M5-L2 Problem 2 (6 Points)

Stress-strain measurements have been collected for many samples across many parts, resulting in much noisier data than would come from a tensile test, for example. Your job is to train an ensemble of decision trees that can predict stress for an input strain.

Scikit-Learn's `RandomForestRegressor()` has several parameters that you will experiment with below.

Run each cell; then, experiment with different settings of the `RandomForestRegressor()` to answer the questions at the end.

```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
%matplotlib inline
from ipywidgets import interact, interactive, fixed, interact_manual,
Layout, FloatSlider, Dropdown

# Load the data
y = np.array([133.18473289, 366.12422297, 453.70990214, 479.37136253,
238.16361712,  39.91719443, 282.21638562, 292.65795577, 452.3018357 ,
513.74698695, 218.15682352, 246.89907722, 288.01585801, 496.79161385,
513.33226691, 424.08833145, 348.82218375, 416.3219439 , 377.13994489,
369.19256451, 473.34491909, 439.30614707, 294.35282781, 480.91717688,
296.48549884, 179.54014001, 207.18389616, 183.07319414, 120.82807145,
533.60761691, 580.56296671, 386.6089496 , 419.26095887, 281.62811215,
173.98663034, 532.76872944, 480.19236657, 399.04560233, 234.12695309,
67.66845783, 512.31910187, 115.28680775, 401.89425604, 383.0896221 ,
348.80843569,  80.44889501,  64.68281643, 526.95380423, 310.85373168,
307.50969584, 446.45803748, 165.35545741, 414.88737018, 364.63597852,
487.6081401 , 468.15816997, 349.14335436, 332.10442343, 490.53829223,
455.37759943, 296.34199873, 482.30630337])
x = np.array([0.47358185, 0.80005535, 1.10968143, 1.85282726,
0.58177792, 0.24407275, 0.67817621, 0.59768343, 1.39656401,
1.20373001, 0.64022514, 0.51568838, 0.65147781, 1.20059147,
1.83127605, 0.96453862, 0.96392458, 1.34246004, 0.94255129,
0.78008304, 1.86226445, 1.30136524, 0.67180015, 1.39195582,
0.71199128, 0.58129463, 0.56788261, 0.53974967, 0.4527218 ,
1.32972689, 1.69826628, 1.06217982, 0.83887108, 0.92104216,
0.40126339, 1.64047136, 0.98148719, 1.02722597, 0.50128165,
0.18748944, 1.70601479, 0.42319326, 0.85202771, 1.15619305,
0.8703823 , 0.41810514, 0.24339075, 1.43638861, 0.71262321,
0.76776402, 1.08206553, 0.30560831, 1.04197577, 1.26957562,
1.33471511, 1.06236103, 0.70525115, 0.73310256, 1.23735534,
1.27799174, 0.72219864, 1.45629556])
```

```
def plot(n_estimators, max_leaf_nodes, bootstrap):
    n_estimators = [1,10,20,30,40,50,60,70,80,90,100]
[int(n_estimators)]
    max_leaf_nodes = int(max_leaf_nodes)
    model = RandomForestRegressor(n_estimators=n_estimators,
                                  bootstrap=(True if "On" in bootstrap
else False),
                                  max_leaf_nodes=max_leaf_nodes,
                                  random_state=0)
    model.fit(x.reshape(-1,1), y)

    xs = np.linspace(min(x),max(x),500)
    ys = model.predict(xs.reshape(-1,1))

    plt.figure(figsize=(5,3),dpi=150)

plt.scatter(x,y,s=20,color="cornflowerblue",edgecolor="navy",label="Da
ta")
    plt.plot(xs, ys, c="red",linewidth=2,label="Mean prediction")
    for i,dt in enumerate(model.estimators_):
        label = "Tree predictions" if i == 0 else None
        plt.plot(xs, dt.predict(xs.reshape(-1,1)),
c="gray",linewidth=.5,zorder=-1, label = label)

    plt.legend(loc="lower right",prop={"size":8})
    plt.xlabel("Strain, %")
    plt.ylabel("Stress, MPa")
    plt.title(f"Num. estimators: {n_estimators}, Max leaves =
{max_leaf_nodes}, Bootstrapping: {bootstrap}",fontsize=8)
    plt.show()

slider1 = FloatSlider(
    value=2,
    min=0,
    max=10,
    step=1,
    description='# Estimators',
    disabled=False,
    continuous_update=True,
    orientation='horizontal',
    readout=False,
    layout = Layout(width='550px')
)

slider2 = FloatSlider(
    value=5,
    min=2,
    max=25,
    step=1,
    description='Max Leaves',
```

```
    disabled=False,
    continuous_update=True,
    orientation='horizontal',
    readout=False,
    layout = Layout(width='550px')
)

dropdown = Dropdown(
    options=["On (66% of data)", "Off"],
    value="On (66% of data)",
    description='Bootstrap',
    disabled=False,
)


interactive_plot = interactive(
    plot,
    bootstrap = dropdown,
    n_estimators = slider1,
    max_leaf_nodes = slider2
    )
output = interactive_plot.children[-1]
output.layout.height = '500px'

interactive_plot
```

{"model_id":"72be89851b7b4efb9bd4b925eca1cec1","version_major":2,"version_minor":0}

Questions

1. Keep bootstrapping on and set max leaf nodes constant at 3. Describe what happens to the mean prediction as the number of estimators increases.

   As the number of estimators increases, the mean prediction becomes more accurate and stable with less variance.

2. Keep bootstrapping on and set number of estimators constant at 100. Describe what happens to the mean prediction as the leaf node maximum increases.

   As the leaf node maximum increased, the mean prediction started to be more sensitive varying data and overfitting can be observed.

3. Now disable bootstrapping. Notice that all of the predictions are the same -- the gray lines are behind the red. Why is this? (Hint: Think about the number of features in this dataset.)

   As bootstrapping is disabled, the model will use the entire given dataset instead of a subset of the data, thus leading to no change in prediction outcomes.