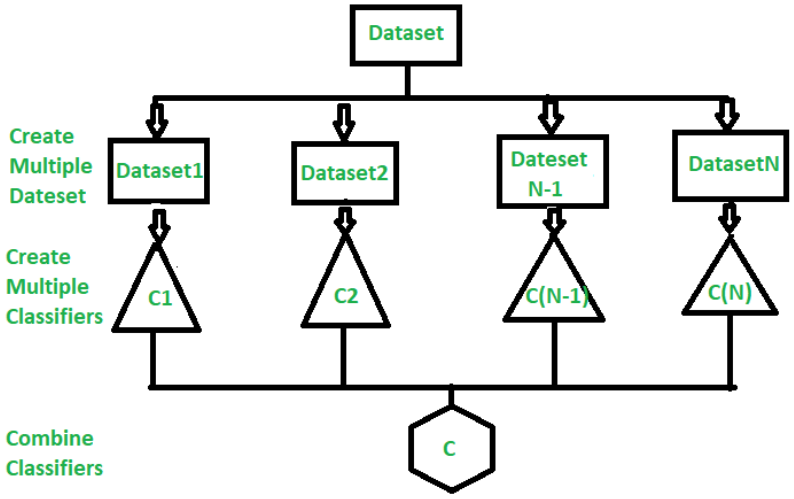
Unit 5: Ensemble Techniques

Introduction to Ensemble Classifiers

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

Advantage: Improvement in predictive accuracy.

Disadvantage: It is difficult to understand an ensemble of classifiers.



Why do ensemble methods work, and what are the three problems identified by Dietterich (2002) that ensembles help to overcome?

Statistical Problem

The Statistical Problem occurs when the **hypothesis space** is too large compared to the amount of available data. In such cases:

- Multiple hypotheses may perform equally well on the training data.

- The learning algorithm arbitrarily selects one of them.
- There is a risk that the selected hypothesis may **not generalize well** to unseen data, resulting in **low accuracy** on new examples.

Computational Problem

The Computational Problem arises when the learning algorithm **cannot guarantee** finding the best hypothesis.

Representational Problem

The Representational Problem arises when the **hypothesis space does not contain a good approximation** of the target class(es).

What are Main Challenges for Developing Ensemble Models?

The main challenge is not to obtain highly accurate base models, but rather to obtain base models which make different kinds of errors. For example, if ensembles are used for classification, high accuracies can be accomplished if different base models misclassify different training examples, even if the bas

Methods for Independently Constructing Ensembles –

Majority Vote

Bagging and Random Forest

Randomness Injection

Feature-Selection Ensembles

Error-Correcting Output Coding

Methods for Coordinated Construction of Ensembles –

Boosting

Stacking

Reliable Classification: Meta-Classifier Approach

Co-Training and Self-Training

Types of Ensemble Classifiers –

1. Bagging
2. Random Forest
3. Boosting

Random Forest:

Random Forest is an extension of bagging. Each classifier in the ensemble is a decision tree classifier and is generated using a random selection of attributes at each node to determine the split. During classification, each tree votes and the most popular class is returned.

Implementation steps of Random Forest –

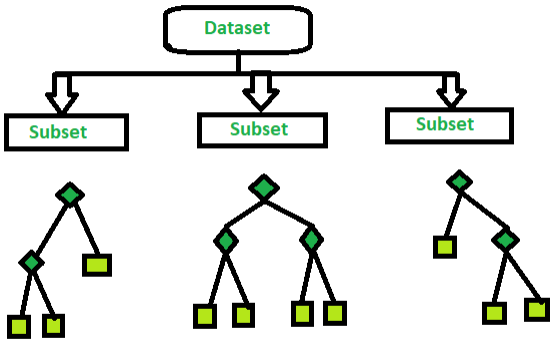
1. **Bootstrap Sampling**: Create multiple subsets from the original dataset by selecting observations with replacement, similar to bagging.

2. **Random Feature Selection**: For each decision tree, a random subset of features is chosen at each node. The best feature from this subset is used to split the node.

3. **Tree Growth**: Each decision tree is grown to its maximum depth, without pruning, allowing the tree to learn as much as possible from the training data.

4. **Ensemble Training**: Repeat the process of creating trees with different random subsets of data and features.

5. **Voting for Final Prediction**: Once all the trees are trained, each tree makes a prediction. The final prediction is based on the majority vote from all the trees in the forest.



Bagging

Bootstrap Aggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the variance and helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.

Description of the Technique

Suppose a set D of d tuples, at each iteration i, a training set Di of d tuples is selected via row sampling with a replacement method (i.e., there can be repetitive elements from different d tuples) from D (i.e., bootstrap). Then a classifier model Mi is learned for each training set D < i. Each classifier Mi returns its class prediction. The bagged classifier M* counts the votes and assigns the class with the most votes to X (unknown sample).

Implementation Steps of Bagging

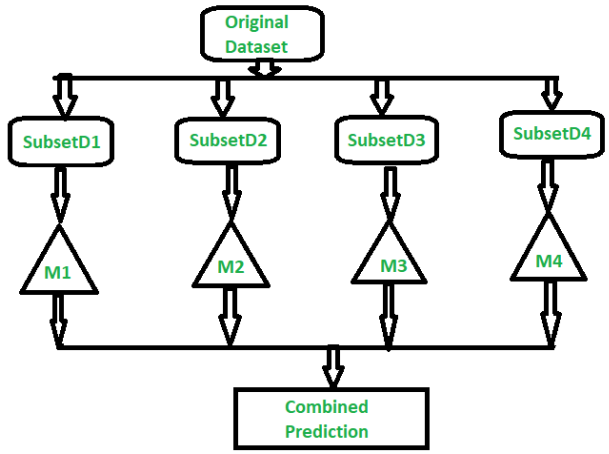
Step 1: Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.

Step 2: A base model is created on each of these subsets.

Step 3: Each model is learned in parallel with each training set and independent of each other.

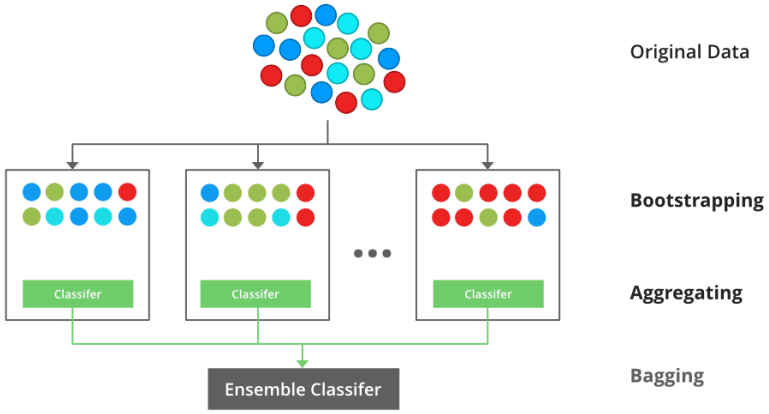
Step 4: The final predictions are determined by combining the predictions from all the models.

An illustration for the concept of bootstrap aggregating (Bagging)



Example of Bagging

The Random Forest model uses Bagging, where decision tree models with higher variance are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.



Boosting

Boosting is an ensemble modeling technique designed to create a strong classifier by combining multiple weak classifiers. The process involves building models sequentially, where each new model aims to correct the errors made by the previous ones.

- In **Boosting**, the process starts by training an initial model on the training data. Subsequent models are trained to correct the errors made by the previous ones.
- Boosting assigns weights to the data points in the original dataset.
- **Misclassified instances** receive **higher weights** to emphasize their importance in the next model's training.
- **Correctly classified instances** receive **lower weights** to reduce their impact.
- Training on weighted data: The subsequent model learns from the weighted dataset, focusing its attention on harder-to-learn examples (those with higher weights).
- This iterative process continues until:
  - The entire training dataset is accurately predicted, or
  - A predefined maximum number of models is reached.

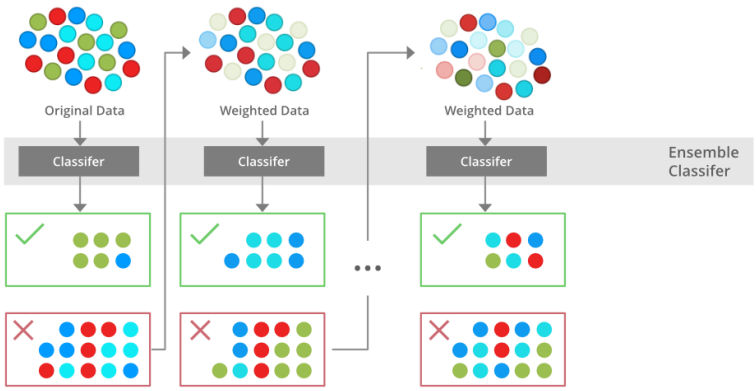
Boosting Algorithms

There are several boosting algorithms. The original ones, proposed by Robert Schapire and Yoav Freund were not adaptive and could not take full advantage of the weak learners. Schapire and Freund then developed AdaBoost, an adaptive boosting algorithm that won the prestigious Gödel Prize. AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple “weak classifiers” into a single “strong classifier”.

Algorithm:

1. Initialise the dataset and assign equal weight to each of the data point.
2. Provide this as input to the model and identify the wrongly classified data points.
3. Increase the weight of the wrongly classified data points and decrease the weights of correctly classified data points. And then normalize the weights of all data points.
4. if (got required results)
   Goto step 5
   Else
   Goto step 2
5. End

An illustration presenting the intuition behind the boosting algorithm, consisting of the parallel learners and weighted dataset.



Similarities Between Bagging and Boosting

Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods.

1. Both are ensemble methods to get N learners from 1 learner.
2. Both generate several training data sets by random sampling.
3. Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
4. Both are good at reducing variance and provide higher stability.

Differences between Bagging and Boosting techniques:-

Bagging	Boosting
The most effective manner of mixing predictions that belong to the same type.	A manner of mixing predictions that belong to different sorts.
The main task of it is decrease the variance but not bias.	The main task of it is decrease the bias but not variance.
Here each of the model is different weight.	Here each of the model is same weight.
Each of the model is built here independently.	Each of the model is built here dependently.

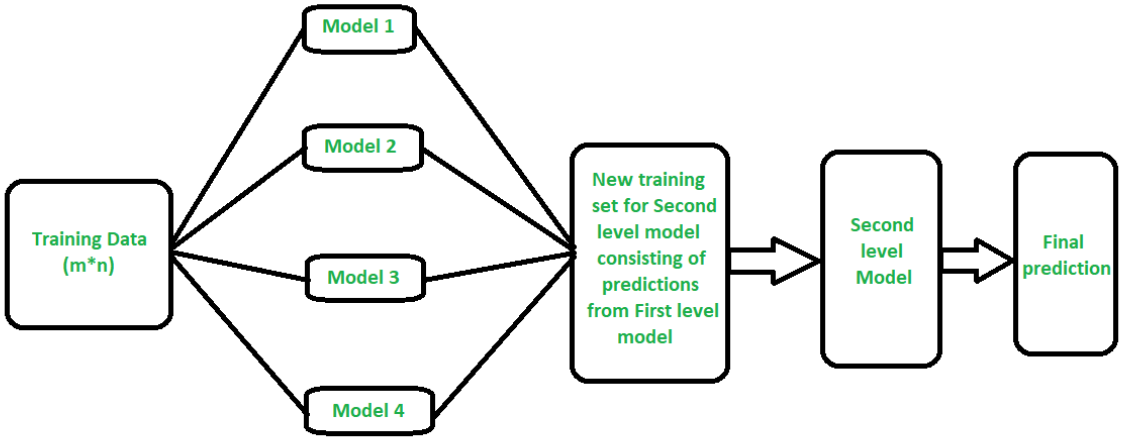
This training records subsets are decided on using row sampling with alternative and random sampling techniques from the whole training dataset.	Each new subset consists of the factors that were misclassified through preceding models.
It is trying to solve by over fitting problem.	It is trying to solve by reducing the bias.
If the classifier is volatile (excessive variance), then apply bagging.	If the classifier is stable and easy (excessive bias) the practice boosting.
In the bagging base, the classifier is works parallelly.	In the boosting base, the classifier is works sequentially.
Example is random forest model by using bagging.	Example is AdaBoost using the boosting technique.

Stacking in Machine Learning

1. Stacking is a way to ensemble multiple classifications or regression model. There are many ways to ensemble models, the widely known models are Bagging or Boosting.
2. Bagging allows multiple similar models with high variance are averaged to decrease variance. Boosting builds multiple incremental models to decrease the bias, while keeping variance small.

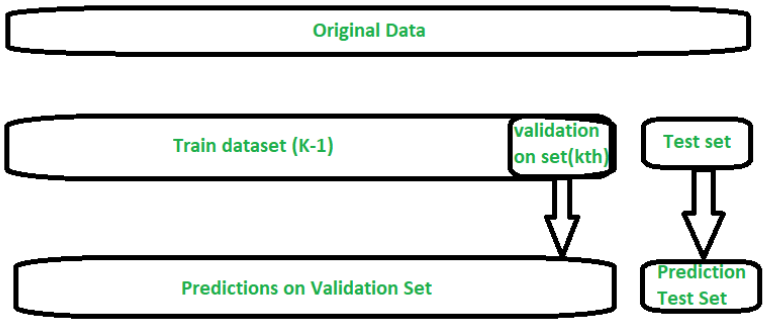
Stacking (sometimes called Stacked Generalization) is a different paradigm. The point of stacking is to explore a space of different models for the same problem. The idea is that you can attack a learning problem with different types of models which are capable to learn some part of the problem, but not the whole space of the problem. So, you can build multiple different learners and you use them to build an intermediate prediction, one prediction for each learned model. Then you add a new model which learns from the intermediate predictions the same target.

This final model is said to be stacked on the top of the others, hence the name. Thus, you might improve your overall performance, and often you end up with a model which is better than any individual intermediate model. Notice however, that it does not give you any guarantee, as is often the case with any machine learning technique**.**



How stacking works?

1. We split the training data into K-folds just like K-fold cross-validation.
2. A base model is fitted on the K-1 parts and predictions are made for Kth part.
3. We do for each part of the training data.
4. The base model is then fitted on the whole train data set to calculate its performance on the test set.
5. We repeat the last 3 steps for other base models.
6. Predictions from the train set are used as features for the second level model.
7. Second level model is used to make a prediction on the test set.



Voting Classifier:

The **Scikit-Learn Voting Classifier** combines multiple machine learning algorithms to create a prediction model, often improving overall model performance.

An ensemble learning approach combines many base models to get a more effective and precise model. The theory behind this approach is that we may decrease bias and variation while also increasing performance by combining the predictions of many models.

Voting Classifier

A voting classifier is a machine learning model that gains experience by training on a collection of several models and forecasts an output (class) based on the class with the highest likelihood of becoming the output. To forecast the output class based on the largest majority of votes, it averages the results of each classifier provided into the voting classifier. The concept is to build a single model that learns from various models and predicts output based on their aggregate majority of votes for each output class, rather than building separate specialized models and determining the accuracy for each of them.

There are primarily two different types of voting classifiers:

Hard Voting:

In **hard voting**, the predicted output class is the class with the highest majority of votes, i.e., the class with the **highest probability** of being predicted by each classifier. For example, let's say classifiers predicted the output classes as (**Cat**, **Dog**, **Dog**). As the classifiers predicted class "**Dog**" a maximum number of times, we will proceed with **Dog** as our final prediction.

Soft Voting:

In **soft voting**, the **average probabilities** of the classes determine the final prediction. For example, let's say the probabilities of the class being a **"dog"** are (0.30, 0.47, 0.53), and the probabilities for a **"cat"** are (0.20, 0.32, 0.40).

- The **average probability** for "dog" is **0.4333**.
- The **average probability** for "cat" is **0.3067**.

- From this, we can confirm the final prediction to be **dog**, as it has the **highest average probability**.