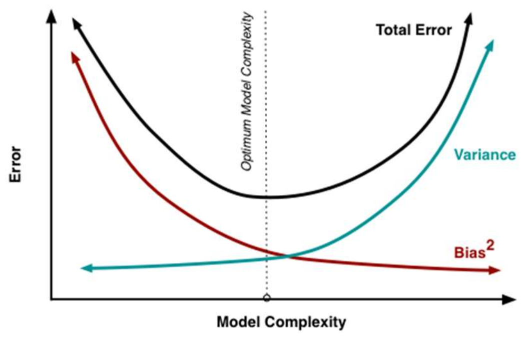
ML – HW1

Amirhossein Izadi – 99243018

Q1. The flexibility/complexity of classification and regression trees is decided by the tree depth:

- To obtain a small bias the tree needs to be grown deep;
- but this results in a high variance! (It means overfitting in the model)



Solution → Ensemble methods:

Combine and average multiple trees:

- Bagging

The problem with bagging: the B bootstrapped dataset are correlated;

Hence, the variance reduction could be diminished

- Random forest

To solve the problem with decision tree as you can see above, we can use bagging or random forest. But use of bagging as I mentioned earlier still problematic for us. A random forest is constructed by bagging, but for each split in each tree only a random subset of $q \leq p$ features are considered as splitting variables. The following list refers to the things that the random forest does to solve decision tree overfitting problem

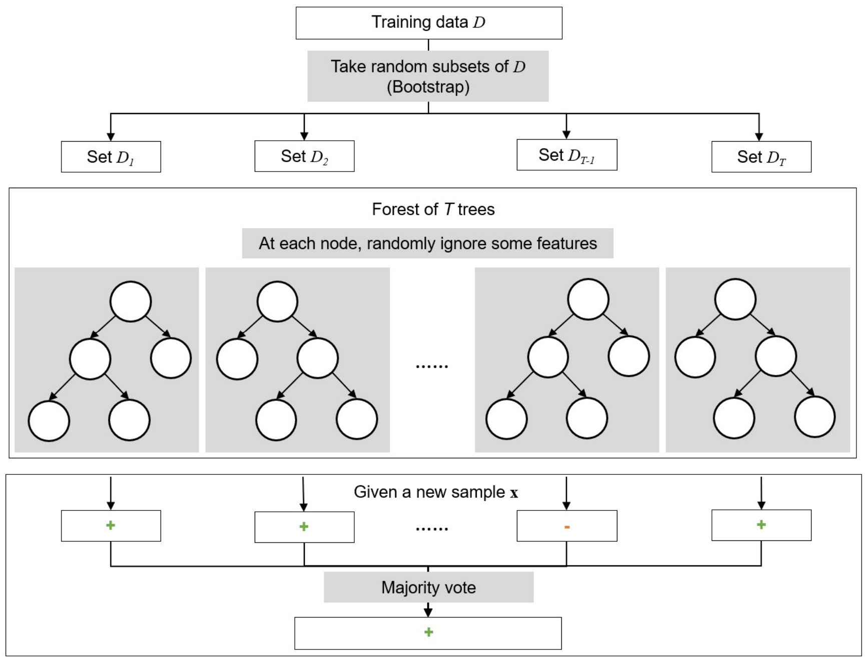
1. **Bagging (Bootstrap Aggregating):** Random Forest builds multiple decision trees by training each tree on a random subset of the training data, sampled with replacement (bootstrap sampling). This process helps reduce overfitting by introducing diversity among the trees. Since each tree sees a different subset of the data, they are less likely to overfit to the noise in any single subset.

2. **Feature Randomization:** In addition to sampling data, Random Forest also introduces randomness in feature selection. For each split in the tree, a random subset of features is considered. This helps prevent individual trees from becoming too specialized and overfitting to specific features.

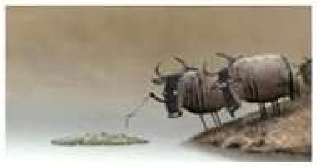
3. **Voting or Averaging:** In the case of classification tasks, Random Forest combines the predictions of multiple trees through voting, while for regression tasks, it averages the predictions. This ensemble approach tends to be more robust and generalizes better than a single decision tree.

4. **Controlled Random Forest Hyperparameters:** Based on this simple explanation of the random forest model there are multiple hyperparameters that we can tune while loading an instance of the random forest model which helps us to prune overfitting.

   - ***max_depth:*** *This controls how deep or the number of layers deep we will have our decision trees up to.*
   - ***n_estimators:*** *This controls the number of decision trees that will be there in each layer. This and the previous parameter solves the problem of overfitting up to a great extent.*
   - ***criterion:*** *While training a random forest data is split into parts and this parameter controls how these splits will occur.*
   - ***min_samples_leaf:*** *This determines the minimum number of leaf nodes.*
   - ***min_samples_split:*** *This determines the minimum number of samples required to split the code.*
   - ***max_leaf_nodes:*** *This determines the maximum number of leaf nodes.*


Training data D
Take random subsets of D (Bootstrap)
Set D_1
Set D_2
Set D_{T-1}
Set D_T
Forest of T trees
At each node, randomly ignore some features
......
Given a new sample $\mathbf{x}$
+
+
......
-
+
Majority vote
+

Q2.



	Be eaten	Not to be eaten
Crocodile	38/100	26/100
Not a crocodile	14/100	22/100

A)

$$H(Y|X=x) \quad = \quad -\sum_{y \in Y} p(y|x) \log_2 p(y|x)$$

P(X=Crocodile) ***H(Y|X=Crocodile)*** + P(X=Not a Crocodile)***H(Y|X =Not a Crocodile)***
= 0.64*0.97 + 0.36*0.96 ≈ 0.967

H(Y|X =Not a Crocodile) **:** - [(38/64) log_2(38/64) + (26/64) log_2(26/64) ≈ 0.97
H(Y|X =Not a Crocodile): - [(14/36) log_2(14/36) + (22/36) log_2(22/36) ≈ 0.96

B)

$$IG\ (Y|X) = H(Y) - H(Y|X\ )$$

- [(0.52) log_2(0.52) + (0.48) log_2(0.48)] + [(38/64) log_2(38/64) + (26/64) log_2(26/64)]

= (0.49 + 0.5) − (0.45 + 0.53) = 0.99 − 0.97 ≈ 0.02 bits

C)

$$H(Y|X = \mathrm{x}\ )$$

- [(26/48) log_2(26/48) + (22/48) log_2(22/48)] ≈ 0.99