DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT

Udayapura, Kanakapura Road, Bengaluru - 560082, Karnataka.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Academic Year – 2022-2023

Lab Manual

Machine Learning

10AIL66

CONTENT

MACHINE LEARNING LABORATORY
(Effective from the academic year 2018 -2019)
SEMESTER – VI

Subject Code	18AIL66	**CIE Marks**	40
Number of Contact Hours/Week	0:2:2	**SEE Marks**	60
Total Number of Lab Contact Hours		**Exam Hours**	3 Hrs

Credits – 2

Course Learning Objectives: This course will enable students to:

- Implement and evaluate ML algorithms in Python/Java programming language.

Descriptions (if any):

1. The programs can be implemented in either JAVA or Python.
2. Data sets can be taken from standard repository such as UCI

Installation procedure of the required software must be demonstrated, carried out in groups and documented in the journal.

Programs List:

1.	Implement and demonstratethe**FIND-Salgorithm** for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file and show the output for test cases. Develop an interactive program by Compareing the result by implementing **LIST THEN ELIMINATE** algorithm.
2	For a given set of training data examples stored in a .CSV file, implement and demonstrate the **Candidate-Elimination**algorithm. Output a description of the set of all hypotheses consistent with the training examples.
3	Demonstrate Pre processing (Data Cleaning, Integration and Transformation) activity on suitable data: For example: Identify and Delete **Rows that Contain Duplicate Data** by considering an appropriate dataset. Identify and Delete **Columns That Contain a Single Value** by considering an appropriate dataset.
4	Demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge toclassify a new sample.
5	Demonstrate the working of the Random forest **algorithm**. Use an appropriate data set for building and apply this knowledge toclassify a new sample.
6	Implement the **naïve Bayesian classifier** for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.
7	Assuming a set of documents that need to be classified, use the **naïve Bayesian** Classifier model to perform this task. Calculate the accuracy, precision, and recall for your data set.
8	Construct a**Bayesian network** considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.
9	Demonstrate the working of EM algorithm to cluster a set of data stored in a .CSV file.
10	Demonstrate the working of SVM classifier for a suitable data set

1. Implement and demonstratethe**FIND-Salgorithm** for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file and show the output for test cases. Develop an interactive program by Comparing the result by implementing **LIST THEN ELIMINATE** algorithm.

Program:

```
import pandas as pd
import numpy as np
data = pd.read_csv('/content/drive/MyDrive/ML_1.csv')
attribute=np.array(data)[:,:-1]
target=np.array(data)[:,-1]
def train(att,tar):
    for i,val in enumerate(tar):
        if val=='yes':
            specific_h=att[i].copy()
            break
    for i,val in enumerate(att):
        if tar[i]=='yes':
            for x in range(len(specific_h)):
                if val[x]!=specific_h[x]:
                    specific_h[x]='?'
                else:
                    pass
    return specific_h
print(train(attribute,target))
```

Dataset :

Sky	**AirTemp**	**Humidity**	**wind**	**Water**	**Forecast**	**Enjoy sport**
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Output:

```
['sunny' 'warm' '?' 'strong' '?' '?']
```

2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the **Candidate-Elimination**algorithm. Output a description of the set of all hypotheses consistent with the training examples.

Program:

```
import numpy as np
import pandas as pd

data = pd.read_csv('/content/drive/MyDrive/ML_1.csv')
concepts = np.array(data.iloc[:,:-1])
print("\nInstances are:\n",concepts)
target = np.array(data.iloc[:,-1])
print("\nTarget Values are: ",target)

def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("\nInitialization of specific_h and genearal_h")
    print("\nSpecific Boundary: ", specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print("\nGeneric Boundary: ",general_h)

    for i, h in enumerate(concepts):
        print("\nInstance", i+1 , "is ", h)
        if target[i] == "yes":
            print("Instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "no":
            print("Instance is Negative ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'

        print("Specific Bundary after ", i+1, "Instance is ", specific_h)
        print("Generic Boundary after ", i+1, "Instance is ", general_h)
        print("\n")

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h

s_final, g_final = learn(concepts, target)

print("Final Specific_h: ", s_final, sep="\n")
print("Final General_h: ", g_final, sep="\n")
```

Dataset :

Sky	**AirTemp**	**Humidity**	**wind**	**Water**	**Forecast**	**Enjoy sport**
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Output:

```
Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?','?',
'?']]
```

3. Demonstrate Pre processing (Data Cleaning, Integration and Transformation) activity on suitable data:
For example:
Identify and Delete **Rows that Contain Duplicate Data** by considering an appropriate dataset.
Identify and Delete **Columns That Contain a Single Value** by considering an appropriate dataset.

Program:

```
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/ML_1.csv')
df=pd.dataframe(data)
# data cleaning – row wise
# only the repetitive rows are removed but in the original data it will be not deleted
df.drop_duplicates(subset=’sky’, keep=’first’)
df
# only the repetitive rows are removed and in the original data it will be deleted
df.drop_duplicates(subset=’sky’, keep=’first’,inplace=True)
df
# To change the index values
df.drop_duplicates(subset=’sky’, keep=’first’, ignore_index=True)
df
# data cleaning – column wise
# using index values
df.drop(df.columns[[2,3]],axis=1)
#using iloc
df.drop(df.iloc[:, 2:4], axis=1)
# similarly the integration and the transformation operation can be carried on the input dataset
some of the examples are as mentioned below
#insert – df.insert(0,column_name,np.random)
#melt- pd.melt
#concat- pd.concat(df.df2, axis=0,ignore_index=true)
#merge- df1.merge(df2, on=’column_name’)
#get dummies pd.get_dummies(df)
#pivot tables  - df.pivot_table_average
```

Dataset : enjoy sport dataset, but make sure enough rows are repetitive rows are inserted

Output: depends on the operation carried out

4. Demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Program:

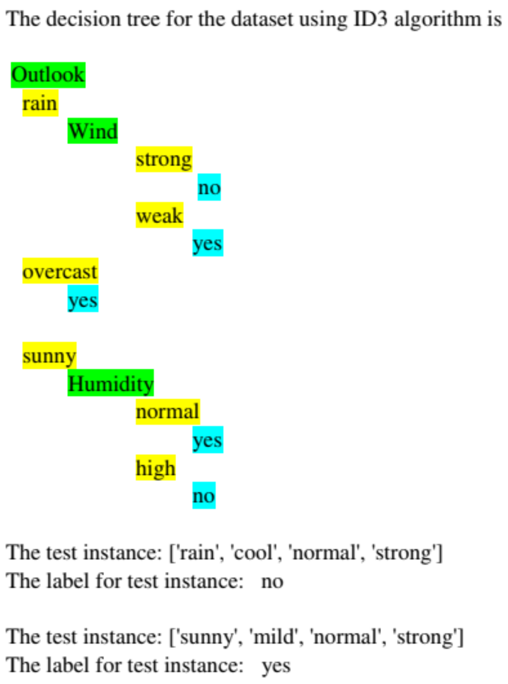
```
import math
import csv
def load_csv(filename):
lines=csv.reader(open(filename,"r"));
dataset = list(lines)
headers = dataset.pop(0)
return dataset,headers
class Node:
def __init__(self,attribute):
self.attribute=attribute
self.children=[]
self.answer=""
def subtables(data,col,delete):
dic={}
coldata=[row[col] for row in data]
attr=list(set(coldata))
counts=[0]*len(attr)
r=len(data)
c=len(data[0])
for x in range(len(attr)):
for y in range(r):
if data[y][col]==attr[x]:
counts[x]+=1
for x in range(len(attr)):
dic[attr[x]]=[[0 for i in range(c)] for j in
range(counts[x])]
pos=0
for y in range(r):
if data[y][col]==attr[x]:
if delete:
del data[y][col]
dic[attr[x]][pos]=data[y]
pos+=1
return attr,dic
def entropy(S):
attr=list(set(S))
if len(attr)==1:
return 0
counts=[0,0]
for i in range(2):
```

```
counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)
sums=0
for cnt in counts:
sums+=-1*cnt*math.log(cnt,2)
return sums
def compute_gain(data,col):
attr,dic = subtables(data,col,delete=False)
total_size=len(data)
entropies=[0]*len(attr)
ratio=[0]*len(attr)
total_entropy=entropy([row[-1] for row in data])
for x in range(len(attr)):
ratio[x]=len(dic[attr[x]])/(total_size*1.0)
entropies[x]=entropy([row[-1] for row in
dic[attr[x]]])
total_entropy-=ratio[x]*entropies[x]
return total_entropy
def build_tree(data,features):
lastcol=[row[-1] for row in data]
if(len(set(lastcol)))==1:
node=Node("")
node.answer=lastcol[0]
return node
n=len(data[0])-1
gains=[0]*n
for col in range(n):
gains[col]=compute_gain(data,col)
split=gains.index(max(gains))
node=Node(features[split])
fea = features[:split]+features[split+1:]
attr,dic=subtables(data,split,delete=True)
for x in range(len(attr)):
child=build_tree(dic[attr[x]],fea)
node.children.append((attr[x],child))
return node
def print_tree(node,level):
if node.answer!="":
print(" "*level,node.answer)
return
print(" "*level,node.attribute)
for value,n in node.children:
print(" "*(level+1),value)
print_tree(n,level+2)
```

```
def classify(node,x_test,features):
if node.answer!="":
print(node.answer)
return
pos=features.index(node.attribute)
for value, n in node.children:
if x_test[pos]==value:
classify(n,x_test,features)
'''Main program'''
dataset,features=load_csv("data3.csv")
node1=build_tree(dataset,features)
print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node1,0)
testdata,features=load_csv("data3_test.csv")
for xtest in testdata:
print("The test instance:",xtest)
print("The label for test instance:",end=" ")
classify(node1,xtest,features)
```

Dataset : create dataset

Output:



```
The decision tree for the dataset using ID3 algorithm is

Outlook
 rain
     Wind
          strong
               no
          weak
              yes
 overcast
     yes

 sunny
     Humidity
          normal
              yes
          high
              no

The test instance: ['rain', 'cool', 'normal', 'strong']
The label for test instance:   no

The test instance: ['sunny', 'mild', 'normal', 'strong']
The label for test instance:   yes
```

5. Demonstrate the working of the Random forest **algorithm**. Use an appropriate data set for building and apply this knowledge to classify a new sample.

Program:

```
from sklearn import datasets
iris = datasets.load_iris()
print(iris.target_names)
print(iris.feature_names)
X, y = datasets.load_iris( return_X_y = True)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
data = pd.DataFrame({‘sepallength’: iris.data[:, 0], ’sepalwidth’: iris.data[:, 1], ’petallength’:
iris.data[:, 2], ’petalwidth’: iris.data[:, 3],’species’: iris.target})
print(data.head())
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
from sklearn import metrics
print()
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
clf.predict([[3, 3, 2, 2]])
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train, y_train)
import pandas as pd
feature_imp = pd.Series(clf.feature_importances_, index =
iris.feature_names).sort_values(ascending = False)
feature_imp
```

Dataset : iris dataset

Output:

```
ACCURACY OF THE MODEL: 0.9238095238095239
petal width (cm)     0.458607
petal length (cm)    0.413859
sepal length (cm)    0.103600
sepal width (cm)     0.023933
dtype: float64
```

6. Implement the **naïve Bayesian classifier** for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Program:

```
from sklearn.datasets import load_iris
iris = load_iris()

X = iris.data
y = iris.target

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)

from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test,
y_pred)*100)
```

Dataset : iris dataset

Output:

```
Gaussian Naive Bayes model accuracy(in %): 95.0
```

7. Assuming a set of documents that need to be classified, use the **naive Bayesian Classifier** model to perform this task. Calculate the accuracy, precision, and recall for your data set.

Program:

```
import pandas as pd
 msg=pd.read_csv('naivetext1.csv',names=['message','label'])
print('The dimensions of the dataset',msg.shape)
 msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
 y=msg.labelnum
print(X)
 print(y)
 from sklearn.model_selection import train_test_split
 xtrain,xtest,ytrain,
ytest=train_test_split(X,y)
 print(xtest.shape)
 print(xtrain.shape)
print(ytest.shape)
print(ytrain.shape)
 from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain)
 xtest_dtm=count_vect.transform(xtest)
 from sklearn.naive_bayes import MultinomialNB
 clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
 from sklearn import metrics
print('Accuracy metrics')
 print('Accuracy of the classifer is',metrics.accuracy_score(ytest,predicted))
print('Confusion matrix')
 print(metrics.confusion_matrix(ytest,predicted))
print('Recall and Precison ')
 print(metrics.recall_score(ytest,predicted))
print(metrics.precision_score(ytest,predicted))
```

Dataset : naïvetext.csv

Output:

```
Accuracy metrics
Accuracy of the classifer is 0.8
Confusion matrix
[[3 1] [0 1]]
Recall and Precison 1.0 0.5
```

8. Construct a **Bayesian network** considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

Program:

```
 import numpy as np
 from urllib.request import urlopen
 import urllib
import pandas as pd
 from pgmpy.inference import VariableElimination
 from pgmpy.models import BayesianModel
 from pgmpy.estimators import MaximumLikelihoodEstimator,  BayesianEstimator
names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal',
'heartdisease']
heartDisease = pd.read_csv('heart.csv', names = names)
 heartDisease = heartDisease.replace('?', np.nan)
model = BayesianModel([('age', 'trestbps'), ('age', 'fbs'), ('sex', 'trestbps'), ('exang',
'trestbps'),('trestbps','heartdisease'),('fbs','heartdisease'),('heartdisease','restecg'), ('heartdisease','thalach'),
('heartdisease','chol')])
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)
from pgmpy.inference import VariableElimination
HeartDisease_infer = VariableElimination(model)
q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'age': 37, 'sex' :0})
print(q['heartdisease'])
```

Dataset : heart.csv

Output:

heartdisease	phi(heartdisease)
heartdisease_0	0.5593
heartdisease_1	0.4407

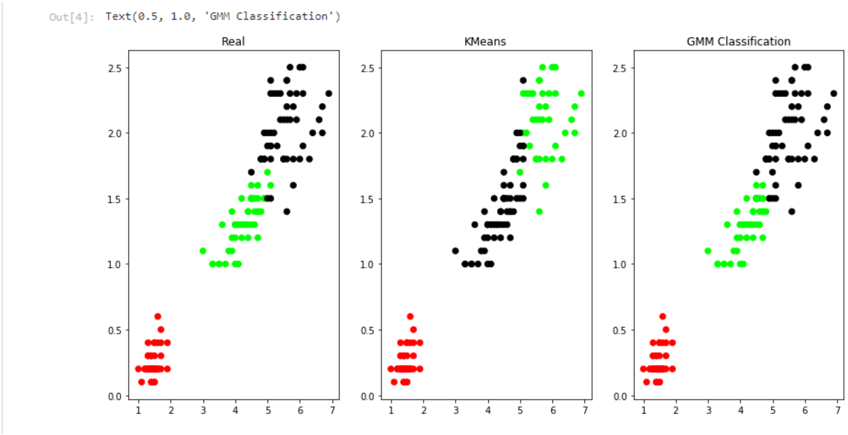
9. Demonstrate the working of EM algorithm to cluster a set of data stored in a .CSV file

Program:

```
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.mixture import GaussianMixture
from sklearn.datasets import load_iris
import sklearn.metrics as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset=load_iris()
# print(dataset)
X=pd.DataFrame(dataset.data)
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y=pd.DataFrame(dataset.target)
y.columns=['Targets']
# print(X)
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT
plt.subplot(1,3,1)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)
plt.title('Real')

# K-PLOT
plt.subplot(1,3,2)
model=KMeans(n_clusters=3)
model.fit(X)
predY=np.choose(model.labels_,[0,1,2]).astype(np.int64)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[predY],s=40)
plt.title('KMeans')

# GMM PLOT
scaler=preprocessing.StandardScaler()
scaler.fit(X)
xsa=scaler.transform(X)
xs=pd.DataFrame(xsa,columns=X.columns)
gmm=GaussianMixture(n_components=3)
gmm.fit(xs)
y_cluster_gmm=gmm.predict(xs)
plt.subplot(1,3,3)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm],s=40)
plt.title('GMM Classification')
```

Dataset : iris dataset

Output:



```
Out[4]:  Text(0.5, 1.0, 'GMM Classification')
```

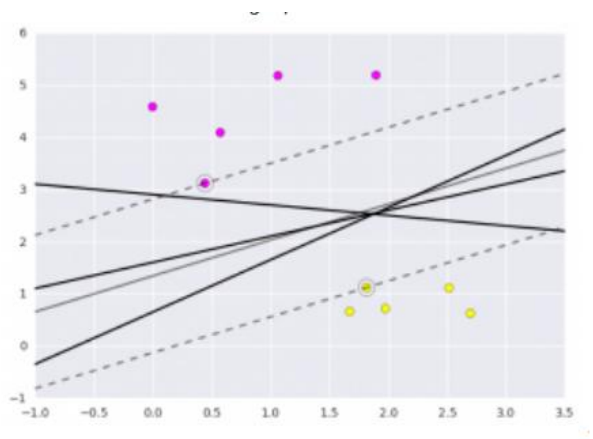
10. Demonstrate the working of SVM classifier for a suitable data set

Program:

```
# importing scikit learn with make_blobs
from sklearn.datasets.samples_generator import make_blobs

# creating datasets X containing n_samples
# Y containing two classes
X, Y = make_blobs(n_samples=500, centers=2,
                                        random_state=0, cluster_std=0.40)
import matplotlib.pyplot as plt
# plotting scatters
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring');
plt.show()
# creating linspace between -1 to 3.5
xfit = np.linspace(-1, 3.5)

# plotting scatter
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring')

# plot a line between the different sets of data
for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
          yfit = m * xfit + b
          plt.plot(xfit, yfit, '-k')
          plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
color='#AAAAAA', alpha=0.4)
plt.xlim(-1, 3.5);
plt.show()
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# reading csv file and extracting class column to y.
x = pd.read_csv("C:\...\cancer.csv")
a = np.array(x)
y = a[:,30] # classes having 0 and 1
# extracting two features
x = np.column_stack((x.malignant,x.benign))
# 569 samples and 2 features
x.shape
print (x),(y)
# import support vector classifier
# "Support Vector Classifier"
from sklearn.svm import SVC
clf = SVC(kernel='linear')
# fitting x samples and y classes
clf.fit(x, y)
clf.predict([[120, 990]])
clf.predict([[85, 550]])
```

Dataset : cancer.csv

Output:


6
5
4
3
2
1
0
−1
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5