day4

December 17, 2022

[1]:
```
#FAECES DATA
```

[2]:
```
import numpy
import scipy
import pandas as pd
from scipy.io import loadmat
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import sklearn
from sklearn import svm
import random
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪ConfusionMatrixDisplay
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.utils import resample
from sklearn import metrics
from sklearn.metrics import accuracy_score
from statistics import mean
import sklearn
from sklearn import svm
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from scipy.stats import ttest_ind
```

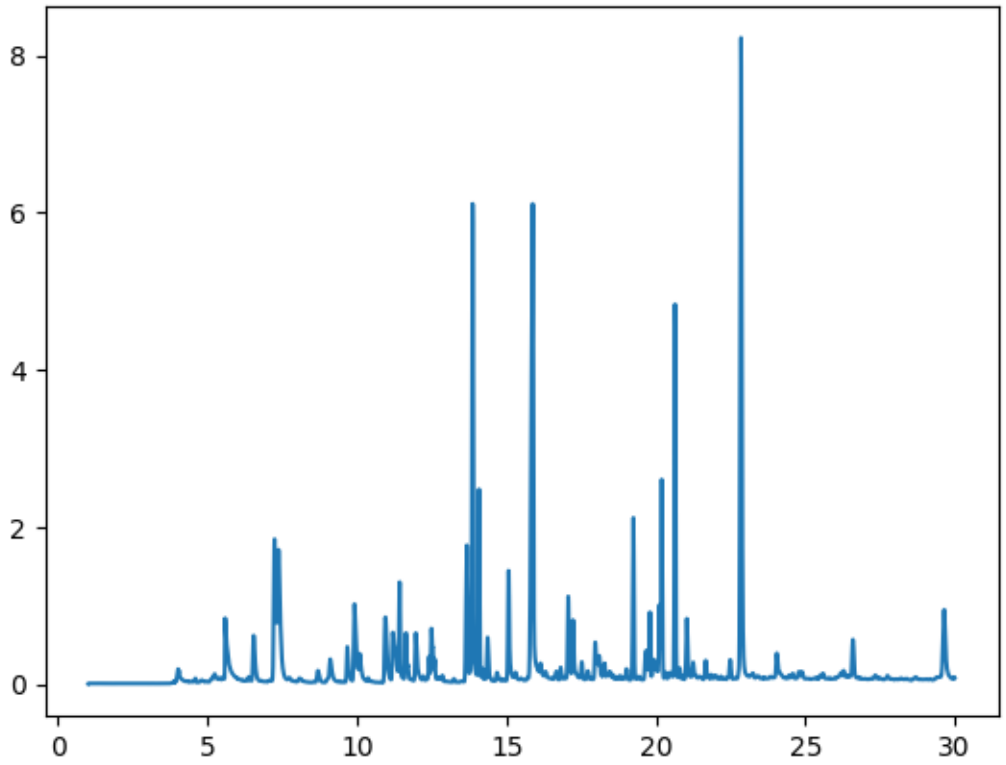
[3]:
```
data =loadmat('BWG_FA_CDvCTRL.mat') #load data
```

[4]:
```
def gcparser(mat):
    """
    Extracts essential data from a Matlab formatted GCMS object loaded
    by sio.loadmat and wrangles this into a pandas dataframe

    Parameters:
```

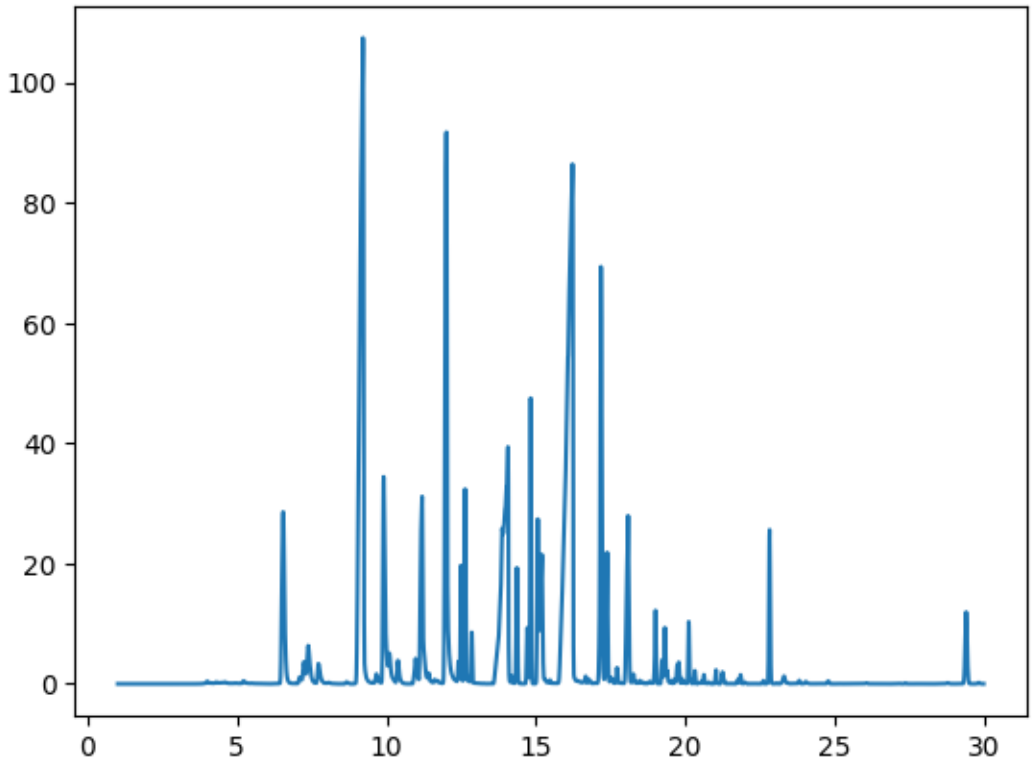
```
    mat (dict): Dictionary produced by loading a file using sio.loadmat

    Return:
    DataFrame: Total ion counts (TIC) arranged by samples (columns) and
               retention time (rows)

    """
    data = np.transpose(mat['XTIC'])  #XTIC is a matix of measured values from␣
↪GC-MS
    sample_names = mat['SAM']  # SAM contains the name of each sample
    sample_names = np.hstack(np.hstack(sample_names)).tolist()  # convert␣
↪nested numpy arrays into a list
    RT = mat['RT']  # RT is retention time (in minutes)
    RT = np.hstack(np.hstack(RT)).tolist()  # convert nested numpy arrays into␣
↪a list
    y = mat['CLASS']  #CLASS is the diagnosis of each sample (in this casse␣
↪1=control; 2=CD)
    y = np.hstack(y).tolist()  # convert nested numpy arrays into a list
    # put pieces back together in a pandas dataframe
    return pd.DataFrame(data, columns=sample_names, index=RT)
```

[5]:
```
table = gcparser(data)
x = table.transpose() #transposed data
```

[6]:
```
plt.plot(table['W1077_FA_CTRL']) #visualise chromotograms(plot of ion count as␣
 ↪a function of time
plt.show()
```



[7]:
```
plt.plot(table['W304_FA_CD'])
plt.show()
```



[8]:
```
zz= data['CLASS']

yy=[] # yy is training data
for i in zz:
     if i==1:
        yy.append("control")
     else:
        yy.append("diseased")
```

[9]:
```
z= data['CLASS']

y=[]
for i in z:
     if i==1:
        y.append("control")
     else:
        y.append("diseased")

random.shuffle(y) #shuffled training data for permtation testing
```

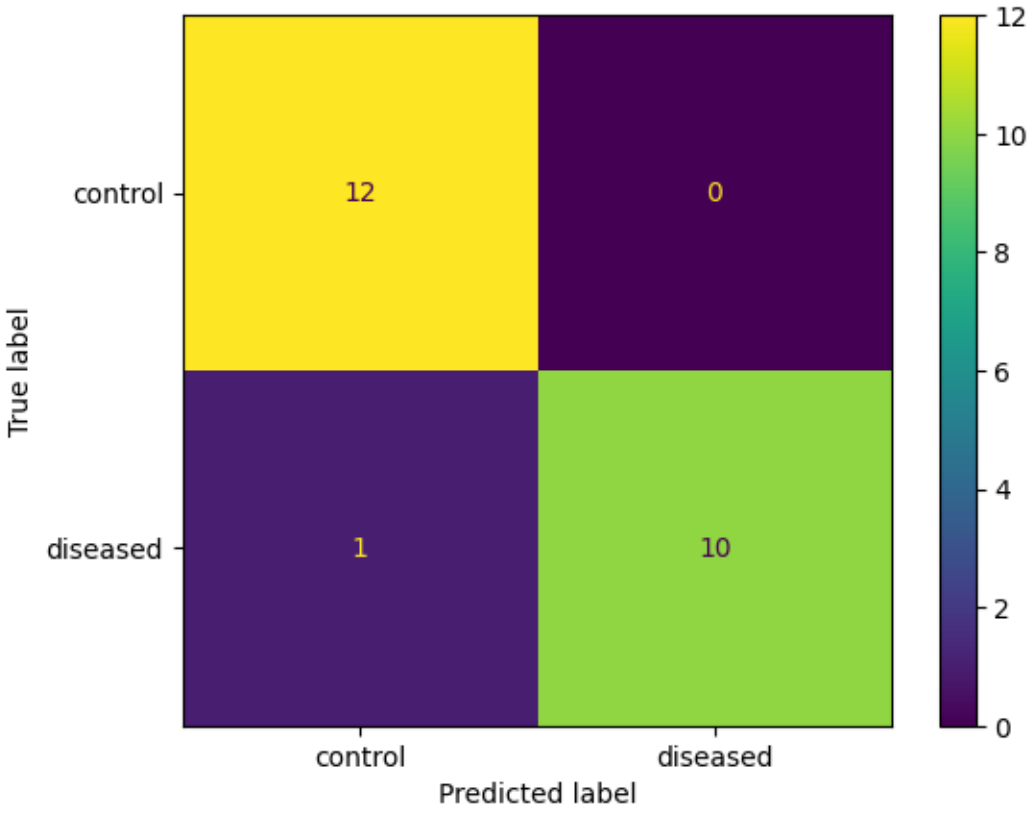
[10]:
```
#SUPPORT VECTOR MACHINE MODEL
```

[11]:
```
clf= svm.SVC()
clf.fit(x,yy) #model training
```

[11]:
```
SVC()
```

[12]:
```
predicted =clf.predict(x)
```

[13]:
```
ConfusionMatrixDisplay.from_predictions(yy, predicted) #visulaisation of␣
↪confusion matrix
plt.show()
```



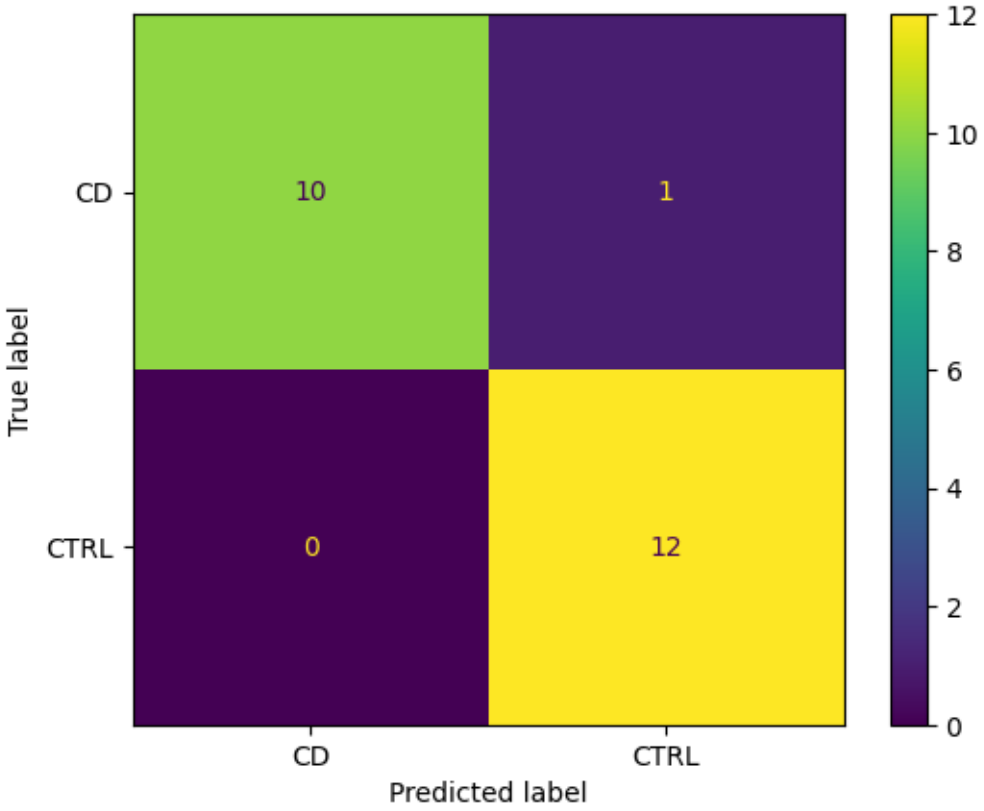
[14]:
```
#LEAVE ONE OUT
```

[15]:
```
for sample in x.index:
    x_train=x.drop(sample)
    x_test= x.loc[sample]
    y_train= [s.split('_')[2] for s in x_train.index]
    y_test = sample.split('_')[2]

    samplenames= x.index
    X= x
```

```
    Y=[s.split('_')[2] for s in samplenames]

    clf =svm.SVC()
    clf.fit(X,Y) #test data and training data
```

[16]:
```
predict= clf.predict(X)
```

[17]:
```
ConfusionMatrixDisplay.from_predictions(Y, predict)
plt.show()
```



[18]:
```
#BOOTSTRAP SVM EVALUATION
```

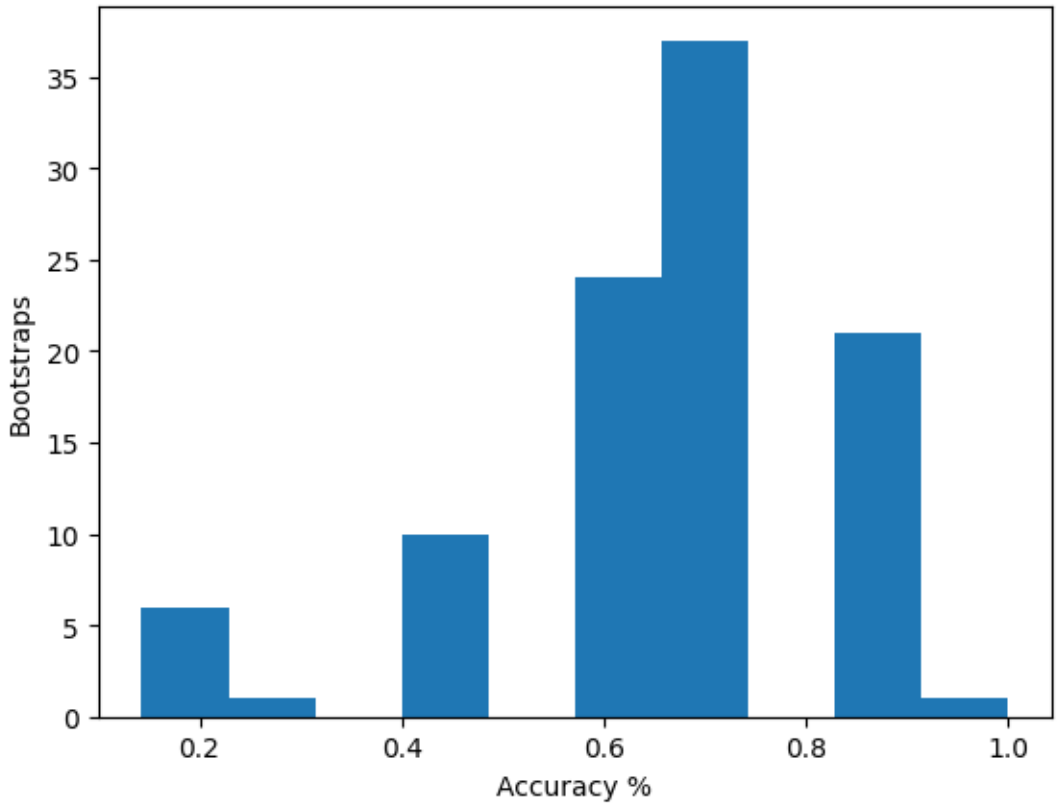
[19]:
```
n_iterations = 100
accuracy=[]

for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(x,yy, train_size= 0.7,␣
 ↪test_size=0.3)
```

```
    model = svm.SVC()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

    accuracy.append(metrics.accuracy_score(Y_test, predictions) )
    avg= mean(accuracy)
```

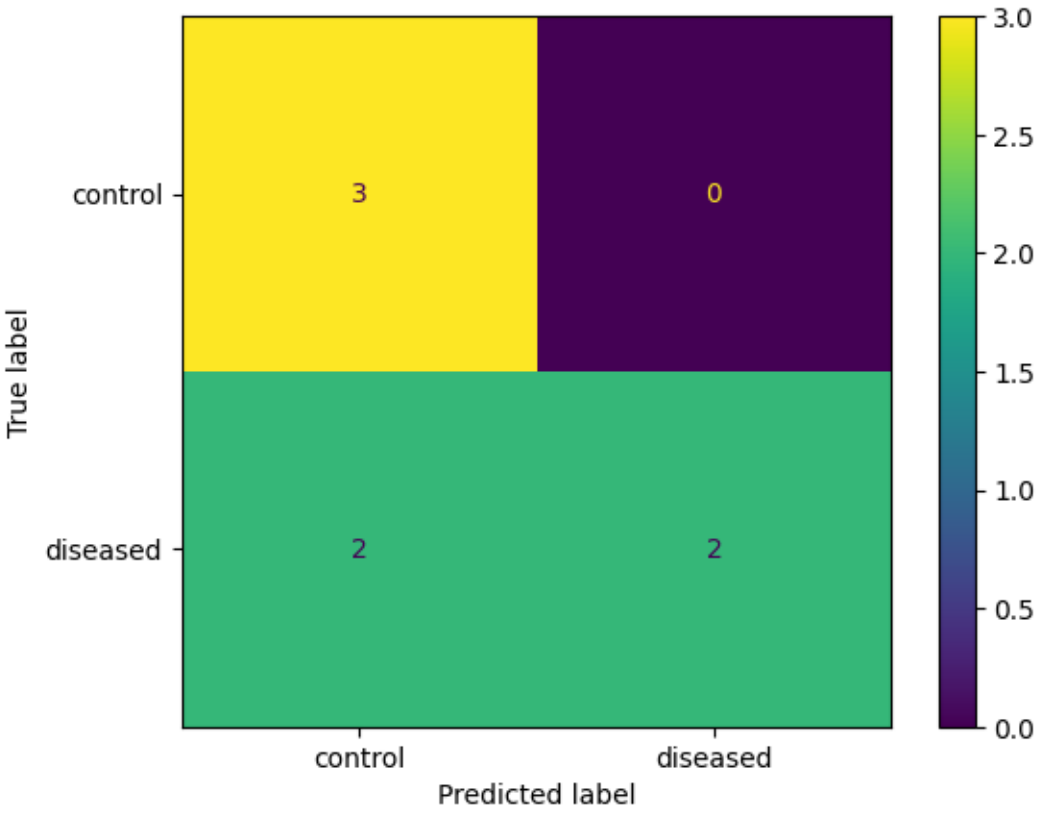
[20]:
```
avg
```

[20]: 0.6457142857142857

[21]:
```
plt.hist(accuracy) #distribution of accuracy of each of the 100 classifiers
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[22]:
```
bootstrap_pred= model.predict(X_test)
ConfusionMatrixDisplay.from_predictions(Y_test, bootstrap_pred)
plt.show()
```



[23]:
```
cm= confusion_matrix(Y_test, bootstrap_pred)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  50 %
specificity:  100 %
```

[24]:
```
#PERMUTATION TESTING
```

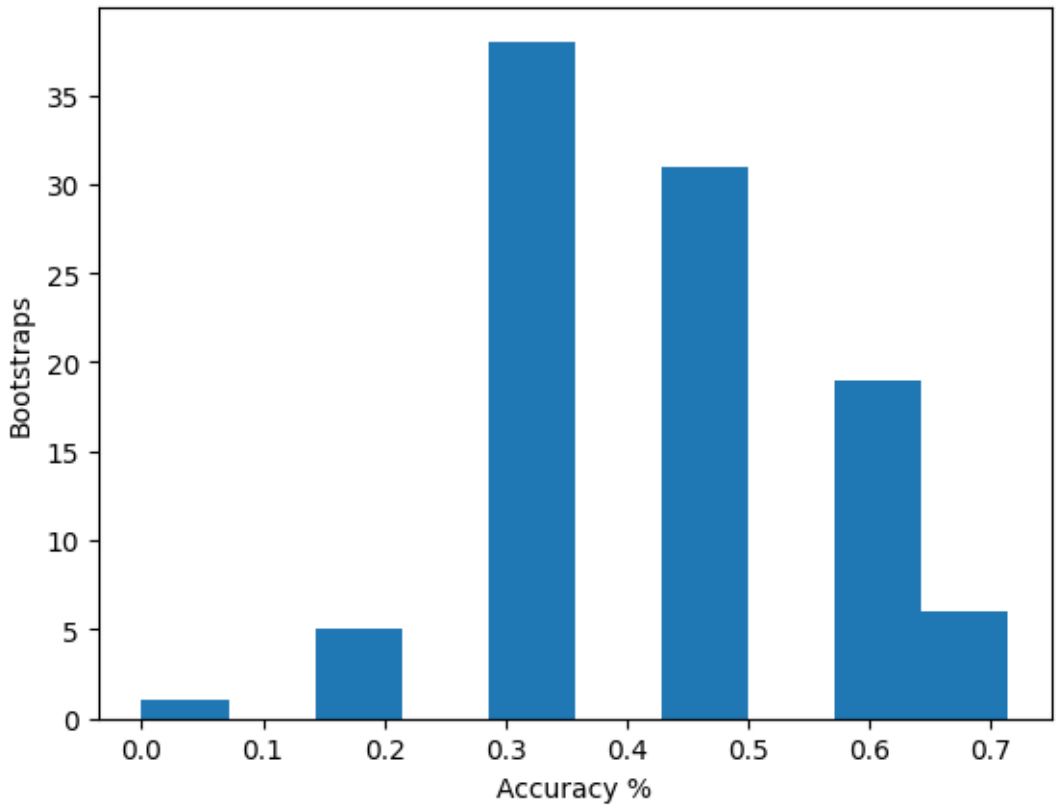
[25]:
```
accuracy1=[]

for i in range(n_iterations):
```

```
   X_train, X_test, Y_train, Y_test= train_test_split(x,y, train_size= 0.7,␣
↪test_size=0.3, shuffle=True)



   model = svm.SVC()
   model.fit(X_train, Y_train)
   predictions= model.predict(X_test)

   accuracy1.append(metrics.accuracy_score(Y_test, predictions))
   avg= mean(accuracy1) #average across 100 bootstraps
```

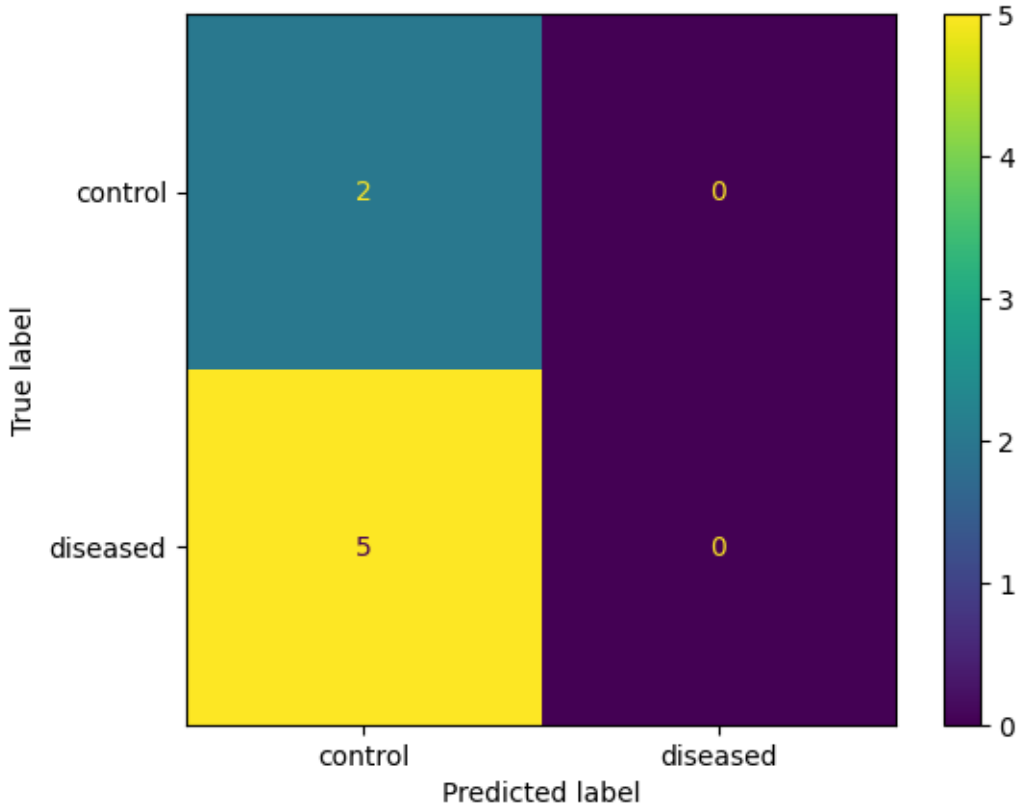
[26]:
```
avg
```

[26]: 0.39999999999999997

[27]:
```
plt.hist(accuracy1)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[28]:
```
wrong_pred= model.predict(X_test)
ConfusionMatrixDisplay.from_predictions(Y_test, wrong_pred)
plt.show()
```

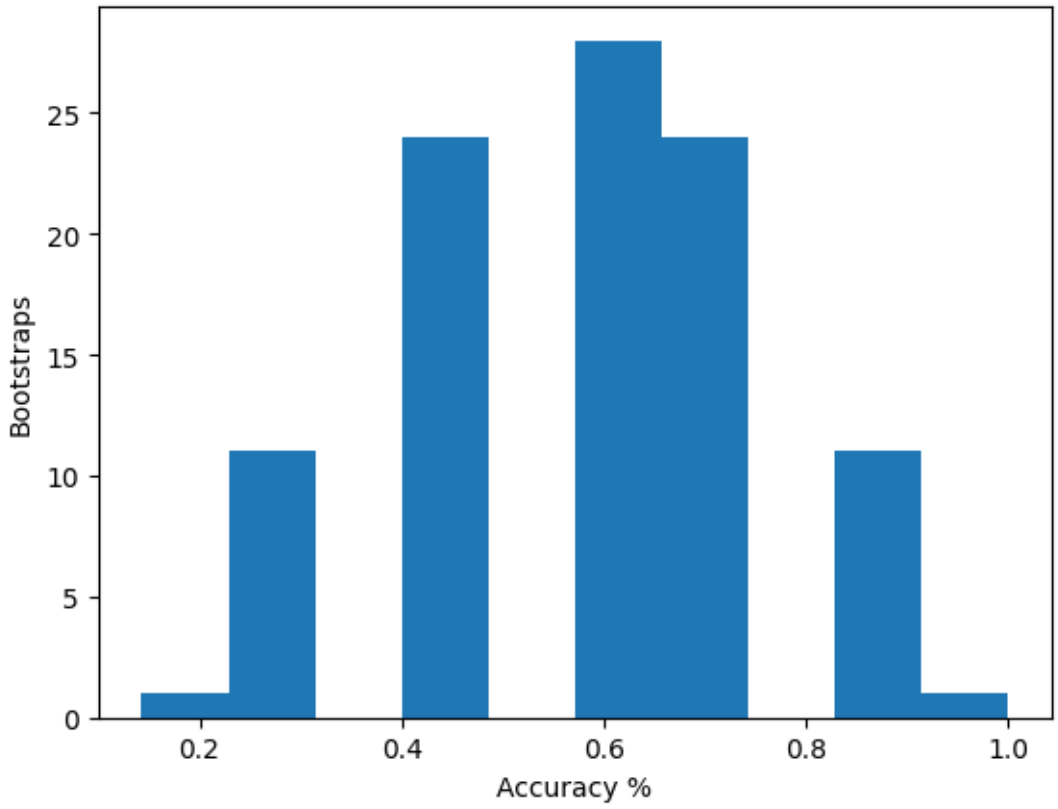


[29]:
```
accuracyfr=[]

for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(x,y, train_size= 0.7,␣
 ↪test_size=0.3, shuffle=True)


    model = RandomForestClassifier()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

    accuracyfr.append(metrics.accuracy_score(Y_test, predictions))
    avg= mean(accuracyfr)
```

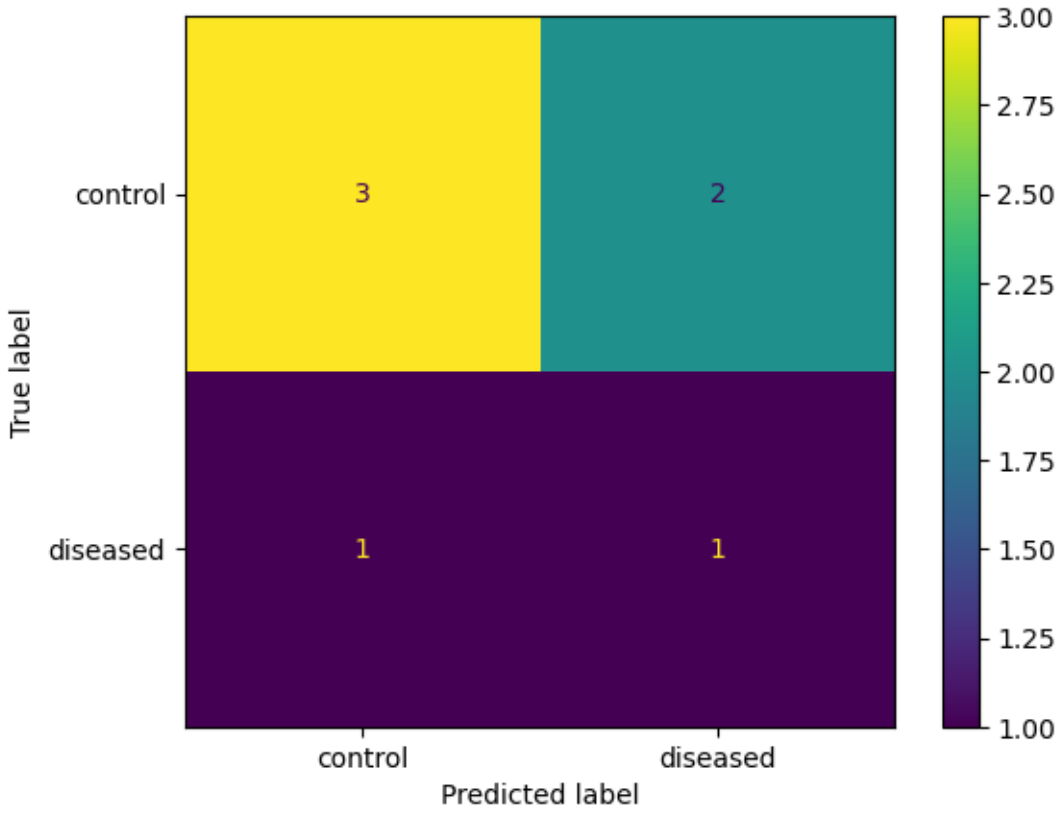
[30]:
```
avg
```

[30]: 0.5714285714285714

[31]:
```
plt.hist(accuracyfr)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[32]:
```
wrong_pred2= model.predict(X_test)
ConfusionMatrixDisplay.from_predictions(Y_test, wrong_pred2)
plt.show()
```



[33]:
```
cm= confusion_matrix(Y_test, wrong_pred2)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  50 %
specificity:  60 %
```

[34]:
```
#RANDOM FORESTS BOOTSTRAP
```

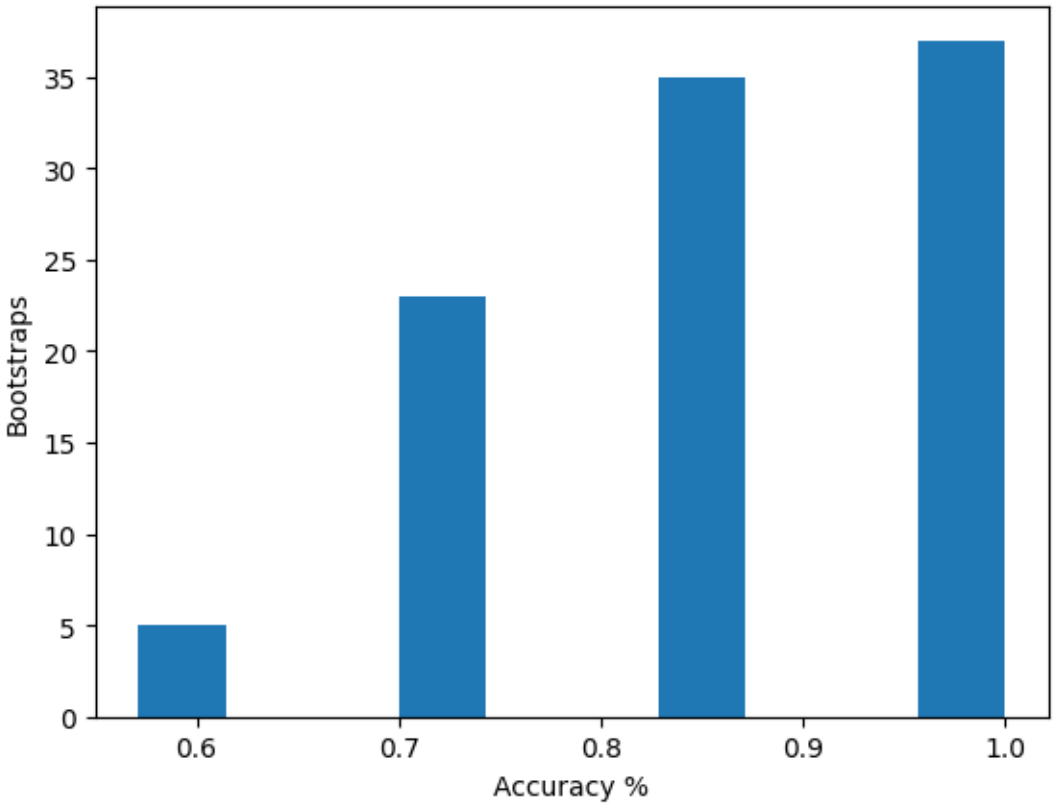
[35]:
```
accuracy2=[]

for i in range(n_iterations):
```

```
   X_train, X_test, Y_train, Y_test= train_test_split(x,yy, train_size= 0.7,␣
↪test_size=0.3)

   model = RandomForestClassifier()
   model.fit(X_train, Y_train)
   predictions= model.predict(X_test)

   #avg3= metrics.accuracy_score(YY_test, predictions3) #average across 100␣
↪bootstraps

   accuracy2.append(metrics.accuracy_score(Y_test, predictions))
   avg= mean(accuracy2)
```

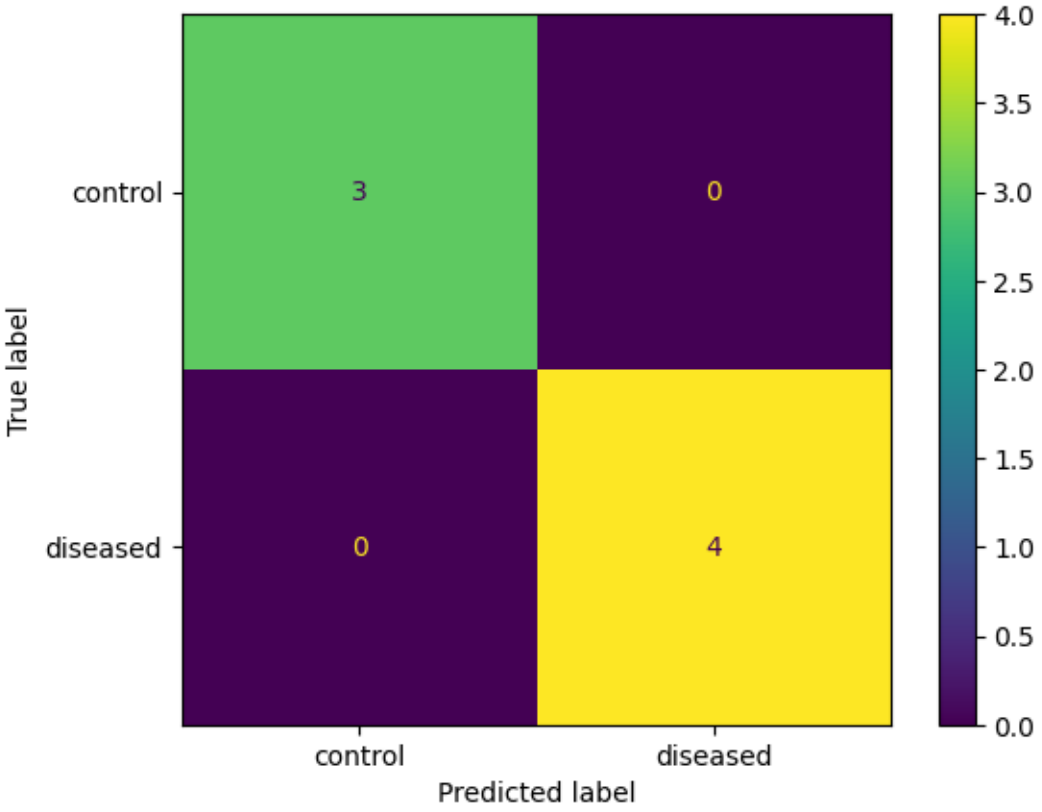
[36]:
```
avg
```

[36]: 0.8628571428571429

[37]:
```
plt.hist(accuracy2)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[38]:
```
r_f= model.predict(X_test)
ConfusionMatrixDisplay.from_predictions(Y_test, r_f)
plt.show()
```



[39]:
```
cm= confusion_matrix(Y_test, r_f)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  100 %
specificity:  100 %
```

[40]:
```
#BLOOD
```

[93]:
```
blooddata =loadmat('BWG_BL_CDvCTRL.mat')
```

[94]:
```
blooddf = gcparser(blooddata)
bloodx= blooddf.transpose()
```

[95]:
```
zz= blooddata['CLASS']

yy=[] # yy is training data
for i in zz:
     if i==1:
        yy.append("control")
     else:
        yy.append("diseased")
```

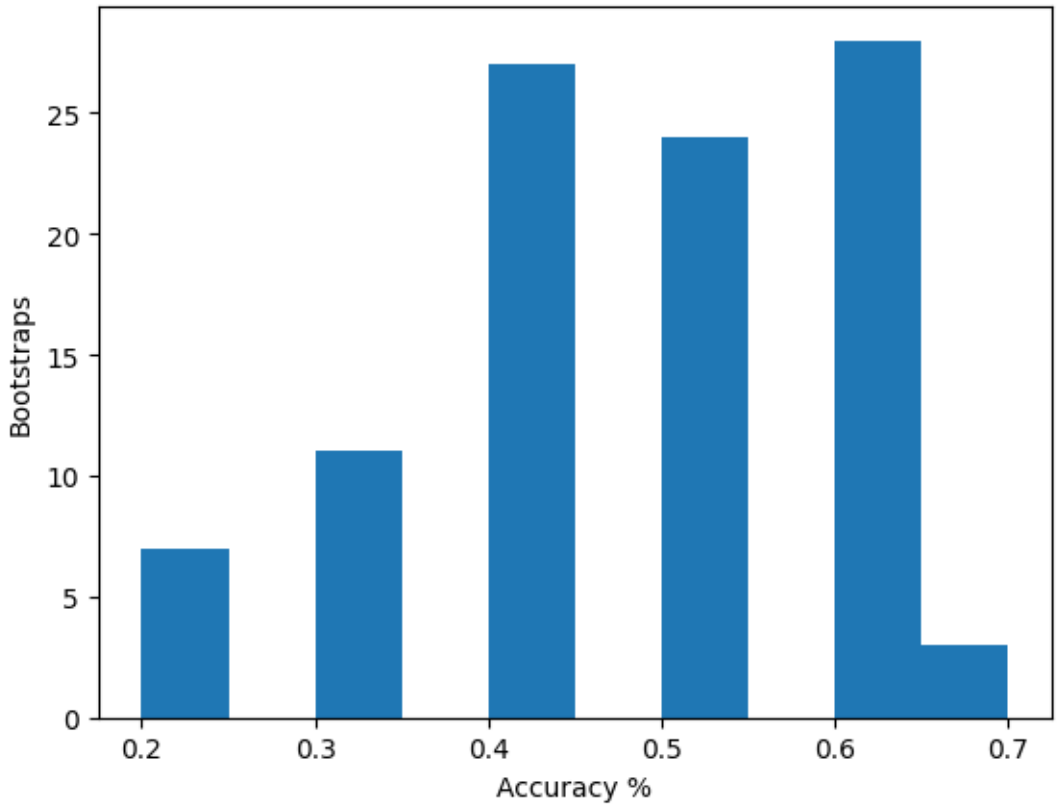
[44]:
```
#SVM BOOTSTRAP
```

[96]:
```
bloodaccuracy=[]
for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(bloodx,yy, train_size= 0.
 ↪7, test_size=0.3)

    model = svm.SVC()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

     #average across 100 bootstraps
    bloodaccuracy.append(metrics.accuracy_score(Y_test, predictions) )
    avg= mean(bloodaccuracy)
```

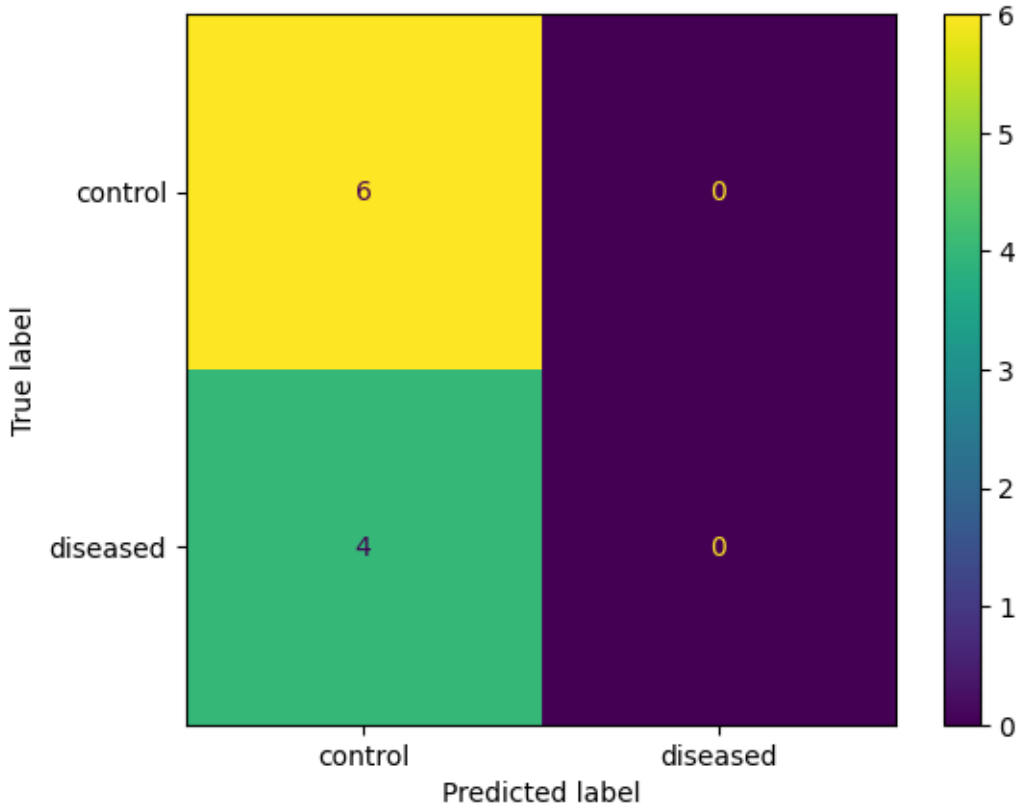
[97]:
```
avg
```

[97]:
```
0.464
```

[98]:
```
plt.hist(bloodaccuracy)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[99]:
```
bloodpred= model.predict(X_test)
ConfusionMatrixDisplay.from_predictions(Y_test, bloodpred)
plt.show()
```



[100]:
```
cm= confusion_matrix(Y_test, bloodpred)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  0 %
specificity:  100 %
```

[50]:
```
#RANDOM FOREST CLASSIFIER BOOTSTRAP
```

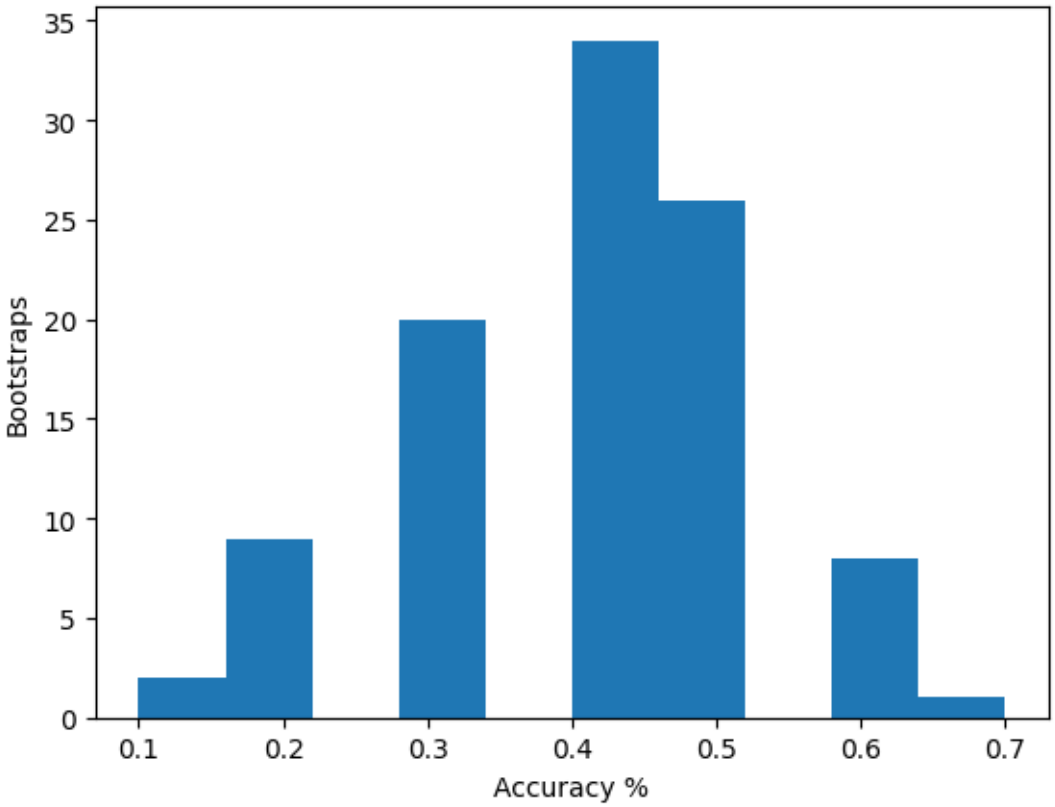
[101]:
```
bloodaccuracy1=[]
for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(bloodx,yy, train_size= 0.
 ↪7, test_size=0.3)
```

```
    model = RandomForestClassifier()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

    bloodaccuracy1.append(metrics.accuracy_score(Y_test, predictions) )
    avg= mean(bloodaccuracy1)
```

[102]:
```
avg
```

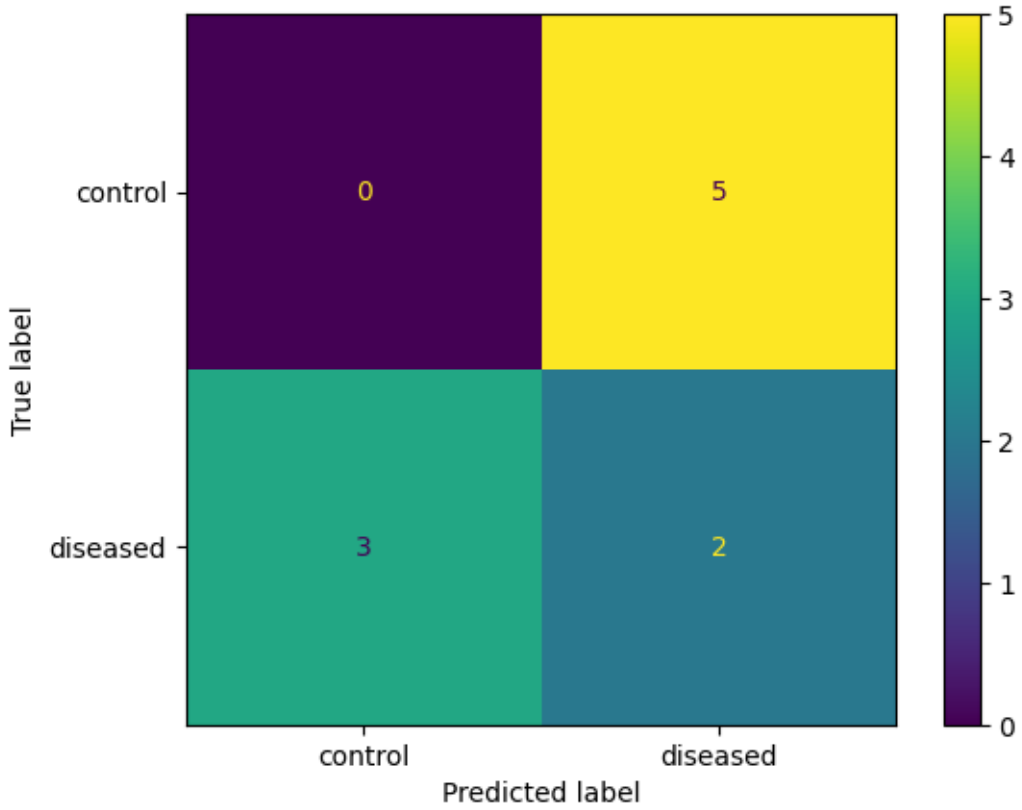
[102]: 0.434

[53]:
```
plt.hist(bloodaccuracy1)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[54]:
```
bloodpred1= model.predict(X_test)
```

```
ConfusionMatrixDisplay.from_predictions(Y_test, bloodpred1) #better than leave␣
↪on out and sum
plt.show()
```



[55]:
```
cm= confusion_matrix(Y_test, bloodpred1)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  40 %
specificity:  0 %
```

[56]:
```
#URINE
```

[57]:
```
urinedata =loadmat('BWG_UR_CDvCTRL.mat')
```

[58]:
```
urinedf = gcparser(urinedata)
urinex= urinedf.transpose()
```

[59]:
```
zz= urinedata['CLASS']

yy=[] # yy is training data
for i in zz:
     if i==1:
        yy.append("control")
     else:
        yy.append("diseased")
```

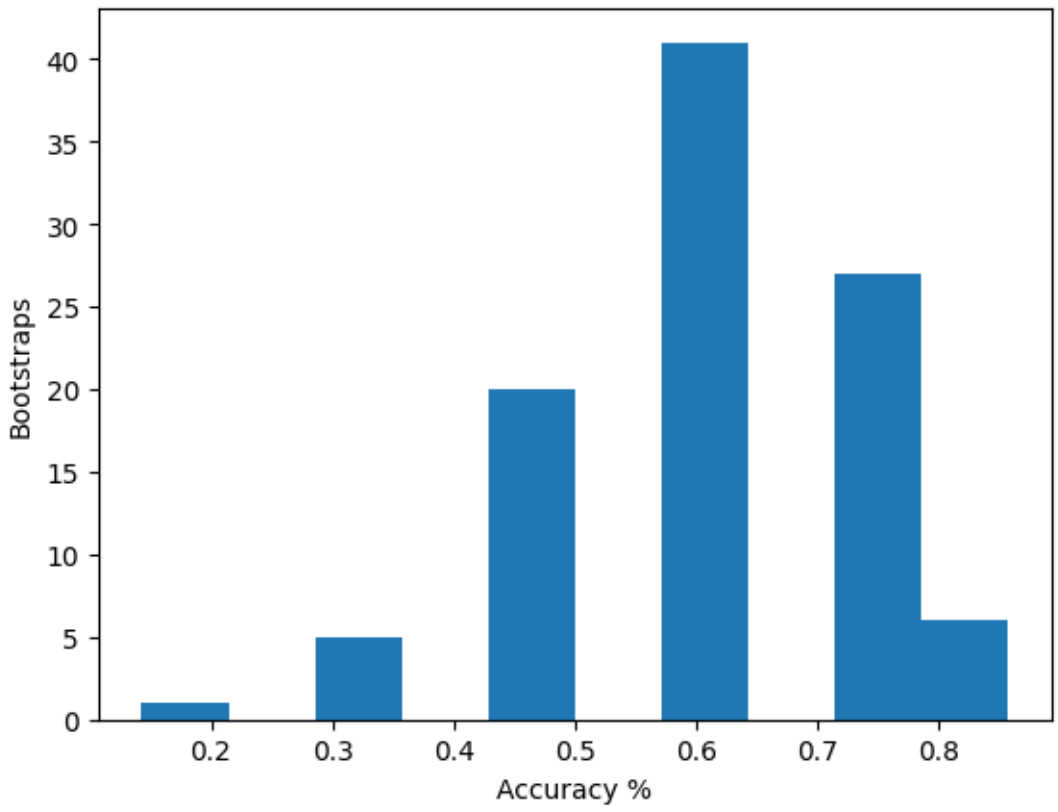
[60]:
```
#SVM BOOTSTRAP
```

[61]:
```
urineaccuracy=[]
for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(urinex,yy, train_size= 0.
 ↪7, test_size=0.3)

    model = svm.SVC()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

    urineaccuracy.append(metrics.accuracy_score(Y_test, predictions) )
    avg= mean(urineaccuracy)
```

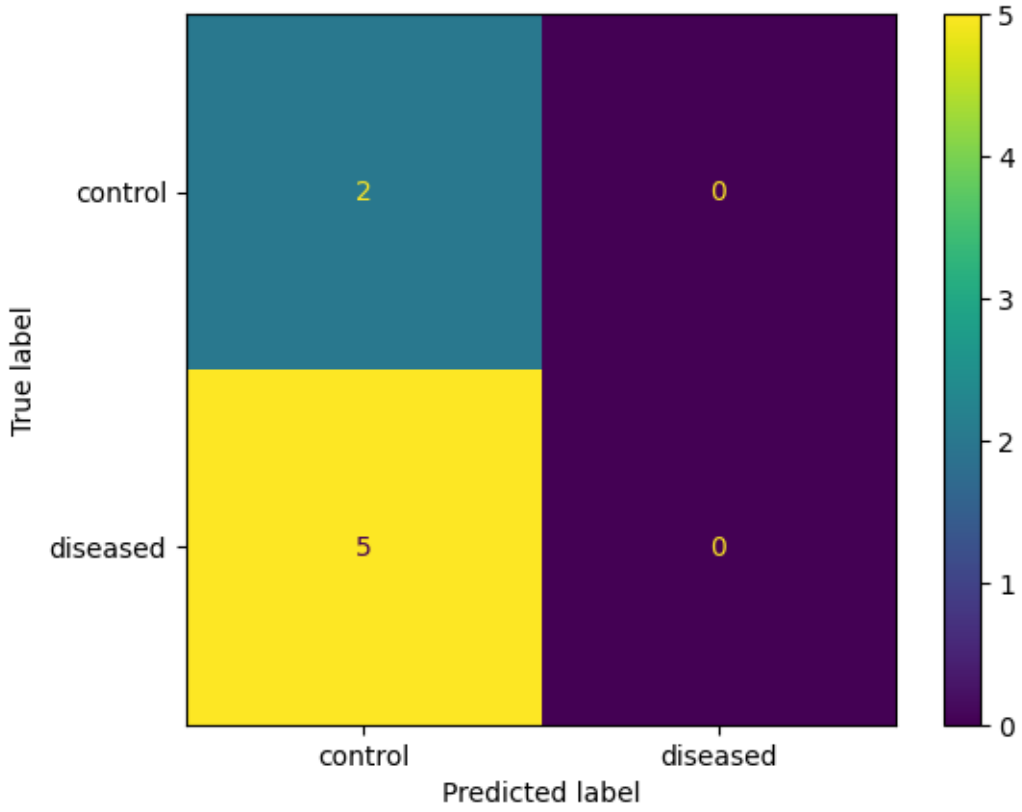
[62]:
```
avg
```

[62]:
```
0.58
```

[63]:
```
plt.hist(urineaccuracy)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[64]:
```
urinepred= model.predict(X_test)
ConfusionMatrixDisplay.from_predictions(Y_test, urinepred)
plt.show()
```



[65]:
```
cm= confusion_matrix(Y_test, urinepred)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  0 %
specificity:  100 %
```

[66]:
```
#RANDOM FOREST CLASSIFIER BOOTSTRAP
```

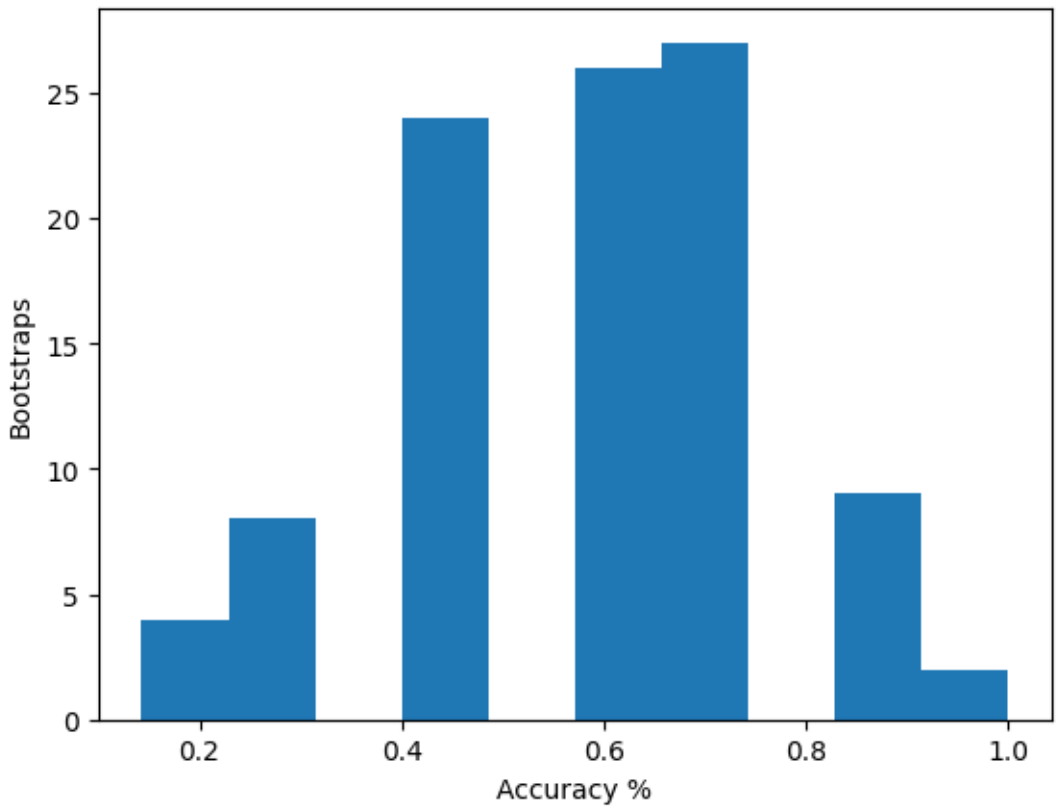
[67]:
```
urineaccuracy1=[]
for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(urinex,yy, train_size= 0.
↪7, test_size=0.3)
```

```
    model = RandomForestClassifier()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

    urineaccuracy1.append(metrics.accuracy_score(Y_test, predictions) )
    avg= mean(urineaccuracy1)
```

[68]:
```
avg
```

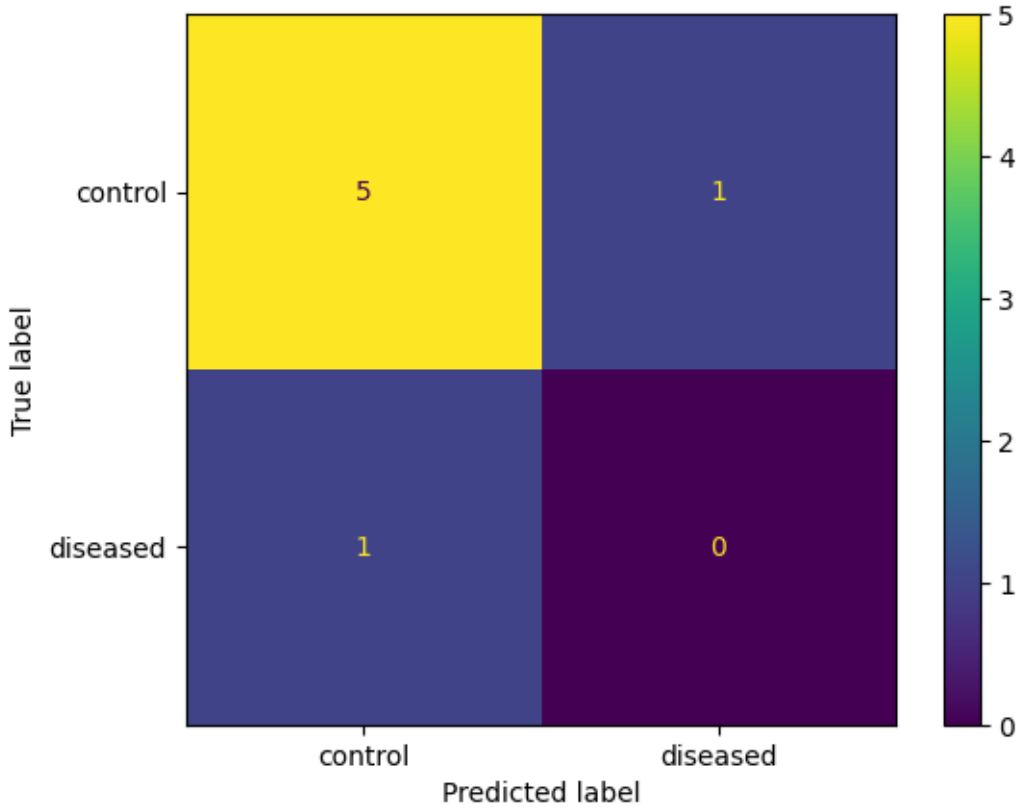
[68]: 0.57

[69]:
```
plt.hist(urineaccuracy1)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[70]:
```
urinepred1= model.predict(X_test)
```

```
ConfusionMatrixDisplay.from_predictions(Y_test, urinepred1) #better than leave␣
↪on out and sum
plt.show()
```



[71]:
```
cm= confusion_matrix(Y_test, urinepred1)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  0 %
specificity:  83 %
```

[72]:
```
#BREATH
```

[73]:
```
breathdata =loadmat('BWG_BR_CDvCTRL.mat')
```

[74]:
```
breathdf = gcparser(breathdata)
breathx= breathdf.transpose()
```

[75]:
```
zz= breathdata['CLASS']

yy=[] # yy is training data
for i in zz:
     if i==1:
        yy.append("control")
     else:
        yy.append("diseased")
```

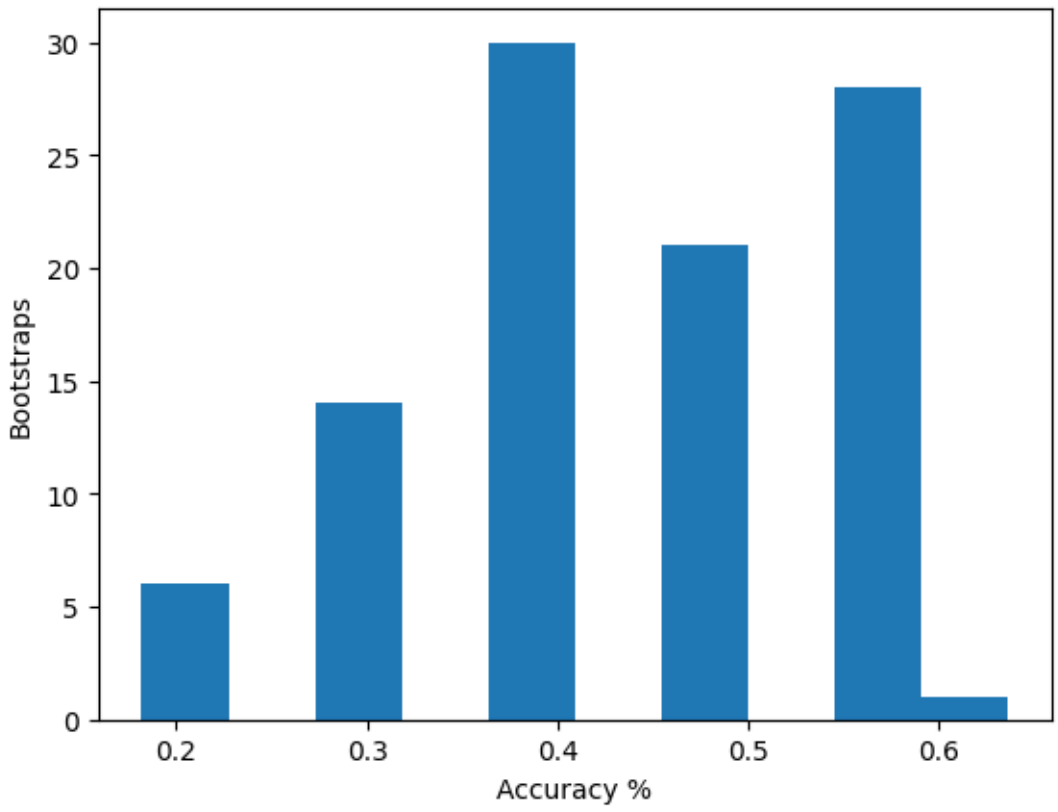
[76]:
```
#SVM BOOTSTRAP
```

[77]:
```
breathaccuracy=[]

for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(breathx,yy, train_size=␣
 ↪0.7, test_size=0.3)

    model = svm.SVC()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

    breathaccuracy.append(metrics.accuracy_score(Y_test, predictions))
    avg= mean(breathaccuracy)
```

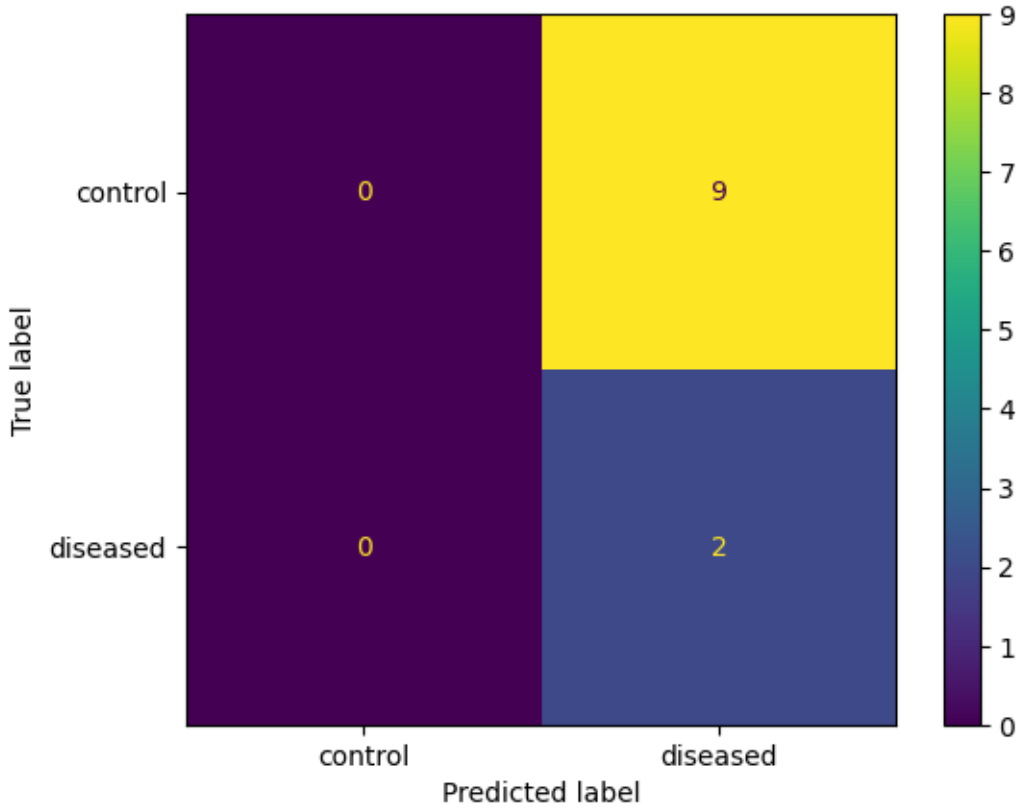
[78]:
```
avg
```

[78]:
```
0.4127272727272727
```

[79]:
```
plt.hist(breathaccuracy)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[80]:
```
breathpred= model.predict(X_test)
ConfusionMatrixDisplay.from_predictions(Y_test, breathpred)
plt.show()
```



[81]:
```
cm= confusion_matrix(Y_test, breathpred)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  100 %
specificity:  0 %
```

[82]:
```
#RANDOM FOREST CLASSIFIER
```

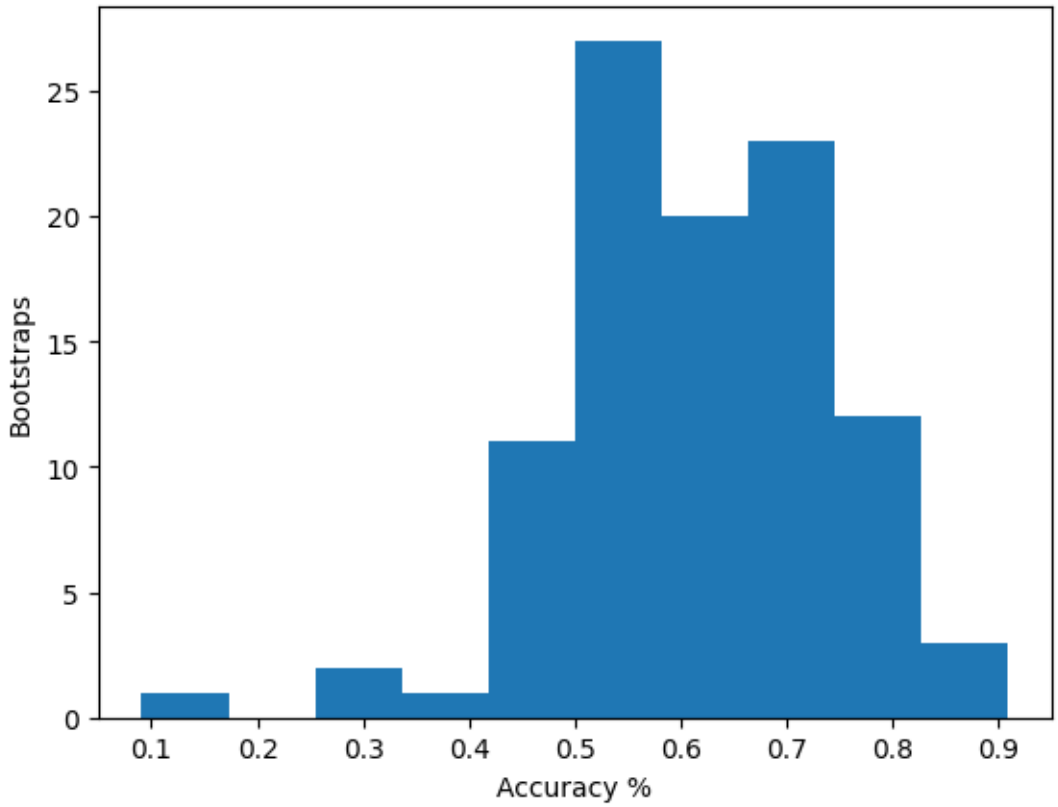
[83]:
```
breathaccuracy1=[]
for i in range(n_iterations):

    X_train, X_test, Y_train, Y_test= train_test_split(breathx,yy, train_size=␣
 ↪0.7, test_size=0.3)
```

```
    model = RandomForestClassifier()
    model.fit(X_train, Y_train)
    predictions= model.predict(X_test)

    breathaccuracy1.append(metrics.accuracy_score(Y_test, predictions) )
    avg= mean(breathaccuracy1)
```

[84]:
```
avg
```

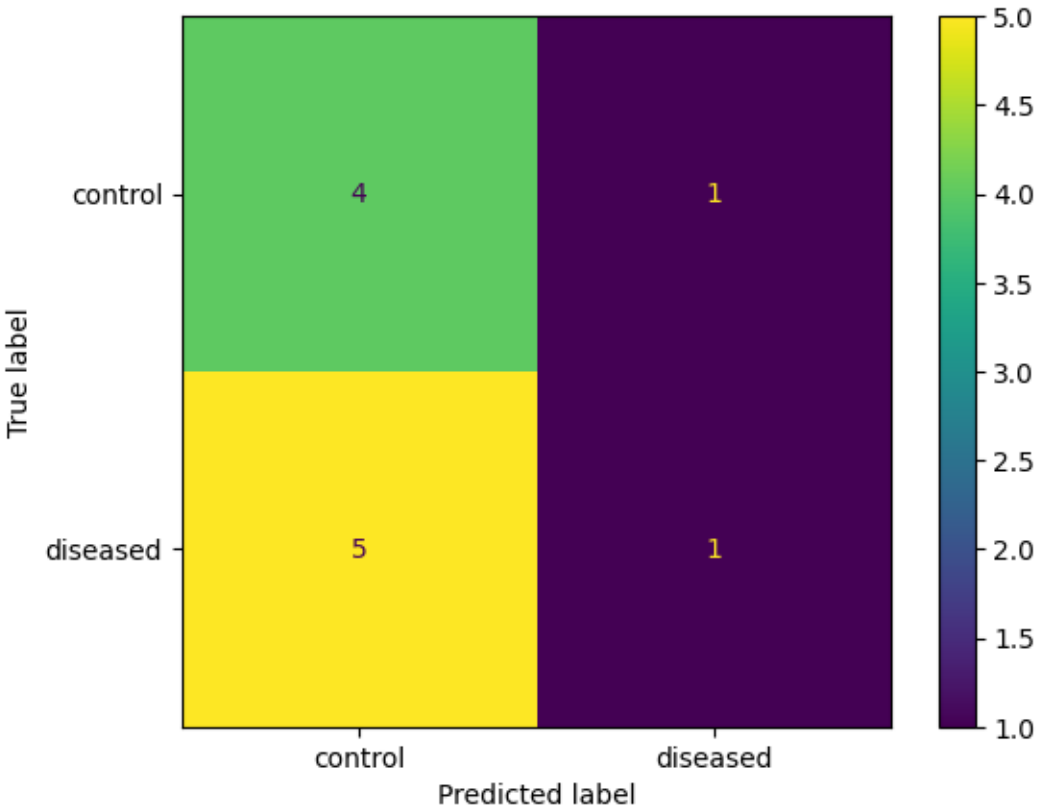
[84]: 0.6272727272727273

[85]:
```
plt.hist(breathaccuracy1)
plt.ylabel('Bootstraps')
plt.xlabel('Accuracy %')
plt.show()
```



[86]:
```
breathpred1= model.predict(X_test)
```

```
ConfusionMatrixDisplay.from_predictions(Y_test, breathpred1) #better than leave␣
↪on out and sum
plt.show()
```



[87]:
```
cm= confusion_matrix(Y_test, breathpred1)
TN= cm[0,0]
FP= cm[0,1]
FN= cm[1,0]
TP= cm[1,1]

sens = (TP/(TP+FN))*100
print('sensitivity: ', round(sens),'%')
spec =(TN/(TN+FP))*100
print('specificity: ', round(spec),'%')
```

```
sensitivity:  17 %
specificity:  80 %
```

[88]:
```
faeces_ttest= ttest_ind(accuracy, accuracy2) #sig
faeces_ttest
```

[88]:
```
Ttest_indResult(statistic=-9.54986910905066, pvalue=5.132039318042005e-18)
```

[103]:
```
blood_ttest= ttest_ind(bloodaccuracy, bloodaccuracy1) #not sig
blood_ttest
```

[103]:
```
Ttest_indResult(statistic=1.7248643247205713, pvalue=0.08611238093677644)
```

[90]:
```
urine_ttest= ttest_ind(urineaccuracy, urineaccuracy1) #not sig
urine_ttest
```

[90]:
```
Ttest_indResult(statistic=0.42182160083986914, pvalue=0.673612860316273)
```

[91]:
```
breath_ttest= ttest_ind(breathaccuracy, breathaccuracy1)
breath_ttest #sig
```

[91]:
```
Ttest_indResult(statistic=-11.802384561002423, pvalue=1.093379754117222e-24)
```

[92]:
```
perm_ttest= ttest_ind(accuracy2, accuracyfr) #sig
perm_ttest
```

[92]:
```
Ttest_indResult(statistic=13.27358892110159, pvalue=3.5330344015965584e-29)
```