Ass1: - KNN

```
from sklearn import datasets

import numpy

iris=datasets.load_iris()

print(iris.target_names)

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.3)

from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=5)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

#scaler.fit(x_train)

x_train = scaler.fit_transform(x_train)

x_test = scaler.transform(x_test)

knn.fit(x_train,y_train)

y_pred=knn.predict(x_test)

from sklearn import metrics

print("correct predictions", metrics.accuracy_score(y_test,y_pred))

print("wrong predictions", 1-metrics.accuracy_score(y_test,y_pred))

print("confusion matrix \n",metrics.confusion_matrix(y_test,y_pred))

import matplotlib.pyplot as plt

plt.plot(y_test,y_pred)

plt.bar(y_test,y_pred)
```

Output: -

```
correct predictions 0.95

wrong predictions 0.050000000000000044
confusion matrix
 [[20  0  0]
 [ 0 17  1]
 [ 0  2 20]]
```

Ass2: - Candidate Elimination

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv('candidate_elimination.csv'))
print(data)
concepts = np.array(data.iloc[:,0:-1])
target = np.array(data.iloc[:,-1])
print(target)
print(concepts)
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("Initialization of specific_h and general_h")
    print("specific_h: ",specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print("general_h: ",general_h)
    print("concepts: ",concepts)
    for i, h in enumerate(concepts):
        if target[i] == "yes":
            for x in range(len(specific_h)):
                #print("h[x]",h[x])
                if h[x] != specific_h[x]:
                    specific_h[x] = '?'
                    general_h[x][x] = '?'
        if target[i] == "no":
            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("\nSteps of Candidate Elimination Algorithm: ",i+1)
    print("Specific_h: ",i+1)
    print(specific_h,"\n")
    print("general_h :", i+1)
    print(general_h)
    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    print("\nIndices",indices)
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h
s_final,g_final = learn(concepts, target)
print("\nFinal Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

Output: -

```
general_h : 4
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Indices [2, 3, 4, 5]
Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

Ass5: - Naïve Bayes

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
data = pd.read_csv('Decision_Tree.csv')
print("The first 5 values of data is :\n",data.head())
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)
le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)
le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)
le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)
print("\nNow the Train data is :\n",X.head())
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
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)
classifier = GaussianNB()
classifier.fit(X_train,y_train)
from sklearn.metrics import accuracy_score
```

```
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))

from sklearn.metrics import confusion_matrix

print(confusion_matrix(classifier.predict(X_test),y_test))
```

Output: -

Accuracy is: 0.8

```
[[1 1]
 [0 3]]
```

Ass6: -

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn import metrics

msg=pd.read_csv('Ass6.csv',names=['message','label'])

print('The dimensions of the dataset',msg.shape)

msg

msg['labelnum']=msg.label.map({'pos':1,'neg':0})

X=msg.message

y=msg.labelnum

X

#splitting the dataset into train and test data

xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.35)

print ('\n the total number of Training Data :',ytrain.shape)

print ('\n the total number of Test Data :',ytest.shape)

cv = CountVectorizer()

xtrain_dtm = cv.fit_transform(xtrain)

xtest_dtm=cv.transform(xtest)
```

```
print('\n The words or Tokens in the text documents \n')
print(cv.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_names())
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision', metrics.precision_score(ytest,predicted))
print('\n The value of Recall', metrics.recall_score(ytest,predicted))
```

Output: -

```
Accuracy of the classifier is 0.8571428571428571
 Confusion matrix
[[3 0]
 [1 3]]
 The value of Precision 1.0
 The value of Recall 0.75
```

Ass7: - Bayesian Network

```
import numpy as np
from urllib.request import urlopen
import urllib
import matplotlib.pyplot as plt # Visuals
import seaborn as sns
import sklearn as skl
import pandas as pd
heartDisease = pd.read_csv( 'heart.csv')
```

```
heartDisease.head()
del heartDisease['ca']
del heartDisease['slope']
del heartDisease['thal']
del heartDisease['oldpeak']
heartDisease = heartDisease.replace('?', np.nan)
heartDisease.dtypes
heartDisease.columns
from pgmpy.models import BayesianModel
from pgmpy.estimators import MaximumLikelihoodEstimator, BayesianEstimator
model = BayesianModel([('age', 'trestbps'), ('age', 'fbs'), ('sex', 'trestbps'), ('sex', 'trestbps'),
                       ('exang', 'trestbps'),('trestbps','heartdisease'),('fbs','heartdisease'),
                      ('heartdisease','restecg'),('heartdisease','thalach'),('heartdisease','chol')])
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)
print(model.get_cpds('age'))
from pgmpy.inference import VariableElimination
HeartDisease_infer = VariableElimination(model)
q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'age': 28},joint=False)
print(q['heartdisease'])
q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'chol': 100},joint=False)
print(q['heartdisease'])
```

Output: -

heartdisease	phi(heartdisease)
heartdisease(0)	0.4058
heartdisease(1)	0.5942

heartdisease	phi(heartdisease)
heartdisease(0)	0.0000
heartdisease(1)	1.0000

Ass8: - K-Means

```
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()
X=pd.DataFrame(dataset.data)
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y=pd.DataFrame(dataset.target)
y.columns=['Targets']
plt.figure(figsize=(14,7))
```

```
colormap=np.array(['red','lime','black'])
colormap
#Real plot
plt.subplot(1,2,1)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)
plt.title('Real')
# K-PLOT
plt.subplot(1,2,2)
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')
sm.accuracy_score(y, model.labels_)
sm.confusion_matrix(y, model.labels_)
```

Output: -

```
Accuracy 0.80
array([[ 0, 50,  0],
       [48,  0,  2],
       [14,  0, 36]], dtype=int64)
```

Logistic Regression: -

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset=pd.read_csv('Logistic.csv')
dataset
x=dataset.iloc[:,[2,3]].values
```

```
y=dataset.iloc[:,4].values
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
#scaler.fit(x_train)
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
from sklearn.linear_model import LogisticRegression
Classifier=LogisticRegression(random_state=0)
Classifier.fit(x_train,y_train)
y_pred=Classifier.predict(x_test)
print(y_pred)
from sklearn.metrics import accuracy_score
print("Accuracy ",accuracy_score(y_test,y_pred))
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

Output: -

Accuracy 0.6666666666666666

```
confusion matrix
 [[0 0]
 [1 2]]
```

Linear Regression: -

```
from sklearn import datasets
wine=datasets.load_wine()
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(wine.data,wine.target,test_size=0.3)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
from sklearn.metrics import r2_score,mean_squared_error
score=r2_score(y_test,y_pred)
print(score)
mse=mean_squared_error(y_test,y_pred)
print(mse)
```

Output: -

r^2: - 0.9059591403557081

Mean Square Error: - 0.0552119175963744

Chi-Square:

```
from sklearn import datasets
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
iris=datasets.load_iris()
x=iris.data
y=iris.target
```

```
x = x.astype(int)
chi_feature=SelectKBest(chi2,k=2)

x_kbestfeature=chi_feature.fit_transform(x,y)
print('Original feature number:', x.shape[1])
print('Reduced feature number:', x_kbestfeature.shape[1])
```

Output:

```
Original feature number: 4
Reduced feature number: 2
```

Import csv file using pandas:

```
import pandas as pd
data=pd.read_csv("Practical.csv")
print(data)
```

Splitting Training and Testing Data on CSV file:

```
import pandas as pd
data=pd.read_csv("Practical.csv")
data
X = data.iloc[:, :-1].values
y = data.iloc[:, 5].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
X_train
```

X_test

y_train

y_test

Splitting Training and Testing Data on imported dataset:

```
from sklearn import datasets
iris=datasets.load_iris()
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.3)
```