COURSE OUTCOME-3

PROGRAM NO-1

Aim: Program to implement text classification using Support vector machine

```
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]
```

```
{"type":"string"}
```

```
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 buffet... Cine there got amore wat...


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. Text FA to 87121 to receive entry question(std txt
rate)T&C's apply 08452810075over18's


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'd like some fun you up for it still? Tb ok! XxX std chgs
to send, £1.50 to rcv


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


7 ham As per your request 'Melle Melle (Oru Minnaminunginte Nurungu
Vettam)' has been set as your callertune for all Callers. Press *9 to
copy your friends Callertune


8 spam     WINNER!! As a valued network customer you have been
```

```
selected to receivea £900 prize reward! To claim call 09061701461.
Claim code KL341. Valid 12 hours only.


9 spam     Had your mobile 11 months or more? U R entitled to Update
to the latest colour mobiles with camera for Free! Call The Mobile
Update Co FREE on 08002986030
```

```
messages[0]
```

{"type":"string"}

```
import pandas as pd
```

```
messages=pd.read_csv('/content/SMSSpamCollection',sep='\
t',names=['label','message'])
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 format using
#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
```

{"type":"string"}

```
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
```

{"type":"string"}

```
#removing stopwords
#for this, we need to download stopword's corpus from nltk.corpus import stopwords
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.words('english')]
```

```
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("english")]
```

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

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

```
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: FutureWarning: Function get_feature_names is
deprecated; get_feature_names is deprecated in 1.0 and will be removed
in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

{"type":"string"}

```
#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 only ...
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'],messages['label'])
```

```
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 token integer
            ('tfidf',TfidfTransformer()),
#integer counts to weighted TF-IDF score
            ('classifier',MultinomialNB()),
#train on TF-IDF vectors w/Naive Bayes
])

pipeline.fit(msg_test,label_train)
```

```
Pipeline(steps=[('bow',
                 CountVectorizer(analyzer=<function text_process at
0x7f1c627f2c20>)),
                ('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)),
#strings to token int
                      ('tfidf',TfidfTransformer()),
#integer counts to weighted TF-IDF score

('classifier',svm.SVC(C=1.0,kernel='linear',degree=3,gamma='auto'))
])

pipeline1.fit(msg_test,label_train)
```

```
Pipeline(steps=[('bow',
                 CountVectorizer(analyzer=<function text_process at
0x7f1c627f2c20>)),
                ('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 is executed successsfully and obtained the output.

PROGRAM NO-2

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
```

```
data.data.shape
```

```
(150, 4)
```

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

```
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 (cm)']
```

```
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
```

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,random_state = 50,
test_size = 0.25)
```

```
#default criterion is Gini
classifier = DecisionTreeClassifier()
classifier.fit(xtrain,ytrain)
```

```
DecisionTreeClassifier()
```

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

```
from sklearn.metrics import accuracy_score
print('Accuracy on train data using
Gini:',accuracy_score(ytrain,classifier.predict(xtrain)))
print('Accuracy on test data using
Gini:',accuracy_score(ytest,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(xtrain,ytrain)
y_pred_entropy = classifier_entropy.predict(xtest)
print('Accuracy on train data using entropy',
accuracy_score(ytrain,classifier.predict(xtrain)))
print('Accuracy on test data using entropy',
accuracy_score(ytest,y_pred_entropy))
```

```
Accuracy on train data using entropy 1.0
Accuracy on test data using entropy 0.9473684210526315
```

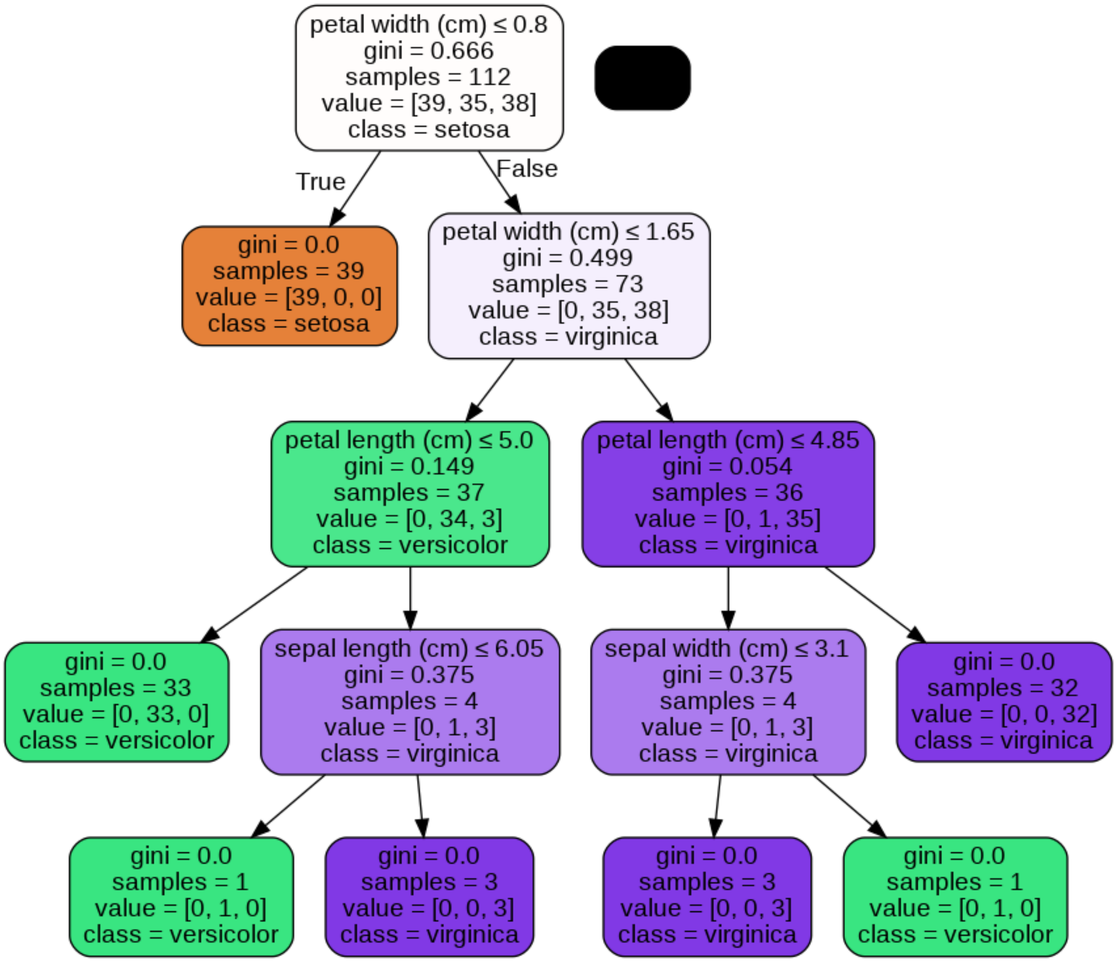
```
#change criterion to entropy with min_samples to 50. Default value is
2
classifier_entropy1 = DecisionTreeClassifier(criterion = 'entropy',
min_samples_split=50)  #min_samples_split. min_samples_split
represents

#the minimum number of samples required to split an internal node.
classifier_entropy1.fit(xtrain,ytrain)
y_pred_entropy1 = classifier_entropy1.predict(xtest)
print('Accuracy on train data using entropy',
accuracy_score(y_true=ytrain, y_pred=
classifier_entropy1.predict(xtrain)))
print('Accuracy on test data using entropy', accuracy_score(y_true =
ytest, y_pred = y_pred_entropy1))
```

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

```
#visualise the decision tree
from sklearn.tree import export_graphviz     #for visualization
from six import StringIO   #python 2,3 compatibility package, when the
StringIO object is created
                           #it is initialized by passing a string to
the constructor. If no string is passed the StringIO wil start empty.
from IPython.display import Image     #Python is an interactive shell
that is built with python
import pydotplus   #Python interface to Graphviz's Dot language.

dot_data = StringIO()
#try using classifer,classifer_entropy and classifier_entropy1 as
first parameter below.
export_graphviz(classifier, out_file = dot_data, filled = True,
rounded = True, special_characters = True, feature_names
=data.feature_names, class_names = data.target_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



Result: The program is executed successfully and obtained the output.

PROGRAM NO-3

Aim: Program to implement k-means clustering technique using any standard dataset available in the public domain

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

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

```
df.head()
```

```
                             Private  Apps  ...  Expend  Grad.Rate
Abilene Christian University     Yes  1660  ...    7041         60
Adelphi University               Yes  2186  ...   10527         56
Adrian College                   Yes  1428  ...    8735         54
Agnes Scott College              Yes   417  ...   19016         59
Alaska Pacific University        Yes   193  ...   10922         15

[5 rows x 18 columns]
```

```
df.info()
```

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

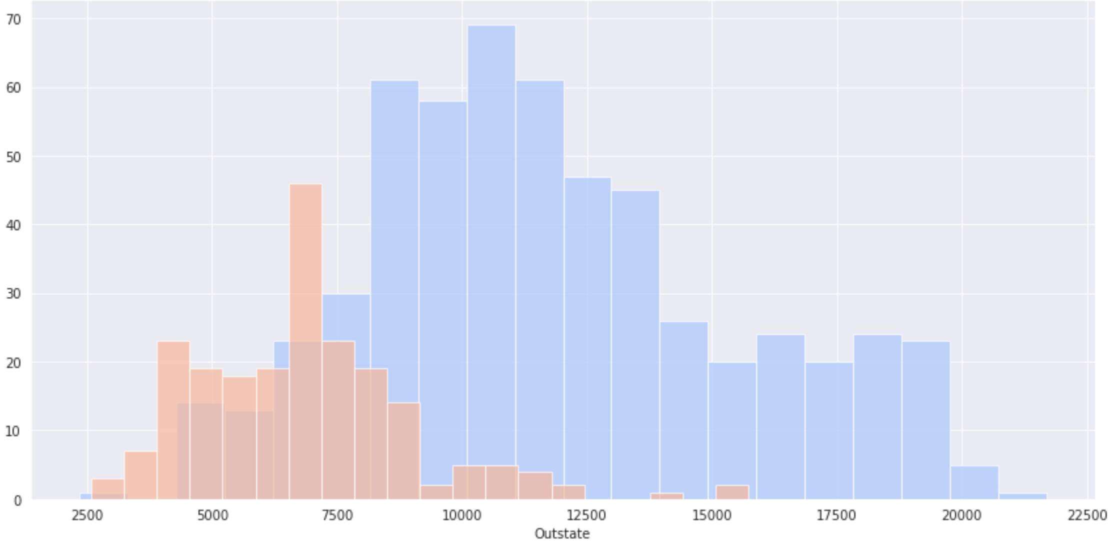
```
df.describe()
```

```
                Apps        Accept  ...        Expend  Grad.Rate
count    777.000000    777.000000  ...    777.000000  777.00000
mean    3001.638353   2018.804376  ...   9660.171171   65.46332
std     3870.201484   2451.113971  ...   5221.768440   17.17771
min       81.000000     72.000000  ...   3186.000000   10.00000
25%      776.000000    604.000000  ...   6751.000000   53.00000
50%     1558.000000   1110.000000  ...   8377.000000   65.00000
75%     3624.000000   2424.000000  ...  10830.000000   78.00000
max    48094.000000  26330.000000  ...  56233.000000  118.00000

[8 rows x 17 columns]
```

```
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 `size` parameter has been renamed to `height`; please
update your code.
  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  ...  Expend
Grad.Rate  cluster
Abilene Christian University     Yes  1660    1232  ...    7041
60        1
Adelphi University               Yes  2186    1924  ...   10527
56        1
Adrian College                   Yes  1428    1097  ...    8735
54        1
Agnes Scott College              Yes   417     349  ...   19016
59        1
Alaska Pacific University        Yes   193     146  ...   10922
15        1

[5 rows x 19 columns]
```

```
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
```

Result: The program is executed successfully and obtained the output.