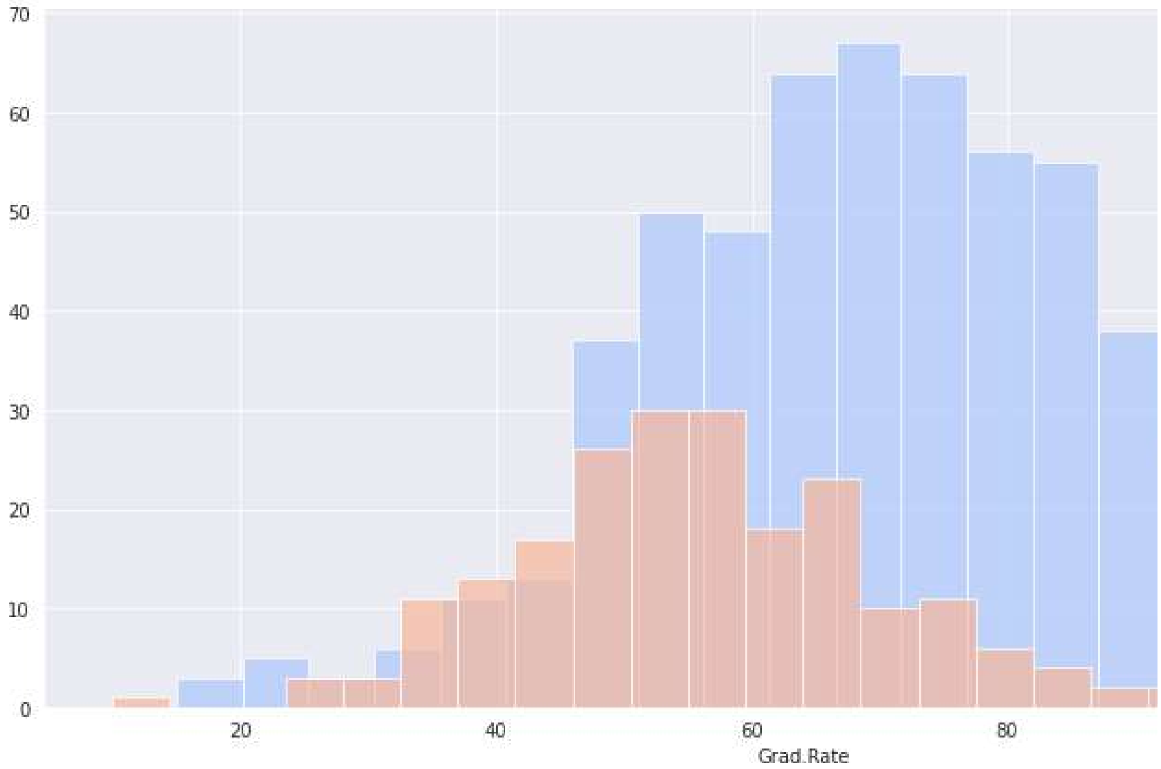
```
sns.set_style('darkgrid')
g = sns.FacetGrid(df,hue="Private",palette='coolwarm',size=6,aspect=2)
g = g.map(plt.hist,'Outstate',bins=20,alpha=0.7)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning
  warnings.warn(msg, UserWarning)
```



```
sns.set_style('darkgrid')
g = sns.FacetGrid(df,hue="Private",palette='coolwarm',size=6,aspect=2)
g = g.map(plt.hist,'Grad.Rate',bins=20,alpha=0.7)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning
  warnings.warn(msg, UserWarning)
```



```
df[df['Grad.Rate'] > 100]
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad
Cazenovia College	Yes	3847	3433	527	9	35	1010

```
df['Grad.Rate']['Cazenovia College'] = 100
```

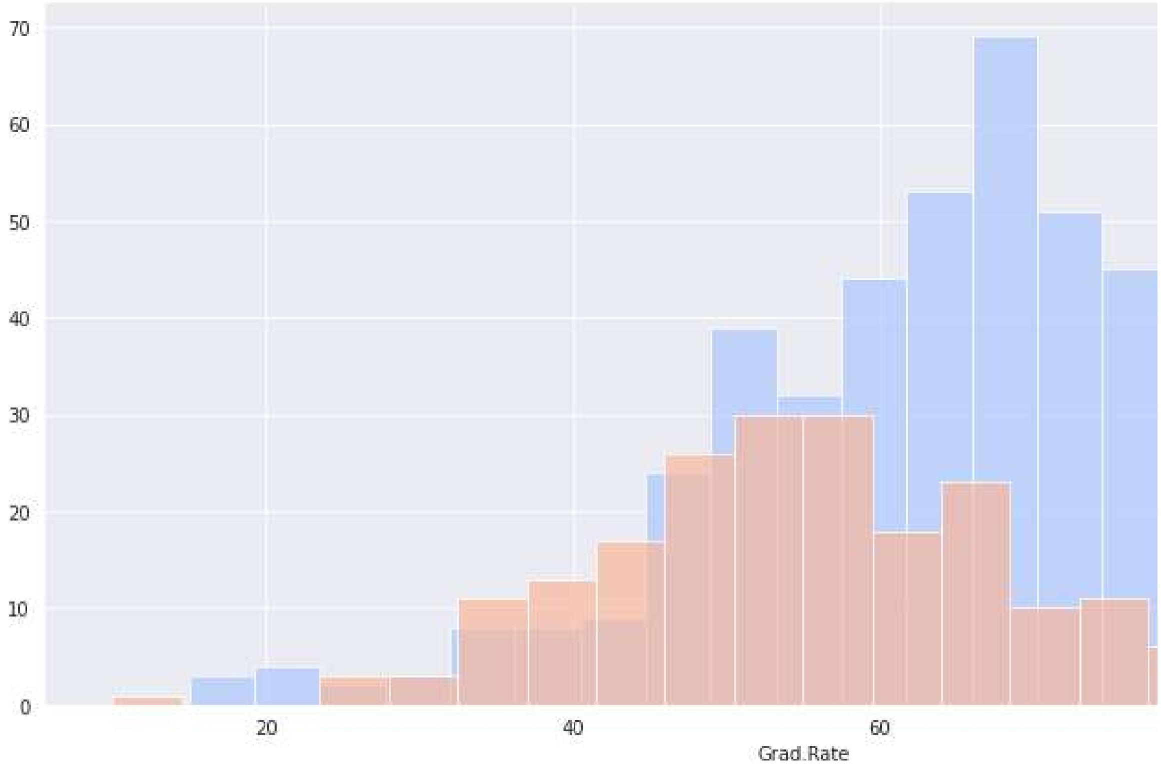
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWith(
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
  """Entry point for launching an IPython kernel.
```

```
df[df['Grad.Rate'] > 100]
```

Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Underg

```
sns.set_style('darkgrid')
g = sns.FacetGrid(df,hue="Private",palette='coolwarm',size=6,aspect=2)
g = g.map(plt.hist,'Grad.Rate',bins=20,alpha=0.7)
```

```
    /usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning
      warnings.warn(msg, UserWarning)
```



```
from sklearn.cluster import KMeans
```

```
kmeans = KMeans(n_clusters=2)
```

```
kmeans.fit(df.drop('Private',axis=1))
```

```
    KMeans(n_clusters=2)
```

```
kmeans.cluster_centers_
```

```
    array([[1.81323468e+03, 1.28716592e+03, 4.91044843e+02, 2.53094170e+01,
            5.34708520e+01, 2.18854858e+03, 5.95458894e+02, 1.03957085e+04,
            4.31136472e+03, 5.41982063e+02, 1.28033632e+03, 7.04424514e+01,
            7.78251121e+01, 1.40997010e+01, 2.31748879e+01, 8.93204634e+03,
            6.50926756e+01],
           [1.03631389e+04, 6.55089815e+03, 2.56972222e+03, 4.14907407e+01,
             7.02037037e+01, 1.30619352e+04, 2.46486111e+03, 1.07191759e+04,
             4.64347222e+03, 5.95212963e+02, 1.71420370e+03, 8.63981481e+01,
             9.13333333e+01, 1.40277778e+01, 2.00740741e+01, 1.41705000e+04,
             6.75925926e+01]])
```

EVALUATION

```
def converter(cluster):
    if cluster=='Yes':
        return 1
    else:
        return 0
```

```
df['Cluster'] = df['Private'].apply(converter)
```

```
df.head()
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad
Abilene Christian University	Yes	1660	1232	721	23	52	2885
Adelphi University	Yes	2186	1924	512	16	29	2683
Adrian College	Yes	1428	1097	336	22	50	1036
Agnes Scott College	Yes	417	349	137	60	89	510
Alaska Pacific University	Yes	193	146	55	16	44	249

```
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(df['Cluster'],kmeans.labels_))
print(classification_report(df['Cluster'],kmeans.labels_))
```

```
[[138  74]
 [531  34]]
              precision    recall  f1-score   support

           0       0.21      0.65      0.31       212
           1       0.31      0.06      0.10       565

    accuracy                           0.22       777
   macro avg       0.26      0.36      0.21       777
weighted avg       0.29      0.22      0.16       777
```

RESULT:

The program executed successfully and obtained the output.