NYCU Introduction to Machine Learning, Homework 3

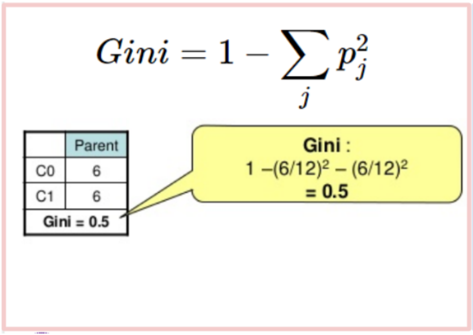
Deadline: Nov. 15, 23:59

Part. 1, Coding (80%):

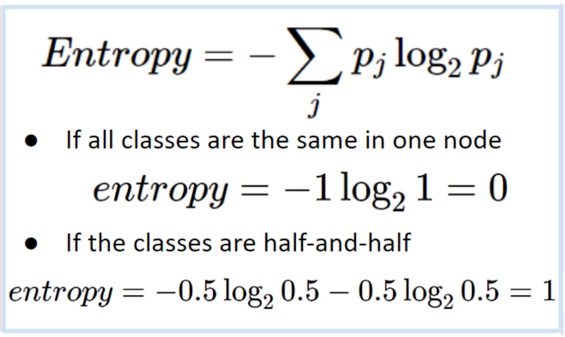
In this coding assignment, you need to implement the Decision Tree, AdaBoost and Random Forest algorithm by using only NumPy, then train your implemented model by the provided dataset and test the performance with testing data. Find the sample code and data on the GitHub page
https://github.com/NCTU-VRDL/CS_CS20024/tree/main/HW3

Please note that only <u>NumPy</u> can be used to implement your model, you will get no points by simply calling sklearn.tree.DecsionTreeClassifier.

1. (5%) Gini Index or Entropy is often used for measuring the "best" splitting of the data. Please compute the Entropy and Gini Index of this array `np.array([1,2,1,1,1,1,2,2,1,1,2])` by the formula below. (More details on page 5 of the hw3 slides, 1 and 2 represent class1 and class 2, respectively)



$$Gini = 1 - \sum_j p_j^2$$



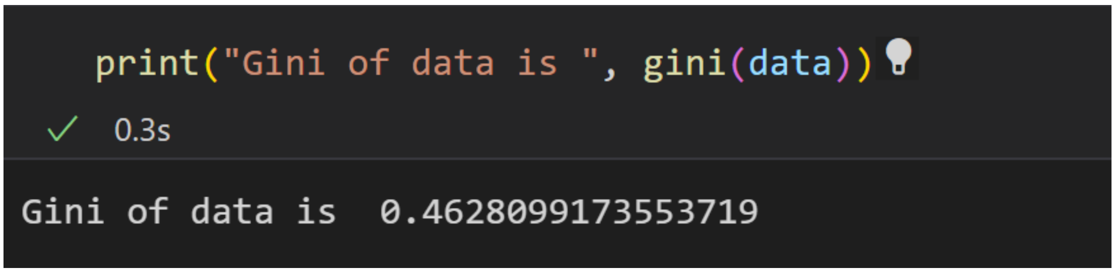
$$Entropy = -\sum_j p_j \log_2 p_j$$

- If all classes are the same in one node

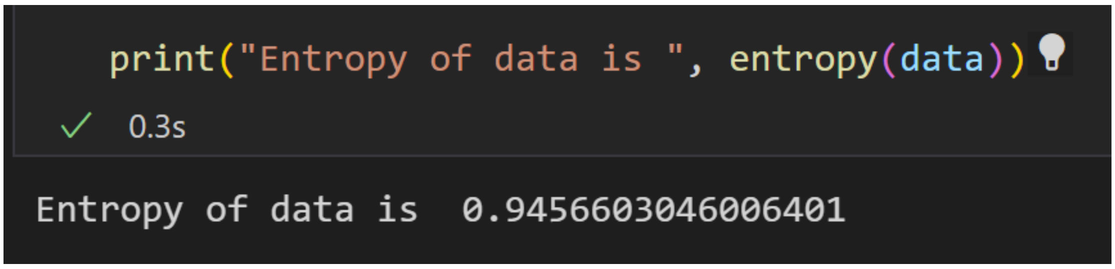
$$entropy = -1\log_2 1 = 0$$

- If the classes are half-and-half

$$entropy = -0.5\log_2 0.5 - 0.5\log_2 0.5 = 1$$

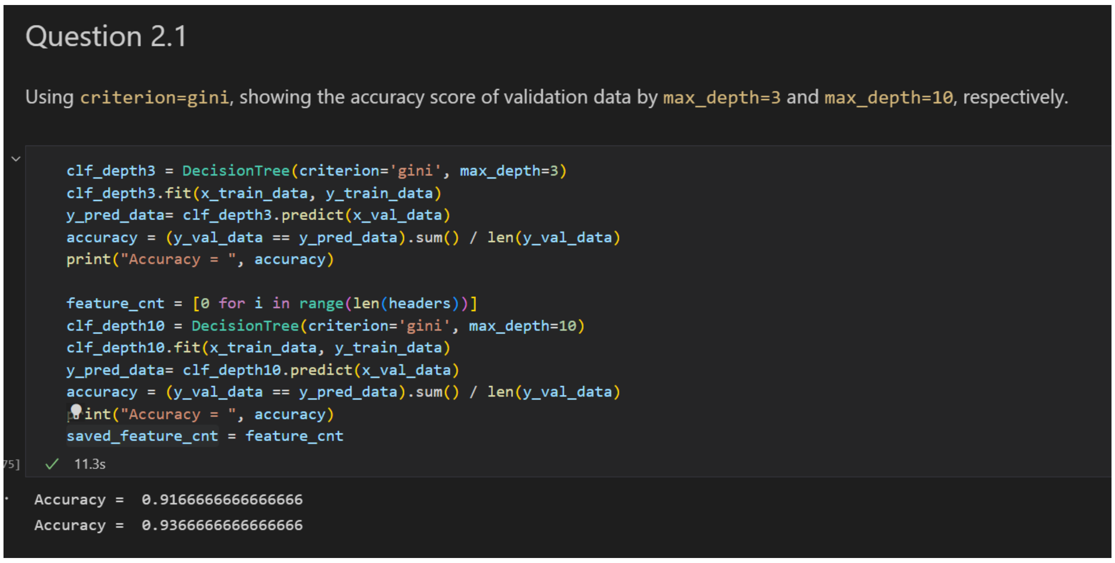


```
print("Gini of data is ", gini(data))
✓ 0.3s

Gini of data is  0.4628099173553719
```



```
print("Entropy of data is ", entropy(data))
✓ 0.3s

Entropy of data is  0.9456603046006401
```

2. (10%) Implement the Decision Tree algorithm (CART, Classification and Regression Trees) and train the model by the given arguments, and print the accuracy score on the test data. You should implement **two arguments** for the Decision Tree algorithm, 1) **Criterion**: The function to measure the quality of a split. Your model should support "gini" for the Gini impurity and "entropy" for the information gain.
2) **Max_depth**: The maximum depth of the tree. If Max_depth=None, then nodes are expanded until all leaves are pure. Max_depth=1 equals split data once
   1. Using Criterion='gini', showing the accuracy score of test data by Max_depth=3 and Max_depth=10, respectively.



Question 2.1

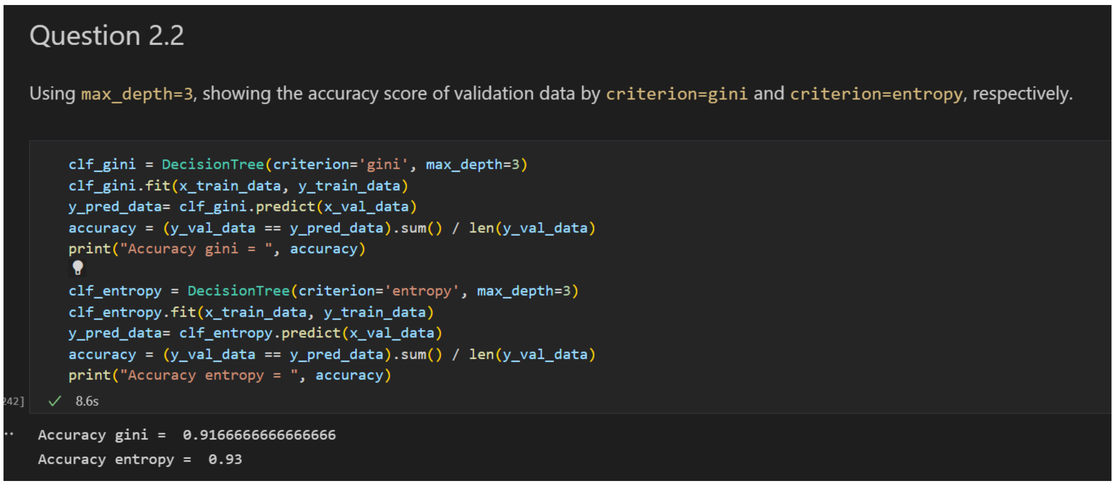
Using `criterion=gini`, showing the accuracy score of validation data by `max_depth=3` and `max_depth=10`, respectively.

```
clf_depth3 = DecisionTree(criterion='gini', max_depth=3)
clf_depth3.fit(x_train_data, y_train_data)
y_pred_data= clf_depth3.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy = ", accuracy)

feature_cnt = [0 for i in range(len(headers))]
clf_depth10 = DecisionTree(criterion='gini', max_depth=10)
clf_depth10.fit(x_train_data, y_train_data)
y_pred_data= clf_depth10.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy = ", accuracy)
saved_feature_cnt = feature_cnt
```

11.3s

```
Accuracy =  0.9166666666666666
Accuracy =  0.9366666666666666
```

   2. Using Max_depth=3, showing the accuracy score of test data by Criterion='gini' and Criterion='entropy', respectively.



Question 2.2

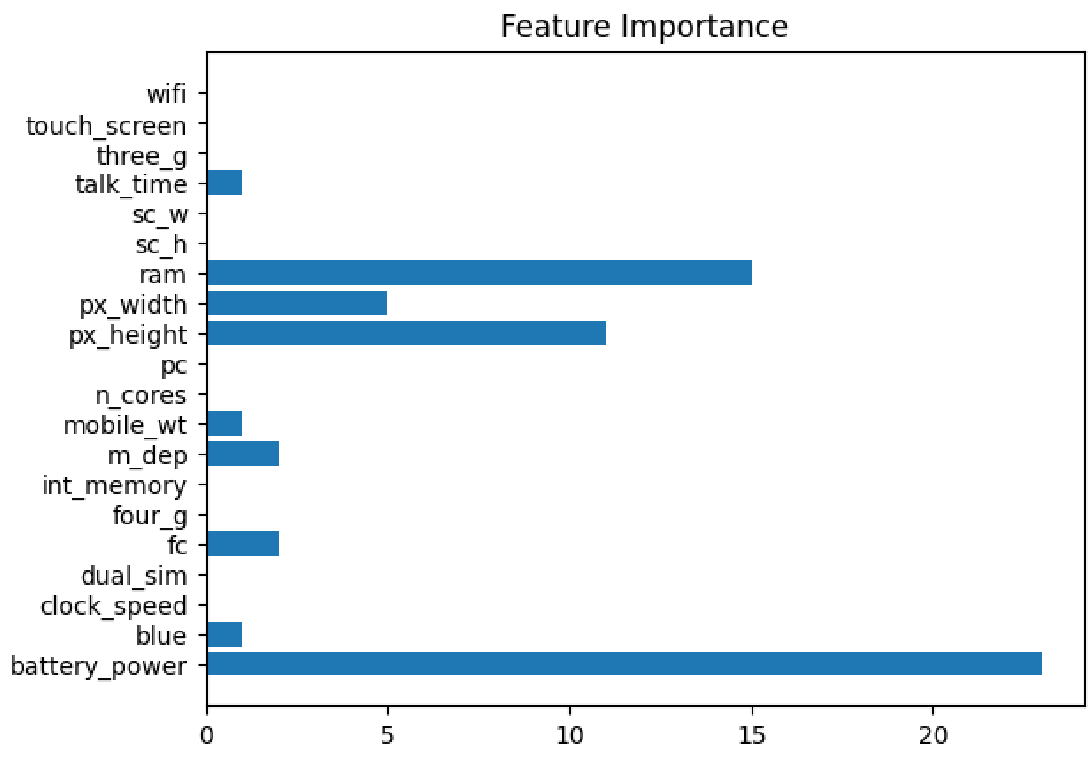
Using `max_depth=3`, showing the accuracy score of validation data by `criterion=gini` and `criterion=entropy`, respectively.

```
clf_gini = DecisionTree(criterion='gini', max_depth=3)
clf_gini.fit(x_train_data, y_train_data)
y_pred_data= clf_gini.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy gini = ", accuracy)

clf_entropy = DecisionTree(criterion='entropy', max_depth=3)
clf_entropy.fit(x_train_data, y_train_data)
y_pred_data= clf_entropy.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy entropy = ", accuracy)
```

8.6s

```
Accuracy gini =  0.9166666666666666
Accuracy entropy =  0.93
```

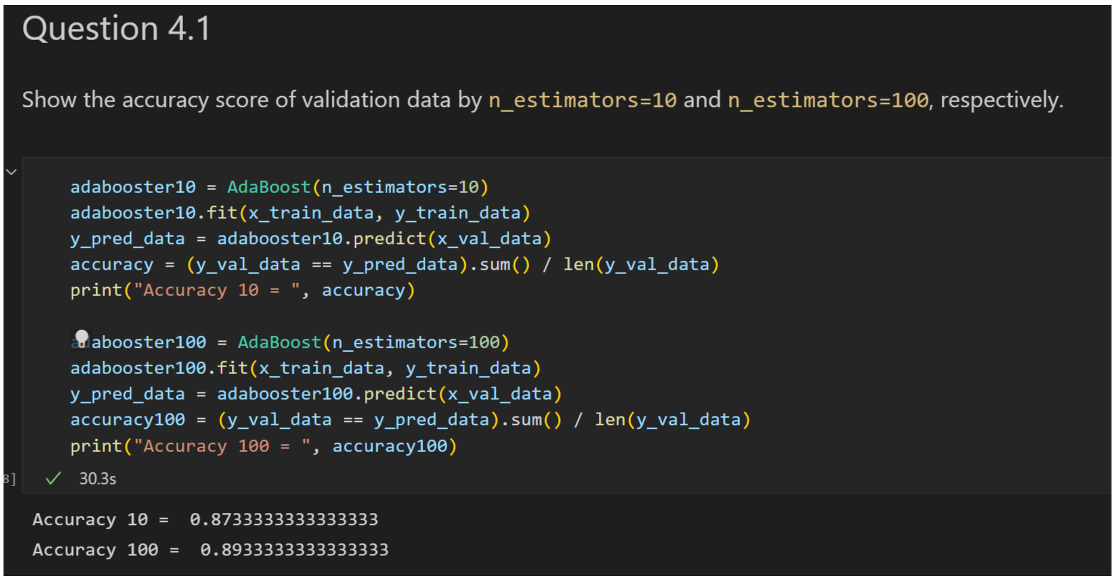
3. (5%) Plot the feature importance of your Decision Tree model. You can use the model from Question 2.1, max_depth=10. (You can use simply counting to get the feature importance instead of the formula in the reference, more details on the sample code. **Matplotlib** is allowed to be used)



4. (15%) Implement the AdaBoost algorithm by using the CART you just implemented from question 2. You should implement **one argument** for the AdaBoost.

   1) **N_estimators**: The number of trees in the forest.

      1. Showing the accuracy score of test data by n_estimators=10 and n_estimators=100, respectively.



Question 4.1

Show the accuracy score of validation data by `n_estimators=10` and `n_estimators=100`, respectively.

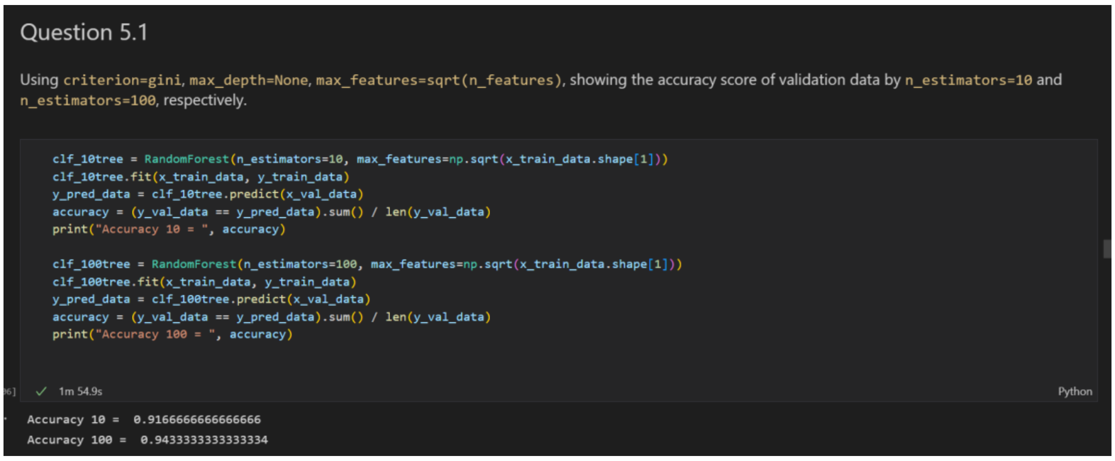
```python
adabooster10 = AdaBoost(n_estimators=10)
adabooster10.fit(x_train_data, y_train_data)
y_pred_data = adabooster10.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy 10 = ", accuracy)

adabooster100 = AdaBoost(n_estimators=100)
adabooster100.fit(x_train_data, y_train_data)
y_pred_data = adabooster100.predict(x_val_data)
accuracy100 = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy 100 = ", accuracy100)
```

✓ 30.3s

```
Accuracy 10 =  0.8733333333333333
Accuracy 100 =  0.8933333333333333
```

5. (15%) Implement the Random Forest algorithm by using the CART you just implemented from question 2. You should implement **three arguments** for the Random Forest.
   1) **N_estimators**: The number of trees in the forest.
   2) **Max_features**: The number of features to consider when looking for the best split
   3) **Bootstrap**: Whether bootstrap samples are used when building trees

   1. Using Criterion='gini', Max_depth=None, Max_features=sqrt(n_features), Bootstrap=True**,** showing the accuracy score of test data by n_estimators=10 and n_estimators=100, respectively.



Question 5.1

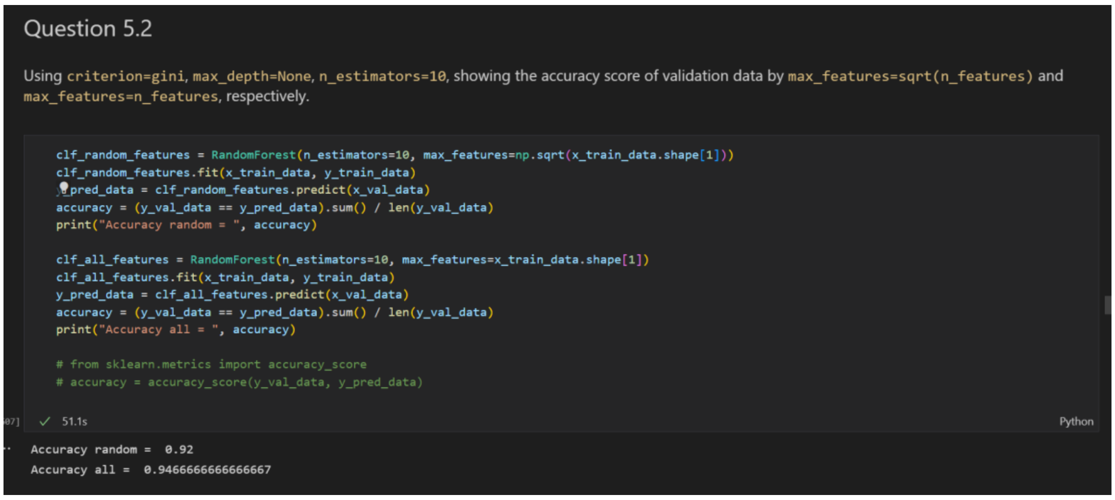
Using `criterion=gini`, `max_depth=None`, `max_features=sqrt(n_features)`, showing the accuracy score of validation data by `n_estimators=10` and `n_estimators=100`, respectively.

```python
clf_10tree = RandomForest(n_estimators=10, max_features=np.sqrt(x_train_data.shape[1]))
clf_10tree.fit(x_train_data, y_train_data)
y_pred_data = clf_10tree.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy 10 = ", accuracy)

clf_100tree = RandomForest(n_estimators=100, max_features=np.sqrt(x_train_data.shape[1]))
clf_100tree.fit(x_train_data, y_train_data)
y_pred_data = clf_100tree.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy 100 = ", accuracy)
```

✓ 1m 54.9s Python

```
Accuracy 10 =  0.9166666666666666
Accuracy 100 =  0.9433333333333334
```

   2. Using Criterion='gini', Max_depth=None, N_estimators=10, Bootstrap=True**,** showing the accuracy score of test data by Max_features=sqrt(n_features) and Max_features=n_features, respectively.



Question 5.2

Using criterion=gini, max_depth=None, n_estimators=10, showing the accuracy score of validation data by max_features=sqrt(n_features) and max_features=n_features, respectively.

```python
clf_random_features = RandomForest(n_estimators=10, max_features=np.sqrt(x_train_data.shape[1]))
clf_random_features.fit(x_train_data, y_train_data)
y_pred_data = clf_random_features.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy random = ", accuracy)

clf_all_features = RandomForest(n_estimators=10, max_features=x_train_data.shape[1])
clf_all_features.fit(x_train_data, y_train_data)
y_pred_data = clf_all_features.predict(x_val_data)
accuracy = (y_val_data == y_pred_data).sum() / len(y_val_data)
print("Accuracy all = ", accuracy)

# from sklearn.metrics import accuracy_score
# accuracy = accuracy_score(y_val_data, y_pred_data)
```

✓ 51.1s Python

```
Accuracy random =  0.92
Accuracy all =  0.9466666666666667
```

6. (20%) Tune the hyperparameter, perform feature engineering or implement more powerful ensemble methods to get a higher accuracy score. Please note that only the ensemble method can be used. The neural network method is not allowed.

Accuracy	**Your scores**
acc > 0.975	**20 points**
0.95 < acc <= 0.975	**15 points**
0.9 < acc <= 0.95	**10 points**
acc < 0.9	**0 points**

詳見.ipynb 檔案

Part. 2, Questions (30%):

Q1.
(a) Because a decision tree is designed to perfectly fit all samples in the training data.
(b) Yes, it is possible. As long as there does not exist two samples with exactly the same features but different labels, it is possible that the decision tree reach a 100% accuracy in the training set.
(c) **Maximum depth**: Stop splitting the tree at some point. **Minimum samples split:** For each node, only split when its samples are more than minimum sample threshold. **Ensemble methods**: Use methods like random forest and adaboost to combine several decision trees, therefore reduce the risk of overfitting.

Q2.
a. True, the weights go up following the update equation.
b. True, the weighted training error tends to increase in each iteration t because the weak classifiers are forced by the AdaBoost algorithm to classify more hard samples.
c. False, if the training data cannot be separated by the linear combination of the type of weak learner it uses, it cannot achieve zero training error. For instance, XOR data distribution.

Q3.
a.
For tree model A, the misclassification rates equal (200 + 0) / 800 = 1/4, 200 is for the first leaf node and 0 is for the second leaf node.
For tree model B, the misclassification rates equal (100 + 100) / 800 = 1/4, the first 100 is for the first leaf node and second 100 is for the second leaf node.
Both of them are 1/4, thus, equal.
b.
Entropy for A:
The Entropy for A ‘s **first** leaf node:
Entropy = - 2/3 log2 (2/3) – 1/3 log2 (1/3) = 0.91829583405
where p1 = 400 / (200+400) = 2/3 and p2 = 200 / (200+400) = 1/3
The Entropy for A ‘s **second** leaf node:
Entropy = - 1* log2 (1) – 0 = 0
where p1 = 200 / (200+0) = 1 and p2 = 0

Total Entropy for A = 6/8 * 0. 91829583405 + 2/8 * 0 = **0.68872187553**, where 6/8 and 2/8 are the ratio of the number of data in the first and the second leaf nodes

---

Entropy for B:
The Entropy for B ‘s **first** leaf node:
Entropy = - 3/4 log2 (3/4) – 1/4 log2 (1/4) = 0.81127812445
where p1 = 300 / (300+100) = 3/4 and p2 = 100 / (300+100) = 1/4
The Entropy for B ‘s **second** leaf node:
Entropy = – 1/4 log2 (1/4) - 3/4 log2 (3/4) = 0.81127812445
where p1 = 100 / (100+300) = 1/4 and p2 = 300 / (100+300) = 3/4

Total Entropy for B = 4/8 *0.81127812445 + 4/8 *0.81127812445 = **0.81127812445**, where 4/8 and 4/8 are the ratio of the number of data in the first and the second leaf nodes

---

Gini for A:

As we have already computed the p1 and p2 for the first and second leaf node of A above, ginis will be directly compute without recomputing those p1 and p2.

The Gini for A ‘s **first** leaf node:

$1 – (2/3)^2 – (1/3)^2 = 0.44444444444$

The Gini for A ‘s second leaf node:

$1 – (1)^2 – (0)^2 = 0$

Total Gini for A = 6/8 * 0.44444444444 + 2/8 * 0 = **0.33333333333,** where 6/8 and 2/8 are the ratio of the number of data in the first and the second leaf nodes

---

Gini for B:

As we have already computed the p1 and p2 for the first and second leaf node of B above, ginis will be directly compute without recomputing those p1 and p2.

The Gini for B ‘s **first** leaf node:

$1 – (3/4)^2 – (1/4)^2 = 0.375$

The Gini for B ‘s **second** leaf node:

$1 – (1/4)^2 – (3/4)^2 = 0.375$

Total Gini for B = 4/8 * 0.375 + 4/8 * 0.375 = **0.375**, where 4/8 and 4/8 are the ratio of the number of data in the first and the second leaf nodes