I pledge my honor that I have abided by the Stevens Honors System. Krisitn Kim

Voting Classifiers

1. Explain the role of each line that generated the law_of_large_numbers_plot figure.

heads_proba = 0.51

- Setting the probability of getting head equal to 51% meaning there is a slightly high chance of getting head in this situation

coin_tosses = (np.random.rand(10000, 10) < heads_proba).astype(np.int32)

- np.random.rand(10000, 10) generates random numbers in range of 0 to 1 in 10000 by 10 format like below

```
[[0.00204744 0.20385025 0.93914564 ... 0.35193953 0.93378842 0.7194601 ]
 [0.66236636 0.76968986 0.56127996 ... 0.32546115 0.00134555 0.21595991]
 [0.50013564 0.20702939 0.17729528 ... 0.15576058 0.96011425 0.65956966]
 ...
 [0.84359807 0.40485925 0.04114056 ... 0.23818149 0.38758964 0.51361451]
 [0.59459316 0.83703046 0.60830217 ... 0.63935956 0.41593274 0.27789693]
 [0.18190028 0.00196975 0.73393513 ... 0.96336605 0.0949099  0.74339814]]
```

- < heads_proba changes those numbers to boolean values
- .astype(np.int32) change them to 0 or 1

cumulative_heads_ratio = np.cumsum(coin_tosses, axis=0) / np.arange(1, 10001).reshape(-1, 1)

- return the cumulative sum of the elements along a given axis
- axis =0 means that sum over rows for each of the 3 columns
- return evenly spaced values within a given interval so in this case, 1 to 10000
- Then divide cum sum by the above array to get a ratio
- Just to mention here, converts it into -1 in reshape function is used when you don't know or want to explicitly tell the dimension of that axis

plt.figure(figsize=(8,3.5))

- Setting the size of figure

plt.plot(cumulative_heads_ratio)

- Plotting the cumulative_heads_ratio

plt.plot([0, 10000], [0.51, 0.51], "k--", linewidth=2, label="51%")

- draws a dotted horizontal line at 0.51 with the label displayed

plt.plot([0, 10000], [0.5, 0.5], "k-", label="50%")

- draws a dotted horizontal line at 0.50 with the label displayed

plt.xlabel("Number of coin tosses")
plt.ylabel("Heads ratio")
plt.legend(loc="lower right")

- Placing label on the lower right

plt.axis([0, 10000, 0.42, 0.58])

- Setting both vertical and horizontal axis

save_fig("law_of_large_numbers_plot")

- Saving the figure

plt.show()

I pledge my honor that I have abided by the Stevens Honors System. Krisitn Kim

2. Reading the relevant section in the book, explains how ensemble classifiers can outperform any individual classifier.

According to the law of large numbers mentioned in the textbook, even if the method is correct only 51%, the correct probability climbs up. So, suppose there are a sufficient number of weak learners as long as they are sufficiently diverse and all classifiers are perfectly independent, ensemble classifiers can outperform individual classifiers.

3. What are the classification performances of each individual classifier and voting classifiers with hard / soft voting. Explain the difference between soft voting and hard voting.

For hard voting, the voting classifier slightly outperforms all the individual classifiers

```
LogisticRegression 0.864
RandomForestClassifier 0.896
SVC 0.896
VotingClassifier 0.912
```

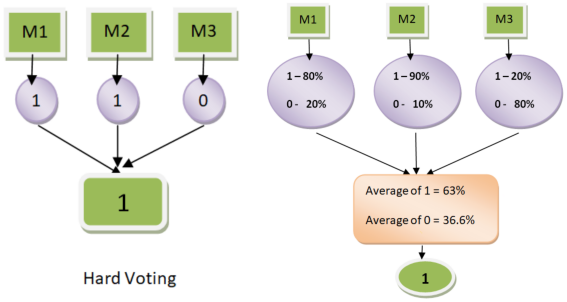
For soft voting, the voting classifier slightly outperforms all the individual classifiers, also it achieves a better accuracy than the hard voting did.

```
LogisticRegression 0.864
RandomForestClassifier 0.896
SVC 0.896
VotingClassifier 0.92
```

In hard voting, every individual classifier votes for a class, and the majority wins. Since this is a majority voting based on different weights, the predicted target label of the ensemble is the mode of the distribution of individually predicted labels. However, soft voting predicts the class with the highest class probability, averaged over all the individual classifiers, If all classifiers are able to estimate class probabilities. For example, because SVMs do not directly provide probability estimates, soft voting can only be used when you set the probability to True like

```
SVC(gamma="scale", probability=True, random_state=42)
```

These figures helped me understand better



I pledge my honor that I have abided by the Stevens Honors System. Krisitn Kim

Bagging and Pasting

4. Add a code block to run the pasting classifier (based on the bagging classifier code) and show each classifier's performance (i.e., decision tree, bagging DT, and pasting DT).

For PastingClassifier, I set bootstrap to False

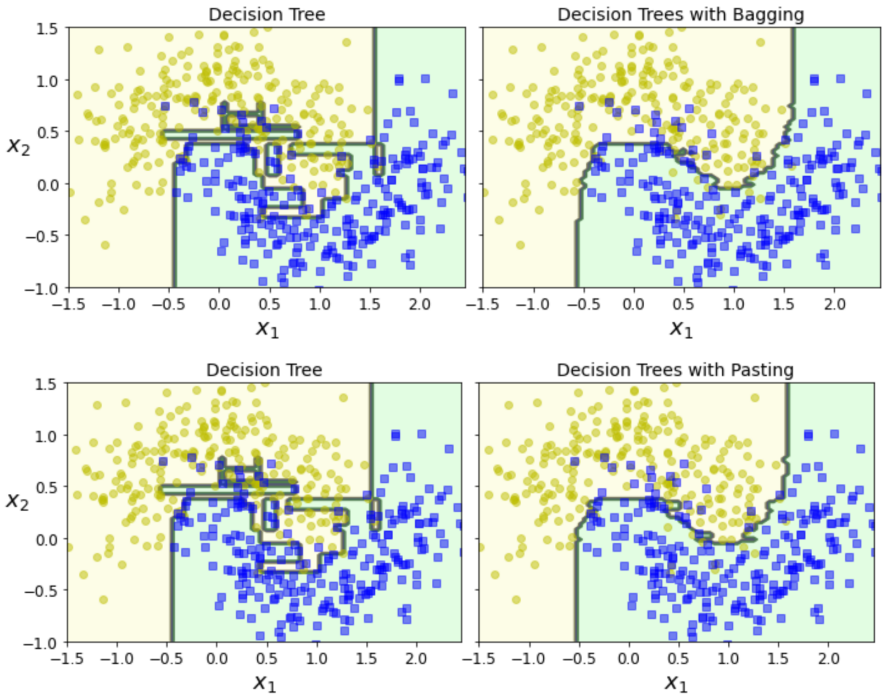
```
bag_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=500,
    max_samples=100, bootstrap=False, random_state=42)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

The results for both baggin DT and pasting DT were very similar, both of them being higher than the accuracy score of the decision tree.

The accuracy scores for y_test and y_pred

- Decision Classifier: 0.856
- Bagging Classifier: 0.904
- Pasting Classifier: 0.920

5. Include a plot of the bagging classifier in the plot (need to modify the last code block and increase ncols). You can download the image from the saved folder in files. Explain overall differences between all models.



Based on the plots, overall, the performance of decision trees with either bagging or pasting is better than the performance of decision trees. The variance of a decision tree is reduced when bagging was used.

Decision trees are a type of supervised learning, while Decision Trees with Bagging/Pasting is a type of ensemble learning. Therefore, in Decision Trees with Bagging/Pasting, the result is found by aggregating the predictions from multiple models. Also, the major difference between bagging and pasting is how they sample the data. In bagging, sampling is performed with replacement while it's performed without replacement in pasting.

Out-of-Bag evaluation

6. What is the Out-of-Bag evaluation score and testing accuracy of the bagging classifier. Is there a relation between these scores? Briefly explain.

Evaluation score: 0.899
Testing accuracy: 0.912

They are very close, and I can say that one should be able to estimate the accuracy on the test set based on the evaluation.

Random Forests

7. Add a code block for Extra Trees as discussed in the book. Does ExtraTreesClassifier produce the same classes as the RandomForestClassifier? What is the performance of each of the classifiers (i.e., Bagging decision tree, Random Forest, Extra Tree)?

ExtraTreesClassifier from sklearn.ensemble is used.

Yes, they have the same classes.

A Random Forest is equivalent to Bagging decision tree.
I expected that Random Forest and Extra Tree perform similarly, and maybe Extra Tree perform slightly better. The evaluation technique of RepeatedStratifiedKFold results in an accuracy score of 0.915 with std of 0.033 for RandomForest, while giving 0.904 and 0.040 for Extra Tree.

I pledge my honor that I have abided by the Stevens Honors System. Krisitn Kim

Feature Importance

8. How does the random forest classifier estimate the importance of features? Can this information be useful while building other supervised learning models? Give an example use with a particular learning model.

> To measure the relative importance of each feature, the classifier looks at how much the tree nodes that use that feature reduce impurity on average (weighted average).
>
> Feature importances is definitely useful in other supervised learning models too, especially when learners want to proceed with feature selection, they give valuable information on features and help them figure out what features are actually important.
>
> For example, in Logistic regression, a supervised learning algorithm, to proceed with feature selection, feature importance will be measured.

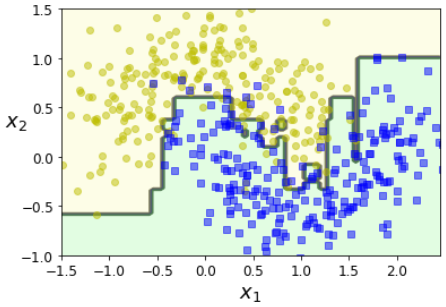
I pledge my honor that I have abided by the Stevens Honors System. Krisitn Kim

AdaBoost

9. What is the classification accuracy of the AdaBoost with a tree depth of 1, 2, 5, and 10? What is the effect of increasing the depth on the decision boundary?

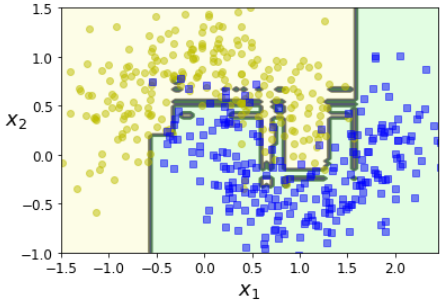
According to visualization of each tree depth,

A tree depth of 1:

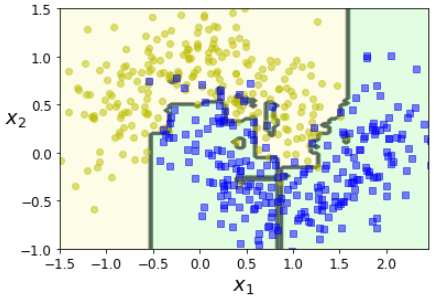


,

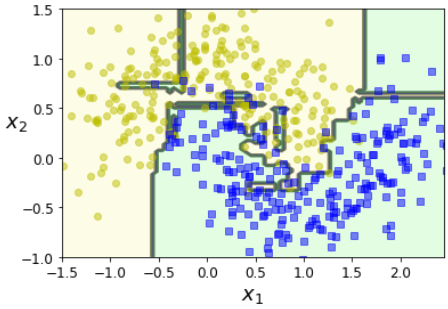
A tree depth of 2:



A tree depth of 5:



A tree depth of 10:



I pledge my honor that I have abided by the Stevens Honors System. Krisitn Kim

I interpreted that as tree depth increases, the boundary gets more accurate, therefore higher accuracy overall. Also, in the beginning, I assumed that the accuracy will increase because in the next layer, previously wrong instances weights get boosted.

However, when I used the method below to find the numeric accuracy,

```
from sklearn.metrics import accuracy_score
y_pred = ada_clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

I found accuracy scores for

- A tree depth of 1: 0.896,
- A tree depth of 2: 0.872,
- A tree depth of 5: 0.88,
- A tree depth of 10: 0.856

Hence, I confirmed that at a certain depth, there might be an overfitting on the training set, and the suggested way from the textbook to resolve it is to reduce the number of estimators or to regularize the base estimator more strongly.

I pledge my honor that I have abided by the Stevens Honors System. Krisitn Kim

Gradient Boosting

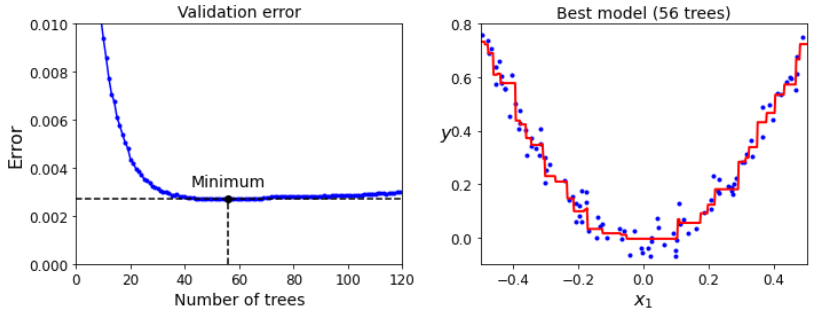
10. Modify the learning rate in Gradient Boosting with Early stopping. What is the tradeoff between the learning rate and the number of estimators? How can we determine the optimal number of trees? Include a snapshot of early_stopping_gbrt_plot.

I modified the learning rate by editing the “learning rate= “ parameter for the classifier.

Lower learning rate lets more estimators since each individual estimator is contributing less, however, it’s not allowed when the learning rate is high.

To find the optimal number of trees for the best result and avoid overfitting training data, there must be adjustments made to learning rate or number of estimators based on the performance. I would use early stopping, specifically by implementing a method called stged_predict() as mentioned in the textbook.

Original snapshot:



When modified the learning rate to 0.9: