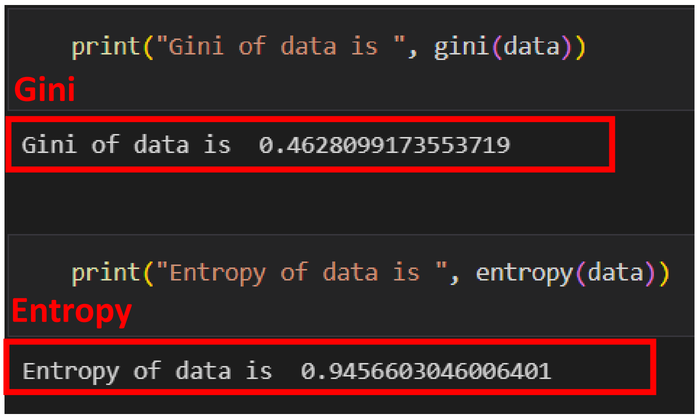
Introduction to Machine Learning HW3 Report

109550159 李驊恩

1. Entropy and Gini Index of this array np.array([1,2,1,1,1,1,2,2,1,1,2]):



```
print("Gini of data is ", gini(data))
```

Gini

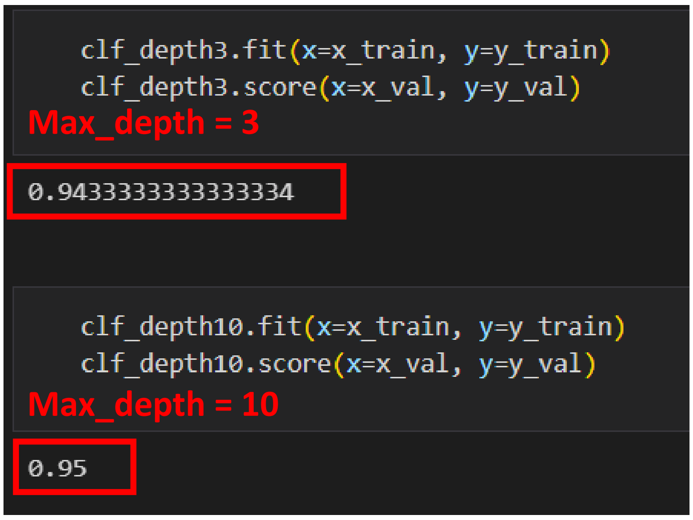
```
Gini of data is  0.4628099173553719
```

```
print("Entropy of data is ", entropy(data))
```

Entropy

```
Entropy of data is  0.9456603046006401
```

2. 2.1 Criterion = gini, Max_depth = 3 and 10



```
clf_depth3.fit(x=x_train, y=y_train)
clf_depth3.score(x=x_val, y=y_val)
```

Max_depth = 3

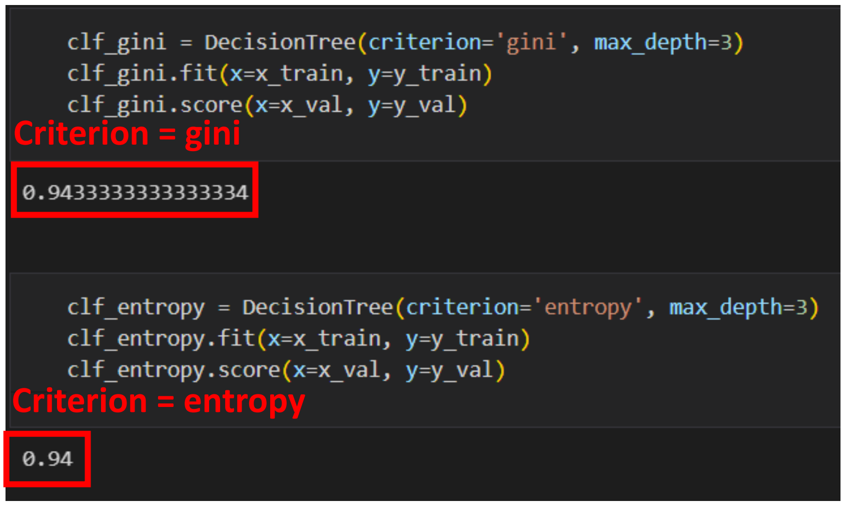
```
0.9433333333333334
```

```
clf_depth10.fit(x=x_train, y=y_train)
clf_depth10.score(x=x_val, y=y_val)
```

Max_depth = 10

```
0.95
```

2.2 Max_depth = 3, Criterion = gini and entropy



```
clf_gini = DecisionTree(criterion='gini', max_depth=3)
clf_gini.fit(x=x_train, y=y_train)
clf_gini.score(x=x_val, y=y_val)
```

Criterion = gini

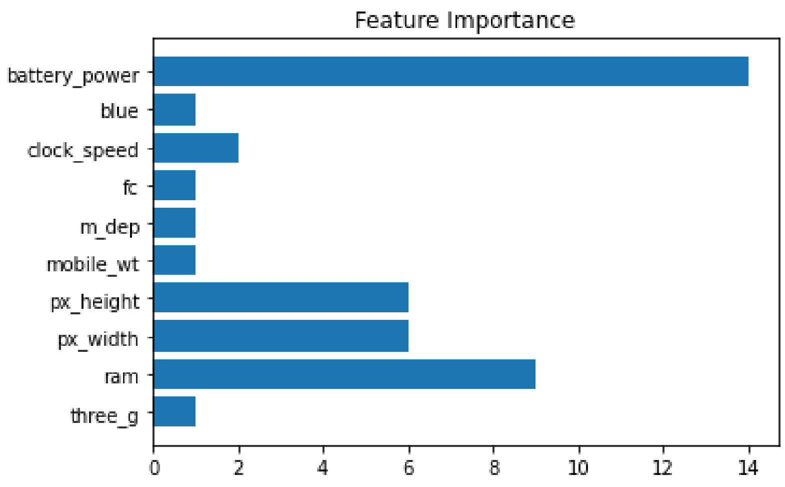
```
0.9433333333333334
```

```
clf_entropy = DecisionTree(criterion='entropy', max_depth=3)
clf_entropy.fit(x=x_train, y=y_train)
clf_entropy.score(x=x_val, y=y_val)
```

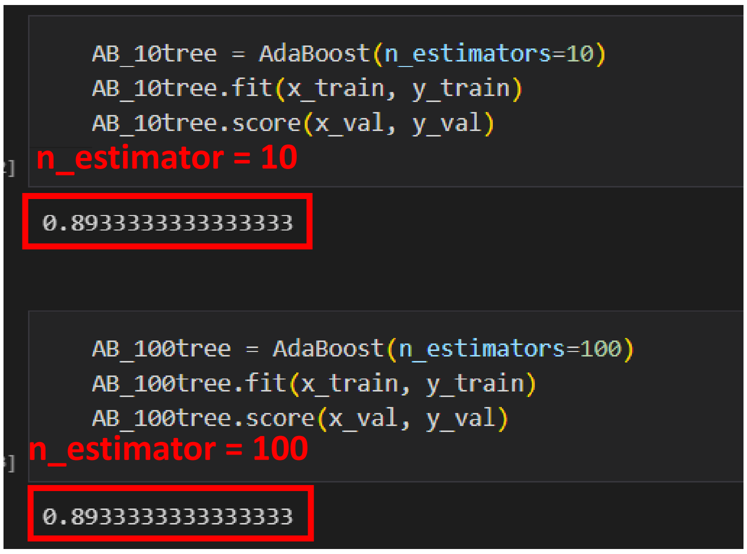
Criterion = entropy

```
0.94
```

3. Feature importance of my Decision Tree model



4. n_estimators=10 and n_estimators=100



```
AB_10tree = AdaBoost(n_estimators=10)
AB_10tree.fit(x_train, y_train)
AB_10tree.score(x_val, y_val)
```

n_estimator = 10

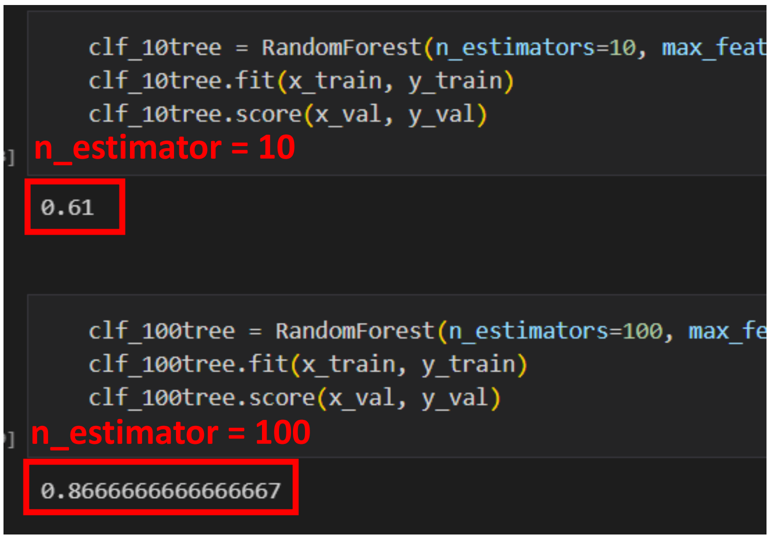
```
0.8933333333333333
```

```
AB_100tree = AdaBoost(n_estimators=100)
AB_100tree.fit(x_train, y_train)
AB_100tree.score(x_val, y_val)
```

n_estimator = 100

```
0.8933333333333333
```

5. 5.1 Criterion='gini', Max_depth=None, Max_features=sqrt(n_features), Bootstrap=True, n_estimator = 10 and 100



```
clf_10tree = RandomForest(n_estimators=10, max_feat
clf_10tree.fit(x_train, y_train)
clf_10tree.score(x_val, y_val)
```

n_estimator = 10

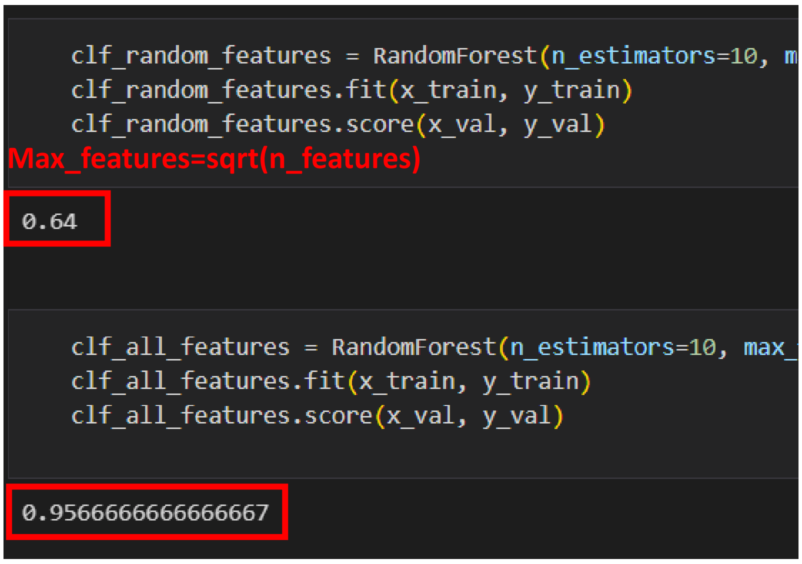
```
0.61
```

```
clf_100tree = RandomForest(n_estimators=100, max_fe
clf_100tree.fit(x_train, y_train)
clf_100tree.score(x_val, y_val)
```

n_estimator = 100

```
0.8666666666666667
```

5.2 Criterion='gini', Max_depth=None, N_estimators=10, Bootstrap=True
Max_features=sqrt(n_features) and Max_features=n_features, respectively



```
clf_random_features = RandomForest(n_estimators=10, m
clf_random_features.fit(x_train, y_train)
clf_random_features.score(x_val, y_val)
```

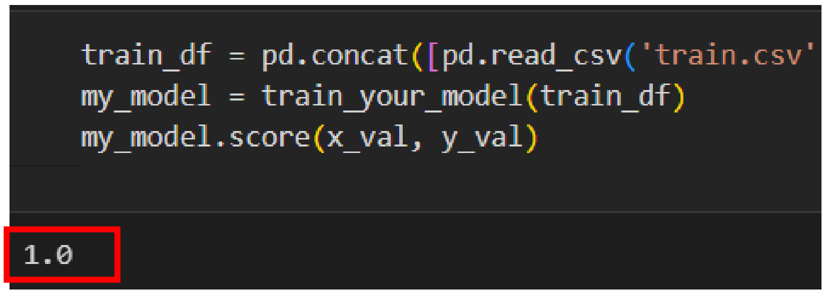
Max_features=sqrt(n_features)

```
0.64
```

```
clf_all_features = RandomForest(n_estimators=10, max_
clf_all_features.fit(x_train, y_train)
clf_all_features.score(x_val, y_val)
```

```
0.9566666666666667
```

6. Accuracy score:



```
train_df = pd.concat([pd.read_csv('train.csv'
my_model = train_your_model(train_df)
my_model.score(x_val, y_val)
```

```
1.0
```

Question Part:

1. Why does a decision tree have a tendency to overfit to the training set? Is it possible for a decision tree to reach a 100% accuracy in the training set? please explain. List and describe at least 3 strategies we can use to reduce the risk of overfitting of a decision tree.

When building a model, if there are too many features and some of the features are irrelevant and not helpful for splitting data, and we do not restrict the growing of the decision tree, the tree will become particular deep. Then our algorithm will make a new node for each feature, looking at every possible split of every independent variable and finally classify all the data correctly, which means that it reached a 100% accuracy in the training data. Thus, it is possible.

Strategies that we can use to avoid overfitting :

1. Limit the remaining data size in a leaf node: For example, we can set a minimum value of the data in a leaf node. If the node try to keep splitting, the next leaf node may remain not enough number of data and hence the tree can not keep growing, which is a way to avoid overfitting.
2. Limit the depth of the tree: For example, we can set a maximum depth of a tree, and once the tree's depth reaches this value, it stops to grow. Thus, it can also help preventing overfitting.
3. Limit the features to consider for split: If we set a maximum value to the features to consider for split, it won't consider some relatively irrelevant feature and therefore it avoids overfitting.

2. **This part consists of three True/False questions. Answer True/False for each question and briefly explain your answer.**
   a. **In AdaBoost, weights of the misclassified examples go up by the same multiplicative factor.**
   b. **In AdaBoost, weighted training error $\boldsymbol{\varepsilon}_t$ of the t_{th} weak classifier on training data with weights D_t tends to increase as a function of t.**
   c. **AdaBoost will eventually give zero training error regardless of the type of weak classifier it uses, provided enough iterations are performed.**

a. **True**. If we check the weight update, we can observe that the AdaBoost algorithm was formulated for weak classifiers that predict ±1 labels. The weights of all misclassified points will be multiplied by $\exp[-\alpha_t y_i h_t(x_i)]$ before normalization. Thus, weights of the misclassified examples go up by the same multiplicative factor.

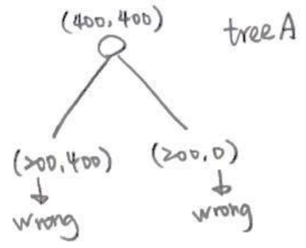
b. **True**. Since the weak classifiers are inclined to classify more difficult examples, there will be some examples that are repeatedly misclassified by the weak classifiers. Those examples' weights will increase. Thus, the weighted training error $\boldsymbol{\varepsilon}_t$ of the t-th weak classifier on the training data is also tend to increase.

c. **False**. If the data is not separable by a linear combination of the weak classifier we are using, AdaBoost can't achieve zero training error.

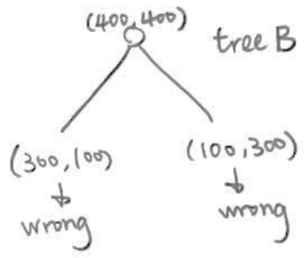
3.

3.

Misclassification rate for tree A: $\frac{200+0}{800}=\frac{1}{4}$



Misclassification rate for tree B: $\frac{100+100}{800}=\frac{1}{4}$



Hence they are equal

Entropy of tree A: $\underline{\frac{600}{800}\times\left(-\frac{400}{600}\log_2\frac{400}{600}-\frac{200}{600}\log_2\frac{200}{600}\right)}+\frac{200}{800}\times\left(-\frac{200}{200}\log_2\frac{200}{200}-\frac{0}{200}\log_2\frac{0}{200}\right)$

$$=\frac{3}{4}\left(-\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}\log_2\frac{1}{3}\right)+\frac{1}{4}\times(0-0)$$

$$=\frac{3}{4}\left[-\frac{2}{3}(1-\log_2 3)-\frac{1}{3}(-\log_2 3)\right]=\frac{3}{4}\log_2 3-\frac{1}{2}\approx 0.689$$ #

Entropy of tree B: $\frac{400}{800}\times\left(-\frac{300}{400}\log_2\frac{300}{400}-\frac{100}{400}\log_2\frac{100}{400}\right)+\frac{400}{800}\times\left(-\frac{300}{400}\log_2\frac{300}{400}-\frac{100}{400}\log_2\frac{100}{400}\right)$

$$=-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}$$

$$=-\frac{3}{4}(\log_2 3-2)-\frac{1}{4}(0-2)=2-\frac{3}{4}\log_2 3\approx 0.811$$ #

Gini of tree A: $\frac{600}{800}\times\left(1-\left(\frac{400}{600}\right)^2-\left(\frac{200}{600}\right)^2\right)+\frac{200}{800}\times\left(1-\left(\frac{200}{200}\right)^2-\left(\frac{0}{200}\right)^2\right)$

$$=\frac{3}{4}\times\frac{4}{9}=\frac{1}{3}\approx 0.333$$ #

Gini of tree B: $\frac{400}{800}\times\left(1-\left(\frac{300}{400}\right)^2-\left(\frac{100}{400}\right)^2\right)+\frac{400}{800}\times\left(1-\left(\frac{300}{400}\right)^2-\left(\frac{100}{400}\right)^2\right)$

$$=1-\left(\frac{3}{4}\right)^2-\left(\frac{1}{4}\right)^2=\frac{6}{16}=\frac{3}{8}=0.375$$ #