DATA SCIENCE
LAB RECORD

COURSE OUTCOME 3

SREELEKSHMI PRATHAPAN
20MCA340

CO3-1

AIM:

Program to implement text classification using Support vector machine.

DATASET:SMSSpamCollection

PROGRAM:

```
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 CorpusHit Enter to continue:
d
  [ ] conll2007..... Dependency Treebanks from CoNLL 2007 (Catalan
                                        and Basque Subset)[ ]
  crubadan..... Crubadan Corpus
```

```
    [ ] dependency_treebank. Dependency Parsed Treebank[ ]
    dolch..... Dolch Word List
    [ ] europarl_raw..... Sample European Parliament Proceedings Parallel
                                        Corpus
    [ ] floresta..... Portuguese Treebank
    [ ] framenet_v15..... FrameNet 1.5
    [ ] framenet_v17..... FrameNet 1.7
    [ ] gazetteers..... Gazeteer Lists
    [ ] genesis..... Genesis Corpus
    [ ] gutenberg..... Project Gutenberg Selections[ ] ieer..... NIST
    IE-ER DATA SAMPLE
    [ ] inaugural..... C-Span Inaugural Address Corpus
    [ ] indian..... Indian Language POS-Tagged Corpus
    [ ] jeita..... JEITA Public Morphologically Tagged Corpus (in
                                        ChaSen format)
    [ ] kimmo..... PC-KIMMO Data Files
    [ ] knbc..... KNB Corpus (Annotated blog corpus)
    [ ] large_grammars..... Large context-free and feature-based grammarsfor parser comparison
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/sample_data/SMSSpamCollect
```

```
print(len(messages))
```

```
5574
```

```
messages[0]
```

```
'ham\tGo until jurong point, crazy.. Available only in bugis n great world la e b
uffet          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 world la e buffe
/n
1 ham      Ok lar... Joking wif u oni...
/n
2 spam     Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to
/n
3 ham      U dun say so early hor... U c already then say...
/n
4 ham      Nah I don't think he goes to usf, he lives around here though
/n
5 spam     FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like som
```

```
/n
6  ham      Even my brother is not like to speak with me. They treat me like aids patent.
/n
7  ham      As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been
/n
8  spam     WINNER!! As a valued network customer you have been selected to receivea £900 p
/n
9  spam     Had your mobile 11 months or more? U R entitled to Update to the latest colour
/n
```

```
messages[0]
```

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

```
import pandas as pd
```

```
messages=pd.read_csv('/content/sample_data/SMSSpamCollection',sep='\t',names=
```

```
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...

```
#convert string into vector format
import string
```

```
#remove punctuations and stopwords
mess="sample messages ! 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',
 's',
 ' ',
 ' ',
'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 messages           Notice it has punctuation '
```

```
#remove stopwords
from nltk.corpus import stopwords
```

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...[nltk_data]
                Unzipping corpora/stopwords.zip.
True
```

```
nopunc.split()
```

```
['sample', 'messages', 'Notice', 'it', 'has', 'punctuation']
```

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

```
['sample', 'messages', 'Notice', 'punctuation']
```

```
#apply to 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.wor
```

```
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...

```
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 into vectors
```

```
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]
mess4
```

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

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

```
print(bow4)
```

```
  (0,   1802)       1
  (0,   4590)       1
  (0,   5193)       1
  (0,   7800)       2
  (0,   8761)       1
  (0,   9971)       1
  (0,   10629)      1
```

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

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

'like'

```
#transform whole image
messages_bow=bow_transformer.transform(messages['message'])
```

```
print('shape of sparse matrix:',messages_bow.shape)
```

shape of sparse matrix: (5572, 11425)

```
messages_bow.nnz
```

50548

```
#term frequency,inverse document frequency
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
```

```
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)
```

```
#pipeline
from sklearn.pipeline import Pipeline
pipeline = Pipeline([('bow',CountVectorizer(analyzer=text_process)),
                     ('tfidf',TfidfTransformer()),
                     ('classifier',MultinomialNB()),
                     ])
```

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

```
Pipeline(steps=[('bow',
                 CountVectorizer(analyzer=<function text_process at 0x7fcddf22f830>)),('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	0.98	1021
spam	0.67	1.00	0.80	94
accuracy			0.96	1115
macro avg	0.83	0.98	0.89	1115
weighted avg	0.97	0.96	0.96	1115

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

```
pipeline1  =  Pipeline([('bow',CountVectorizer(analyzer=text_process)),
                        ('tfidf',TfidfTransformer()),
                        ('classifier',svm.SVC(C=1.0,kernel='linear',degree=3,gam
                        ])
```

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

```
Pipeline(steps=[('bow',
                 CountVectorizer(analyzer=<function text_process at 0x7fcddf22f830>)),('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	976
spam	0.99	1.00	0.99	139
accuracy			1.00	1115
macro avg	0.99	1.00	1.00	1115
weighted avg	1.00	1.00	1.00	1115

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

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

	precision	recall	f1-score	support
ham	1.00	1.00	1.00	976
spam	0.99	1.00	0.99	139
accuracy			1.00	1115
macro avg	0.99	1.00	1.00	1115
weighted avg	1.00	1.00	1.00	1115

CO3-2

AIM:

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

PROGRAM :

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

```
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)', 'petal width (
```

```
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_si
```

```
#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
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
```

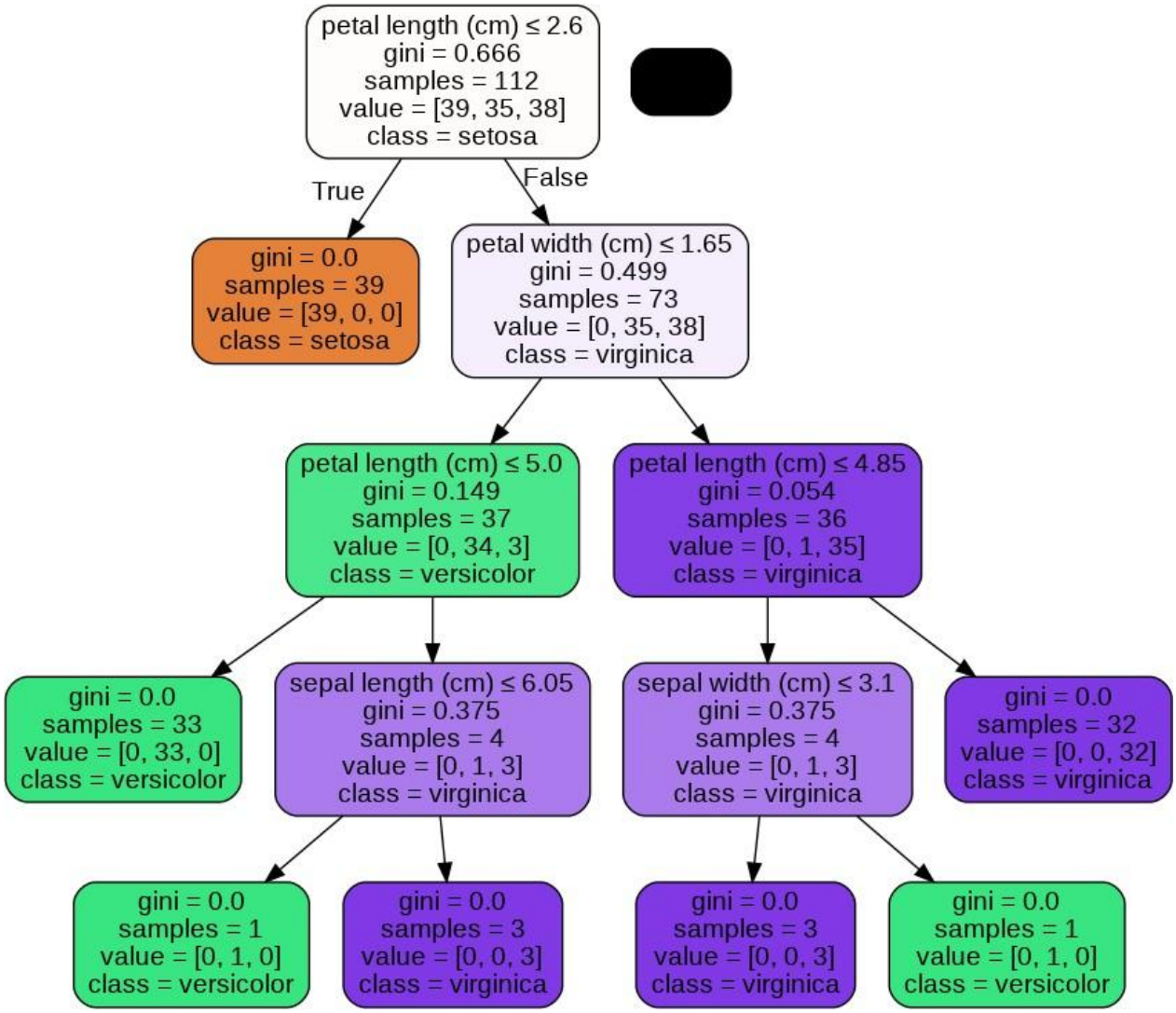
```
#change criterion to entropy
classifier_entropy = DecisionTreeClassifier(criterion='entropy')
classifier_entropy.fit(x_train,y_train)
y_pred_entropy = classifier_entropy.predict(x_test)
print('Accuracy on train data using Gini: ',accuracy_score(y_train,classifier
print('Accuracy on test data using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

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

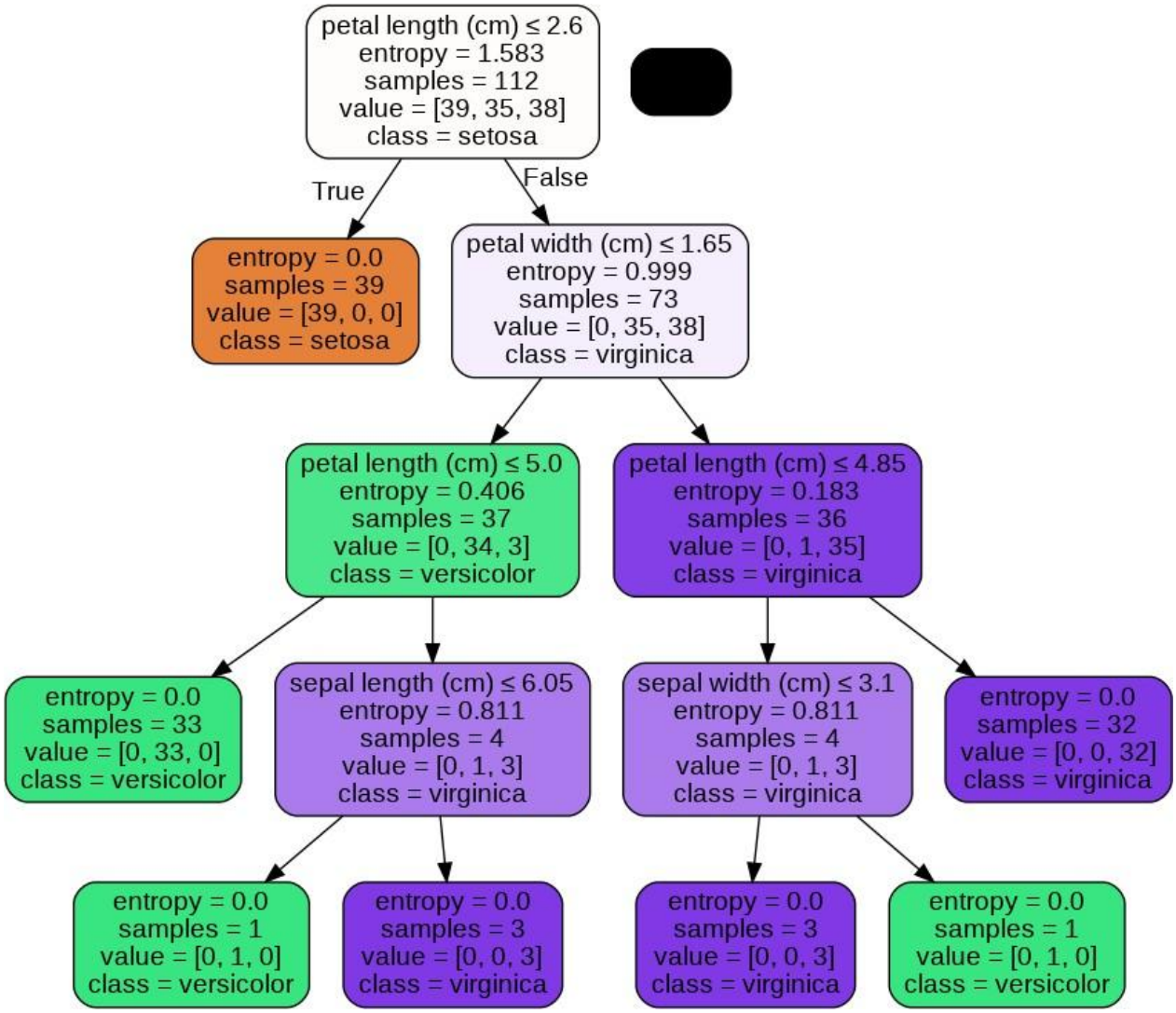
```
classifier_entropy1 = DecisionTreeClassifier(criterion = 'entropy', min_sampl
classifier_entropy1.fit(x_train,y_train)
y_pred_entropy1 = classifier_entropy1.predict(x_test)
print('Accuracy on train data using entropy: ',accuracy_score(y_true = y_trai
print('Accuracy on test data using entropy: ',accuracy_score(y_true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

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

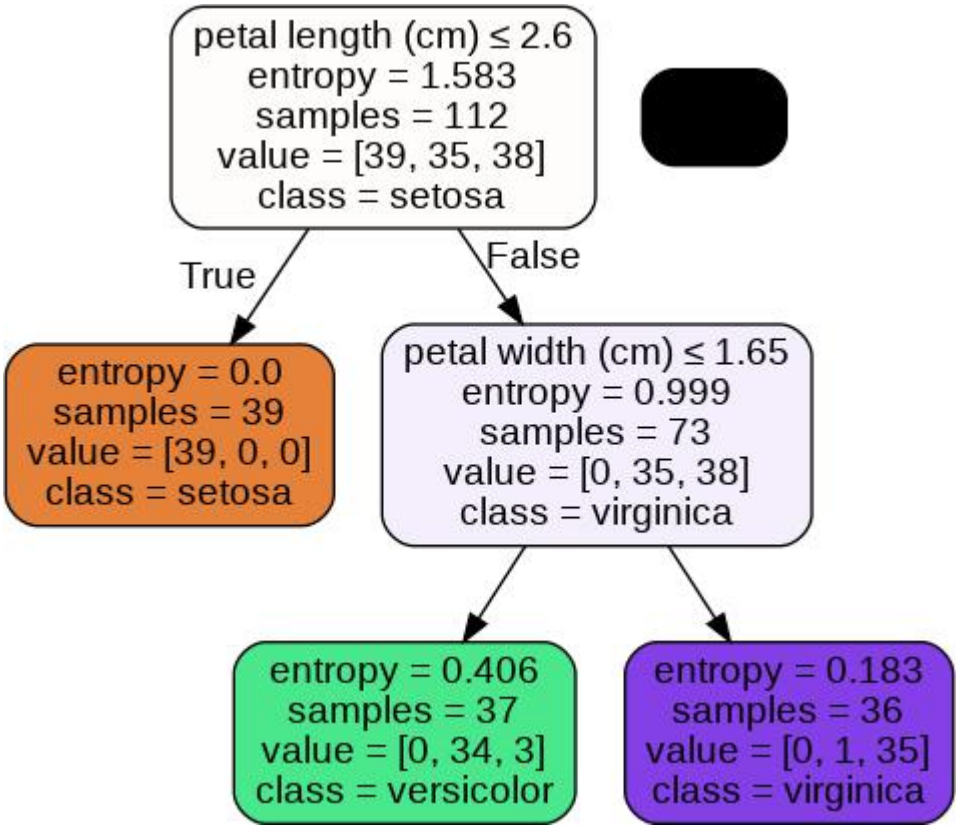
```
#visualize the decision tree

from sklearn.tree import export_graphviz  #for visualization
from six import StringIO   #keep drawing
from IPython.display import Image   #Ipython is an interactive shell
import pydotplus    #python interface to Graphviz's Dot language

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




```
from sklearn.tree import export_graphviz #for visualization
from six import StringIO #keep drawings
from IPython.display import Image #IPython interactive shell
import pydotplus #interface to export lang
dot_data = StringIO()
export_graphviz(classifier_entropy, out_file = dot_data, filled = True, impur
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, roun
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```


petal length (cm) ≤ 2.6
entropy = 1.583
samples = 112
value = [39, 35, 38]
class = setosa
True
False
entropy = 0.0
samples = 39
value = [39, 0, 0]
class = setosa
petal width (cm) ≤ 1.65
entropy = 0.999
samples = 73
value = [0, 35, 38]
class = virginica
entropy = 0.406
samples = 37
value = [0, 34, 3]
class = versicolor
entropy = 0.183
samples = 36
value = [0, 1, 35]
class = virginica

CO3-3

AIM:

Program to implement k-means clustering technique using any standard dataset available in thepublic domain

DATASET:College_Data

PROGRAM:

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

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

```
df.head()
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Und
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()
```

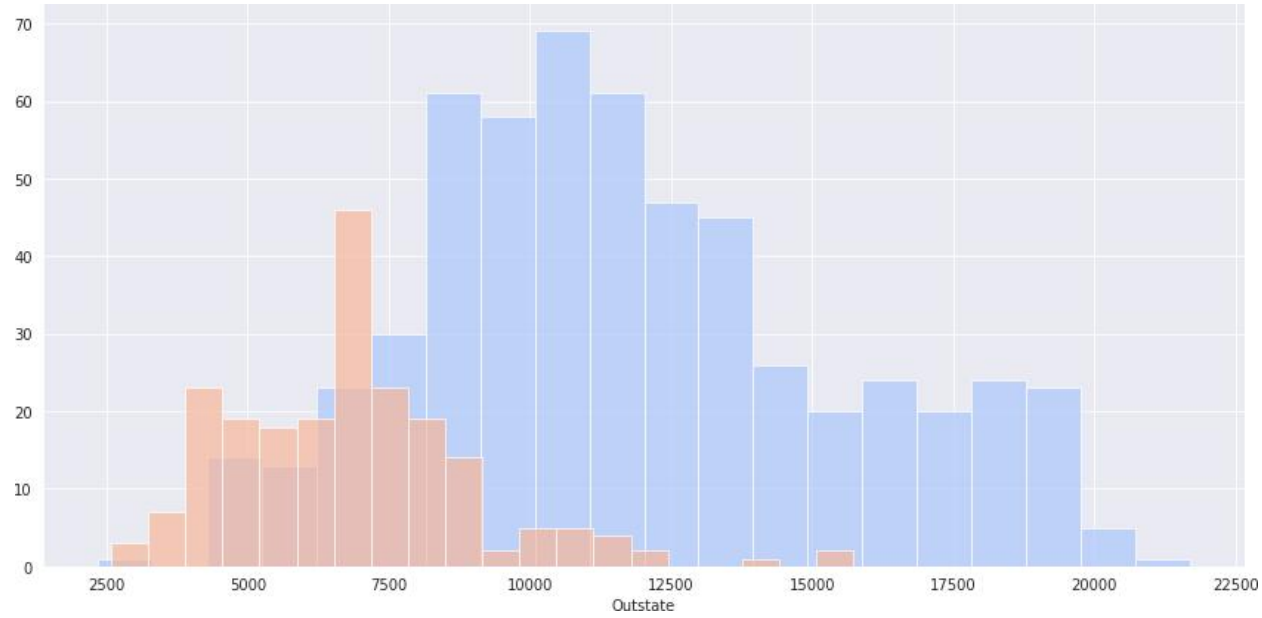
```
<class 'pandas.core.frame.DataFrame'>
Index: 777 entries, Abilene Christian University to York College of PennsylvaniaData columns (total 18 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Private      777 non-null    object 
 1   Apps         777 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: 131.5+ KB
```

```
df.describe()
```

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000

```
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: The `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([[1.03631389e+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],
[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,
6.51195815e+01]])

```

```

```
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	P.Und
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_))
```

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

           0       0.69      0.35      0.46       212
           1       0.79      0.94      0.86       565

    accuracy                           0.78       777
   macro avg       0.74      0.64      0.66       777
weighted avg       0.76      0.78      0.75       777
```