

# Today i am going to learn about Naive Bayes Classifiers

references: <https://youtu.be/mlumJPFvooQ?si=ibvZ3TOr-wMWsTrX> -IP1 introduction

<https://www.youtube.com/watch?v=jS1CKhALUBQ> -Naive bayes intro

<https://www.youtube.com/watch?v=jS1CKhALUBQ> -Real time example on text dataset

<https://www.youtube.com/watch?v=O2L2Uv9pdDA> -Real time example video

## 1.What is naive bayes theorem

$$P(A/B) = (p(B/A) * p(A)) / p(B)$$

B-independent features( $x_1, x_2, x_3, \dots, x_n$ )

A-dependent features( $y$ )

## 2.Mathematical intuition

Y - take different type of values.ex-true or false.

we will find the probability of true and false with respect to given set of independent features.

we will choose the result value based on the higher probability value

## 3.Real time example (binary classification)

refer :4.<https://www.geeksforgeeks.org/naive-bayes-classifiers/>

# Outlook Temperature Play Golf yes\_for\_sunny No\_for\_sunny yes\_for\_rainy no\_for\_rainy yes\_frcast no\_cast 0  
Rainy Hot No 3 2 2 3 4 1 Rainy Hot No 2 Overcast Hot Yes 3 Sunny Mild Yes 4 Sunny Cool Yes 5 Sunny Cool No  
6 Overcast Cool Yes 7 Rainy Mild No 8 Rainy Cool Yes 9 Sunny Mild Yes 10 Rainy Mild Yes 11 Overcast Mild Yes  
12 Overcast Hot Yes 13 Sunny Mild No

steps: 1.frequency table (we need to find how many output class is available for each independent class )

2.Conditional probability table 3.we l find probability of each class with respect to given class

outlook yes NO  $p(y)$   $p(n)$  Rainy 3 2 3/9 3/5 Overcast 4 0 4/9 0/5 sunny 2 3 2/9 =sunny/yes) 3/5=p(sunny/no)  
total=3 ,ty=9 ,tn=5

Temperature yes no  $p(y)$   $p(n)$  HOt 2 2 2/9( $p(\text{hot/yes})$  2/5 = $p(\text{hot/no})$  mild 4 2 4/9 2/5 cool 3 1 3/9 1/9 ty=9,  
tn=5

play prob yes avg  $9/14 = (p(\text{yes}))$  NO avg =  $5/14$

```
In [ ]: # so, we can find the probability table ,
```

we need to take both dependent category, here yes and no, so, we need to find how much probability of yes based on the given independent features and probability of no based on independent features.  $p(\text{yes}/\text{today}) = p(\text{sunny}/\text{yes}) * p(\text{hot}/\text{yes}) * p(\text{yes}) / p(\text{today})$  today = independent features Since,  $P(\text{today})$  is common in both probabilities, we can ignore  $P(\text{today})$  and find proportional probabilities as: so, here  $p(\text{yes}/\text{today}) = (2/9 * 2/9 * 9/14) = 0.03174$  and  $p(\text{no}/\text{today}) = p(\text{sunny}/\text{no}) * p(\text{hot}/\text{no}) * p(\text{no}) = 3/5 * 2/5 * 5/14 = 0.0857$  we need to normalize these values we can take any one,  $p(\text{yes}/\text{today}) = 0.031 / (0.031 + 0.0857) = 0.2656383890317052$   $p(\text{no}/\text{today}) = 0.0857 / (0.031 + 0.0857) = 0.7343616109682948$  so, prob 0.73 for no prob 0.26 for yes so,  $0.73 > 0.26$ , hence we choose no as a result

```
In [11]: p_yes = (2/9 * 2/9 * 9/14)

# normalization
p_yes = 0.031 / (0.031 + 0.0857)
p_yes
```

```
Out[11]: 0.2656383890317052
```

```
In [13]: p_no = 3/5 * 2/5 * 5/14
p_no = 0.0857 / (0.031 + 0.0857)
p_no
```

```
Out[13]: 0.7343616109682948
```

Refer this link for multiclass classification-  
<https://youtu.be/YeD-Ntq96Lo?si=Kw-YVky3utWLD9tZ> (Multiclass classification)

```
In [ ]:
```

## 4. Real time implementation (code implementation)

Refer this link -  
<https://www.youtube.com/watch?v=nHIUYwN-5rM>

```
In [14]: #Spam mail detector
```

## Types of bayes classifier

### 1. Gaussian naive bayes

## 2. Bernoulli naive bayes

## 3. Multinomial naive bayes

## 4. complement naive bayes

Yes, in Naive Bayes classification, there are generally four common methods:

### 1. Gaussian Naive Bayes:

- Assumes that the features follow a normal distribution. It is suitable for continuous data.

### 2. Multinomial Naive Bayes:

- Appropriate for discrete data, such as text data represented as word frequency counts. It is commonly used in natural language processing (NLP) tasks.

### 3. Bernoulli Naive Bayes:

- Designed for binary or Boolean features. It's often used in text classification tasks where the presence or absence of a particular feature is relevant.

### 4. Complement Naive Bayes:

- A variation of Multinomial Naive Bayes that is particularly useful for imbalanced datasets. It adjusts for imbalances by using the complement of each class's probability.

Each method has its own assumptions about the distribution of the data and is suitable for different types of features. The choice of which method to use depends on the nature of your data and the problem you are trying to solve.

In [1]:

In [3]: `pip install scikit-learn`

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn in c:\programdata\anaconda3\lib\site-packages (1.3.0)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:

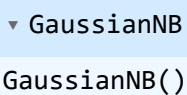
```
#ex. Gaussian naive bayes
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

```
In [42]: # Load data
X,y=load_iris(return_X_y=True)

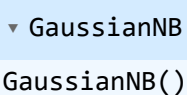
#split data
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.3,random_state=1)
X_train.shape
```

Out[42]: (105, 4)

```
In [29]: #call the model
model=GaussianNB()
model
#fit model
```

Out[29]:  GaussianNB()

```
In [30]: model.fit(X_train,Y_train)
```

Out[30]:  GaussianNB()

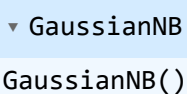
```
In [51]: y_pred=model.predict(X_test)
```

```
In [47]: print("Number of mislabeled points out of a total %d points : %d" % (X_test.shape[0],
Number of mislabeled points out of a total 45 points : 3
```

In [ ]:

## Evaluating A Classification Model

```
In [48]: model
```

Out[48]:  GaussianNB()

## confusion\_matrix

```
In [54]: from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,y_pred)

# here x direction predicted class
# y direction actual class
# ref this link -https://www.analyticsvidhya.com/blog/2021/12/evaluation-of-classif
```

Out[54]: array([[14, 0, 0],  
[ 0, 16, 2],  
[ 0, 1, 12]], dtype=int64)

## Precision

Precision is defined as the ratio of True Positives count to total True Positive count made by the model.  $\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$

```
In [57]: from sklearn.metrics import precision_score  
precision_score(Y_test, y_pred, average=None)
```

```
Out[57]: array([1.          , 0.94117647, 0.85714286])
```

```
In [ ]:
```

## Recall

Recall is defined as the ratio of True Positives count to the total Actual Positive count.  $\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$  Recall is also called "True Positive Rate" or "sensitivity".

```
In [60]: from sklearn.metrics import recall_score  
recall_score(Y_test, y_pred, average=None)
```

```
Out[60]: array([1.          , 0.88888889, 0.92307692])
```

```
In [ ]:
```

**It is called the F1 score. It is the harmonic mean of recall & precision. The harmonic mean is more sensitive to low values, so the F1 will be high only when both precision & recall are high.**

```
In [62]: from sklearn.metrics import f1_score  
f1_score(Y_test, y_pred, average=None)
```

```
Out[62]: array([1.          , 0.91428571, 0.88888889])
```

```
In [ ]:
```

## 5. Advantages

**1. it can handle large no of datasets**

**2. it can handel large no of features**

**3. it converges faster**

(all points works because of computations based on probability)

## 6. Disadvantages

**Co-related features may decrease the performance of the model ,because it may affects probability computation**

In [ ]:

## 7. Feature scaling required or not ?why ?

### NO need to do feature scaling

Naive Bayes algorithms, including Multinomial Naive Bayes and Bernoulli Naive Bayes, are not typically sensitive to feature scaling. These algorithms work with the probabilities of different features occurring given a particular class and make the assumption that features are conditionally independent. Because of this assumption, the relative scales of the features don't affect the model's performance.

Therefore, in most cases, feature scaling is not required for Naive Bayes algorithms. You can usually apply these algorithms directly to your dataset without normalizing or standardizing the features.

However, it's essential to note that if you have other algorithms or methods in your overall workflow that are sensitive to feature scales (such as distance-based methods like k-nearest neighbors or support vector machines), you might still want to scale your features for consistency across your entire process. But for Naive Bayes specifically, it's not a critical preprocessing step.

## 8. Impact of missing values and how?

### Naive bayes robust to missing values

Naive Bayes algorithms, particularly the Multinomial Naive Bayes and Bernoulli Naive Bayes variants commonly used in text classification and other discrete data applications, can handle missing values without requiring imputation.

The reason for this resilience to missing values lies in the nature of the independence assumption that Naive Bayes makes. The "naive" part of Naive Bayes implies that the algorithm assumes all features are independent given the class label. Therefore, the absence

of information (missing values) for one feature does not influence the estimation of the probabilities for other features.

Here are a few points to consider:

**1. Ignoring Missing Values:**

- Naive Bayes can simply ignore the missing values during the training process and still provide reasonable results.

**2. Imputation not Required:**

- Unlike some other machine learning algorithms that might require imputation of missing values before training, Naive Bayes does not necessitate this preprocessing step.

**3. Robustness to Sparse Data:**

- Naive Bayes is also known to be robust to sparse data, which means it can handle datasets with a large number of missing values or zero entries.

While Naive Bayes is robust to missing values, it's always a good practice to carefully analyze the reasons behind missing data in your dataset and consider whether imputation or other strategies might be beneficial for other aspects of your analysis or for the interpretability of your results.

## 9. Naive bayes robust to outliers

Naive Bayes algorithms, including Multinomial Naive Bayes and Bernoulli Naive Bayes, are generally considered to be robust to outliers. The reason for this lies in the nature of the probabilistic model and the assumption of independence among features.

Here are a few reasons why Naive Bayes tends to be robust to outliers:

**1. Independence Assumption:**

- Naive Bayes assumes that features are conditionally independent given the class. This means that the impact of outliers in one feature is limited to that specific feature, and it does not significantly affect the probabilities of other features.

**2. Robustness to Irrelevant Features:**

- Naive Bayes is also known to be robust to irrelevant features. Outliers in irrelevant features are less likely to have a significant impact on the classification process.

**3. Probabilistic Nature:**

- Naive Bayes calculates probabilities based on occurrences of features within each class. Outliers may affect the probability estimates for individual features, but since these probabilities are combined in a multiplicative manner, the overall impact of outliers tends to be mitigated.

While Naive Bayes is generally robust to outliers, it's crucial to note that the performance of any machine learning algorithm can be influenced by the severity and distribution of outliers

in the dataset. Additionally, if outliers are indicative of errors or anomalies in the data, it's important to address and understand them in the context of your specific application.

Preprocessing steps such as outlier detection and handling might still be relevant depending on the nature of your data and the goals of your analysis.

In [ ]:

In [ ]:

## 10.What Are the Basic Assumption?

### Features Are Independent

In [ ]:

## 11.Application

### Sentiment Analysis

### Spam classification

### twitter sentiment analysis

### document categorization