M5-L2 Problem 2

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.

In [1]:
```
# 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, Float

# Load the data
y = np.array([133.18473289, 366.12422297, 453.70990214, 479.37136253, 238.16361712,
x = np.array([0.47358185, 0.80005535, 1.10968143, 1.85282726, 0.58177792, 0.2440727
```

In [2]:
```
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="Data")
    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,

    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}, Boo
    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
```

Out[2]:
```
interactive(children=(FloatSlider(value=2.0, description='# Estimators', layout=La
yout(width='550px'), max=10.…
```

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.

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.

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.)

1. As the number of esimators increases, the accuracy of the model slighty increases.
2. As the leaf node maximum increases, the accuracy of the model generally increases. But after surpassing a certain point the model is overfitting and is heavily
   affected by noise inthe data.
3. For this data set, there's only one feature. And when bootstrap is disabled, the tree doesn't have enough diversity to produce such different results. So the predictions overlap on each other and with the red line.