COM S 474/574 Spring 2020
Introduction to Machine Learning

Homework 4

1 Directions:

- **Due: Thursday April 2, 2020 at 10pm.** Late submissions will be accepted for 24 hours after that time, with a 15% penalty.
- Upload the homework to Canvas as a pdf file. Answers to problems 1-3 can be handwritten, but writing must be neat and the scan should be high-quality image. Other responses should be typed or computer generated.
- Any non-administrative questions must be asked in office hours or (if a brief response is sufficient) Piazza.

2 Problems

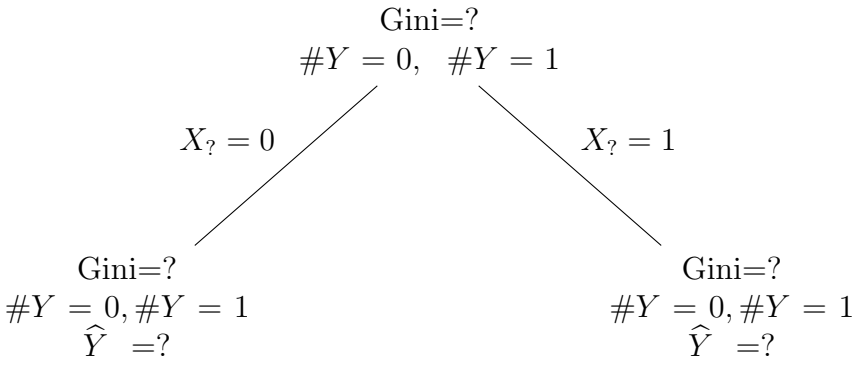
Problem 1. [10 points] Ch. 8 #4 "This question relates ..." Note, in the book's Figure 8.12, the split has <threshold corresponding to the left branch and ≥threshold corresponding to the right branch.

Problem 2. [5 points] Ch. 8 #5 "Suppose we produce ..."

Problem 3. [15 points] Suppose we construct a decision tree (shown below) for the data set in the following table, using the Gini index to select which features to split on. We want to predict $Y_{\text{Heart attack}}$.

Fill in the missing values, including the branches (what feature is split on, such as X_{male}), the Gini index at each node, the number of samples at each node with $Y = 0$ and $Y = 1$, and the predictions at the leaf nodes.

Show your calculations. Also report the training accuracy.



PATIENT ID	CHEST PAIN?	MALE?	SMOKES?	EXERCISES?	HEART ATTACK?
1.	yes	yes	no	yes	yes
2.	yes	yes	yes	no	yes
3.	no	no	yes	no	yes
4.	no	yes	no	yes	no
5.	yes	no	yes	yes	yes
6.	no	yes	yes	yes	no

Problem 4. [40 points] This problem uses the same data sets as homework 3. You should be able to re-use code from homework 3, with only a few modifications. We will compare boosting, bagging, and a single decision tree. If you are using Python, you can call

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

A. Make a scatter plot of the training (only training, not testing) data like in homework 3, with the decision tree classification function plotted using colors. Make plots for `max_depths` of 1, 2, 3, 4, and 10. In Python, you can call

```
clf=DecisionTreeClassifier(criterion='gini',splitter='best',max_depth=
    depth)
```

B. Make plots of testing and training accuracies as a function of the depth, with the depth ranging from 1 to 10. Report the best depth.

Remark: We are setting a parameter for the maximum depth, but the trees may stop growing before reaching that limit. Your plots are based on the maximum depth parameter.

C. Next we will do bagging, where we randomly sample from the original data, fit a tree to that data, repeat, and then combine the predictions of all the trees. In Python, you can call

```
clf=RandomForestClassifier(bootstrap=True,n_estimators=num_trees,max_
    features=None,criterion='gini',max_depth=depth)
```

Make a scatter plot of the training data (only training, not testing) like in homework 3, with the bagged decision tree classification function (fit using training data) plotted using colors. Make plots for `max_depths` of 1, 2, 3, 4, and 10. For each depth, use the best `num_trees` value for that particular depth, from the set

```
numtreerange=[1,5,10,25,50,100,200].
```

Remark: even though the function is called `RandomForestClassifier`, setting `max_features=None` results in allowing each tree to use any of the features (all two of them), which is just bagging.

D. Also make plots of testing and training accuracies as a function of the depth (using the best `num_trees` value for that depth), with the depth ranging from 1 to 10. Report the best depth and corresponding number of trees.

E. Next we will do boosting, where we adaptively build trees to learn from the previous trees' mistakes. In Python, you can call

```
clf=GradientBoostingClassifier(learning_rate=rate,n_estimators=num_
    trees,max_depth=depth)
```

Make a scatter plot of the training data (only training, not testing) like in HW 3, with the boosted decision tree classification function (fit using training data) plotted using colors. Make plots for `max_depths` of 1, 2, 3, 4, and 10. For each depth, use the best number of trees and the best learning rate for that particular depth, from the sets

```
numtreerange=[1,5,10,25,50,100,200].
learnraterange=np.logspace(-3,0,15,base=10)
```

F. Also make plots of testing and training accuracies as a function of the depth (use the best number of trees and the best learning rate for that particular depth), with the depth ranging from 1 to 10. Report the best depth and `num_trees` value.

G. In about 4-6 sentences, comment on how the decision boundaries differ between single trees, bagged trees, and boosted trees, how the depth affects (or not) the performance, and how the regions and performance compare to methods from homework 3.

Problem 5. [30 points] The previous problem allowed us to visualize the prediction function since we only had two features. Now let's explore what happens on a data set with more features. You should be able to use most of the same code as in the previous problem.

Download the files `digits-train.csv` and `digits-test.csv`. These come from a digit recognition data-set from `https://archive.ics.uci.edu/ml/datasets/Multiple+Features`. The first column in both files is the digit ('0', '1', ..., '9') that we want to predict from the other features.

Note, for this data set, we will use most of the samples for testing. (This will allow us to see a bigger contrast between the methods and have an accurate sense of how well the methods will do on new data.)

A. Make plots of testing and training accuracies of a single decision tree, as a function of the depth, with the depth ranging from 1 to 20. Report the best depth.

B. Make plots of testing and training accuracies of bagging as a function of the depth (using the best `num_trees` value for that depth), with the depth ranging from 1 to 20. Report the best depth and corresponding number of trees.
Like in the previous problem, use

```
clf=RandomForestClassifier(bootstrap=True,n_estimators=num_trees,max_
    features=None,criterion='gini',max_depth=depth)
```

For each depth, use the best number of trees for that particular depth, from the set

```
numtreerange=[1,5,10,25,50,100,200].
```

C. Next we will do random forests. Make plots of testing and training accuracies of random forests as a function of the depth (using the best `num_trees` value for that depth), with the depth ranging from 1 to 20. Report the best depth and corresponding number of trees. Use

```
clf=RandomForestClassifier(bootstrap=True,n_estimators=num_trees,max_
    features='sqrt',criterion='gini',max_depth=depth)
```

The argument `max_features='sqrt'` results in a random, small subset of features being considered for splits in each tree. For each depth, use the best number of trees for that particular depth, from the sets

```
numtreerange=[1,5,10,25,50,100,200].
```

D. Make plots of testing and training accuracies for boosting as a function of the depth (use the best number of trees and the best learning rate for that particular depth). Report the best depth and `num_trees` value. Use

```
clf=GradientBoostingClassifier(learning_rate=rate,n_estimators=num_
    trees,max_depth=depth)
```

For each depth, use the best number of trees and the best learning rate for that particular depth, from the sets

```
depthrange=range(1,6)
numtreerange=[50,100,150].
learnraterange=np.logspace(-2,0,10,base=10)
```

E. In about 3-5 sentences, comment on how the performance of the different methods and how they varied as the maximum depth changed. (Although you do not have to hand it in, if you re-plot the training accuracy curves of all methods together, and re-plot the testing accuracy curves of all methods together, this may help your analysis.) Comment also on if there were any noticeable differences in how long each took to complete.