In [49]:
```
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from mlxtend.plotting import plot_decision_regions
```

In [2]:
```
dataset = pd.read_csv('dataset_mod.csv')
```

In [3]:
```
datasetmod = dataset.copy(deep = True)
datasetmod[['Age','Gender','sudden weight loss','weakness','class']] = dataset
```

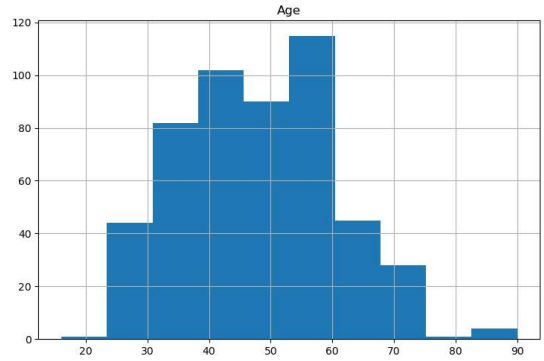
In [4]:
```
datasetmod.isnull().sum()
```

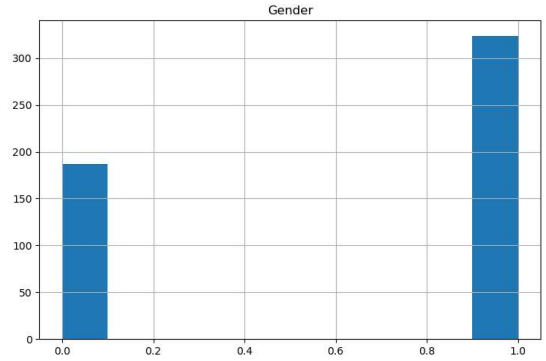
Out[4]:
```
Age                   8
Gender                9
sudden weight loss    6
weakness              7
Obesity               4
class                 9
dtype: int64
```

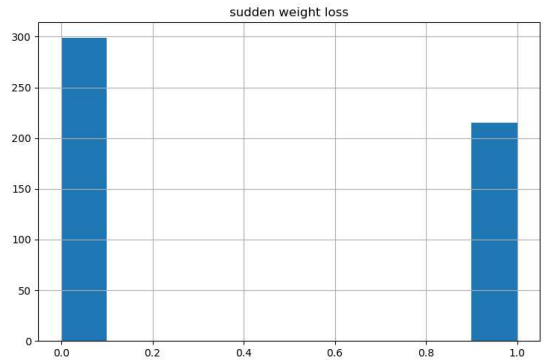
In [5]:
```
dataset['Gender'].replace(['Male', 'Female'],
                          [1, 0], inplace=True)
dataset['sudden weight loss'].replace(['Yes', 'No'],
                          [1, 0], inplace=True)
dataset['weakness'].replace(['Yes', 'No'],
                          [1, 0], inplace=True)
dataset['Obesity'].replace(['Yes', 'No'],
                          [1, 0], inplace=True)
dataset['class'].replace(['Positive', 'Negative'],
                          [1, 0], inplace=True)
```

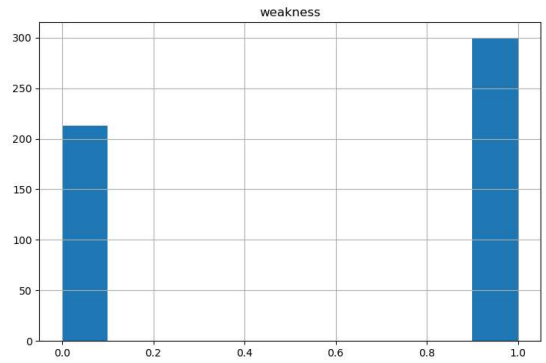
In [6]:
```
datasetmod['Gender'].replace(['Male', 'Female'],
                             [1, 0], inplace=True)
datasetmod['sudden weight loss'].replace(['Yes', 'No'],
                             [1, 0], inplace=True)
datasetmod['weakness'].replace(['Yes', 'No'],
                             [1, 0], inplace=True)
datasetmod['Obesity'].replace(['Yes', 'No'],
                             [1, 0], inplace=True)
datasetmod['class'].replace(['Positive', 'Negative'],
                             [1, 0], inplace=True)
```

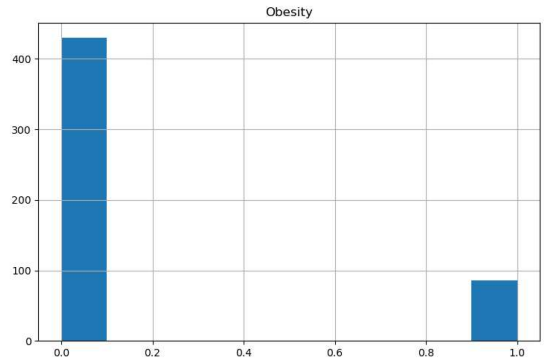
In [7]:
```
p = dataset.hist(figsize = (20, 20))
```

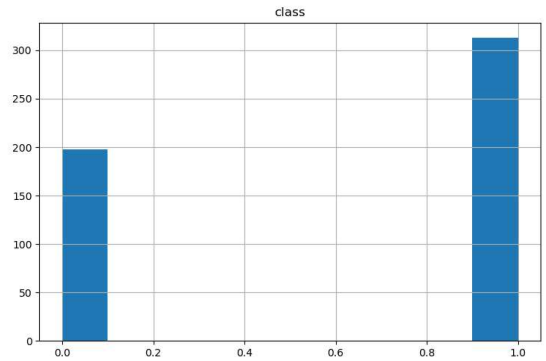










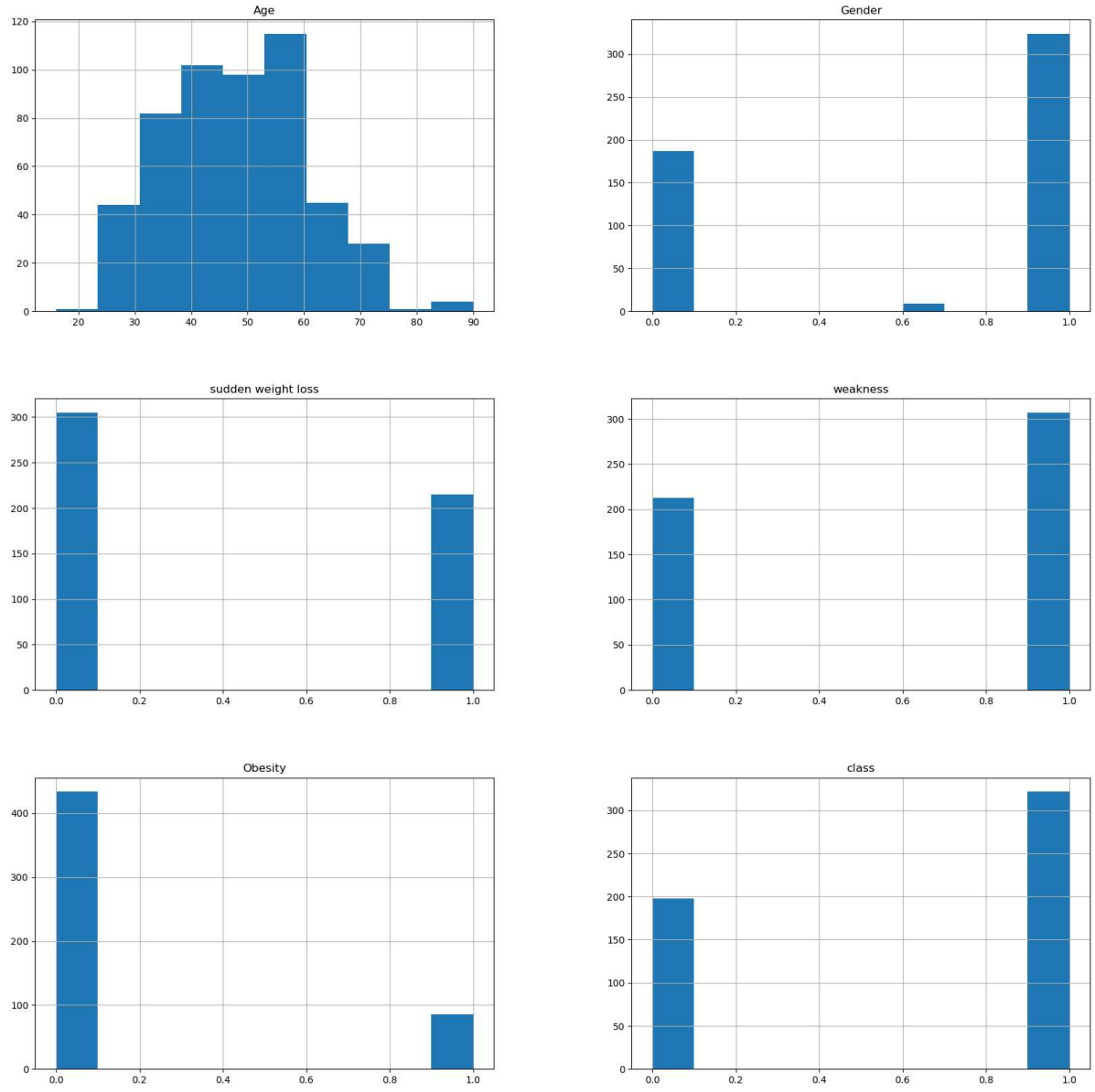


In [8]:
```
datasetmod['Age'].fillna(datasetmod['Age'].mean(), inplace = True)
datasetmod['Gender'].fillna(datasetmod['Gender'].mean(), inplace = True)
datasetmod['sudden weight loss'].fillna(datasetmod['sudden weight loss'].media
datasetmod['weakness'].fillna(datasetmod['weakness'].median(), inplace = True)
datasetmod['Obesity'].fillna(datasetmod['Obesity'].median(), inplace = True)
datasetmod['class'].fillna(datasetmod['class'].median(), inplace = True)
```

In [9]:
```
datasetmod.isnull().sum()
```

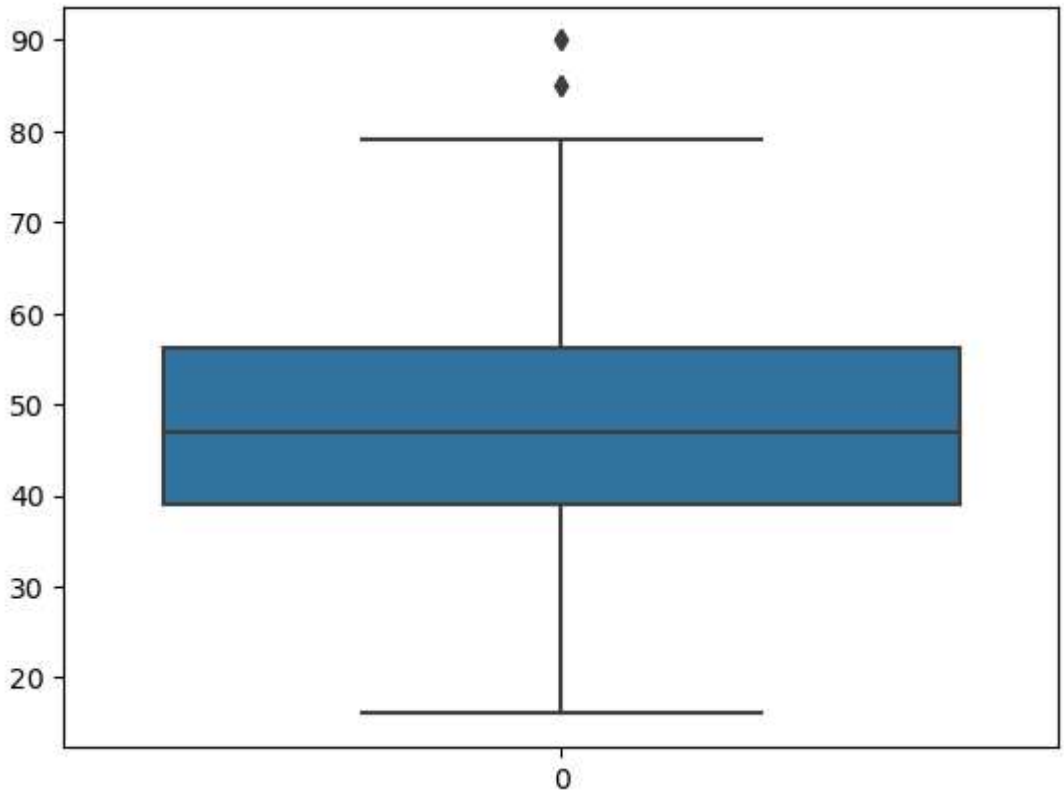
Out[9]:
```
Age                   0
Gender                0
sudden weight loss    0
weakness              0
Obesity               0
class                 0
dtype: int64
```

In [10]:
```
p = datasetmod.hist(figsize = (20, 20))
```

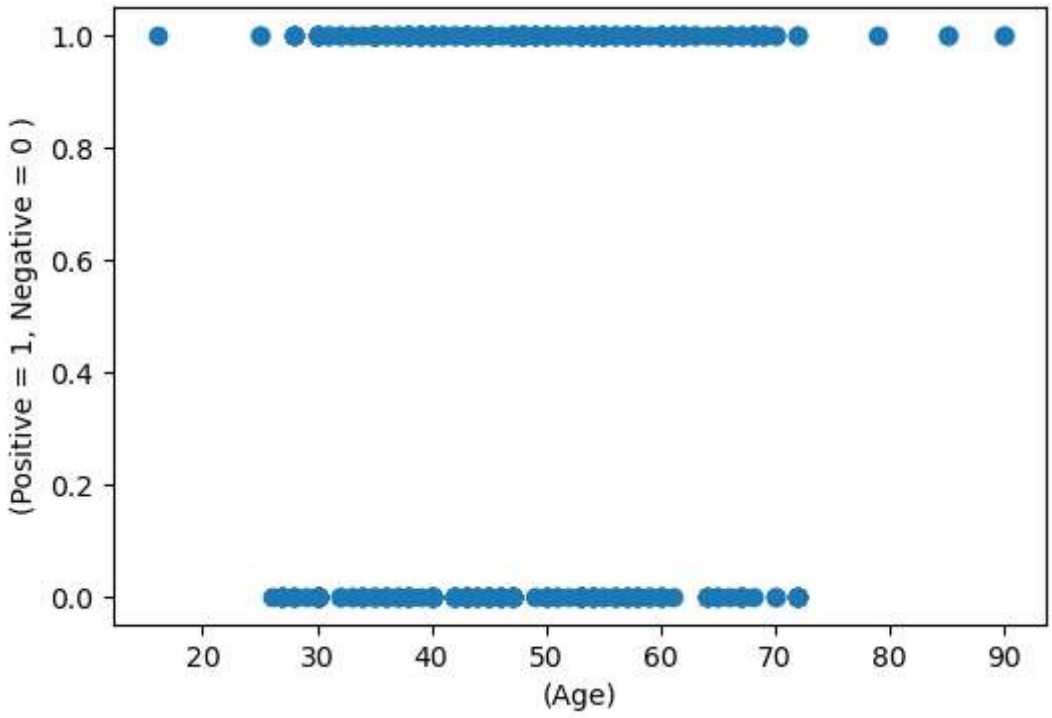


In [11]:
```
sns.boxplot(dataset['Age'])
```

Out[11]: <Axes: >



In [12]:
```
fig, ax = plt.subplots(figsize = (6,4))
ax.scatter(dataset['Age'],dataset['class'])

ax.set_xlabel('(Age)')

ax.set_ylabel('(Positive = 1, Negative = 0 )')
plt.show()
```



In [13]:
```
Q1 = dataset['Age'].quantile(0.25)
Q3 = dataset['Age'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR
print(IQR)
print (lower)
print (upper)
```

```
17.25
13.125
82.125
```

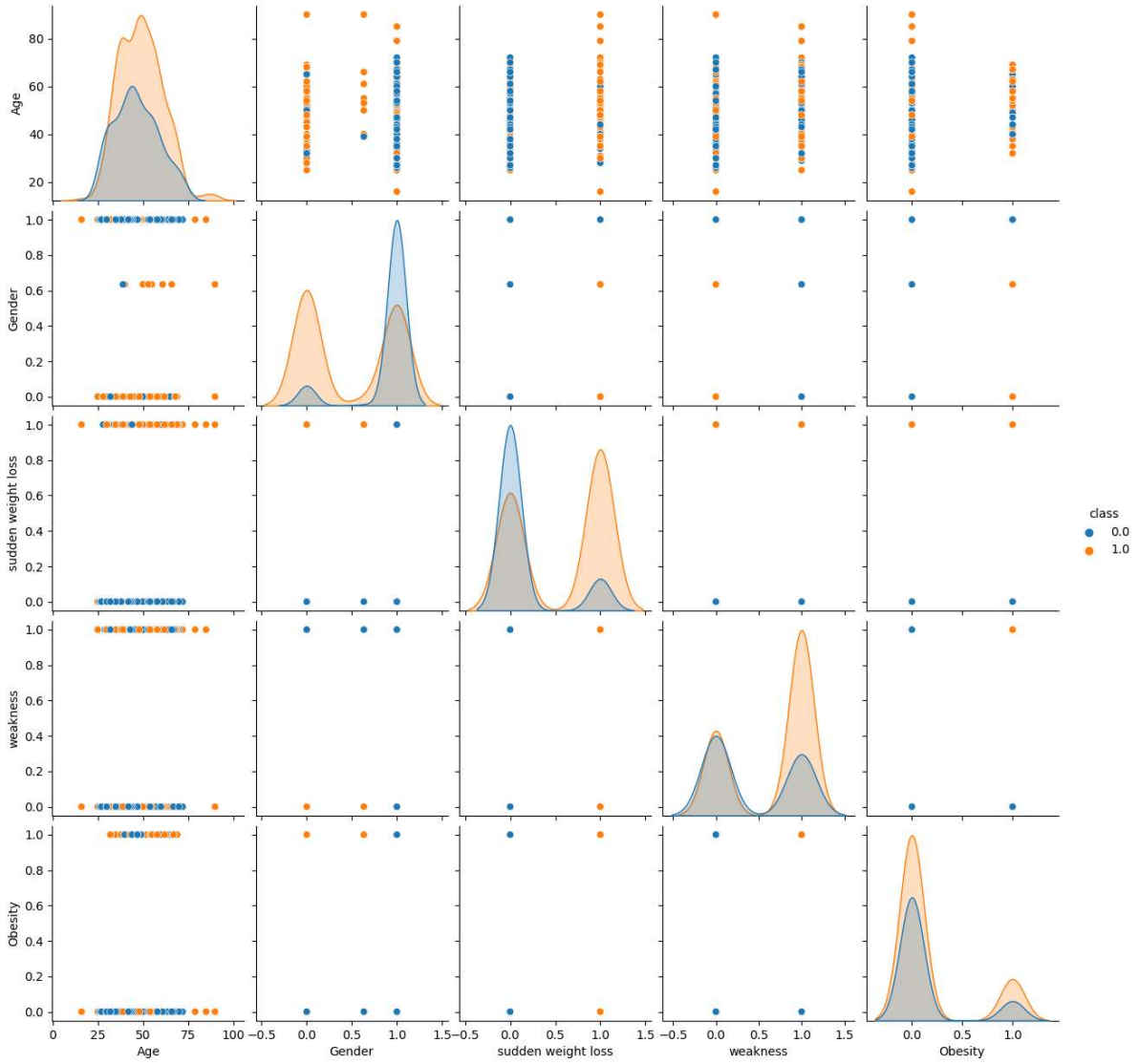
In [14]:
```
upper_array = np.where(dataset['Age']>=upper)[0]
lower_array = np.where(dataset['Age']<=lower)[0]
```

In [15]:
```
dataset.drop(index=upper_array, inplace=True)
dataset.drop(index=lower_array, inplace=True)
```

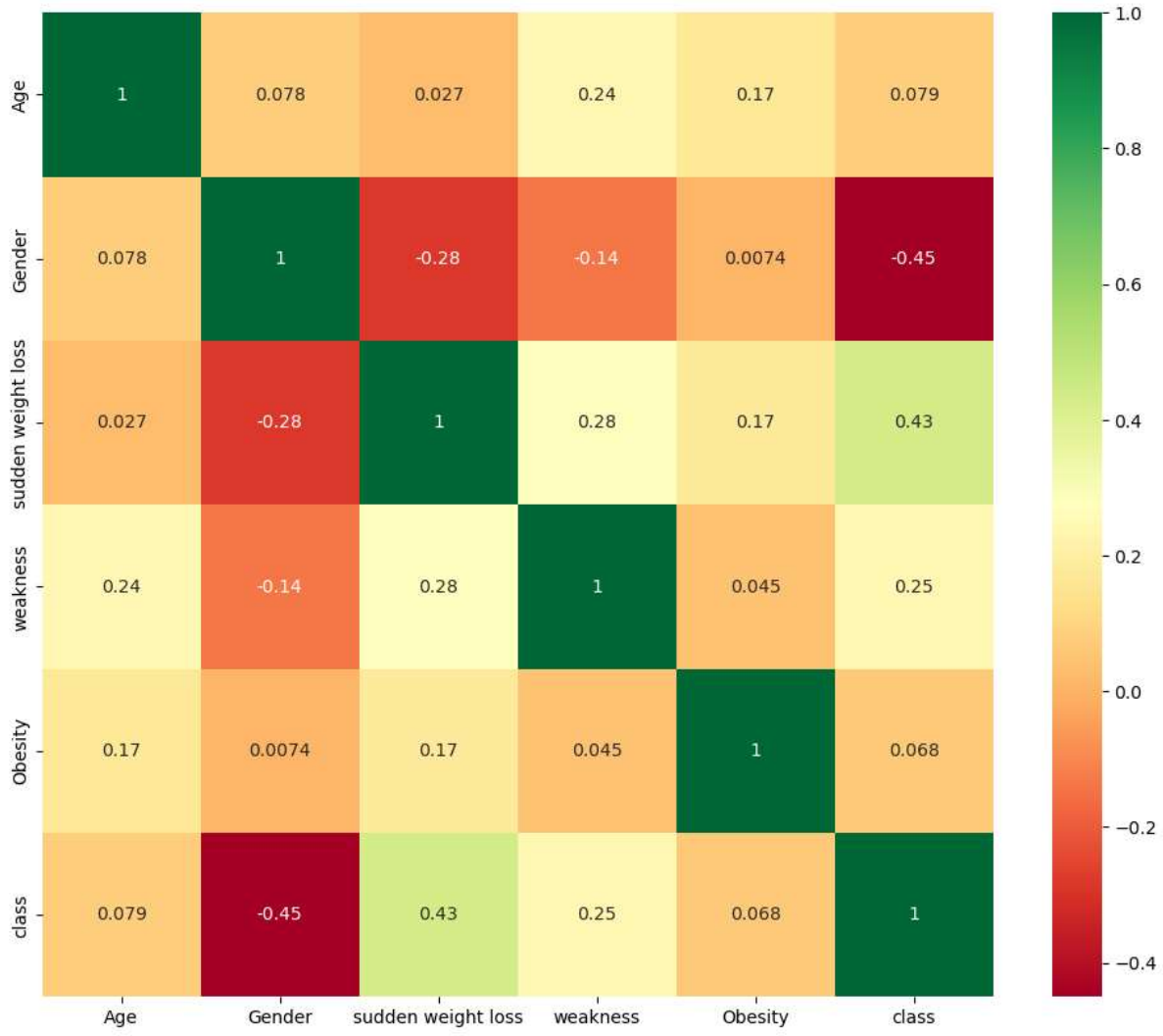
In [16]:
```
dataset.shape[0]
```

Out[16]:
```
516
```

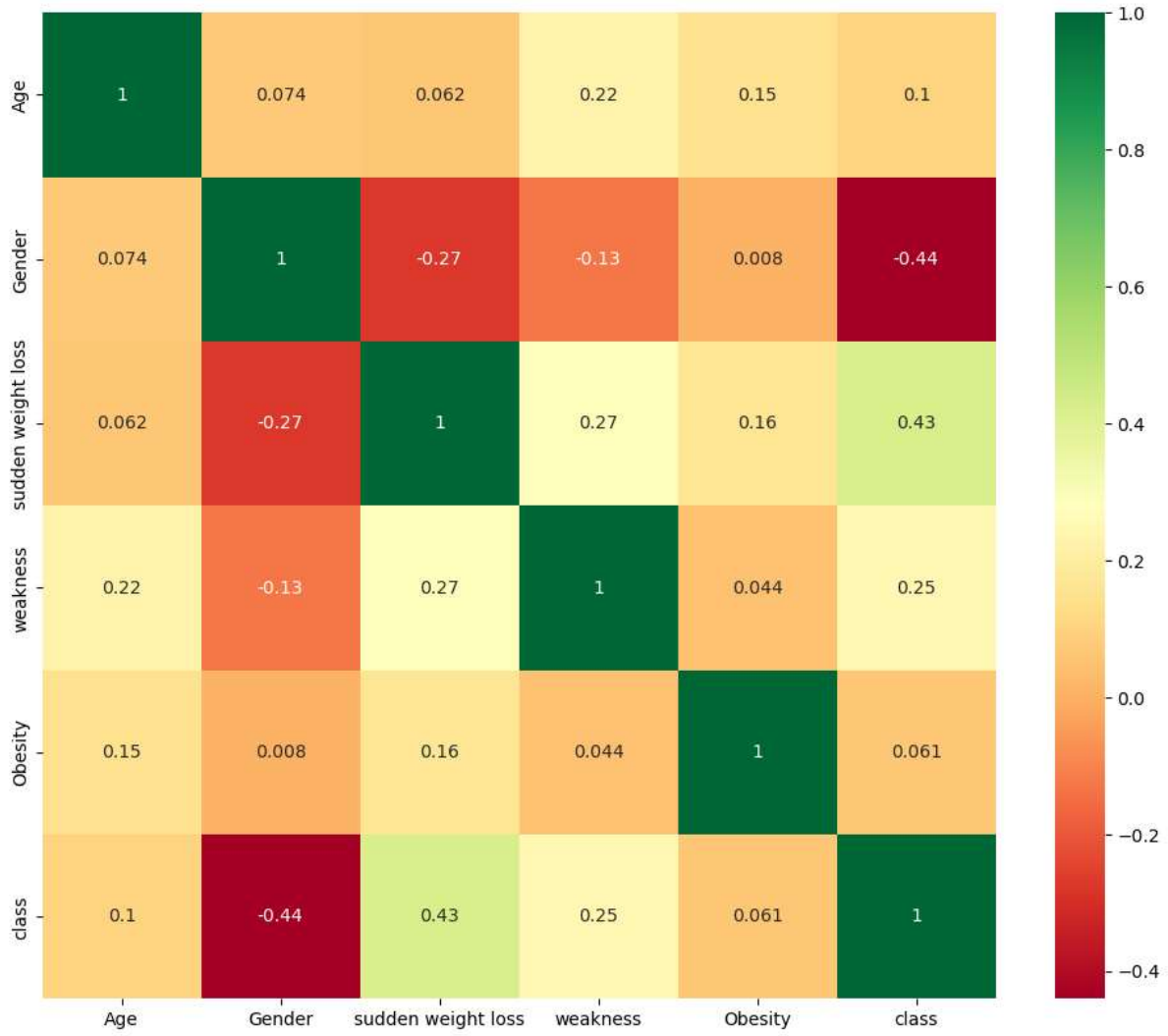
In [17]:
```
p=sns.pairplot(datasetmod, hue = 'class')
```



In [18]:
```
plt.figure(figsize=(12,10))
p=sns.heatmap(dataset.corr(), annot=True,cmap ='RdYlGn')
```



In [19]:
```
plt.figure(figsize=(12,10))
p=sns.heatmap(datasetmod.corr(), annot=True,cmap ='RdYlGn')
```



In [20]:
```
XSC = StandardScaler()
X = pd.DataFrame(XSC.fit_transform(datasetmod.drop(["class"], axis = 1), ),
                 columns = ['Age', 'Gender', 'sudden weight loss', 'weakness', 'O
```

In [21]:
```
X.head()
```

Out[21]:

	Age	Gender	sudden weight loss	weakness	Obesity
0	-0.658964	0.766372	-0.839594	0.832953	2.246444
1	0.834644	0.766372	-0.839594	0.832953	-0.445148
2	-0.575986	0.766372	-0.839594	0.832953	-0.445148
3	-0.244073	0.766372	1.191052	0.832953	-0.445148
4	1.000601	0.766372	1.191052	0.832953	2.246444

In [22]:
```
y = datasetmod['class']
```

In [23]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,random_stat
```

In [24]:
```
test_scores = []
train_scores = []

for i in range(1,29):

    knn = KNeighborsClassifier(i)
    knn.fit(X_train,y_train)

    train_scores.append(knn.score(X_train,y_train))
    test_scores.append(knn.score(X_test,y_test))
```

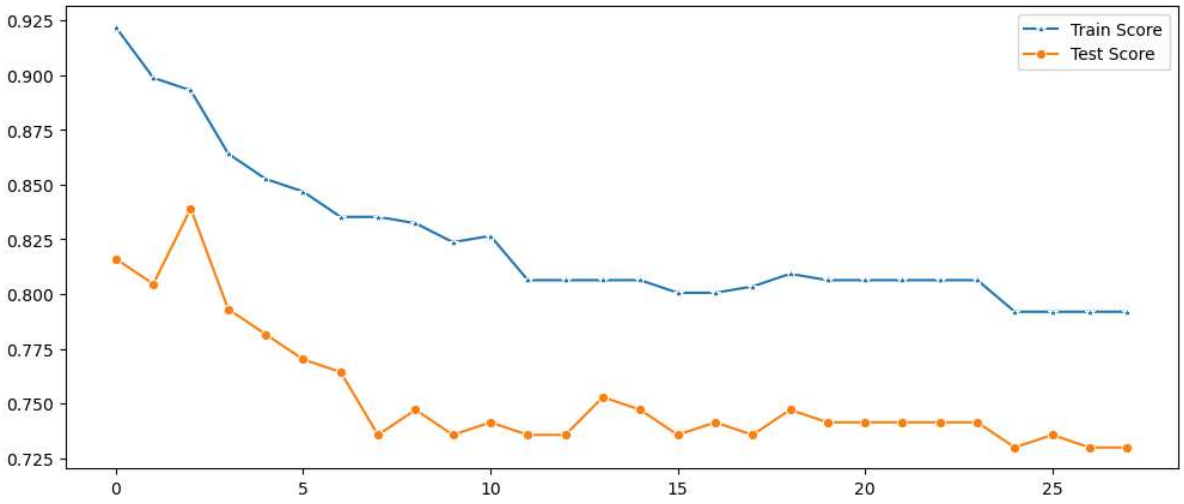
In [25]:
```
max_train_score = max(train_scores)
train_scores_ind = [i for i, v in enumerate(train_scores) if v == max_train_sc
print('Max train score {} % and k = {}'.format(max_train_score*100,list(map(la
```

```
Max train score 92.1965317919075 % and k = [1]
```

In [26]:
```
max_test_score = max(test_scores)
test_scores_ind = [i for i, v in enumerate(test_scores) if v == max_test_score
print('Max test score {} % and k = {}'.format(max_test_score*100,list(map(lambd
```

```
Max test score 83.9080459770115 % and k = [3]
```

In [27]:
```
plt.figure(figsize=(12, 5))

p = sns.lineplot(train_scores, marker='*', label='Train Score')
p = sns.lineplot(test_scores,marker='o',label='Test Score')
```

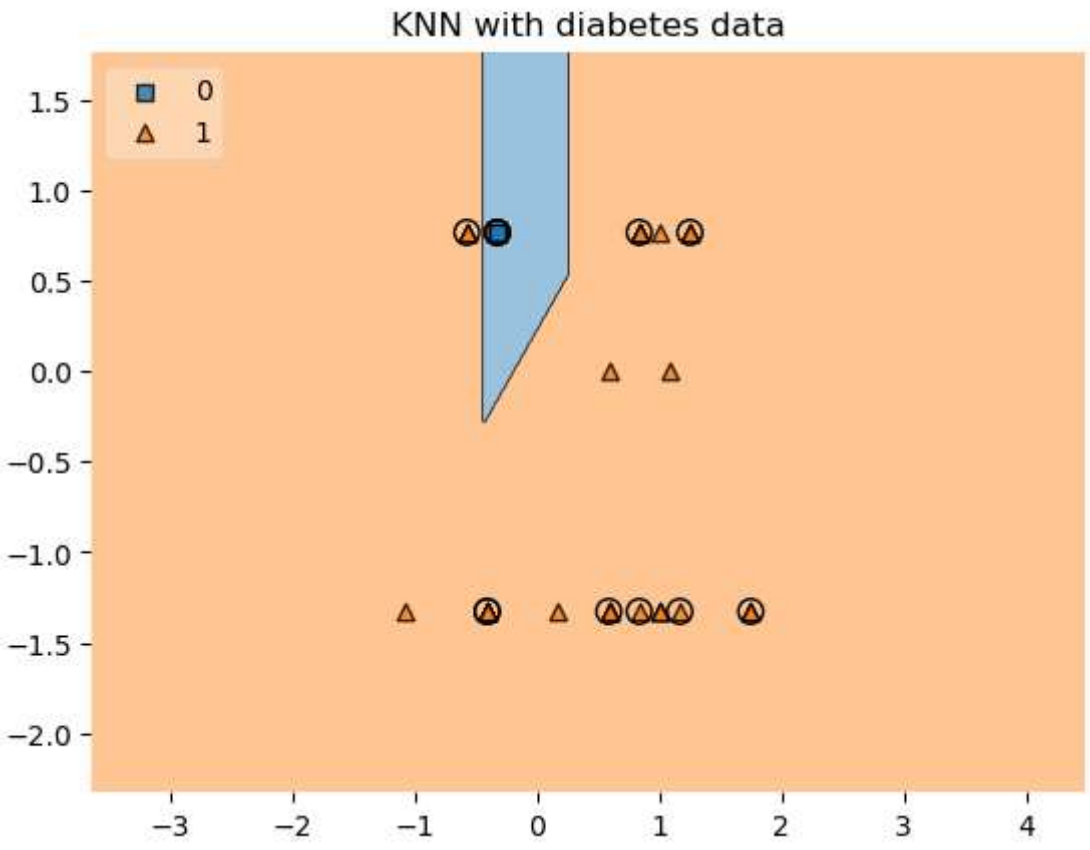


In [28]:
```
knn = KNeighborsClassifier(1)

knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[28]: 0.8160919540229885

In [38]:
```
y = y.astype(np.int_)
value = 20000
width  =20000

plot_decision_regions(X.values, y.values, clf = knn, legend =2,filler_feature_
                      filler_feature_ranges={2: width, 3: width, 4: width},
                      X_highlight=X_test.values)
plt.title("KNN with diabetes data")
plt.show()
```

```
C:\Users\aliii\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning:
X does not have valid feature names, but KNeighborsClassifier was fitted with
feature names
  warnings.warn(
```



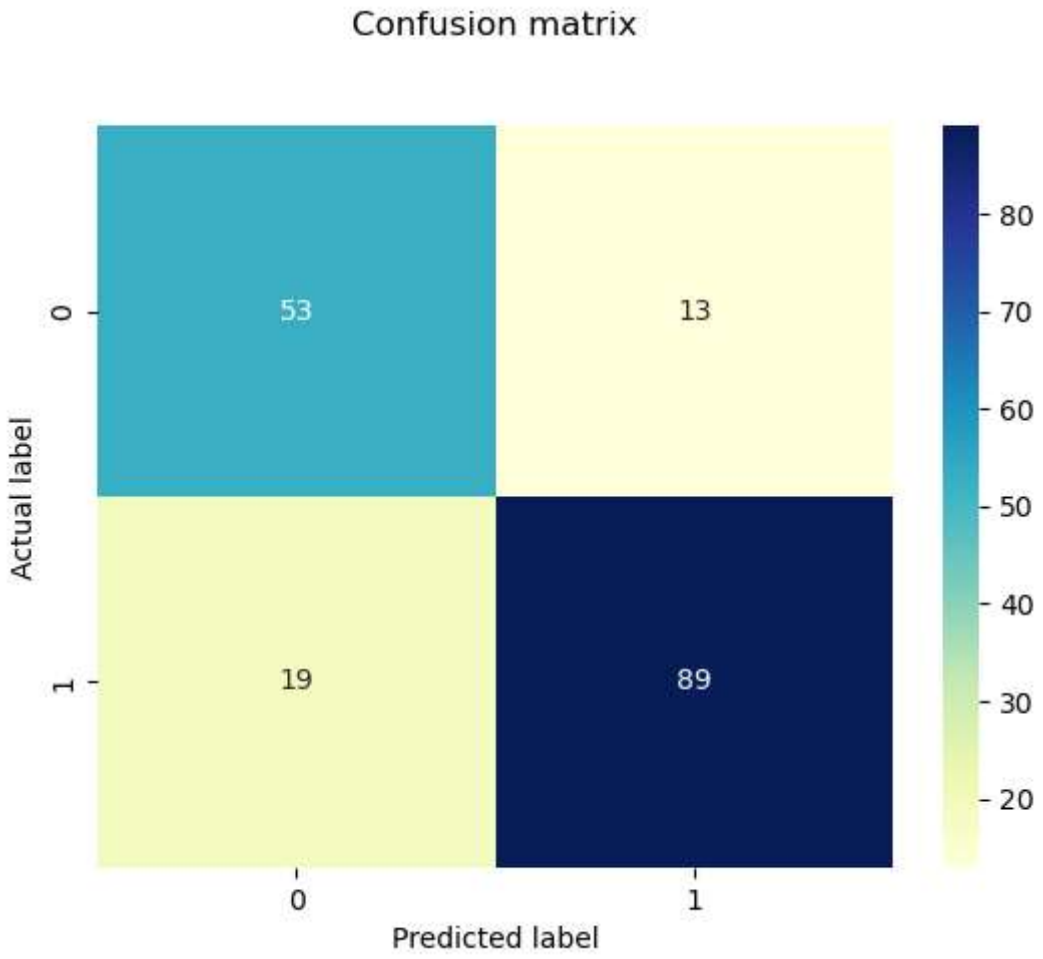
In [39]:
```
y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins
```

Out[39]:

True \ Predicted	0.0	1.0	All
0.0	53	13	66
1.0	19	89	108
All	72	102	174

In [41]:
```
y_pred = knn.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[41]: Text(0.5, 23.52222222222222, 'Predicted label')

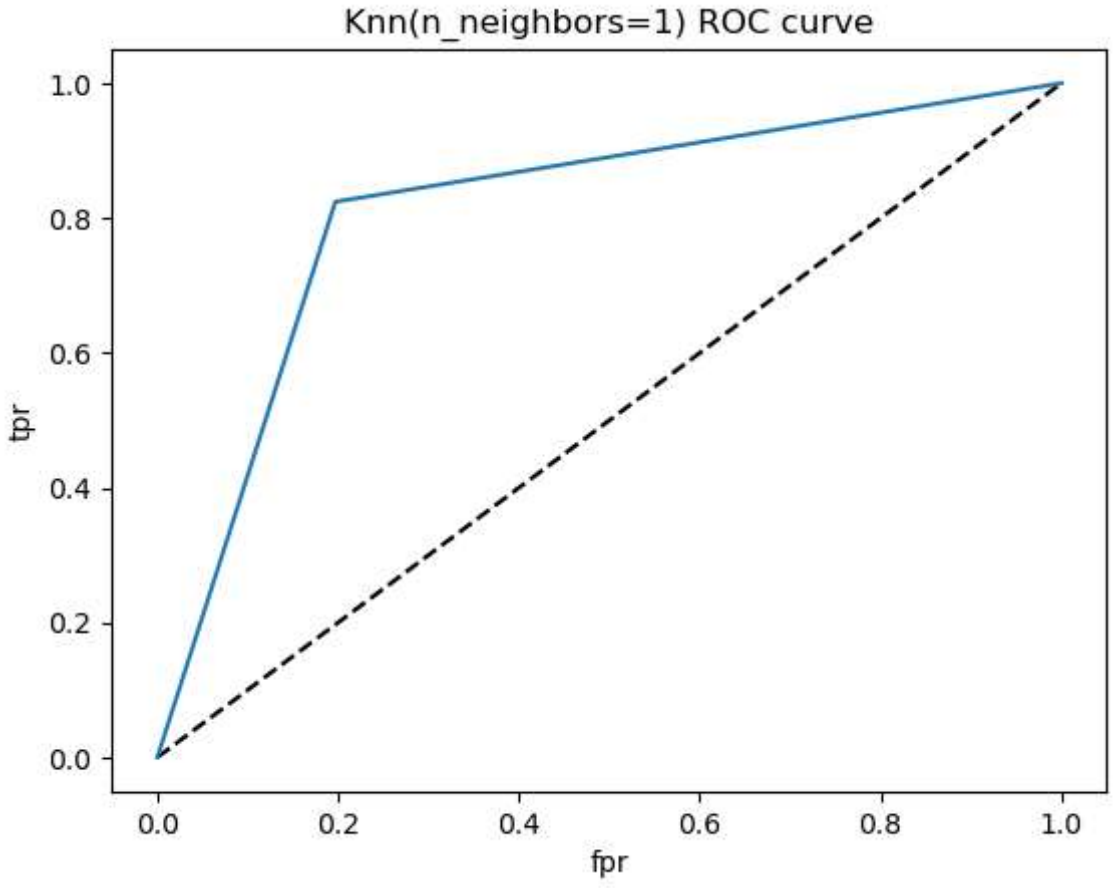


In [43]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.74      0.80      0.77        66
         1.0       0.87      0.82      0.85       108

    accuracy                           0.82       174
   macro avg       0.80      0.81      0.81       174
weighted avg       0.82      0.82      0.82       174
```

In [45]:
```
y_pred_proba = knn.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
```

In [47]:
```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr, label='Knn')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('Knn(n_neighbors=1) ROC curve')
plt.show()
```



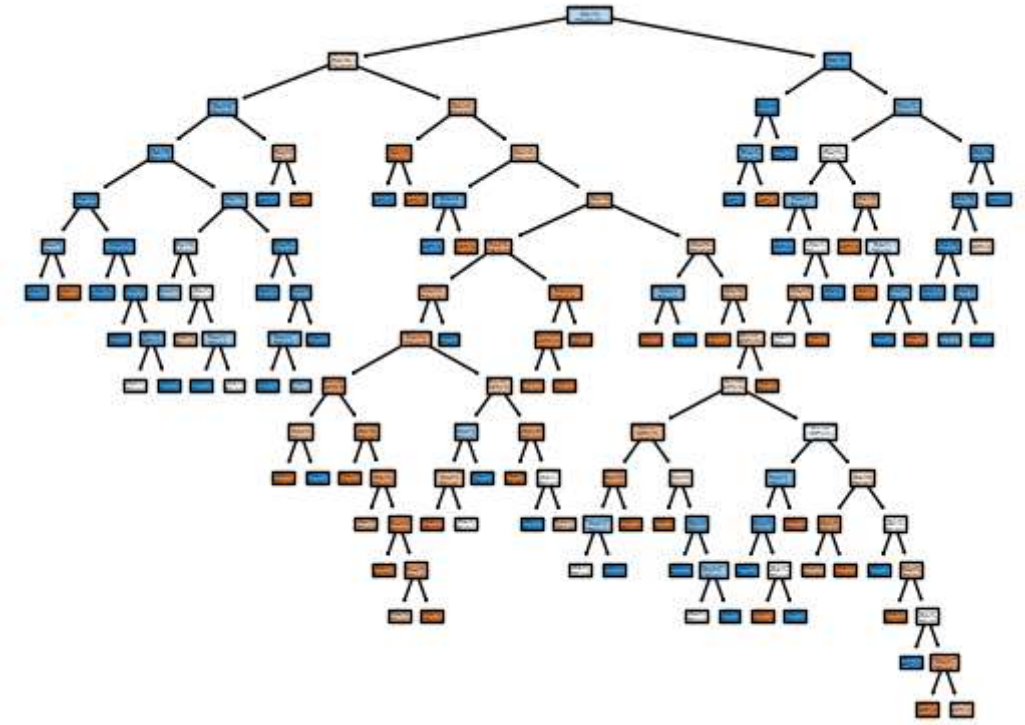
In [50]:
```
decision_tree = tree.DecisionTreeClassifier()
```

In [51]:
```
decision_tree.fit(X_train,y_train)
```

Out[51]:
```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [53]:
```
plt.figure()
tree.plot_tree(decision_tree, feature_names=X_train.columns, class_names=['Yes
plt.show()
```



In [ ]: