M5-L1 Problem 2 (6 Points)

Now we will provide a 2D classification dataset and you will learn to use sklearn's decision tree classifier on the data.

First, run the following cell to load the data and import decision tree tools.

- Input: X, size 80×2
- Output: y, size 80

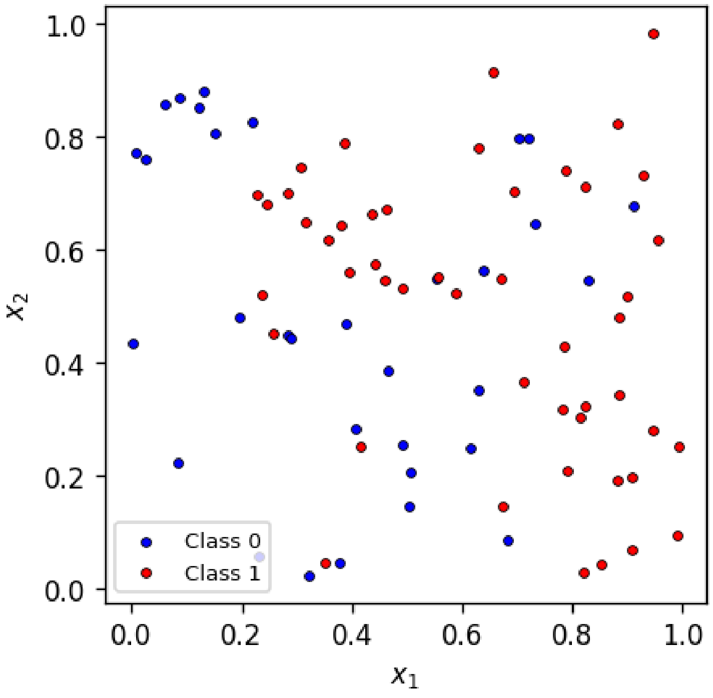
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
from matplotlib.colors import ListedColormap

y = np.array([1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1,
1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0])
x1 = np.array([6.73834679e-01, 3.57095269e-01, 4.42510505e-01,
8.48412660e-02, 2.17890220e-01, 4.60241400e-01, 7.87609761e-01,
7.20097577e-01, 8.81896387e-01, 3.05941324e-01, 3.88219250e-01,
7.10044376e-01, 9.27250328e-01, 2.43837089e-01, 5.95789013e-02,
4.91198192e-01, 1.51655961e-01, 6.13809025e-01, 3.95723003e-01,
5.55833098e-01, 4.62360874e-01, 8.83678959e-01, 4.16099641e-01,
9.46254162e-01, 5.51854839e-01, 4.63910645e-01, 4.07507369e-01,
8.52476098e-04, 5.87336538e-01, 6.81185355e-01, 6.29008279e-01,
1.96662091e-01, 3.76311610e-01, 3.16277339e-01, 2.56410886e-01,
1.30402898e-01, 9.91131913e-01, 7.80540215e-01, 4.35788740e-01,
3.22648602e-01, 7.01992141e-01, 1.22742024e-01, 9.07070546e-01,
8.70998784e-02, 8.14737827e-01, 2.56563996e-02, 6.38786620e-01,
9.09495514e-01, 2.83605500e-01, 9.92281843e-01, 8.84983935e-01,
2.82535401e-01, 3.51902502e-01, 3.85510606e-01, 9.08504747e-01,
9.45943000e-01, 8.18720088e-01, 8.22720940e-01, 8.51050202e-01,
5.06850808e-01, 7.31154379e-01, 7.84164014e-01, 6.30222156e-01,
9.53644588e-01, 4.90604436e-01, 2.36871523e-01, 6.70092986e-01,
3.81385827e-01, 8.97776618e-01, 8.81222406e-01, 8.24001410e-01,
6.93123693e-01, 7.90115238e-01, 6.56975559e-01, 2.30069955e-01,
2.90401258e-01, 7.92101141e-03, 2.28748706e-01, 8.28434414e-01,
5.03178362e-01])
x2 = np.array([0.14784469, 0.61647661, 0.57595235, 0.2232836 ,
0.82559199, 0.54569237, 0.73986085, 0.79782627, 0.82160469,
0.74537515, 0.46966765, 0.36512663, 0.73218711, 0.67966439,
0.85628818, 0.5325947 , 0.80458211, 0.24922691, 0.560076  ,
0.55214334, 0.67065618, 0.47970432, 0.25138818, 0.9830899 ,
0.5498764 , 0.38548435, 0.28514957, 0.43461184, 0.52278175,
0.08819936, 0.77946808, 0.48184639, 0.04768255, 0.64917397,
0.4532573 , 0.8799674 , 0.09534969, 0.31860112, 0.66189135,
0.02451146, 0.79680498, 0.85089439, 0.19792231, 0.86776139,
0.3038833 , 0.75953865, 0.5644305 , 0.67669664, 0.44999576,
0.25310745, 0.34467416, 0.70163484, 0.04647378, 0.7900774 ,
0.06895479, 0.27997123, 0.0308624 , 0.71039115, 0.04362167,
0.20736501, 0.64479502, 0.42872118, 0.35341853, 0.61623213,
0.25638276, 0.5216159 , 0.54970855, 0.64398701, 0.51780879,
0.19366846, 0.32399839, 0.70226861, 0.21057736, 0.91378165,
0.05743309, 0.44419594, 0.77169446, 0.69745565, 0.54526859,
0.14609322])
X = np.vstack([x1, x2]).T

def plot_data(X,y):
    colors=["blue","red"]
    for i in range(2):

plt.scatter(X[y==i,0],X[y==i,1],s=12,c=colors[i],edgecolors="black",li
newidths=.5,label=f"Class {i}")
        plt.xlabel("$x_1$")
        plt.ylabel("$x_2$")
        plt.legend(loc="lower left",prop={'size':8})

plt.figure(figsize=(4,4),dpi=120)
plot_data(X,y)
plt.show()
```



Create and fit a decision tree classifier

Create an instance of a `DecisionTreeClassifier()` with `max_depth` of 5. Fit this to the data `X`, `y`.

For more details, consult: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

```
# YOUR CODE GOES HERE
dt = DecisionTreeClassifier(max_depth=5)
dt.fit(X,y)
```

```
DecisionTreeClassifier(max_depth=5)
```

Making new predictions using your model

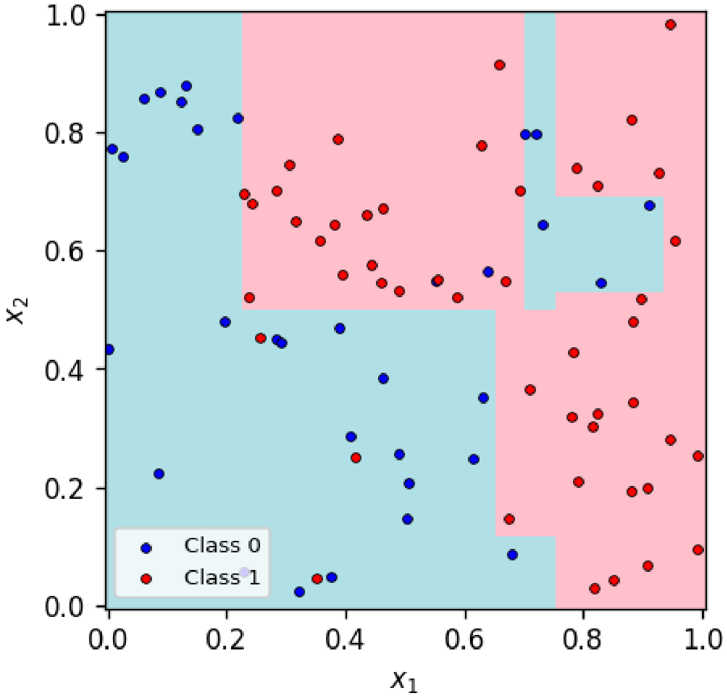
Now use the decision tree you trained to evaluate on the meshgrid of points `X_test` as indicated below. The code here will generate a plot showing the decision boundaries created by the model.

```
vals = np.linspace(0,1,100)
x1grid, x2grid = np.meshgrid(vals, vals)

X_test = np.vstack([x1grid.flatten(), x2grid.flatten()]).T


# YOUR CODE GOES HERE
# compute a prediction, `pred` for the input `X_test`
pred = dt.predict(X_test)

plt.figure(figsize=(4,4),dpi=120)
bgcolors = ListedColormap(["powderblue","pink"])
plt.pcolormesh(x1grid, x2grid, pred.reshape(x1grid.shape),
shading="nearest",cmap=bgcolors)
plot_data(X,y)
plt.show()
```



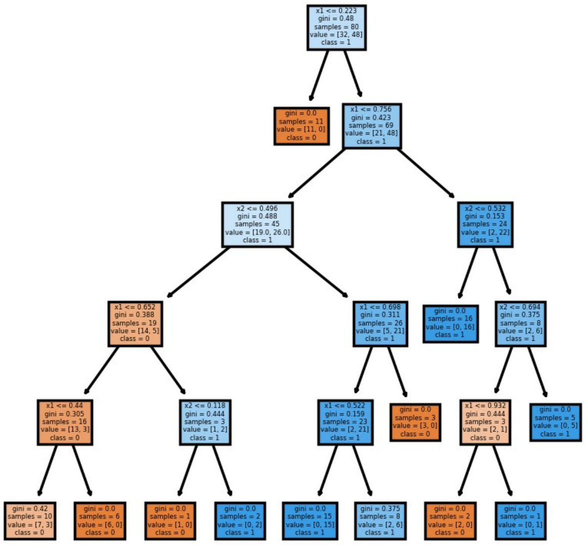
Visualizing the decision tree

The `plot_tree()` function (https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html) can generate a simple visualization of your decision tree model. Try out this function below:

```
plt.figure(figsize=(4,4),dpi=250)

# YOUR CODE GOES HERE
# plot_tree(dt)
plot_tree(dt, feature_names= ["x1","x2"], class_names= ["0","1"],
impurity= True, filled= True)

plt.show()
```


x1 <= 0.223
gini = 0.48
samples = 80
value = [32, 48]
class = 1
gini = 0.0
samples = 11
value = [11, 0]
class = 0
x1 <= 0.756
gini = 0.423
samples = 69
value = [21, 48]
class = 1
x2 <= 0.496
gini = 0.488
samples = 45
value = [19.0, 26.0]
class = 1
x2 <= 0.532
gini = 0.153
samples = 24
value = [2, 22]
class = 1
x1 <= 0.652
gini = 0.388
samples = 19
value = [14, 5]
class = 0
x1 <= 0.698
gini = 0.311
samples = 26
value = [5, 21]
class = 1
gini = 0.0
samples = 16
value = [0, 16]
class = 1
x2 <= 0.694
gini = 0.375
samples = 8
value = [2, 6]
class = 1
x1 <= 0.44
gini = 0.305
samples = 16
value = [13, 3]
class = 0
x2 <= 0.118
gini = 0.444
samples = 3
value = [1, 2]
class = 1
x1 <= 0.522
gini = 0.159
samples = 23
value = [2, 21]
class = 1
gini = 0.0
samples = 3
value = [3, 0]
class = 0
x1 <= 0.932
gini = 0.444
samples = 3
value = [2, 1]
class = 0
gini = 0.0
samples = 5
value = [0, 5]
class = 1
gini = 0.42
samples = 10
value = [7, 3]
class = 0
gini = 0.0
samples = 6
value = [6, 0]
class = 0
gini = 0.0
samples = 1
value = [1, 0]
class = 0
gini = 0.0
samples = 2
value = [0, 2]
class = 1
gini = 0.0
samples = 15
value = [0, 15]
class = 1
gini = 0.375
samples = 8
value = [2, 6]
class = 1
gini = 0.0
samples = 2
value = [2, 0]
class = 0
gini = 0.0
samples = 1
value = [0, 1]
class = 1