Day 9

Agenda :

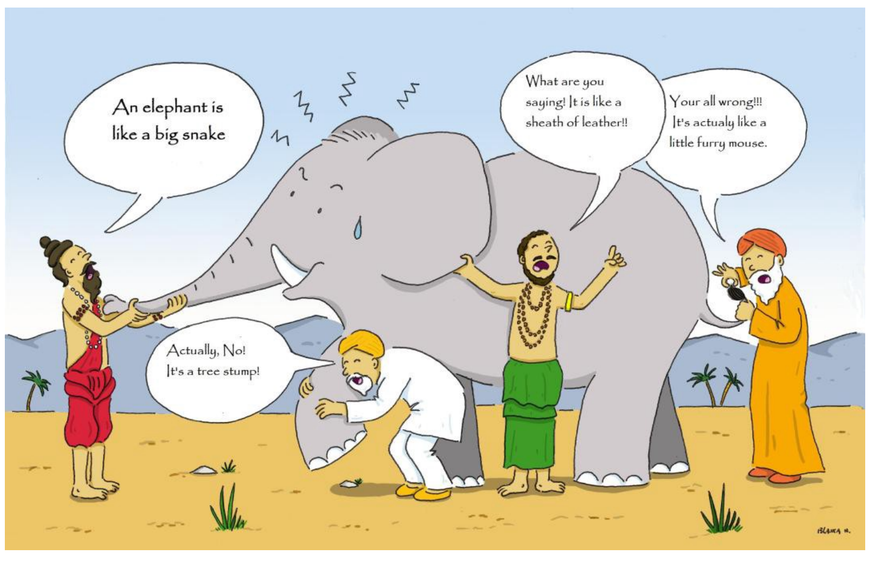
1. Introduction to Ensemble Learning
2. Bias and Variance Tradeoff
3. Basic Ensemble Techniques
    - 4.1 Max Voting
    - 4.2 Averaging
4. Advanced Ensemble Techniques
    - 5.1 Bagging
    - 5.2 Boosting
    - 5.3 Stacking
5. Algorithms based on Bagging and Boosting
    - 6.1 Random Forest
    - 6.2 Gradient Boosting(GBDT)
    - 6.3 XGBOOST

Ensemble Models

--- Ask your friends to rate a movie and decide either you should go and watch or not

--- Ensemble learning is a general meta approach to machine learning that seeks better predictive performance by combining the predictions from multiple models.

--- When we try to predict the target variable using any machine learning technique, the main causes of difference in actual and predicted values are noise, variance, and bias. Ensemble helps to reduce these factors (except noise, which is irreducible error).



Simple Ensemble Techniques

1. Max Voting :

--- The max voting method is generally used for classification problems.
--- In this technique, multiple models are used to make predictions for each data point.
--- The predictions by each model are considered a 'vote'. The predictions that we get from the majority of the models are used as the final prediction.

The result of max voting would be something like this:

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

2. Averaging :

--- Similar to the max voting technique, multiple predictions are made for each data point in averaging.
--- In this method, we take an average of predictions from all the models and use it to make the final prediction.
--- Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4.4

Advanced Ensemble techniques

1. Bagging

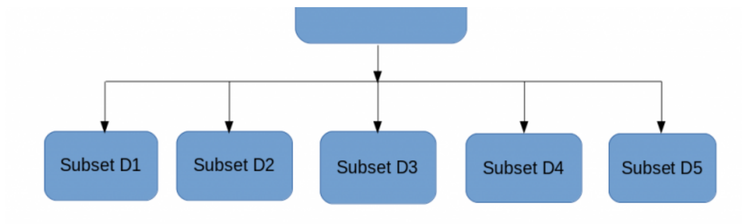
Here's a question: If you create all the models on the same set of data and combine it, will it be useful? There is a high chance that these models will give the same result since they are getting the same input. So how can we solve this problem? One of the techniques is bootstrapping.

Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, with replacement. The size of the subsets is the same as the size of the original set.

1. How bootstrapping of data is happening give me an example.
2. What do you mean by bootstrapped sample and original data set having same size.(Optional)

ORIGINAL DATA

Bootstrapping of Data Example: Imagine we have a list of numbers: 5, 8, 2, 1, 6, 9, 3, 7, 4, and 10.



1. Multiple subsets are created from the original dataset, selecting observations with replacements.
2. A base model (weak model) is created on each of these subsets.
3. The models run in parallel and are independent of each other.
4. The final predictions are determined by combining the predictions from all the models.

Key points:

1. We do sampling with replacement like we can use any data any number of times in our data.
2. Bagging's beauty is reduced variance with low Bias.
3. classification: Majority Vote
   Regression : Mean/Median/Average

4. Model - Bias(Don't touch) Variance(Reduce)
5. Row sampling

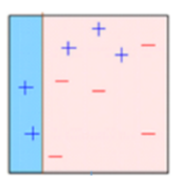
2. Boosting

Whats : If a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results? Such situations are taken care of by boosting.

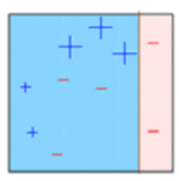
--- Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model.
--- The succeeding models are dependent on the previous model.

Steps:

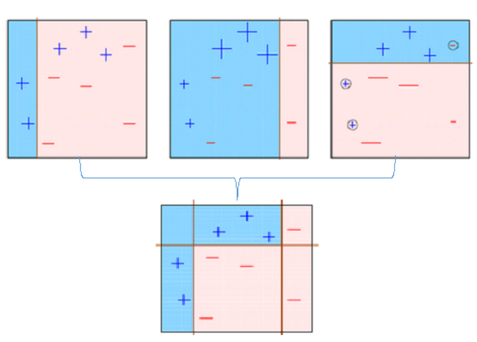
1. A subset is created from the original dataset.
2. Initially, all data points are given equal weights.
3. A base model is created on this subset.
4. This model is used to make predictions on the whole dataset



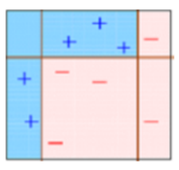
5. Errors are calculated using the actual values and predicted values.
6. The observations which are incorrectly predicted, are given higher weights.
   (Here, the three misclassified blue-plus points will be given higher weights)
7. Another model is created and predictions are made on the dataset.
   (This model tries to correct the errors from the previous model)



8. Similarly, multiple models are created, each correcting the errors of the previous model.
9. The final model (strong learner) is the weighted mean of all the models (weak learners).



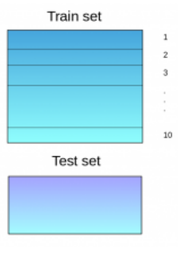
--- Thus, the boosting algorithm combines a number of weak learners to form a strong learner.
--- The individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.



3. Stacking

Stacking is an ensemble learning technique that uses predictions from multiple models (for the example decision tree, knn or svm) to build a new model. This model is used for making predictions on the test set.

1. The train set is split into 10 parts.



Bootstrapping involves creating multiple subsets of this list. Here's an example to help you understand:

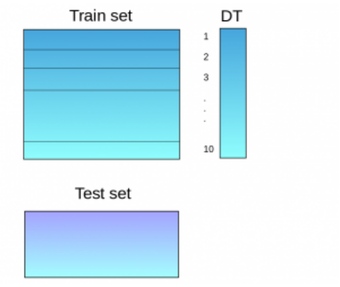
Step 1: We randomly select a number from the list. Let's say we choose the number 2.
Step 2: We put the number 2 back into the original list.
Step 3: We repeat steps 1 and 2 several times (let's say four times) to create a bootstrap sample.
After four iterations of bootstrapping, our bootstrap sample could look like this: 2, 9, 3, 6.

It's like picking a number from a bag, writing it down, putting it back, and then repeating the process multiple times. This way, we create a new list of numbers.

Bootstrapped Sample and Original Dataset Size: When we say the bootstrapped sample and the original dataset have the same size, it means the new list we created through bootstrapping is the same length as the original list.

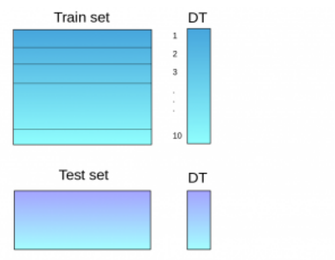
Let's consider our previous example with the original list of numbers. If we had 10 numbers in the original list, we would create a bootstrapped sample of 10 numbers as well. During the bootstrapping process, we randomly select numbers from the original list, and some numbers may be picked more than once while others may not be picked at all.

2. A base model (suppose a decision tree) is fitted on 9 parts and predictions are made for the 10th part. This is done for each part of the train set.

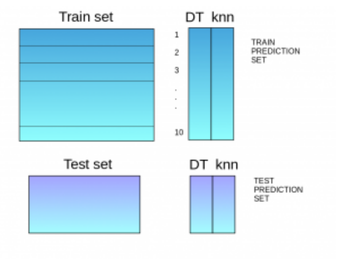


3. The base model (in this case, decision tree) is then fitted on the whole train dataset.

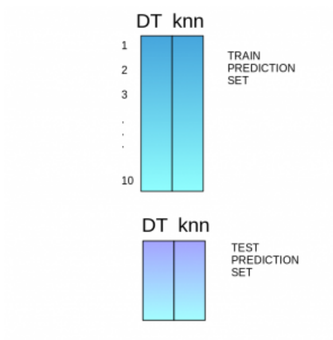
4. Using this model, predictions are made on the test set.



5. Steps 2 to 4 are repeated for another base model (say knn) resulting in another set of predictions for the train set and test set.



6. The predictions from the train set are used as features to build a new model.



7. This model is used to make final predictions on the test prediction set.

Algorithms based on Bagging and Boosting

1. Random Forest :

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

--- Random Forest uses the bagging technique.
--- RF = DT + Bagging + Column Sampling
--- Aggregation : Majority Vote, Mean or Median

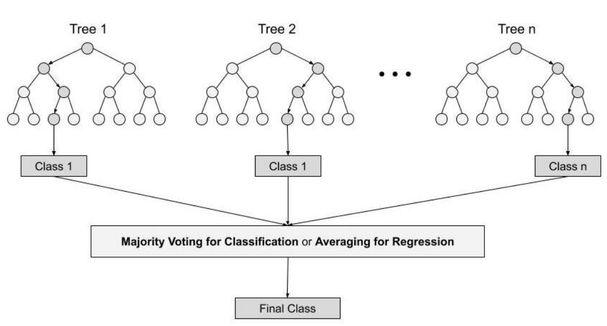
Steps involved in random forest algorithm:

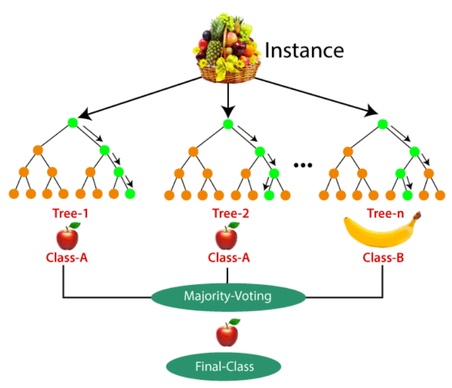
Step 1: In Random forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.





Important Features of Random Forest

1. Diversity - Not all attributes/variables/features are considered while making an individual tree, each tree is different.

2. Immune to the curse of dimensionality - Since each tree does not consider all the features, the feature space is reduced.

3. Parallelization - Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

4. Stability- Stability arises because the result is based on majority voting/ averaging.

Important Hyperparameters

Following hyperparameters increases the predictive power:

1. n_estimators– number of trees the algorithm builds before averaging the predictions.

2. max_features– maximum number of features random forest considers splitting a node.

3. mini_samples_leaf– determines the minimum number of leaves required to split an internal node.

Following hyperparameters increases the speed:

1. n_jobs– it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.

2. random_state– controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.

3. oob_score – OOB means out of the bag. It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out-of-bag samples.

2. Gradient Boosting :

--- Gradient boosting is a method standing out for its prediction speed and accuracy, particularly with large and complex datasets.

--- There are mainly two types of error, bias error and variance error. Gradient boost algorithm helps us minimize bias error of the model

Following is a sample from a random dataset where we have to predict the car price based on various features. The target column is price and other features are independent features.

Row No.	Cylinder Number	Car Height	Engine Location	Price
1	Four	48.8	Front	12000
2	Six	48.8	Back	16500
3	Five	52.4	Back	15500
4	Four	54.3	Front	14000

Step -1 The first step in gradient boosting is to build a base model to predict the observations in the training dataset. For simplicity we take an average of the target column and assume that to be the predicted value as shown below:

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1
1	Four	48.8	Front	12000	14500
2	Six	48.8	Back	16500	14500
3	Five	52.4	Back	15500	14500
4	Four	54.3	Front	14000	14500

Step-2 The next step is to calculate the pseudo residuals which are (observed value – predicted value)

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1	Residual 1
1	Four	48.8	Front	12000	14500	-2500
2	Six	48.8	Back	16500	14500	2000
3	Five	52.4	Back	15500	14500	1000
4	Four	54.3	Front	14000	14500	-500

Step 3 In the next step, we will build a model on these pseudo residuals and make predictions. Why do we do this? Because we want to minimize these residuals and minimizing the residuals will eventually improve our model accuracy and prediction power. So, using the Residual as a target and the original feature Cylinder number, cylinder height, and Engine location we will generate new predictions. Note that the predictions, in this case, will be the error values, not the predicted car price values since our target column is an error now.
Let's say hm(x) is our DT made on these residuals.

Step- 4 In this step we find the output values for the predictions. Those predictions are not the actual values as Y_true.

Step -5 This is finally the last step where we have to update the predictions of the previous model. It can be updated as

Update the model:

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x)$$

New Prediction = **Previous Prediction** + **Learning Rate** * **The tree made on Residuals**

Algorithm 1: Gradient Boosting Decision Tree for Classification [44]

1. Initialise $F_{k0}(x) = 0, k = 1,2 \cdots K$.
2. For $m = 1$ to M, where M is the maximum iteration number.
   a. Set $p_k(x)$ as represented by Equation 2.
   b. For $k = 1,2 \cdots K$:
      i. Compute $r_{ikm} = y_{ik} - p_k(x_i), i = 1,2 \cdots n$ (n is the total number of observations).
      ii. Fit a decision tree for the new target γ_{ikm} giving each terminal region, $R_{jkm}, j = 1,2 \cdots J_m$.
      iii. Compute

$$\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{x \in R_{jkm}} r_{ikm}}{\sum_{x \in R_{jkm}} |r_{ikm}|(1-|r_{ikm}|)}, \qquad j = 1,2 \cdots J_m$$

      iv. Update $F_{km}(x) = F_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jkm} I(x \in R_{jkm})$, where $I(x \in R_{jkm})$ is the index function.
3. Return $\hat{F}_x(x) = F_{kM}(x), k = 1,2 \cdots K$

3. XGBOOST:

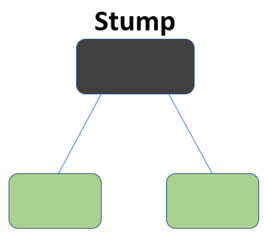
Install XGBOOST : sudo pip install xgboost

XGBOOST == GBDT + Row Sampling + Column Sampling

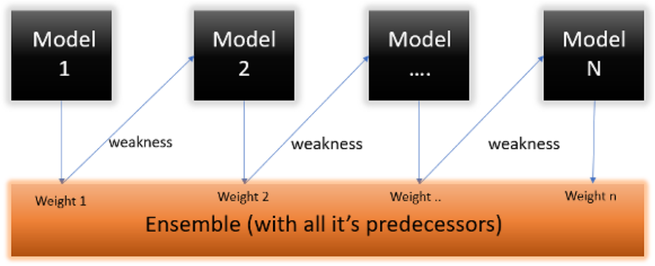
-- Xgboost is implementation of GBDT and RF

4. Adaboost

AdaBoost is decision trees with one level which means Decision trees with only 1 split. .These trees are also called Decision Stumps



What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points with higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.



Step 1. First of all, these data points will be assigned some weights. Initially, all the weights will be equal.

Row No.	Gender	Age	Income	Illness	Sample Weights
1	Male	41	40000	Yes	1/5
2	Male	54	30000	No	1/5
3	Female	42	25000	No	1/5
4	Female	40	60000	Yes	1/5
5	Male	46	50000	Yes	1/5

Step 2 – We start by seeing how well "Gender" classifies the samples and will see how the variables (Age, Income) classify the samples.

We'll create a decision stump for each of the features and then calculate the Gini Index of each tree. The tree with the lowest Gini Index will be our first stump.

Here in our dataset, let's say Gender has the lowest gini index, so it will be our first stump.

Step 3 – We'll now calculate the "Amount of Say" or "Importance" or "Influence" for this classifier in classifying the data points using this formula:

Here in our dataset, let's assume there is 1 wrong output, so our total error will be 1/5, and the alpha (performance of the stump) will be:

$$\frac{1}{2}\log\frac{1-Total\ Error}{Total\ Error}$$

$$Performance\ of\ the\ stump\ =\ \frac{1}{2}\log_e(\frac{1-Total\ Error}{Total\ Error})$$

$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{1-\frac{1}{5}}{\frac{1}{5}}\right)$$

$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{0.8}{0.2}\right)$$

$$\alpha\ =\ \frac{1}{2}\log_e(4)\ =\ \frac{1}{2}*(1.38)$$

$$\alpha\ =\ 0.69$$

Step 4. After finding the importance of the classifier and total error, we need to finally update the weights, and for this, we use the following formula:

$$New\ sample\ weight\ =\ old\ weight\ *\ e^{\pm Amount\ of\ say\ (\alpha)}$$

The amount of, say (alpha) will be negative when the sample is correctly classified.

The amount of, say (alpha) will be positive when the sample is miss-classified.

New weights for correctly classified samples are:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(-0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 0.502\ =\ 0.1004$$

For wrongly classified samples, the updated weights will be:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 1.994\ =\ 0.3988$$

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004
2	Male	54	30000	No	1/5	0.1004

3	Female	42	25000	No	1/5	0.1004
4	Female	40	60000	Yes	1/5	0.3988
5	Male	46	50000	Yes	1/5	0.1004

We know that the total sum of the sample weights must be equal to 1, but here if we sum up all the new sample weights, we will get 0.8004. To bring this sum equal to 1, we will normalize these weights by dividing all the weights by the total sum of updated weights, which is 0.8004. So, after normalizing the sample weights, we get this dataset, and now the sum is equal to 1.

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004/0.8004 =0.1254
2	Male	54	30000	No	1/5	0.1004/0.8004 =0.1254
3	Female	42	25000	No	1/5	0.1004/0.8004 =0.1254
4	Female	40	60000	Yes	1/5	0.3988/0.8004 =0.4982
5	Male	46	50000	Yes	1/5	0.1004/0.8004 =0.1254

Step 5 – Now, we need to make a new dataset to see if the errors decreased or not. For this, we will remove the "sample weights" and "new sample weights" columns and then, based on the "new sample weights," divide our data points into buckets.

Row No.	Gender	Age	Income	Illness	New Sample Weights	Buckets
1	Male	41	40000	Yes	0.1004/0.8004= 0.1254	0 to 0.1254
2	Male	54	30000	No	0.1004/0.8004= 0.1254	0.1254 to 0.2508
3	Female	42	25000	No	0.1004/0.8004= 0.1254	0.2508 to 0.3762
4	Female	40	60000	Yes	0.3988/0.8004= 0.4982	0.3762 to 0.8744
5	Male	46	50000	Yes	0.1004/0.8004= 0.1254	0.8744 to 0.9998

Step 6 – We are almost done. Now, what the algorithm does is selects random numbers from 0-1.Since incorrectly classified records have higher sample weights, the probability of selecting those records is very high.

Suppose the 5 random numbers our algorithm take is 0.38,0.23,0.26,0.40,0.55.

Row No.	Gender	Age	Income	Illness
1	Female	40	60000	Yes
2	Male	54	30000	No
3	Female	42	25000	No
4	Female	40	60000	Yes
5	Female	40	60000	Yes

This comes out to be our new dataset, and we see the data point, which was wrongly classified, has been selected 3 times because it has a higher weight.

Step 7 – Now this act as our new dataset, and we need to repeat all the above steps i.e.

1. Assign equal weights to all the data points.
2. Find the stump that does the best job classifying the new collection of samples by finding their Gini Index and selecting the one with the lowest Gini index.
3. Calculate the "Amount of Say" and "Total error" to update the previous sample weights.
4. Normalize the new sample weights