Naïve Bayes Classifier Algorithm

* Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem(Probability of hypothesis)** 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" \t "_blank)

.

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.

33.5K**P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.

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

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

**Problem**: If the weather is sunny, then the Player should play or not?

**Solution**: To solve this, first consider the below dataset:

|  |  |  |
| --- | --- | --- |
|  | **Outlook** | **Play** |
| **0** | Rainy | Yes |
| **1** | Sunny | Yes |
| **2** | Overcast | Yes |
| **3** | Overcast | Yes |
| **4** | Sunny | No |
| **5** | Rainy | Yes |
| **6** | Sunny | Yes |
| **7** | Overcast | Yes |
| **8** | Rainy | No |
| **9** | Sunny | No |
| **10** | Sunny | Yes |
| **11** | Rainy | No |
| **12** | Overcast | Yes |
| **13** | Overcast | Yes |

**Frequency table for the Weather Conditions:**

|  |  |  |
| --- | --- | --- |
| Weather | Yes | No |
| Overcast | 5 | 0 |
| Rainy | 2 | 2 |
| Sunny | 3 | 2 |
| Total | 10 | 5 |

**Likelihood table weather condition:**

|  |  |  |  |
| --- | --- | --- | --- |
| Weather | No | Yes |  |
| Overcast | 0 | 5 | 5/14= 0.35 |
| Rainy | 2 | 2 | 4/14=0.29 |
| Sunny | 2 | 3 | 5/14=0.35 |
| All | 4/14=0.29 | 10/14=0.71 |  |

**Applying Bayes'theorem:**

**P(Yes|Sunny)= P(Sunny|Yes)\*P(Yes)/P(Sunny)**

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3\*0.71/0.35= **0.60**

**P(No|Sunny)= P(Sunny|No)\*P(No)/P(Sunny)**

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5\*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

**Hence on a Sunny day, Player can play the game.**

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:

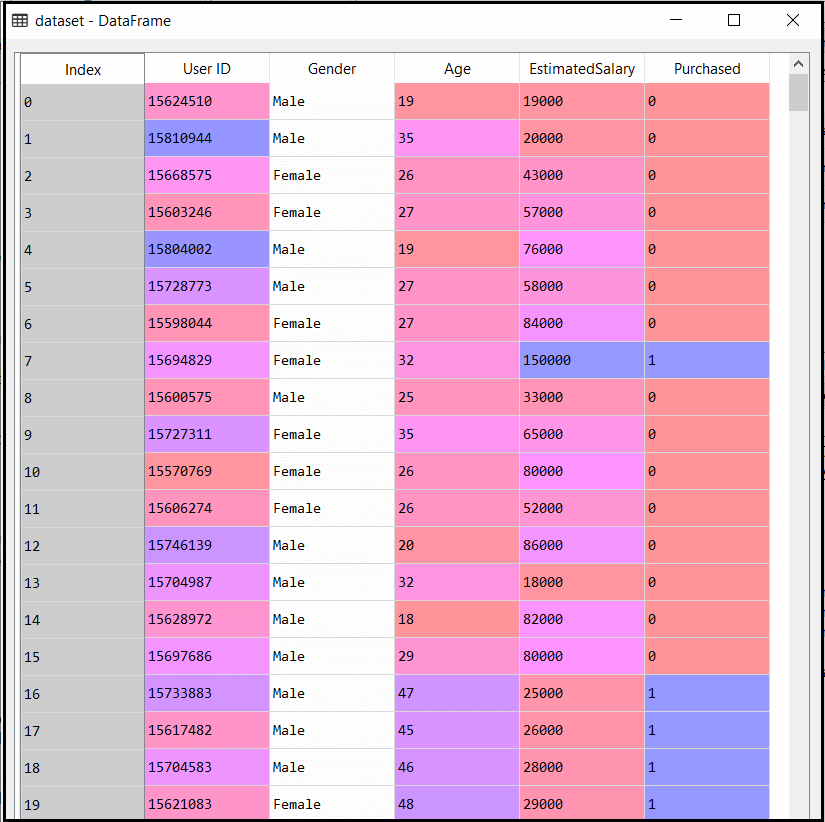
In this step, we will pre-process/prepare the data so that we can use it efficiently in our code. It is similar as we did in [data-pre-processing](https://www.javatpoint.com/data-preprocessing-machine-learning" \t "_blank)

. The code for this is given below:

1. Importing the libraries
2. **import** numpy as nm
3. **import** matplotlib.pyplot as mtp
4. **import** pandas as pd
6. # Importing the dataset
7. dataset = pd.read\_csv('user\_data.csv')
8. x = dataset.iloc[:, [2, 3]].values
9. y = dataset.iloc[:, 4].values
11. # Splitting the dataset into the Training set and Test set
12. **from** sklearn.model\_selection **import** train\_test\_split
13. x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.25, random\_state = 0)
15. # Feature Scaling
16. **from** sklearn.preprocessing **import** StandardScaler
17. sc = StandardScaler()
18. x\_train = sc.fit\_transform(x\_train)
19. x\_test = sc.transform(x\_test)

In the above code, we have loaded the dataset into our program using "**dataset = pd.read\_csv('user\_data.csv')**. The loaded dataset is divided into training and test set, and then we have scaled the feature variable.

The output for the dataset is given as:



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:

1. # Fitting Naive Bayes to the Training set
2. **from** sklearn.naive\_bayes **import** GaussianNB
3. classifier = GaussianNB()
4. classifier.fit(x\_train, y\_train)

In the above code, we have used the **GaussianNB classifier** to fit it to the training dataset. We can also use other classifiers as per our requirement.

**Output:**

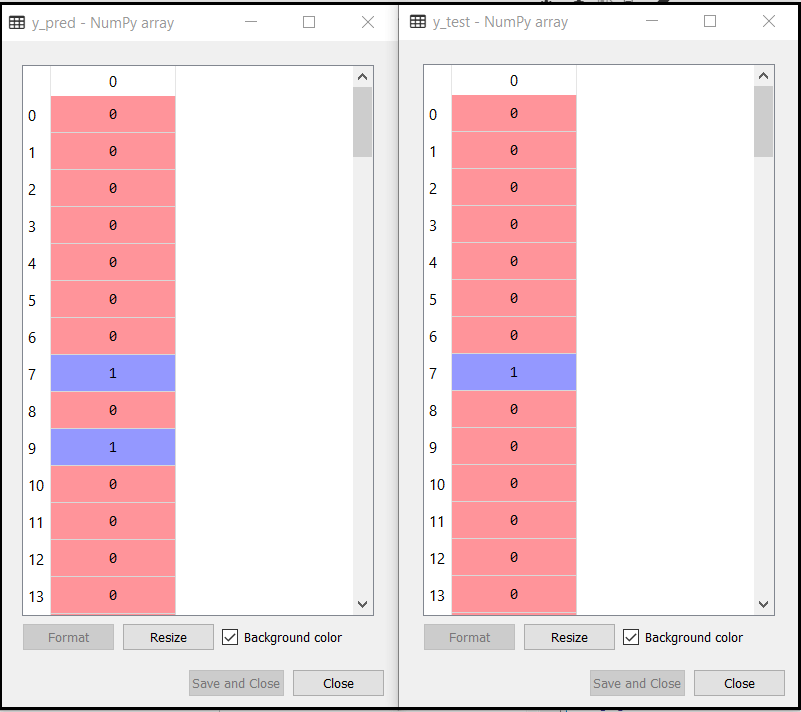
Out[6]: GaussianNB(priors=None, var\_smoothing=1e-09)

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.

1. # Predicting the Test set results
2. y\_pred = classifier.predict(x\_test)

**Output:**



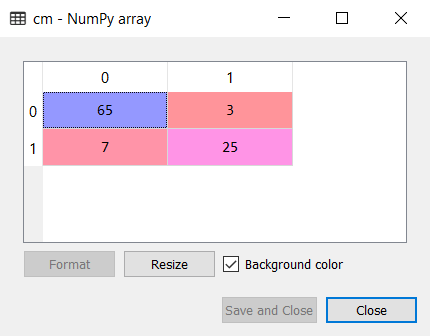
The above output shows the result for prediction vector **y\_pred** and real vector y\_test. We can see that some predications are different from the real values, which are the incorrect predictions.

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:

1. # Making the Confusion Matrix
2. **from** sklearn.metrics **import** confusion\_matrix
3. cm = confusion\_matrix(y\_test, y\_pred)

**Output:**



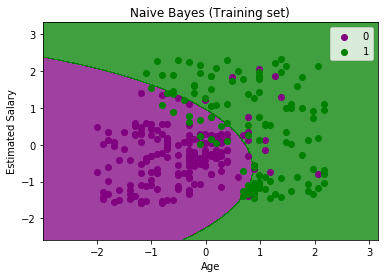
As we can see in the above confusion matrix output, there are 7+3= 10 incorrect predictions, and 65+25=90 correct predictions.

5) Visualizing the training set result:

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

1. # Visualising the Training set results
2. **from** matplotlib.colors **import** ListedColormap
3. x\_set, y\_set = x\_train, y\_train
4. X1, X2 = nm.meshgrid(nm.arange(start = x\_set[:, 0].min() - 1, stop = x\_set[:, 0].max() + 1, step = 0.01),
5. nm.arange(start = x\_set[:, 1].min() - 1, stop = x\_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
7. alpha = 0.75, cmap = ListedColormap(('purple', 'green')))
8. mtp.xlim(X1.min(), X1.max())
9. mtp.ylim(X2.min(), X2.max())
10. **for** i, j **in** enumerate(nm.unique(y\_set)):
11. mtp.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],
12. c = ListedColormap(('purple', 'green'))(i), label = j)
13. mtp.title('Naive Bayes (Training set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

**Output:**

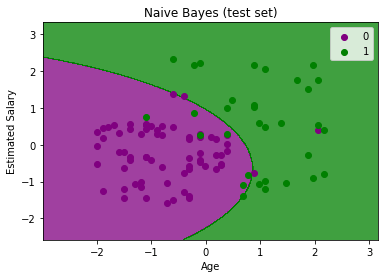


In the above output we can see that the Naïve Bayes classifier has segregated the data points with the fine boundary. It is Gaussian curve as we have used **GaussianNB** classifier in our code.

6) Visualizing the Test set result:

1. # Visualising the Test set results
2. **from** matplotlib.colors **import** ListedColormap
3. x\_set, y\_set = x\_test, y\_test
4. X1, X2 = nm.meshgrid(nm.arange(start = x\_set[:, 0].min() - 1, stop = x\_set[:, 0].max() + 1, step = 0.01),
5. nm.arange(start = x\_set[:, 1].min() - 1, stop = x\_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
7. alpha = 0.75, cmap = ListedColormap(('purple', 'green')))
8. mtp.xlim(X1.min(), X1.max())
9. mtp.ylim(X2.min(), X2.max())
10. **for** i, j **in** enumerate(nm.unique(y\_set)):
11. mtp.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],
12. c = ListedColormap(('purple', 'green'))(i), label = j)
13. mtp.title('Naive Bayes (test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

**Output:**



The above output is final output for test set data. As we can see the classifier has created a Gaussian curve to divide the "purchased" and "not purchased" variables. There are some wrong predictions which we have calculated in Confusion matrix. But still it is pretty good classifier.