Unit -3

**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.

VOTING CLASSIFIERS

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.  
It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

Voting Classifier supports two types of votings.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction

.

1. **Soft Voting**: In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.

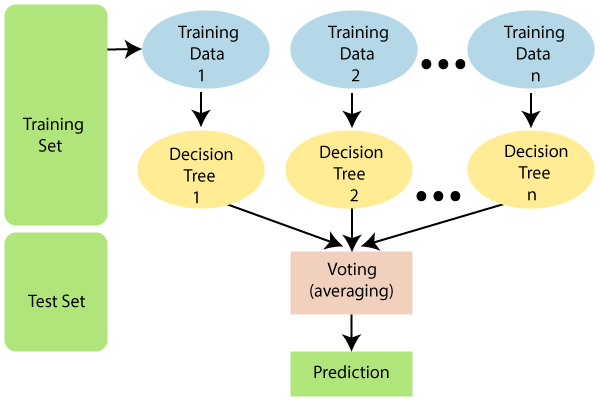
Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the***

***predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The below diagram explains the working of the Random Forest algorithm:



USES OF RANDOM FOREST :

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.

## How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

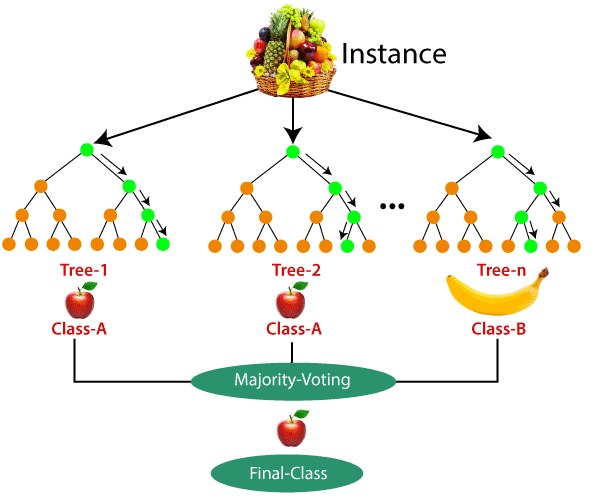
**Step-4:** Repeat Step 1 & 2.

**Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

**Example:** Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision.

Consider the below image:



## Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

## Advantages of Random Forest

* Random Forest is capable of performing both Classification and Regression tasks.
* It is capable of handling large datasets with high dimensionality.
* It enhances the accuracy of the model and prevents the overfitting issue.

## Disadvantages of Random Forest

* Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

# Stacking in Machine Learning

There are many ways to ensemble models in machine learning, such as Bagging, Boosting, and stacking. ***Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance.*** Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance.

ENSEMBLE LEARNING

Ensemble learning is one of the most powerful machine learning techniques that use the combined output of two or more models/weak learners and solve a particular computational intelligence problem. E.g., a Random Forest algorithm is an ensemble of various decision trees combined.

***"An ensembled model is a machine learning model that combines the predictions from two or more models.”***

There are 3 most common ensemble learning methods in machine learning. These are as follows:

* Bagging
* Boosting
* Stacking

However, we will mainly discuss Stacking on this topic.

### 1. Bagging

Bagging is a method of ensemble modeling, which is primarily used to solve supervised machine learning problems. It is generally completed in two steps as follows:

* **Bootstrapping:** It is a random sampling method that is used to derive samples from the data using the replacement procedure. In this method, first, random data samples are fed to the primary model, and then a base learning algorithm is run on the samples to complete the learning process.
* **Aggregation:** This is a step that involves the process of combining the output of all base models and, based on their output, predicting an aggregate result with greater accuracy and reduced variance.

**Example:** In the Random Forest method, predictions from multiple decision trees are ensembled parallelly. Further, in regression problems, we use an average of these predictions to get the final output, whereas, in classification problems, the model is selected as the predicted class.

### 2. Boosting

Boosting is an ensemble method that enables each member to learn from the preceding member's mistakes and make better predictions for the future. Unlike the bagging method, in boosting, all base learners (weak) are arranged in a sequential format so that they can learn from the mistakes of their preceding learner. Hence, in this way, all weak learners get turned into strong learners and make a better predictive model with significantly improved performance.

We have a basic understanding of ensemble techniques in machine learning and their two common methods, i.e., bagging and boosting. Now, let's discuss a different paradigm of ensemble learning, i.e., Stacking.

DIFFERENCE

| **Bagging** | **Boosting** |
| --- | --- |
| Various training data subsets are randomly drawn with replacement from the whole training dataset. | Each new subset contains the components that were misclassified by previous models. |
| Bagging attempts to tackle the over-fitting issue. | Boosting tries to reduce bias. |
| If the classifier is unstable (high variance), then we need to apply bagging. | If the classifier is steady and straightforward (high bias), then we need to apply boosting. |
| Every model receives an equal weight. | Models are weighted by their performance. |
| Objective to decrease variance, not bias. | Objective to decrease bias, not variance. |
| It is the easiest way of connecting predictions that belong to the same type. | It is a way of connecting predictions that belong to the different types. |
| Every model is constructed independently. | New models are affected by the performance of the previously developed model. |

### 3. Stacking

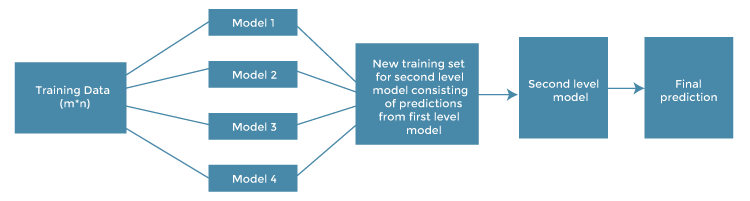
***Stacking is one of the popular ensemble modeling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them with Meta learners, we can predict better predictions for the future.***

In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

Stacking is also known as **a stacked generalization** and is an extended form of the Model Averaging Ensemble technique in which all sub-models equally participate as per their performance weights and build a new model with better predictions. This new model is stacked up on top of the others; this is the reason why it is named stacking.

## Architecture of Stacking

The architecture of the stacking model is designed in such as way that it consists of two or more base/learner's models and a meta-model that combines the predictions of the base models. These base models are called level 0 models, and the meta-model is known as the level 1 model. So, the Stacking ensemble method includes **original (training) data, primary level models, primary level prediction, secondary level model, and final prediction**. The basic architecture of stacking can be represented as shown below the image.



* **Original data:** This data is divided into n-folds and is also considered test data or training data.
* **Base models:** These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level-0) as an output.
* **Level-0 Predictions:** Each base model is triggered on some training data and provides different predictions, which are known as **level-0 predictions.**
* **Meta Model:** The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the **level-1 model**.
* **Level-1 Prediction:** The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models, i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

## Steps to implement Stacking models:

There are some important steps to implementing stacking models in machine learning.

These are as follows:

* Split training data sets into n-folds using the **RepeatedStratifiedKFold** as this is the most common approach to preparing training datasets for meta-models.
* Now the base model is fitted with the first fold, which is n-1, and it will make predictions for the nth folds.
* The prediction made in the above step is added to the x1\_train list.
* Repeat steps 2 & 3 for remaining n-1folds, so it will give x1\_train array of size n,
* Now, the model is trained on all the n parts, which will make predictions for the sample data.
* Add this prediction to the y1\_test list.
* In the same way, we can find x2\_train, y2\_test, x3\_train, and y3\_test by using Model 2 and 3 for training, respectively, to get Level 2 predictions.
* Now train the Meta model on level 1 prediction, where these predictions will be used as features for the model.
* Finally, Meta learners can now be used to make a prediction on test data in the stacking model.

**pasting**

Bagging is to use the same training for every predictor, but to train them on different random subsets of the training set. When sampling is performed with replacement, this method is called bagging (short for bootstrap aggregating). When sampling is performed without replacement, it is called pasting.

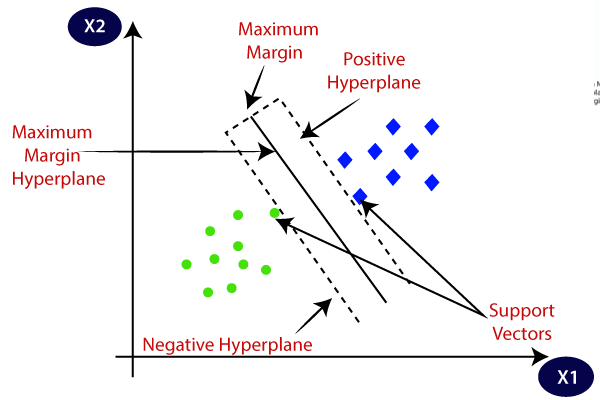
\ **Support Vector Machine:** Linear SVM Classification, Nonlinear SVM Classification SVM Regression, Naïve Bayes Classifiers.

Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



## Types of SVM

**SVM can be of two types:**

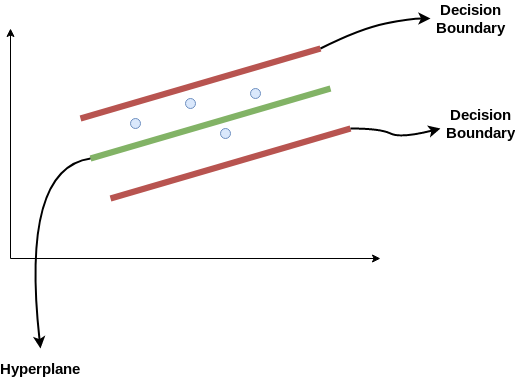
* **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

## Introduction to Support Vector Regression (SVR)

Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems. Let’s spend a few minutes understanding the idea behind SVR.

### The Idea Behind Support Vector Regression

The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample. So let’s now dive deep and understand how SVR works actually.



hyperplane. **Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line.** Our best fit line is the hyperplane that has a maximum number of points.

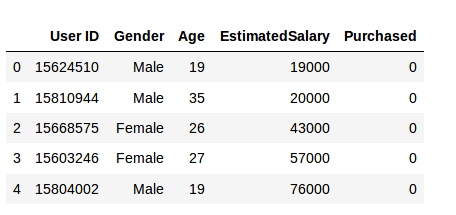
The first thing that we’ll understand is what is the decision boundary (the danger red line above!). Consider these lines as being at any distance, say ‘a’, from the hyperplane. So, these are the lines that we draw at distance ‘+a’ and ‘-a’ from the hyperplane. This ‘a’ in the text is basically referred to as epsilon.

.Our main aim here is to decide a decision boundary at ‘a’ distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line.

Hence, we are going to take only those points that are within the decision boundary and have the least error rate, or are within the Margin of Tolerance. This gives us a better fitting model.

## Implementing Support Vector Regression (SVR) in Python

Time to put on our coding hats! In this section, we’ll understand the use of Support Vector Regression with the help of a dataset. Here, we have to predict the salary of an employee given a few independent variables. A classic HR analytics project!



**Step 1: Importing the libraries**

|  | import numpy as np |
| --- | --- |
|  | import matplotlib.pyplot as plt |
|  | import pandas as pd |

[**view raw**](https://gist.github.com/Alakhator/5ea0601f67f2ecda0f504042c66985f2/raw/eed4547a6f04c23053064c93c5b7b74d0191d0d9/svr1.py)[**svr1.py**](https://gist.github.com/Alakhator/5ea0601f67f2ecda0f504042c66985f2#file-svr1-py)hosted with  by [**GitHub**](https://github.com/)

**Step 2: Reading the dataset**

|  | dataset = pd.read\_csv('Position\_Salaries.csv') |
| --- | --- |
|  | X = dataset.iloc[:, 1:2].values |
|  | y = dataset.iloc[:, 2].values |

[**view raw**](https://gist.github.com/Alakhator/a56458b4cd70a743f6202ef7d475582f/raw/c711116c5e4e4950283bd18d27e167f2fc16a849/sv2.py)[**sv2.py**](https://gist.github.com/Alakhator/a56458b4cd70a743f6202ef7d475582f#file-sv2-py)hosted with  by [**GitHub**](https://github.com/)

**Step 3: Feature Scaling**

A real-world dataset contains features that vary in magnitudes, units, and range. I would suggest performing normalization when the scale of a feature is irrelevant or misleading.

Feature Scaling basically helps to normalize the data within a particular range. Normally several common class types contain the feature scaling function so that they make feature scaling automatically. However, the SVR class is not a commonly used class type so we should perform feature scaling using Python.

|  | from sklearn.preprocessing import StandardScaler |
| --- | --- |
|  | sc\_X = StandardScaler() |
|  | sc\_y = StandardScaler() |
|  | X = sc\_X.fit\_transform(X) |
|  | y = sc\_y.fit\_transform(y) |

[**view raw**](https://gist.github.com/Alakhator/62fe57be04934a13e2b16994fbd7c80f/raw/b77c4854033c5648d5429a73a5d05d9cfc35498d/svr3.py)[**svr3.py**](https://gist.github.com/Alakhator/62fe57be04934a13e2b16994fbd7c80f#file-svr3-py)hosted with  by [**GitHub**](https://github.com/)

**Step 4: Fitting SVR to the dataset**

|  | from sklearn.svm import SVR |
| --- | --- |
|  | regressor = SVR(kernel = 'rbf') |
|  | regressor.fit(X, y) |

[**view raw**](https://gist.github.com/Alakhator/52735c2deda2e508788041c1ad5e60cf/raw/8528b12178591b26144c9040dc179cd74042fa1f/svr4.py)[**svr4.py**](https://gist.github.com/Alakhator/52735c2deda2e508788041c1ad5e60cf#file-svr4-py)hosted with  by [**GitHub**](https://github.com/)

Kernel is the most important feature. There are many types of kernels – linear, Gaussian, etc. Each is used depending on the dataset. To learn more about this, read this: [Support Vector Machine (SVM) in Python and R](https://courses.analyticsvidhya.com/courses/support-vector-machine-svm-in-python-and-r?utm_source=blog&utm_medium=support-vector-regression-tutorial-for-machine-learning)

**Step 5. Predicting a new result**

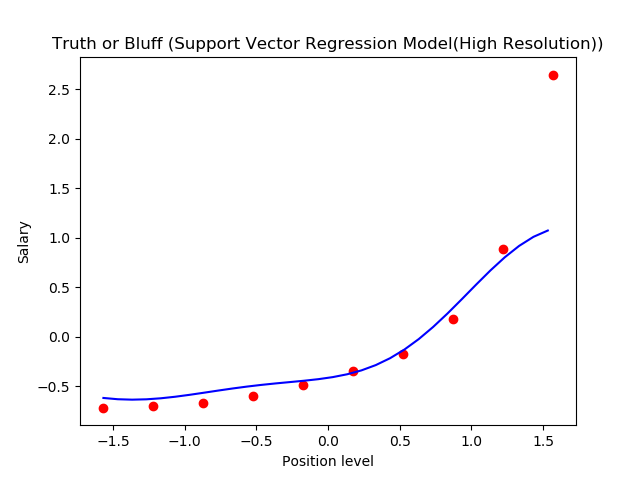
|  | y\_pred = regressor.predict(6.5) |
| --- | --- |
|  | y\_pred = sc\_y.inverse\_transform(y\_pred) |

[**view raw**](https://gist.github.com/Alakhator/a31ea5396d941e702e54623f11e85078/raw/93f3cc45beb039c64d52c3b52ffd0f6b9a22339e/svr5.py)[**svr5.py**](https://gist.github.com/Alakhator/a31ea5396d941e702e54623f11e85078#file-svr5-py)hosted with  by [**GitHub**](https://github.com/)

So, the prediction for y\_pred(6, 5) will be 170,370.

**Step 6.** **Visualizing the SVR results (for higher resolution and smoother curve)**

|  | X\_grid = np.arange(min(X), max(X), 0.01) #this step required because data is feature scaled. |
| --- | --- |
|  | X\_grid = X\_grid.reshape((len(X\_grid), 1)) |
|  | plt.scatter(X, y, color = 'red') |
|  | plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue') |
|  | plt.title('Truth or Bluff (SVR)') |
|  | plt.xlabel('Position level') |
|  | plt.ylabel('Salary') |
|  | plt.show() |



This is what we get as output- the best fit line that has a maximum number of points. Quite accurate!

Naïve Bayes Classifier Algorithm

* Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
* It is mainly used in *text classification* that includes a high-dimensional training dataset.
* Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
* **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
* Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

## Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

* **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
* **Bayes**: It is called Bayes because it depends on the principle of [Bayes' Theorem](https://www.javatpoint.com/bayes-theorem-in-artifical-intelligence)

.

## Bayes' Theorem:

* Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
* The formula for Bayes' theorem is given as:

Naïve Bayes Classifier Algorithm

**Where,**

**P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.

**P(B|A) is Likelihood probability**: Probability of the evidence given that the probability of a hypothesis is true.

**P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.

**P(B) is Marginal Probability**: Probability of Evidence.

### Advantages of Naïve Bayes Classifier:

* Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
* It can be used for Binary as well as Multi-class Classifications.
* It performs well in Multi-class predictions as compared to the other Algorithms.
* It is the most popular choice for **text classification problems**.

### Disadvantages of Naïve Bayes Classifier:

* Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

### Applications of Naïve Bayes Classifier:

* It is used for **Credit Scoring**.
* It is used in **medical data classification**.
* It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
* It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

## Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

* **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
* **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.  
  The classifier uses the frequency of words for the predictors.
* **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

## Python Implementation of the Naïve Bayes algorithm:

Now we will implement a Naive Bayes Algorithm using Python. So for this, we will use the "**user\_data**" **dataset**, which we have used in our other classification model. Therefore we can easily compare the Naive Bayes model with the other models.

### Steps to implement:

* Data Pre-processing step
* Fitting Naive Bayes to the Training set
* Predicting the test result
* Test accuracy of the result(Creation of Confusion matrix)
* Visualizing the test set result

### 1) Data Pre-processing step:

In this step, we will pre-process/prepare the data so that we can use it efficiently in our code. It is similar as we did in [data-pre-processing](https://www.javatpoint.com/data-preprocessing-machine-learning)

### 2) Fitting Naive Bayes to the Training Set:

After the pre-processing step, now we will fit the Naive Bayes model to the Training set. Below is the code for it:

### 3) Prediction of the test set result:

Now we will predict the test set result. For this, we will create a new predictor variable **y\_pred**, and will use the predict function to make the predictions

### 4) Creating Confusion Matrix:

Now we will check the accuracy of the Naive Bayes classifier using the Confusion matrix. Below is the code for it:

### 5) Visualizing the training set result:

Next we will visualize the training set result using Naïve Bayes Classifier. Below is the code for it: