Ensemble Model -

Ensemble methods involve group of predictive models to achieve a better accuracy and model stability.

Simple Ensemble Techniques

In this section, we will look at a few simple but powerful techniques, namely:

1. Max Voting
2. Averaging
3. Weighted Averaging

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 as a 'vote'. The predictions which we get from the majority of the models are used as the final prediction.

For example, when you asked 5 of your colleagues to rate your movie (out of 5); we'll assume three of them rated it as 4 while two of them gave it a 5. Since the majority gave a rating of 4, the final rating will be taken as 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.

For example, in the below case, the averaging method would take the average of all the values.

i.e. (5+4+5+4+4)/5 = 4.4

3. Weighted Average

This is an extension of the averaging method. All models are assigned different weights defining the importance of each model for prediction. For instance, if two of your colleagues are critics, while others have no prior experience in this field, then the answers by these two friends are given more importance as compared to the other people.

The result is calculated as [(5*0.23) + (4*0.23) + (5*0.18) + (4*0.18) + (4*0.18)] = 4.41.

Advanced Ensemble techniques

1. Stacking
2. Blending
3. Bagging

4. Boosting

1. Stacking

Stacking is an ensemble learning technique that uses predictions from multiple models (for example decision tree, knn or svm) to build a new model. This model is used for making predictions on the test set. Below is a step-wise explanation for a simple stacked ensemble:
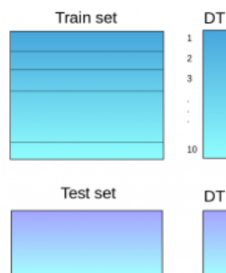
1. The train set is split into 10 parts.
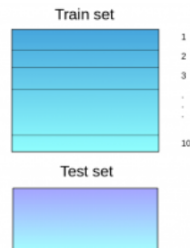
2. A base model (suppose a decision tree) is fitted on 9 parts and predictions are made for the 10th part. This is done for each part of the train set.
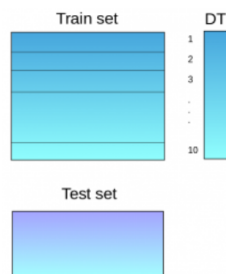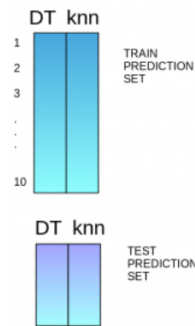

3. The base model (in this case, decision tree) is then fitted on the whole train dataset.
4. Using this model, predictions are made on the test set.

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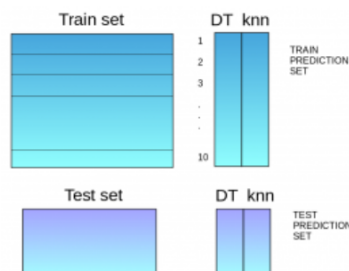
6. The predictions from the train set are used as features to build a new model.

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

2. Blending

Blending follows the same approach as stacking but uses only a holdout (validation) set from the train set to make predictions. In other words, unlike stacking, the predictions are made on the holdout set only. The holdout set and the predictions are used to build a model which is run on the test set. Here is a detailed explanation of the blending process:

1. The train set is split into training and validation sets.

2. Model(s) are fitted on the training set.
3. The predictions are made on the validation set and the test set.
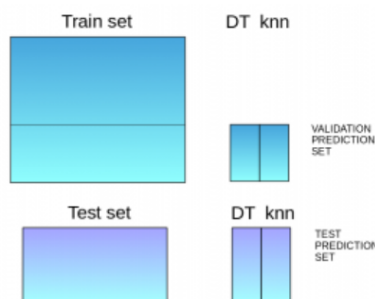
4. The validation set and its predictions are used as features to build a new model.
5. This model is used to make final predictions on the test and meta-features.

3. Bagging -

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.

Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set). The size of subsets created for bagging may be less than the original set.
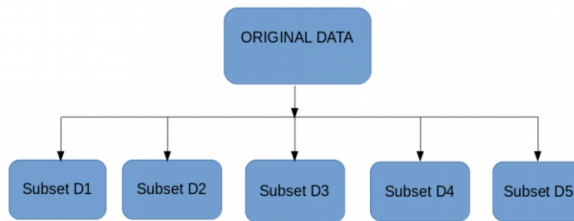
1. Multiple subsets are created from the original dataset, selecting observations with replacement.
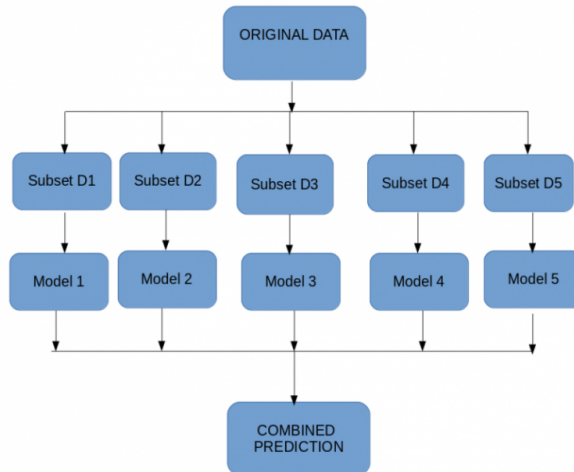2. A base model (weak model) is created on each of these subsets.
3. The models run in parallel and are independent of each other.
4. The final predictions are determined by combining the predictions from all the models.

4. Boosting -

Situation - 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.

Let's understand the way boosting works in the below 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.
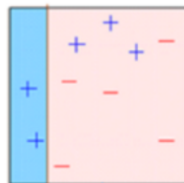
5. Errors are calculated using the actual values and predicted values.
6. The observations which are incorrectly predicted, are given higher weights. (Here, the three misclassified blue-plus points will be given higher weights)
7. Another model is created and predictions are made on the dataset. (This model tries to correct the errors from the previous model)
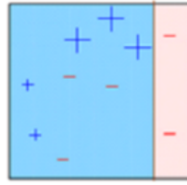
8. Similarly, multiple models are created, each correcting the errors of the previous model.
9. The final model (strong learner) is the weighted mean of all the models (weak learners).
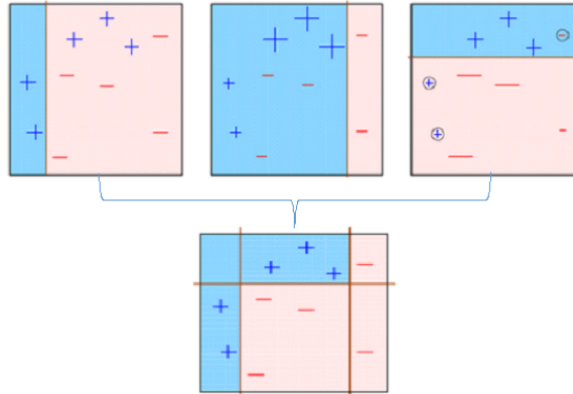
Thus, the boosting algorithm combines a number of weak learners to form a strong learner. The individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.

Algorithms based on Bagging and Boosting

Bagging and Boosting are two of the most commonly used techniques in machine learning. In this section, we will look at them in detail. Following are the algorithms we will be focusing on:

Bagging algorithms-

- Bagging meta-estimator
- Random forest

Boosting algorithms-

- AdaBoost
- GBM
- XGBM
- Light GBM
- CatBoost

Bagging meta-estimator -

Bagging meta-estimator is an ensembling algorithm that can be used for both classification (BaggingClassifier) and regression (BaggingRegressor) problems. It follows the typical bagging technique to make predictions.

Following are the steps for the bagging meta-estimator algorithm:

1. Random subsets are created from the original dataset (Bootstrapping).
2. The subset of the dataset includes all features.
3. A user-specified base estimator is fitted on each of these smaller sets.
4. Predictions from each model are combined to get the final result.

Random Forest -

Random Forest is another ensemble machine learning algorithm that follows the bagging technique. It is an extension of the bagging estimator algorithm. The base estimators in random forest are decision trees. Unlike bagging meta estimator, random forest randomly selects a set of features which are used to decide the best split at each node of the decision tree.

Looking at it step-by-step, this is what a random forest model does:

1. Random subsets are created from the original dataset (bootstrapping).
2. At each node in the decision tree, only a random set of features are considered to decide the best split.
3. A decision tree model is fitted on each of the subsets.
4. The final prediction is calculated by averaging the predictions from all decision trees.

AdaBoost -

Adaptive boosting or AdaBoost is one of the simplest boosting algorithms. Usually, decision trees are used for modelling. Multiple sequential models are created, each correcting the errors from the last model. AdaBoost assigns weights to the observations which are incorrectly predicted and the subsequent model works to predict these values correctly.

Below are the steps for performing the AdaBoost algorithm:

1. Initially, all observations in the dataset are given equal weights.
2. A model is built on a subset of data.
3. Using this model, predictions are made on the whole dataset.
4. Errors are calculated by comparing the predictions and actual values.
5. While creating the next model, higher weights are given to the data points which were predicted incorrectly.
6. Weights can be determined using the error value. For instance, higher the error more is the weight assigned to the observation.
7. This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached.

Gradient Boosting (GBM) -

Gradient Boosting or GBM is another ensemble machine learning algorithm that works for both regression and classification problems. GBM uses the boosting technique, combining a number of weak learners to form a strong learner. Regression trees used as a base learner, each subsequent tree in series is built on the errors calculated by the previous tree.

We will use a simple example to understand the GBM algorithm. We have to predict the age of a group of people using the below data:

ID	Married	Gender	Current City	Monthly Income	Age (target)
1	Y	M	A	51,000	35
2	N	F	B	25,000	24
3	Y	M	A	74,000	38
4	N	F	A	29,000	30
5	N	F	B	37,000	33

1. The mean age is assumed to be the predicted value for all observations in the dataset.
2. The errors are calculated using this mean prediction and actual values of age.

ID	Married	Gender	Current City	Monthly Income	Age (target)	Mean Age (prediction 1)	Residual 1
1	Y	M	A	51,000	35	32	3
2	N	F	B	25,000	24	32	-8
3	Y	M	A	74,000	38	32	6
4	N	F	A	29,000	30	32	-2
5	N	F	B	37,000	33	32	1

3. A tree model is created using the errors calculated above as target variable. Our objective is to find the best split to minimize the error.
4. The predictions by this model are combined with the predictions 1.

ID	Age (target)	Mean Age (prediction 1)	Residual 1 (new target)	Prediction 2	Combine (mean+pred2)
1	35	32	3	3	35
2	24	32	-8	-5	27
3	38	32	6	3	35
4	30	32	-2	-5	27
5	33	32	1	3	35

5. This value calculated above is the new prediction.
6. New errors are calculated using this predicted value and actual value.

ID	Age (target)	Mean Age (prediction 1)	Residual 1 (new target)	Prediction 2	Combine (mean+pred2)	Residual 2 (latest target)
1	35	32	3	3	35	0
2	24	32	-8	-5	27	-3
3	38	32	6	3	35	-3
4	30	32	-2	-5	27	3
5	33	32	1	3	35	-2

7. Steps 2 to 6 are repeated till the maximum number of iterations is reached (or error function does not change).

XGBoost -

XGBoost (extreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm. XGBoost has proved to be a highly effective ML algorithm, extensively used in machine learning competitions and hackathons. XGBoost has high predictive power and is almost 10 times

faster than the other gradient boosting techniques. It also includes a variety of regularization which reduces overfitting and improves overall performance. Hence it is also known as **regularized boosting** technique.

Let us see how XGBoost is comparatively better than other techniques:

1. Regularization -

- Standard GBM implementation has no regularisation like XGBoost.
- Thus XGBoost also helps to reduce overfitting.

2. Parallel Processing -

- XGBoost implements parallel processing and is faster than GBM .
- XGBoost also supports implementation on Hadoop.

3. High Flexibility -

- XGBoost allows users to define custom optimization objectives and evaluation criteria adding a whole new dimension to the model.

4. Handling Missing Values -

- XGBoost has an in-built routine to handle missing values.

5. Tree Pruning -

- XGBoost makes splits up to the max_depth specified and then starts pruning the tree backwards and removes splits beyond which there is no positive gain.

6. Built-in Cross-Validation -

- XGBoost allows a user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run.

Light GBM -

Before discussing how Light GBM works, let's first understand why we need this algorithm when we have so many others (like the ones we have seen above). **Light GBM beats all the other algorithms when the dataset is extremely large.** Compared to the other algorithms, Light GBM takes lesser time to run on a huge dataset.

LightGBM is a gradient boosting framework that uses tree-based algorithms and follows leaf-wise approach while other algorithms work in a level-wise approach pattern. The images below will help you understand the difference in a better way.

Level-wise tree growth

Leaf-wise tree growth
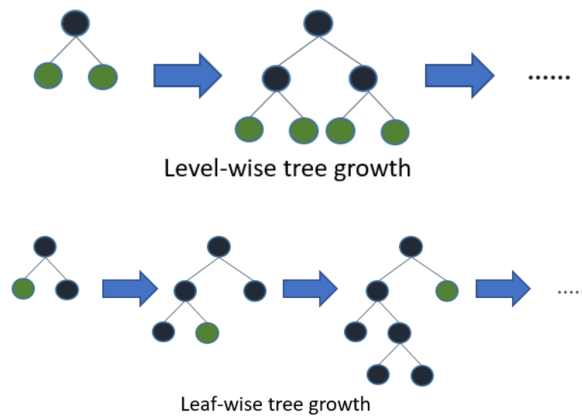
Leaf-wise growth may cause over-fitting on smaller datasets but that can be avoided by using the 'max_depth' parameter for learning.

CatBoost -

Handling categorical variables is a tedious process, especially when you have a large number of such variables. When your categorical variables have too many labels (i.e. they are highly cardinal), performing one-hot-encoding on them exponentially increases the dimensionality and it becomes really difficult to work with the dataset.

CatBoost can automatically deal with categorical variables and does not require extensive data preprocessing like other machine learning algorithms

Refer - https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/ (https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/)