**UNIT III:**

**Ensemble Learning and Random Forests:** Introduction, Voting Classifiers, Bagging and Pasting, Random Forests, Boosting, Stacking.

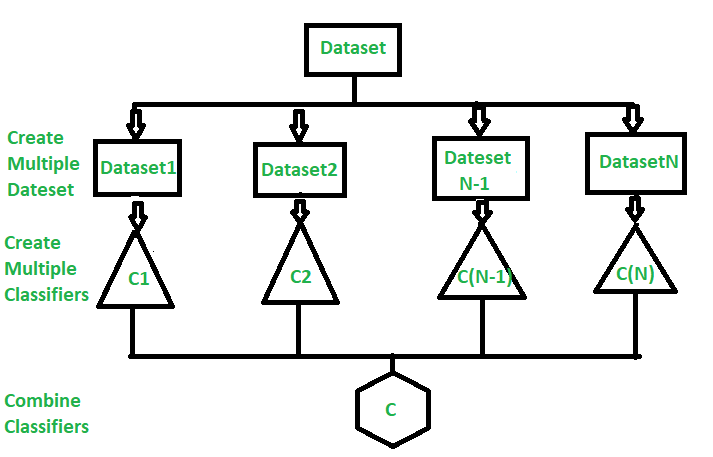
**Support Vector Machine:** Linear SVM Classification, Nonlinear SVM Classification SVM Regression, Naïve Bayes Classifiers.

--------------------------------------------------------------------------------------------------------------------------------------

1. **ENSEMBLE LEARNING:**

Ensemble learning Is a supervised machine learning technique helps enhance the overall performance of machine learning models. The concept behind it is simple. Multiple machine learning models are combined to obtain a more accurate model.

Ensemble learning combines multiple individual models to create a stronger, more accurate predictive model. By leveraging the diverse strengths of different models, ensemble learning aims to mitigate errors, enhance performance, and increase the overall robustness of predictions, leading to improved results across various tasks in machine learning and data analysis.



**How Ensemble Learning Works?**

Ensemble learning is a learning method that consists of combining multiple machine learning models.

A problem in machine learning is that individual models tend to perform poorly. In other words, they tend to have low prediction accuracy. To mitigate this problem, we combine multiple models to get one with a better performance.

The individual models that we combine are known as weak learners. We call them weak learners because they either have a high bias or high variance. Because they either have high bias or variance, weak learners cannot learn efficiently and perform poorly.

Ensemble learning will aim to reduce the variance if we have a weak model with high variance and low bias. This way, the resulting model will be much more balanced, with low bias and variance. Thus, the resulting model will be known as a strong learner. This model will be more generalized than the weak learners. It will thus be able to make accurate predictions.

1. **VOTING CLASSIFIERS:**

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**](https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/)**:** In hard voting, the predicted output class is a 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**](https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/)**:** In this, the average probabilities of the classes determine which one will be the final prediction. For example, let’s say the probabilities of the class being a “dog” is (0.30, 0.47, 0.53) and a “cat” is (0.20, 0.32, 0.40). So, the average for a class dog is 0.4333, and the cat is 0.3067, from this, we can confirm our final prediction to be a dog as it has the highest average probability.

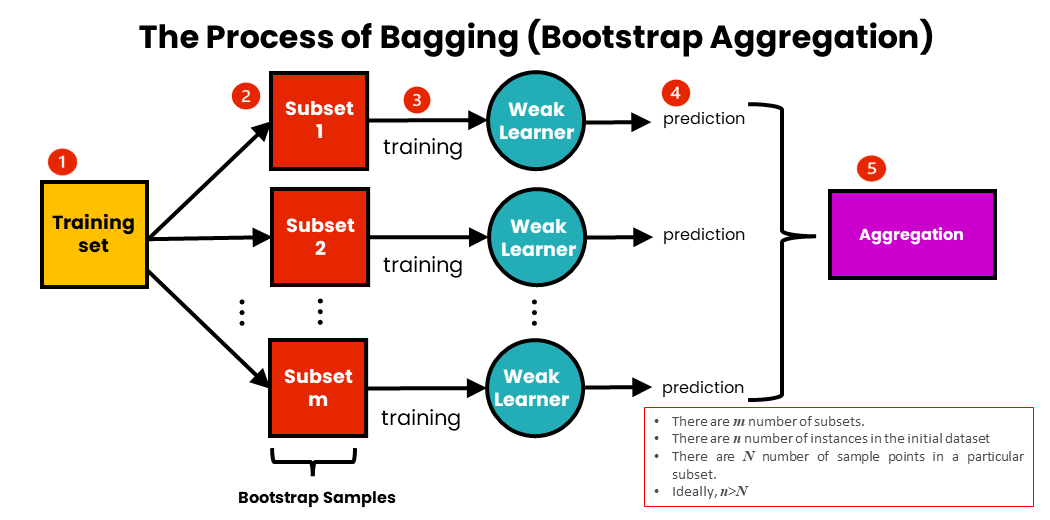
1. **BAGGING:**

Bagging (Bootstrap Aggregation) is used to reduce the variance of a decision tree. Suppose a set D of d tuples, at each iteration *i*, a training set Di of d tuples is sampled with replacement 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).

Sampling without replacement is called pasting.

**Implementation steps of Bagging –**

1. Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
2. A base model is created on each of these subsets.
3. Each model is learned in parallel from each training set and independent of each other.
4. The final predictions are determined by combining the predictions from all the models.

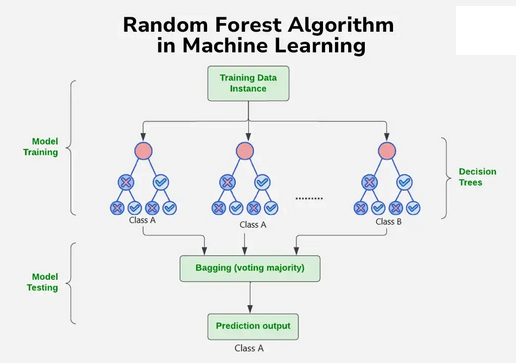


1. **RANDOM FORESTS:**

Random Forest is an extension over 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. If the number of examples in the training set is ‘N’ then take a sample of ‘n’ examples at random with replacement from the original data. This sample will be the training set for generating the tree.
2. If there are ‘M’ input variables, the ‘m’ variables are selected at random and the best split on these ‘m’ features is used to split the node. The value of ‘m’ is constant during generation of various trees in the forest.
3. Each tree is grown to the largest extent possible.
4. The final prediction is given based on the aggregation of predictions from all the trees.



**Strengths of Random Forest:**

1. It takes less training time compared to other algorithms.
2. It predicts the output with high accuracy even for large datasets.
3. It can also maintain accuracy when a large portion of data is missing or contains outliers.

**Weakness of Random Forest:**

1. When use for regression they cannot predict beyond the range in training data.
2. Size of the model is very large and requires hundreds of megabytes of memoryto store.
3. **BOOSTING:**

**Boosting** is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

**Steps in training the Boosting Model:**

1. Initialize 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 weights of the wrongly predicted data points.
4. If got the required results, go to step 5 else go to step 2.
5. End.

****

**Explanation for training the boosting model:**

1. **B1** consists of 10 data points which consist of two types namely plus(+) and minus(-) and 5 of which are plus(+) and the other 5 are minus(-) and each one has been assigned equal weight initially. The first model tries to classify the data points and generates a vertical separator line but it wrongly classifies 3 plus(+) as minus(-).
2. **B2** consists of the 10 data points from the previous model in which the 3 wrongly classified plus(+) are weighted more so that the current model tries more to classify these pluses(+) correctly. This model generates a vertical separator line that correctly classifies the previously wrongly classified pluses(+) but in this attempt, it wrongly classifies three minuses(-).
3. **B3** consists of the 10 data points from the previous model in which the 3 wrongly classified minus(-) are weighted more so that the current model tries more to classify these minuses(-) correctly. This model generates a horizontal separator line that correctly classifies the previously wrongly classified minuses(-).
4. **B4** combines together B1, B2, and B3 in order to build a strong prediction model which is much better than any individual model used.

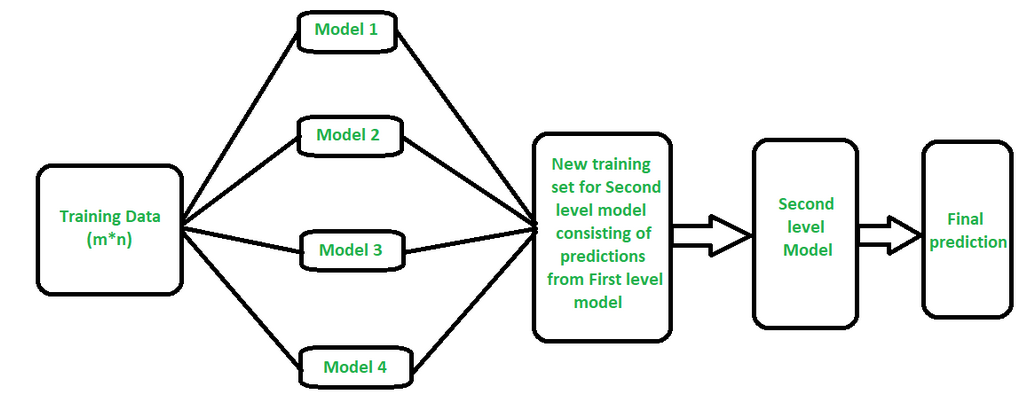
**Advantages of Boosting**

* **Improved Accuracy** – Boosting can improve the accuracy of the model by combining several weak models’ accuracies and averaging them for regression or voting over them for classification to increase the accuracy of the final model.
* **Robustness to Overfitting** – Boosting can reduce the risk of overfitting by reweighting the inputs that are classified wrongly.
* **Better handling of imbalanced data** – Boosting can handle the imbalance data by focusing more on the data points that are misclassified
* **Better Interpretability** – Boosting can increase the interpretability of the model by breaking the model decision process into multiple processes.

**Types Of Boosting Algorithms**

1. **Gradient Boosting –** It is a boosting technique that builds a final model from the sum of several weak learning algorithms that were trained on the same dataset. It operates on the idea of stagewise addition. The first weak learner in the gradient boosting algorithm will not be trained on the dataset; instead, it will simply return the mean of the relevant column. The residual for the first weak learner algorithm’s output will then be calculated and used as the output column or target column for the next weak learning algorithm that will be trained. The second weak learner will be trained using the same methodology, and the residuals will be computed and utilized as an output column once more for the third weak learner, and so on until we achieve zero residuals. The dataset for gradient boosting must be in the form of numerical or categorical data, and the loss function used to generate the residuals must be differential at all times.
2. **Adaboost** – AdaBoost is a boosting algorithm that also works on the principle of the stagewise addition method where multiple weak learners are used for getting strong learners. The value of the alpha parameter, in this case, will be indirectly proportional to the error of the weak learner.
3. **STACKING:**

Stacking explores a space of different models for the same problem. The idea is that we 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, we can build multiple different learners and use them to build an intermediate prediction, one prediction for each learned model. Then 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, overall performance is improved.



**Working of Stacking:**

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.

**Advantages of Stacking:**

1. **Improved Predictive Performance:** Stacking can reduce bias and variance in the final forecast by merging the results of numerous base models, resulting in improved predictive performance.
2. **Model Diversity:** Stacking promotes the usage of varied base models that may be trained using a variety of algorithms, architectures, and hyperparameter settings. This diversity can serve to lessen the danger of overfitting while also making the stacked ensemble more robust to different types of data.
3. **Flexibility:** Stacking is a versatile strategy that may be used to solve a variety of machine learning issues, including classification, regression, and even time series forecasting.
4. **Interpretability:** Stacking can also reveal the significance of many base models and their forecasts for the final prediction.

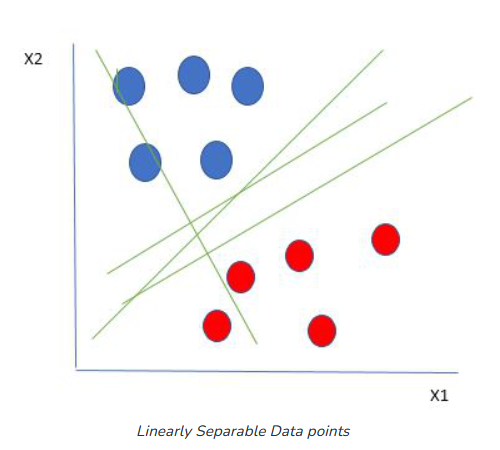
**SUPPORT VECTOR MACHINE**

1. **LINEAR SVM CLASSIFICATION:**

Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.

The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

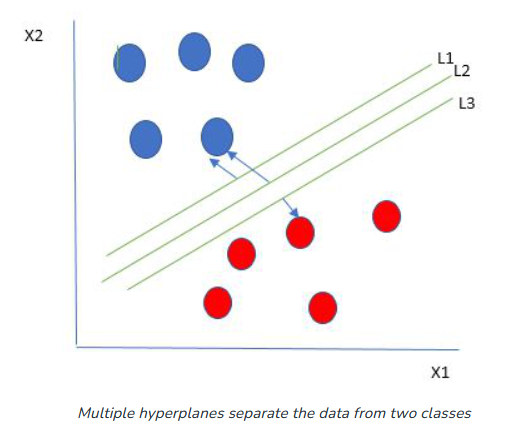
Consider two independent variables x1, x2, and one dependent variable which is either a blue circle or a red circle.

****

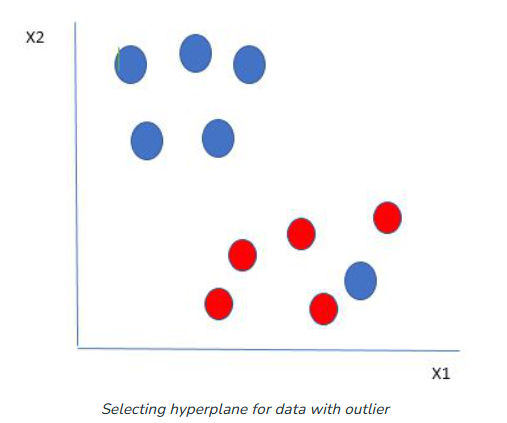
From the figure above it’s very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregate our data points or do a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points?

**Working of SVM:**

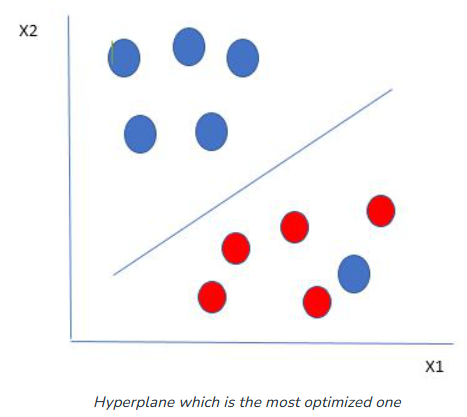
One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.

****

So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the **maximum-margin hyperplane/hard margin**. So from the above figure, we choose L2. Let’s consider a scenario like shown below:

****

Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It’s simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.

****

So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called **soft margins**. When there is a soft margin to the data set, the SVM tries to minimize *(1/margin+∧(∑penalty))*. Hinge loss is a commonly used penalty. If no violations no hinge loss. If violations hinge loss proportional to the distance of violation.

1. **NONLINEAR SVM CLASSIFICATION**

Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

What to do if data are not linearly separable?



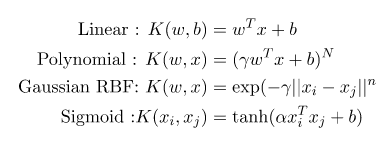
Say, our data is shown in the figure above. SVM solves this by creating a new variable using a **kernel**. We call a point xi on the line and we create a new variable yi as a function of distance from origin o.so if we plot this we get something like as shown below



In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as a kernel.

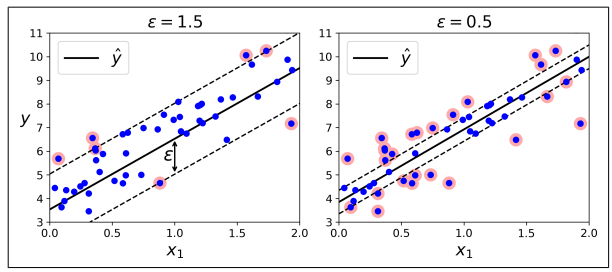
**Popular kernel functions in SVM**

The SVM kernel is a function that takes low-dimensional input space and transforms it into higher-dimensional space, i.e., it converts non separable problems to separable problems. It is mostly useful in non-linear separation problems. Simply put the kernel, does some extremely complex data transformations and then finds out the process to separate the data based on the labels or outputs defined.

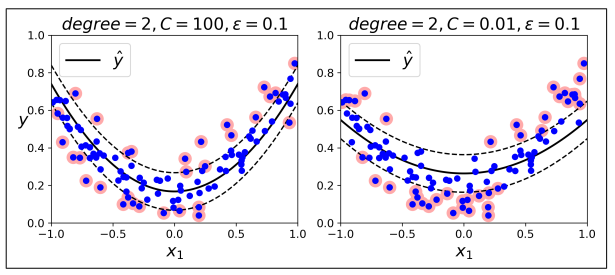


1. **SVM REGRESSION:**

SVM not only supports linear and nonlinear classification, but it also supports linear and nonlinear regression. The trick is to reverse the objective: instead of trying to fit the largest possible street between two classes while limiting margin violations, SVM Regression tries to fit as many instances as possible on the street while limiting margin violations. The width of the street is controlled by a hyperparameter ϵ. The below figure shows two linear SVM Regression models trained on some random linear data, one with a large margin (ϵ = 1.5) and the other with a small margin (ϵ = 0.5).

****

To handle nonlinear regression tasks, a kernelized SVM model is used. For example, Figure below shows SVM Regression on a random quadratic training set, using a 2nd-degree polynomial kernel. There is little regularization on the left plot (i.e., a large C value), and much more regularization on the right plot (i.e., a small C value).

****

1. **NAÏVE BAYES CLASSIFIERS: Refer UNIT-2**