Coursework 1 - Decision Trees Learning

Enter your candidate number here: 660050171

Summary

In this coursework, your task is to develop a machine learning classifier for predicting female patients that at high risk of Diabetes. Your model is to support clinicians in identifying patients who are likely to have “Diabetes”. The dataset has 9 attributes in total including the “target/label” attribute. The full dataset is available on ELE under assessment coursework 1. The dataset consists of the following:

Dataset

1. preg: Number of times pregnant
2. plas: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. pres: Diastolic blood pressure (mm Hg)
4. skin: Triceps skin fold thickness (mm)
5. insu: 2-Hour serum insulin (mu U/ml)
6. mass: Body mass index (weight in kg/(height in m)^2)
7. pedi: Diabetes pedigree function
8. age: Age (years)
9. class: Class variable (0 or 1)

In [2]:
```
import pandas as pd
dia_all = pd.read_csv("diabetes.txt") # This loads the full dataset
                # In the file, attributes are separated by ,
```

In [3]:
```
dia_all.head(10)
```

Out[3]:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive
5	5	116	74	0	0	25.6	0.201	30	tested_negative
6	3	78	50	32	88	31.0	0.248	26	tested_positive
7	10	115	0	0	0	35.3	0.134	29	tested_negative
8	2	197	70	45	543	30.5	0.158	53	tested_positive
9	8	125	96	0	0	0.0	0.232	54	tested_positive

Seperate the inpout (attributes) from target (label)

In [4]:
```
sourcevars = dia_all.iloc[:,:-1] #all rows + all columns except the last one
targetvar = dia_all.iloc[:,-1:] #all rows + only the last column
```

In [5]:
```
sourcevars.head()
```

Out[5]:

	preg	plas	pres	skin	insu	mass	pedi	age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [6]:
```
targetvar.head()
```

Out[6]:

	class
0	tested_positive
1	tested_negative
2	tested_positive
3	tested_negative
4	tested_positive

Your answers

Please clearly highlight each task.

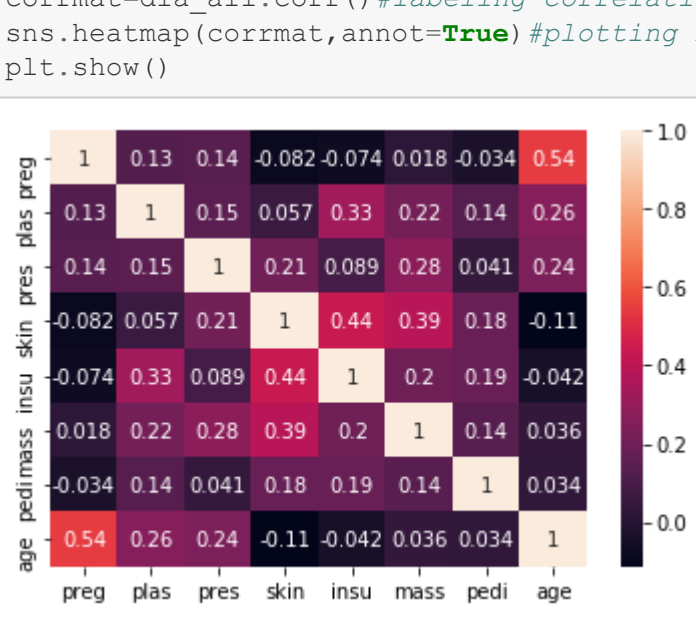
Taks 1.a [plotting mean, median, std and correlation matrix]

In [7]:
```
#your code here
colnames=['preg','plas','pres','skin','insu','mass','pedi','age']#creating a columns list

means= []
median=[]
sd=[]
#creating empty lists for each question
for i in range(8):
    means.append(sourcevars.iloc[:,i].mean())
for i in range(8):
    median.append(sourcevars.iloc[:,i].median())
for i in range(8):
    sd.append( sourcevars.iloc[:,i].std())
# for loop appends to each list the answers

df_info =pd.DataFrame(data=[means,median,sd],index = ['Mean','Median','Std'], columns=colnames)
# create a DataFrame with all 3 lists as data, set index and columns
df_info
```

Out[7]:

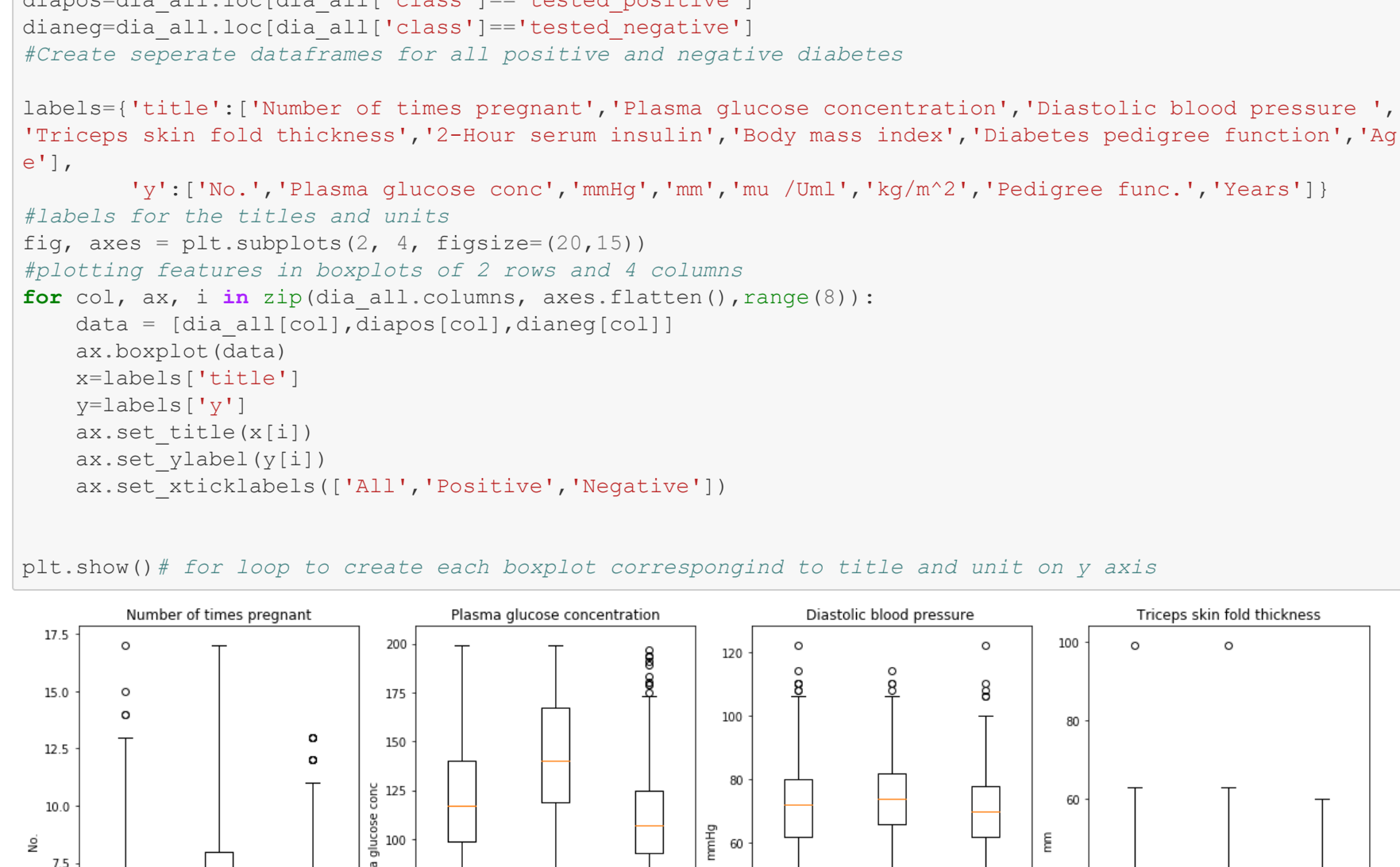
	preg	plas	pres	skin	insu	mass	pedi	age
Mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885
Median	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
Std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
corrmat=dia_all.corr()#labeling correlation matrix
sns.heatmap(corrmat,annot=True)#plotting heatmap, annot allows numbers in heatmap
plt.show()
```



Taks 1.b [Plotting Box plots]

In [9]:
```
import matplotlib.pyplot as plt
diapos=dia_all.loc[dia_all['class']=='tested_positive']
dianeg=dia_all.loc[dia_all['class']=='tested_negative']
#Create seperate dataframes for all positive and negative diabetes

labels={'title':['Number of times pregnant','Plasma glucose concentration','Diastolic blood pressure ',
'Triceps skin fold thickness','2-Hour serum insulin','Body mass index','Diabetes pedigree function','Ag
e'],
        'y':['No.','Plasma glucose conc','mmHg','mm','mu /Uml','kg/m^2','Pedigree func.','Years']}
#labels for the titles and units
fig, axes = plt.subplots(2, 4, figsize=(20,15))
#plotting features in boxplots of 2 rows and 4 columns
for col, ax, i in zip(dia_all.columns, axes.flatten(),range(8)):
    data = [dia_all[col],diapos[col],dianeg[col]]
    ax.boxplot(data)
    x=labels['title']
    y=labels['y']
    ax.set_title(x[i])
    ax.set_ylabel(y[i])
    ax.set_xticklabels(['All','Positive','Negative'])


plt.show()# for loop to create each boxplot correspongind to title and unit on y axis
```



Task 2.a [Building DT of 10 repeats]

In [10]:
```
# your code here ..... and so on
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
import numpy as np
#importing pakages

dt=DecisionTreeClassifier()# setting decion tree to variable
targetvar=targetvar.replace('tested_positive',1)
targetvar=targetvar.replace('tested_negative',0)
#replacing positive for diabetes to 1 and negative for diabetes to 0

X=sourcevars
y=targetvar
accs=[]#accuracy
senss=[]#sensitivity or recall or true positive rate
specs=[]#specificity or true negative rate
precs=[]#precision or PPV
npv=[]#negative predictive value
fnr=[]#false negative rate
fpr=[]#false postivie rate

# relabelling feautres and target, creating empty lists for performance
for i in range(10):

    X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=i, test_size=30)
    #splitting into train(70%) and test(30%), for loop to do this 10 times with different random_state
    dt.fit(X_train,y_train)#fitting into ML model
    y_pred = dt.predict(X_test)
    y_true = y_test
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()#splitting into true negative, false posit
ives etc
    accs.append(dt.score(X_test,y_test))
    senss.append(tp/(tp+fn))
    specs.append(tn/(tn+fp))
    precs.append(tp/(tp+fp))
    npv.append(tn/(tn+fn))
    fnr.append(fn/(fn+tp))
    fpr.append(fp/(fp+tn))
    #appending to empty lists for each iteration

for i in range(10):
    print('Iteration number '+str(i+1))
    print('Accuracy: '+str(accs[i]))
    print('Sensitivity: '+str(senss[i]))
    print('Specificity: '+str(specs[i]))
    print('Precision: '+str(precs[i]))
    print('NPV: '+str(npv[i]))
    print('FNR: '+str(fnr[i]))
    print('FPR: '+str(fpr[i]))
    print('-'*20)
#printing performance of each iteration using for loop
```

```
Iteration number 1
Accuracy: 0.7666666666666667
Sensitivity: 0.6
Specificity: 0.85
Precision: 0.6666666666666666
NPV: 0.8095238095238095
FNR: 0.4
FPR: 0.15
--------------------
Iteration number 2
Accuracy: 0.5666666666666667
Sensitivity: 0.4
Specificity: 0.65
Precision: 0.36363636363636365
NPV: 0.6842105263157895
FNR: 0.6
FPR: 0.35
--------------------
Iteration number 3
Accuracy: 0.6
Sensitivity: 0.6
Specificity: 0.6
Precision: 0.42857142857142855
NPV: 0.75
FNR: 0.4
FPR: 0.4
--------------------
Iteration number 4
Accuracy: 0.7666666666666667
Sensitivity: 0.5
Specificity: 0.9
Precision: 0.7142857142857143
NPV: 0.782608695652174
FNR: 0.5
FPR: 0.1
--------------------
Iteration number 5
Accuracy: 0.8
Sensitivity: 0.8
Specificity: 0.8
Precision: 0.6666666666666666
NPV: 0.8888888888888888
FNR: 0.2
FPR: 0.2
--------------------
Iteration number 6
Accuracy: 0.8
Sensitivity: 0.9
Specificity: 0.75
Precision: 0.6428571428571429
NPV: 0.9375
FNR: 0.1
FPR: 0.25
--------------------
Iteration number 7
Accuracy: 0.8333333333333334
Sensitivity: 0.8
Specificity: 0.85
Precision: 0.7272727272727273
NPV: 0.8947368421052632
FNR: 0.2
FPR: 0.15
--------------------
Iteration number 8
Accuracy: 0.7666666666666667
Sensitivity: 0.7
Specificity: 0.8
Precision: 0.6363636363636364
NPV: 0.8421052631578947
FNR: 0.3
FPR: 0.2
--------------------
Iteration number 9
Accuracy: 0.6
Sensitivity: 0.5
Specificity: 0.65
Precision: 0.4166666666666667
NPV: 0.7222222222222222
FNR: 0.5
FPR: 0.35
--------------------
Iteration number 10
Accuracy: 0.7333333333333333
Sensitivity: 0.7
Specificity: 0.75
Precision: 0.5833333333333334
NPV: 0.8333333333333334
FNR: 0.3
FPR: 0.25
--------------------
```

In [11]:
```
def Mean(lst):
    return sum(lst) / len(lst) # creating a function to calculate mean performance
print('Mean performance of all 10 iterations')
print('Mean Accuracy: ', Mean(accs))
print('Mean Sensitivity: ', Mean(senss))
print('Mean Specificity: ', Mean(specs))
print('Mean Precision: ', Mean(precs))
print('Mean NPV: ', Mean(npv))
print('Mean FNR: ',Mean(fnr))
print('Mean FPR: ', Mean(fpr))
```

```
Mean performance of all 10 iterations
Mean Accuracy:  0.7233333333333333
Mean Sensitivity:  0.6500000000000001
Mean Specificity:  0.76
Mean Precision:  0.5846320346320346
Mean NPV:  0.8145129581199375
Mean FNR:  0.35
Mean FPR:  0.24
```

Task 2.b [Building entropy DT]

In [12]:
```
dt2=DecisionTreeClassifier(criterion='entropy')

X=sourcevars
y=targetvar
accs2=[]#accuracy
senss2=[]#sensitivity or recall or true positive rate
specs2=[]#specificity or true negative rate
precs2=[]#precision or PPV
npv2=[]#negative predictive value
fnr2=[]#false negative rate
fpr2=[]#false postivie rate

# relabelling feautres and target, creating empty lists for performance
for i in range(10):

    X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=i, test_size=30)
    #splitting into train(70%) and test(30%), for loop to do this 10 times with different random_state
    dt2.fit(X_train,y_train)#fitting into ML model
    y_pred = dt.predict(X_test)
    y_true = y_test
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()#splitting into true negative, false posit
ives etc
    accs2.append(dt.score(X_test,y_test))
    senss2.append(tp/(tp+fn))
    specs2.append(tn/(tn+fp))
    precs2.append(tp/(tp+fp))
    npv2.append(tn/(tn+fn))
    fnr2.append(fn/(fn+tp))
    fpr2.append(fp/(fp+tn))
    #appending to empty lists for each iteration

for i in range(10):
    print('Iteration number '+str(i+1))
    print('Accuracy')
    print('Gini: ',accs[i],' Entropy: ',accs2[i])
    print('Sensitivity')
    print('Gini: ',senss[i],' Entropy: ',senss2[i])
    print('Specificity')
    print('Gini: ',specs[i],' Entropy: ',specs2[i])
    print('Precision')
    print('Gini: ',precs[i],' Entropy: ',precs[i])
    print('NPV')
    print('Gini: ',npv[i],' Entropy: ',npv2[i])
    print('FNR')
    print('Gini: ',fnr[i],' Entropy: ',fnr[i])
    print('FPR')
    print('Gini: ',fpr[i],' Entropy: ',fpr[i])
    print('-'*80)
```

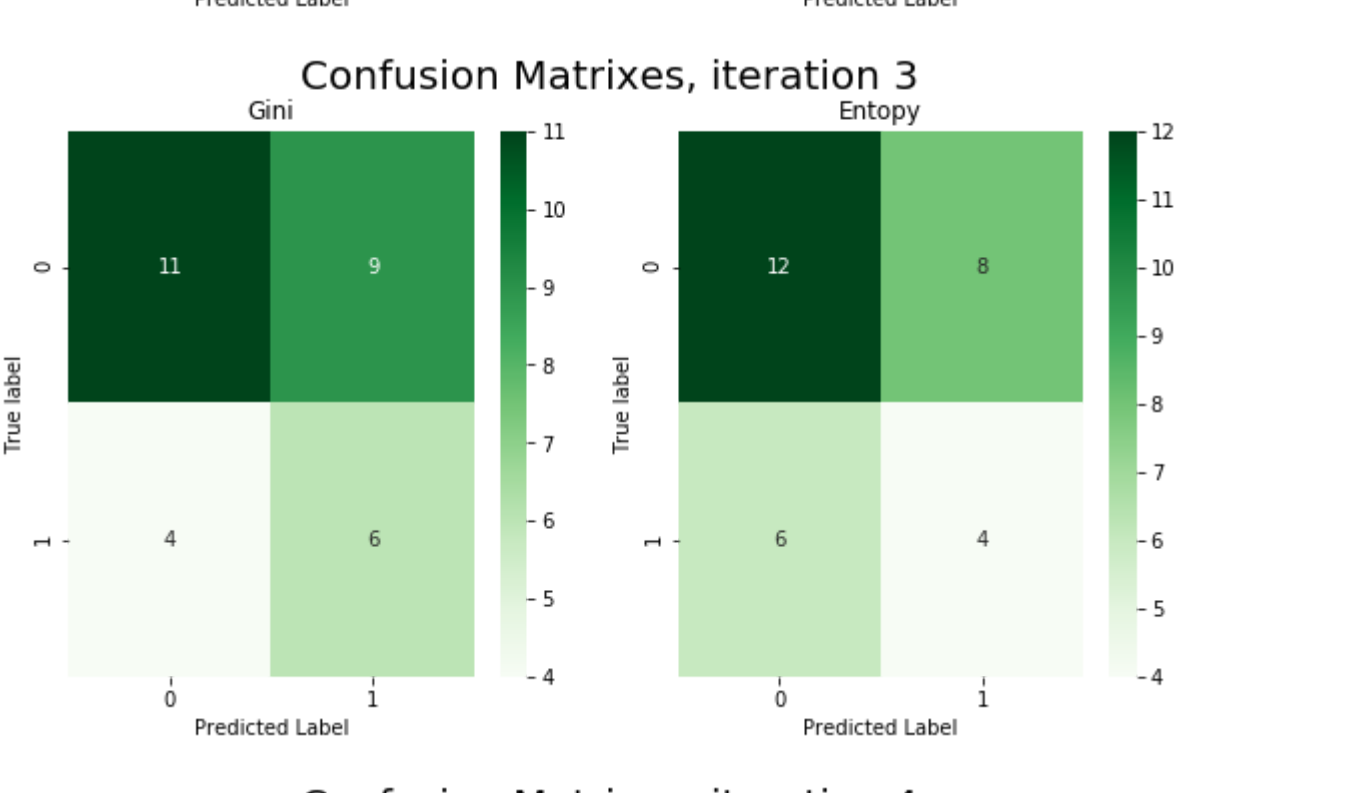
```
Iteration number 1
Accuracy
Gini:  0.7666666666666667  Entropy:  0.9666666666666667
Sensitivity
Gini:  0.6  Entropy:  1.0
Specificity
Gini:  0.85  Entropy:  0.95
Precision
Gini:  0.6666666666666666  Entropy:  0.6666666666666666
NPV
Gini:  0.8095238095238095  Entropy:  1.0
FNR
Gini:  0.4  Entropy:  0.4
FPR
Gini:  0.15  Entropy:  0.15
--------------------------------------------------------------------------------
Iteration number 2
Accuracy
Gini:  0.5666666666666667  Entropy:  1.0
Sensitivity
Gini:  0.4  Entropy:  1.0
Specificity
Gini:  0.65  Entropy:  1.0
Precision
Gini:  0.36363636363636365  Entropy:  0.36363636363636365
NPV
Gini:  0.6842105263157895  Entropy:  1.0
FNR
Gini:  0.6  Entropy:  0.6
FPR
Gini:  0.35  Entropy:  0.35
--------------------------------------------------------------------------------
Iteration number 3
Accuracy
Gini:  0.6  Entropy:  1.0
Sensitivity
Gini:  0.6  Entropy:  1.0
Specificity
Gini:  0.6  Entropy:  1.0
Precision
Gini:  0.42857142857142855  Entropy:  0.42857142857142855
NPV
Gini:  0.75  Entropy:  1.0
FNR
Gini:  0.4  Entropy:  0.4
FPR
Gini:  0.4  Entropy:  0.4
--------------------------------------------------------------------------------
Iteration number 4
Accuracy
Gini:  0.7666666666666667  Entropy:  1.0
Sensitivity
Gini:  0.5  Entropy:  1.0
Specificity
Gini:  0.9  Entropy:  1.0
Precision
Gini:  0.7142857142857143  Entropy:  0.7142857142857143
NPV
Gini:  0.782608695652174  Entropy:  1.0
FNR
Gini:  0.5  Entropy:  0.5
FPR
Gini:  0.1  Entropy:  0.1
--------------------------------------------------------------------------------
Iteration number 5
Accuracy
Gini:  0.8  Entropy:  1.0
Sensitivity
Gini:  0.8  Entropy:  1.0
Specificity
Gini:  0.8  Entropy:  1.0
Precision
Gini:  0.6666666666666666  Entropy:  0.6666666666666666
NPV
Gini:  0.8888888888888888  Entropy:  1.0
FNR
Gini:  0.2  Entropy:  0.2
FPR
Gini:  0.2  Entropy:  0.2
--------------------------------------------------------------------------------
Iteration number 6
Accuracy
Gini:  0.8  Entropy:  1.0
Sensitivity
Gini:  0.9  Entropy:  1.0
Specificity
Gini:  0.75  Entropy:  1.0
Precision
Gini:  0.6428571428571429  Entropy:  0.6428571428571429
NPV
Gini:  0.9375  Entropy:  1.0
FNR
Gini:  0.1  Entropy:  0.1
FPR
Gini:  0.25  Entropy:  0.25
--------------------------------------------------------------------------------
Iteration number 7
Accuracy
Gini:  0.8333333333333334  Entropy:  1.0
Sensitivity
Gini:  0.8  Entropy:  1.0
Specificity
Gini:  0.85  Entropy:  1.0
Precision
Gini:  0.7272727272727273  Entropy:  0.7272727272727273
NPV
Gini:  0.8947368421052632  Entropy:  1.0
FNR
Gini:  0.2  Entropy:  0.2
FPR
Gini:  0.15  Entropy:  0.15
--------------------------------------------------------------------------------
Iteration number 8
Accuracy
Gini:  0.7666666666666667  Entropy:  1.0
Sensitivity
Gini:  0.7  Entropy:  1.0
Specificity
Gini:  0.8  Entropy:  1.0
Precision
Gini:  0.6363636363636364  Entropy:  0.6363636363636364
NPV
Gini:  0.8421052631578947  Entropy:  1.0
FNR
Gini:  0.3  Entropy:  0.3
FPR
Gini:  0.2  Entropy:  0.2
--------------------------------------------------------------------------------
Iteration number 9
Accuracy
Gini:  0.6  Entropy:  1.0
Sensitivity
Gini:  0.5  Entropy:  1.0
Specificity
Gini:  0.65  Entropy:  1.0
Precision
Gini:  0.4166666666666667  Entropy:  0.4166666666666667
NPV
Gini:  0.7222222222222222  Entropy:  1.0
FNR
Gini:  0.5  Entropy:  0.5
FPR
Gini:  0.35  Entropy:  0.35
--------------------------------------------------------------------------------
Iteration number 10
Accuracy
Gini:  0.7333333333333333  Entropy:  0.7333333333333333
Sensitivity
Gini:  0.7  Entropy:  0.7
Specificity
Gini:  0.75  Entropy:  0.75
Precision
Gini:  0.5833333333333334  Entropy:  0.5833333333333334
NPV
Gini:  0.8333333333333334  Entropy:  0.8333333333333334
FNR
Gini:  0.3  Entropy:  0.3
FPR
Gini:  0.25  Entropy:  0.25
--------------------------------------------------------------------------------
```

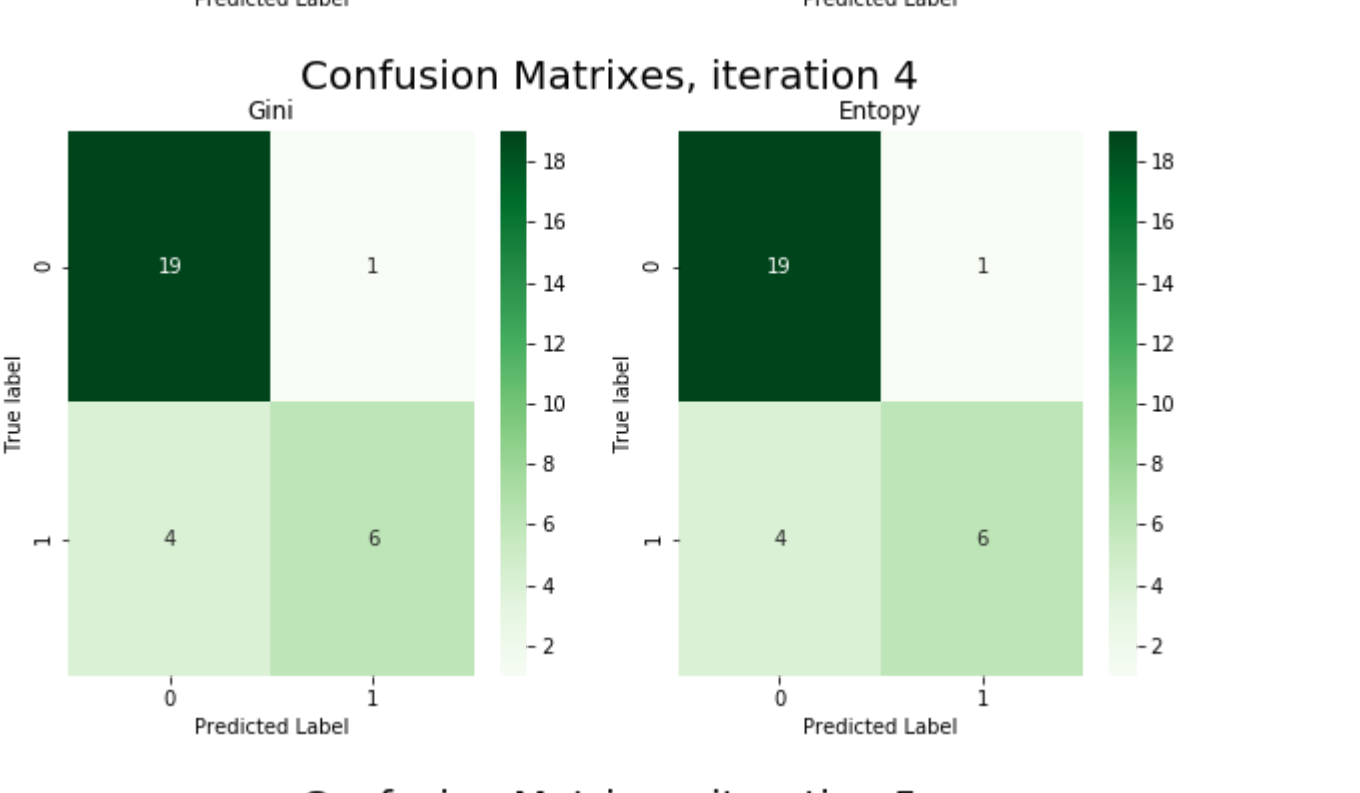
In [13]:
```
#printing means of performance of the 10 iterations
print('Gini Decision Tree')
print('Mean Accuracy: ', Mean(accs))
print('Mean Sensitivity: ', Mean(senss))
print('Mean Specificity: ', Mean(specs))
print('Mean Precision: ', Mean(precs))
print('Mean NPV: ', Mean(npv))
print('Mean FNR: ',Mean(fnr))
print('Mean FPR: ', Mean(fpr))

print('-'*80)
print('Entropy Decision Tree')
print('Mean Accuracy: ', Mean(accs2))
print('Mean Sensitivity: ', Mean(senss2))
print('Mean Specificity: ', Mean(specs2))
print('Mean Precision: ', Mean(precs2))
print('Mean NPV: ', Mean(npv2))
print('Mean FNR: ',Mean(fnr2))
print('Mean FPR: ', Mean(fpr2))
```

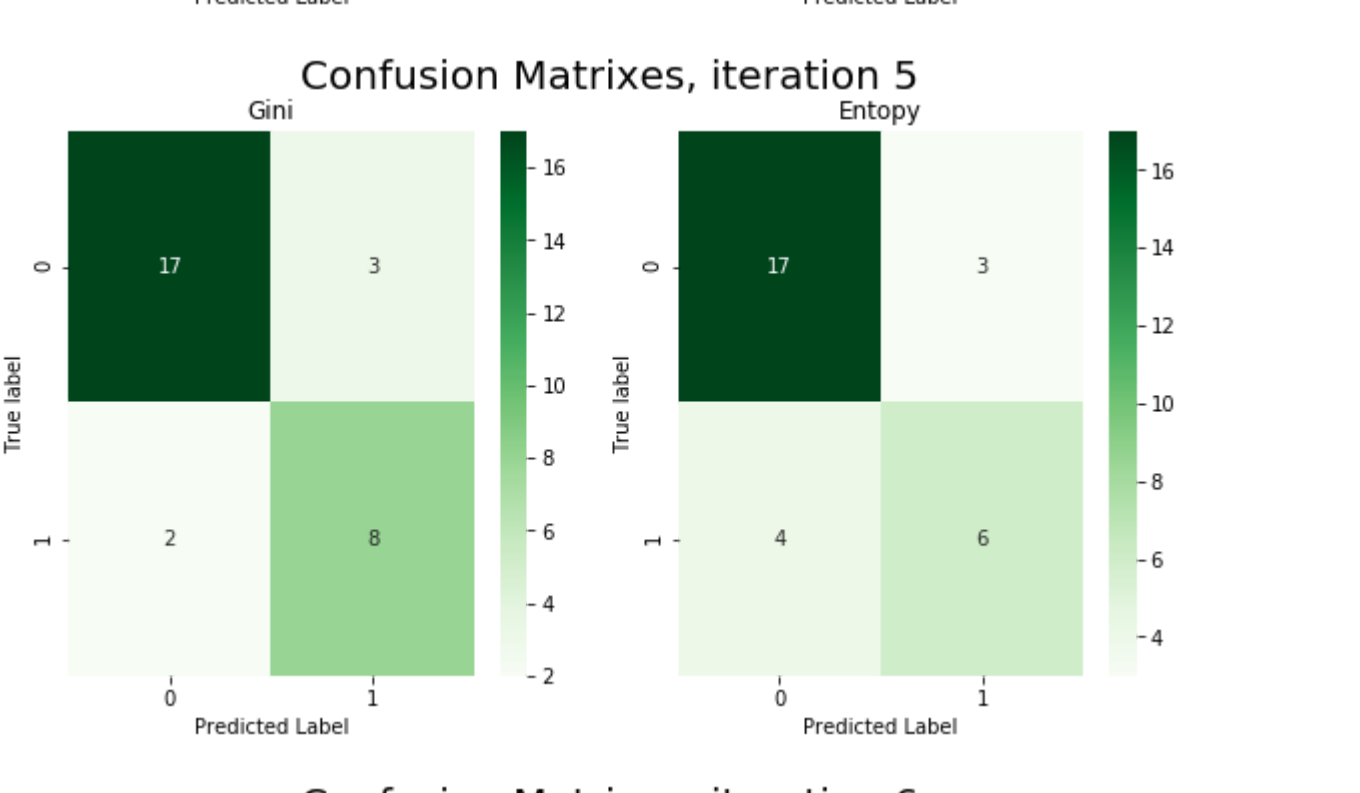
```
Gini Decision Tree
Mean Accuracy:  0.7233333333333333
Mean Sensitivity:  0.6500000000000001
Mean Specificity:  0.76
Mean Precision:  0.5846320346320346
Mean NPV:  0.8145129581199375
Mean FNR:  0.35
Mean FPR:  0.24
--------------------------------------------------------------------------------
Entropy Decision Tree
Mean Accuracy:  0.97
Mean Sensitivity:  0.97
Mean Specificity:  0.97
Mean Precision:  0.9492424242424244
Mean NPV:  0.9833333333333334
Mean FNR:  0.03
Mean FPR:  0.03
```

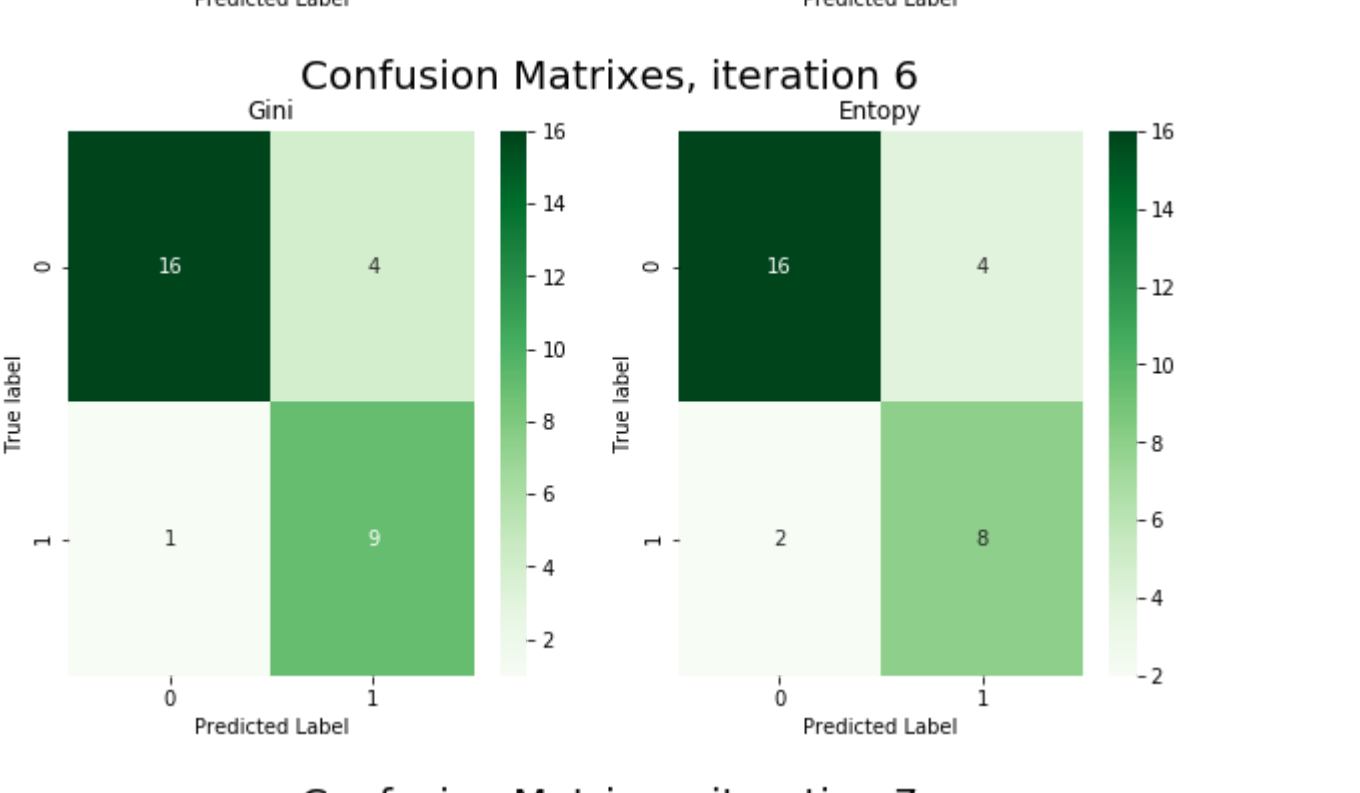
Task 2c [Comparing Gini and Entropy DT]

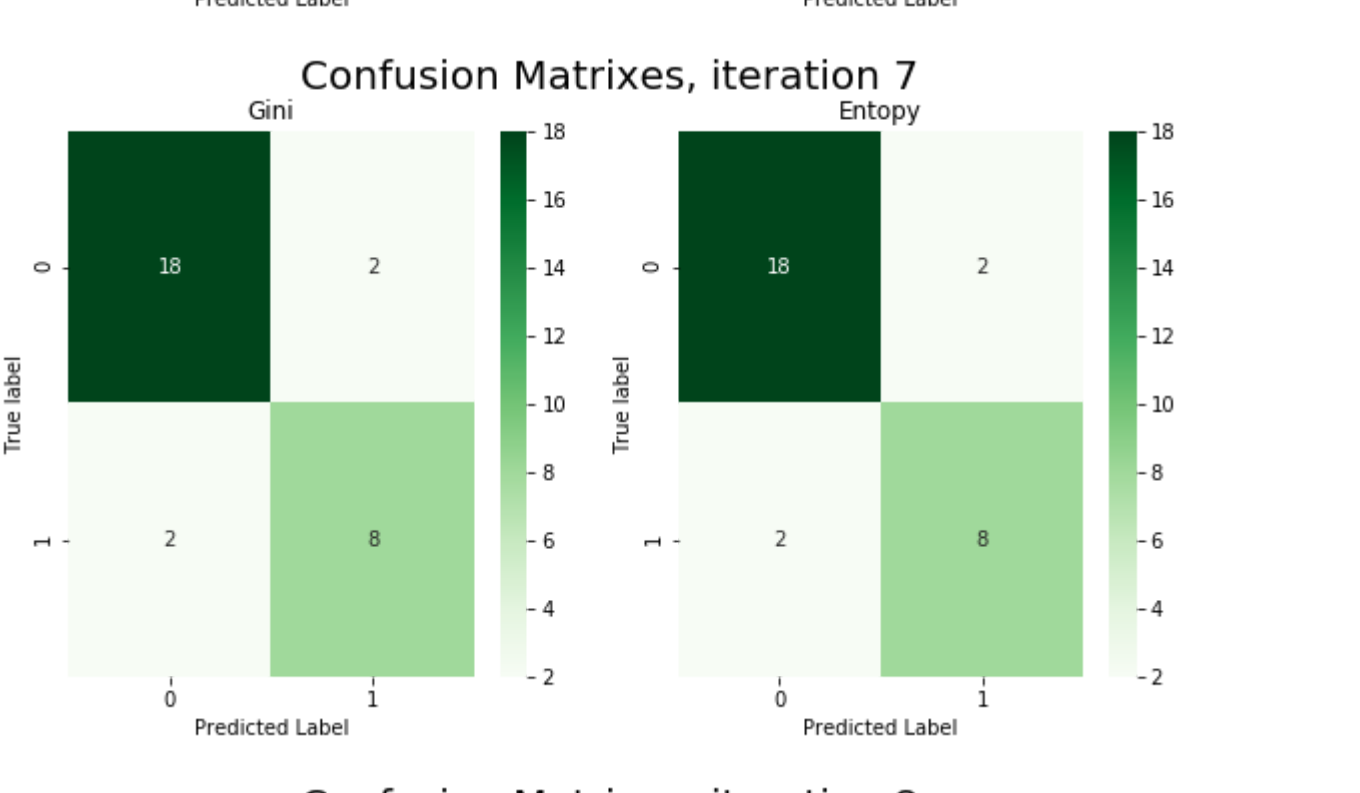
```python
In [14]: from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

#for loop to set iterations
for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=i, test_size=30)
    #fitting to both entropy and gini classifiers
    dt.fit(X_train,y_train)
    dt2.fit(X_train,y_train)
    # confisuon matrix
    y_pred = dt.predict(X_test) #setting the model's predicted score
    y_true = y_test
    cm_dt = confusion_matrix(y_true,y_pred)#plotting on confusion matrix

    y_pred2 = dt2.predict(X_test) #repeat for Entropy
    y_true = y_test
    cm_dt2 = confusion_matrix(y_true,y_pred2)

    plt.figure(figsize=(10,5))#plotting each iteration with confusion matrix of both models

    plt.suptitle("Confusion Matrixes, iteration "+str(i+1),fontsize=20)#

    plt.subplot(1,2,1, title='Gini')
    sns.heatmap(cm_dt,cbar=True,annot=True,cmap="Greens",fmt="d")
    plt.xlabel('Predicted Label')
    plt.ylabel('True label')

    plt.subplot(1,2,2, title='Entopy')
    sns.heatmap(cm_dt2,cbar=True,annot=True,cmap="Greens",fmt="d")
    plt.xlabel('Predicted Label')
    plt.ylabel('True label')
    plt.show()
```

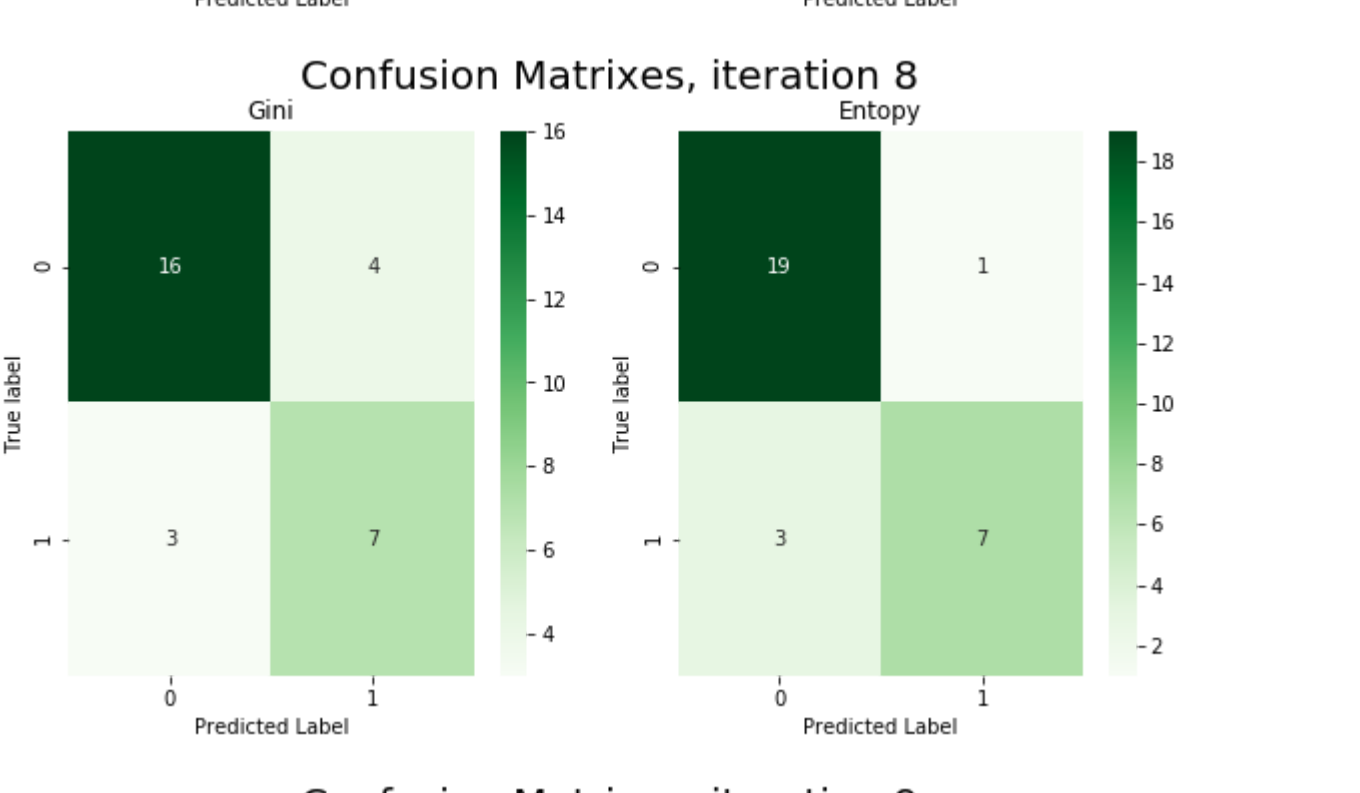


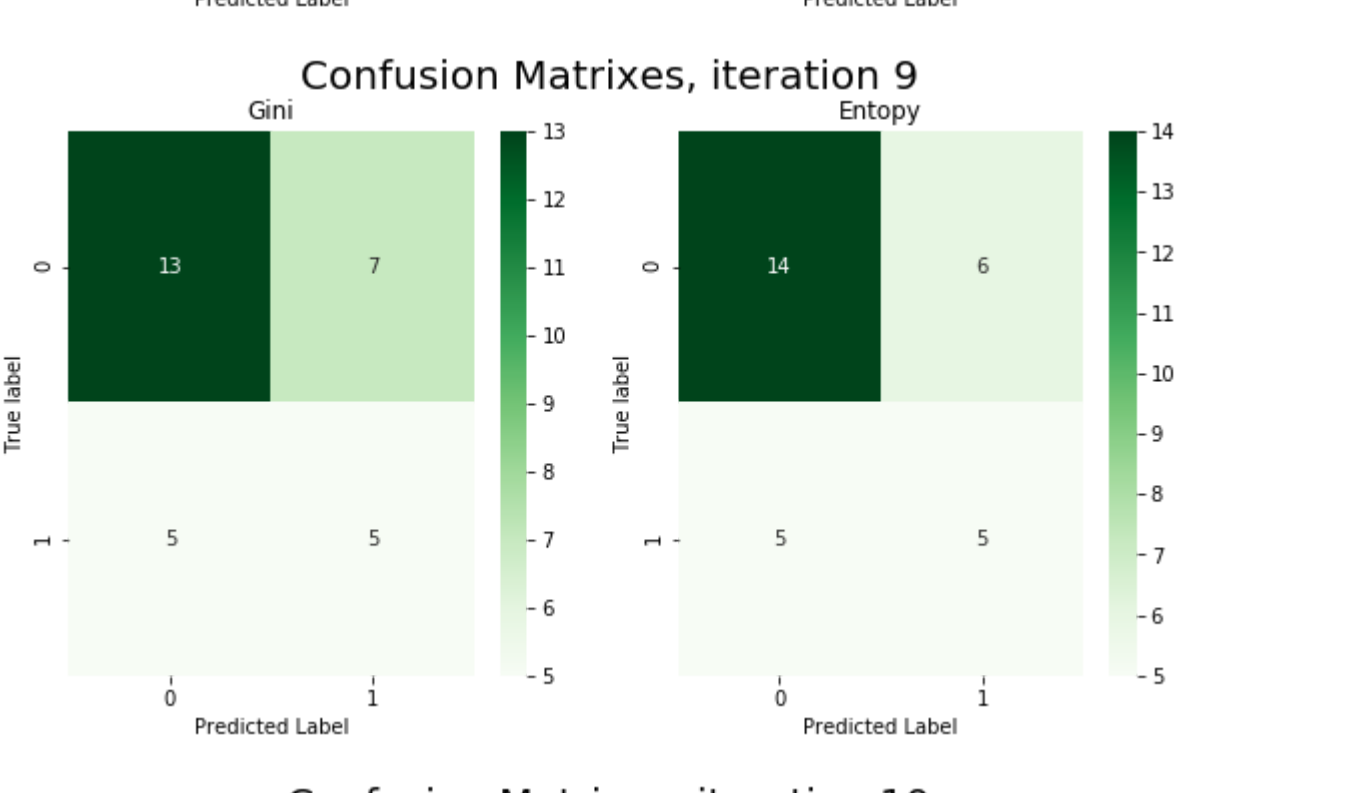


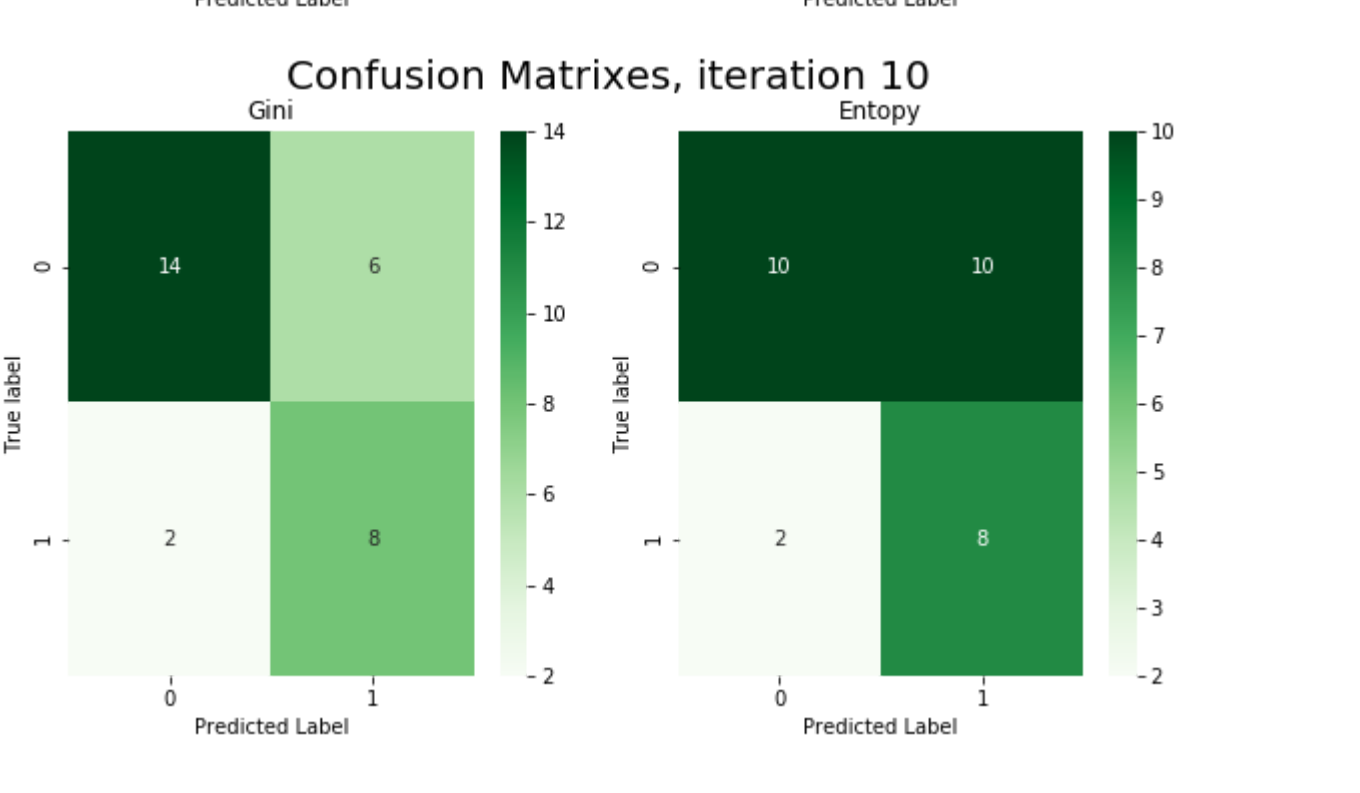










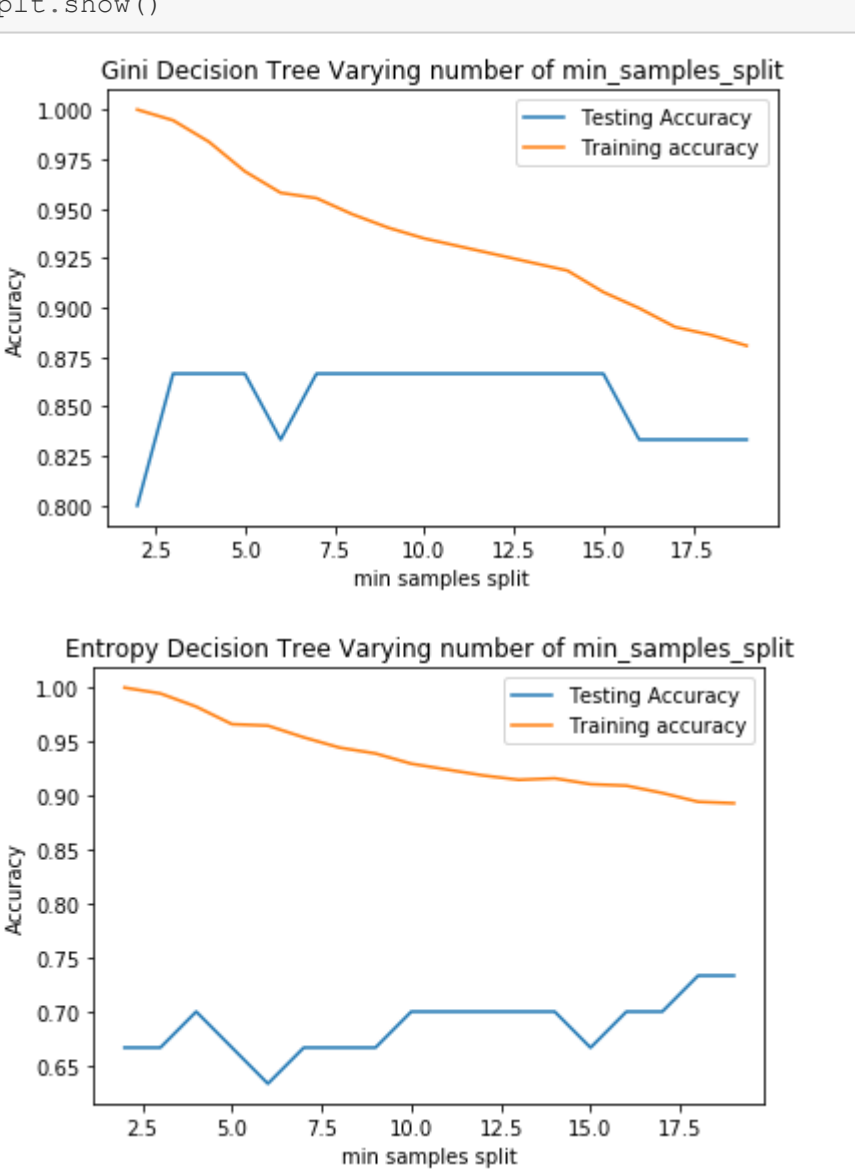


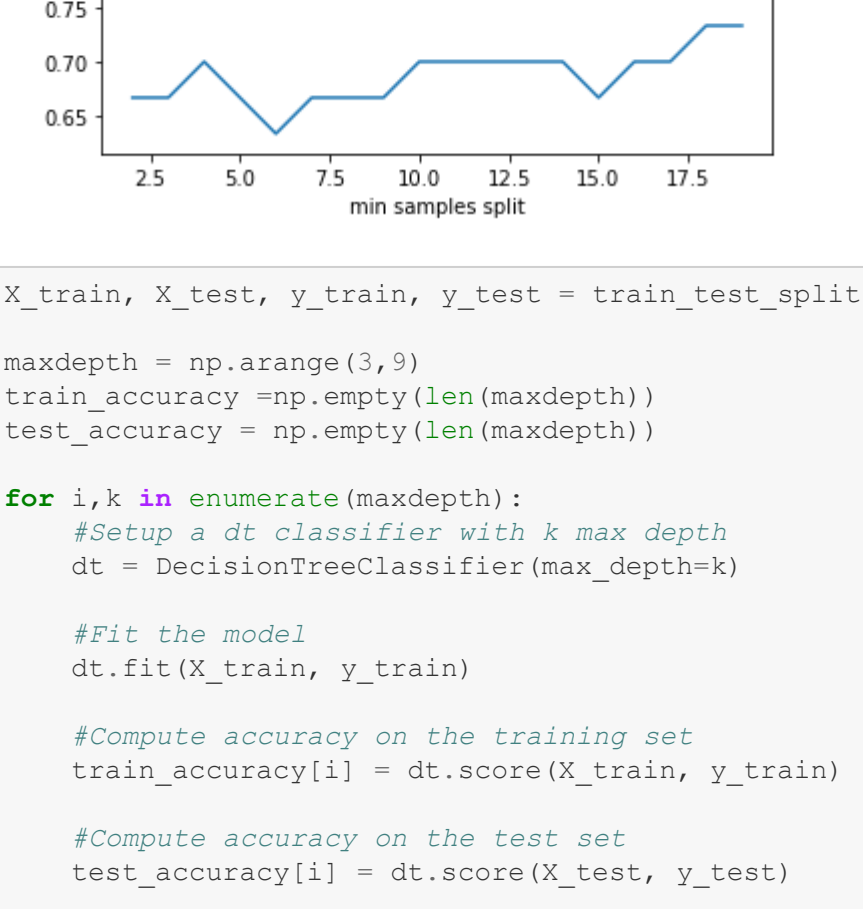
Task 2.d[Standardising data]

Standardising will not change the outcome of the decision tree models. Standardising is useful for models where scale variances between features will affect decision making (e.g. for KNN features on a bigger scale will have more weight) by scaling the features to have a mean of zero. For decision trees the nodes are split based on information gain or gini values of a single feature each split, therefore the scale of the features would not matter.

Task 3

```python
In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42,
                                                    test_size=30)
samplesplit = np.arange(2,20)#range of min_samples_split
train_accuracy =np.empty(len(samplesplit))
test_accuracy = np.empty(len(samplesplit))

for i,k in enumerate(samplesplit):
    #Setup a dt classifier with k min_sample_split)
    dt = DecisionTreeClassifier(min_samples_split=k)

    #Fit the model
    dt.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = dt.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = dt.score(X_test, y_test)

plt.title('Gini Decision Tree Varying number of min_samples_split')
plt.plot(samplesplit, test_accuracy, label='Testing Accuracy')
plt.plot(samplesplit, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('min samples split')
plt.ylabel('Accuracy')
plt.show()
#repeat with when criterion is 'entropy'
for i,k in enumerate(samplesplit):
    #Setup a dt classifier with k min_sample_split)
    dt2 = DecisionTreeClassifier(criterion='entropy',min_samples_split=k)

    #Fit the model
    dt2.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = dt2.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = dt2.score(X_test, y_test)

plt.title('Entropy Decision Tree Varying number of min_samples_split')
plt.plot(samplesplit, test_accuracy, label='Testing Accuracy')
plt.plot(samplesplit, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('min samples split')
plt.ylabel('Accuracy')
plt.show()
```






```python
In [16]: X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42,
                                                    test_size=30)
maxdepth = np.arange(3,9)
train_accuracy =np.empty(len(maxdepth))
test_accuracy = np.empty(len(maxdepth))

for i,k in enumerate(maxdepth):
    #Setup a dt classifier with k max depth
    dt = DecisionTreeClassifier(max_depth=k)

    #Fit the model
    dt.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = dt.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = dt.score(X_test, y_test)

plt.title('Gini Decision Tree Varying number of max_depth')
plt.plot(maxdepth, test_accuracy, label='Testing Accuracy')
plt.plot(maxdepth, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('max depth')
plt.ylabel('Accuracy')
plt.show()

#repeat with criterion = entropy
for i,k in enumerate(maxdepth):
    #Setup a dt classifier with k max depth
    dt2 = DecisionTreeClassifier(max_depth=k)

    #Fit the model
    dt2.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = dt2.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = dt2.score(X_test, y_test)

plt.title('Entropy Decision Tree Varying number of max_depth')
plt.plot(maxdepth, test_accuracy, label='Testing Accuracy')
plt.plot(maxdepth, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('max depth')
plt.ylabel('Accuracy')
plt.show()
```