UNIT 2

Naive Bayes Classifiers

Naive Bayes Classifiers

Naive Bayes classifiers are supervised machine learning algorithms used for classification tasks, based on Bayes' Theorem to find probabilities.

Bayes' Theorem is a fundamental concept in probability theory that describes how to update the probabilities of hypotheses as new evidence becomes available. It's a way to formally incorporate new information into your beliefs about the likelihood of events.

Key Features of Naive Bayes Classifiers

- The main idea behind the Naive Bayes classifier is to use **Bayes' Theorem** to classify data based on the probabilities of different classes given the features of the data. It is used mostly in high-dimensional text classification.
- It is named as "Naive" because it **assumes the presence of one feature does not affect other features.**
- The "Bayes" part of the name refers to for the basis in Bayes' Theorem.

- **The Naive Bayes Classifier** is a simple probabilistic classifier and it has very few number of parameters which are used to build the ML models that can predict at a faster speed than other classification algorithms.
- It is a **probabilistic classifier** because it assumes that one feature in the model is independent of existence of another feature. Each feature contributes to the predictions with no relation between each other.
- **Naïve Bayes Algorithm** is used in situations like spam filtration, Sentimental analysis, classifying articles and many more.

Sample Dataset that **describes the weather conditions for playing a game of golf.** Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for playing golf.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.
Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In the dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In given dataset, the class variable name is **‘Play golf’.**

Assumption of Naive Bayes

The fundamental Naive Bayes assumption is that each feature makes a/an:

- **Feature independence:** This means that when we are trying to classify something, we assume that each feature (or piece of information) in the data does not affect any other feature.
- **Continuous features are normally distributed:** If a feature is continuous, then it is assumed to be normally distributed within each class.
- **Discrete features have multinomial distributions:** If a feature is discrete, then it is assumed to have a multinomial distribution within each class.
- **Features are equally important:** All features are assumed to contribute equally to the prediction of the class label.
- **No missing data:** The data should not contain any missing values.

Naive Bayes is a classification algorithm that predicts the probability of a data point belonging to a particular class given its features.
It's often used for text classification (e.g., spam filtering, sentiment analysis), but it can also be applied to other types of data.

Bayes' Theorem in Naive Bayes:

Example: classify a data point *x* into one of several classes *C*. Bayes' Theorem :

P(C|x) = [P(x|C) * P(C)] / P(x)

Where:

- **P(C|x):** The *posterior probability* of the data point *x* belonging to class *C*. This is what we want to calculate.
- **P(x|C):** The *likelihood* of observing the features *x* given that the data point belongs to class *C*.
- **P(C):** The *prior probability* of class *C*.
- **P(x):** The probability of observing the features *x*. This acts as a normalizing constant.

The "Naive" Assumption:

The problem with directly using this formula is that calculating

$P(x|C)$ can be very difficult, especially when the data point X has many features. This is where the "naive" assumption comes in.

Naive Bayes assumes that the features are *conditionally independent* given the class.
It assumes that the presence or absence of one feature does not affect
the presence or absence of another feature, *given* that we know the class.

$$P(x|C) = P(x_1|C) * P(x_2|C) * ... * P(x_n|C)$$

Where $x_1, x_2, ..., x_n$ are the individual features of x.

Apply Bayes' theorem based on Sample Dataset

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Types of Naive Bayes Model

There are three types of Naive Bayes Model :

- **Gaussian Naive Bayes-**continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values

- **Multinomial Naive Bayes-**used when features represent the frequency of terms (such as word counts) in a document. It is commonly applied in text classification, where term frequencies are important.

- **Bernoulli Naive Bayes-**deals with binary features, where each feature indicates whether a word appears or not in a document. It is suited for scenarios where the presence or absence of terms is more relevant than their frequency. Both models are widely used in document classification tasks

Specific example of classification using Naive Bayes.

Simple text classification problem.

- **Problem:**
- We want to classify emails as either "spam" or "not spam" based on the words they contain.

- **Data:**
- We have the following **training data**
- **Spam:** "free money win prize", "urgent claim your reward", "discount offer limited time“
- **Not Spam:** "meeting tomorrow at 10am", "project update discussion", "lunch together after work"

Steps:

1.Vocabulary: Create a vocabulary of all unique words in the training data:

{"free", "money", "win", "prize", "urgent", "claim", "your", "reward", "discount", "offer", "limited", "time", "meeting", "tomorrow", "at", "10am", "project", "update", "discussion", "lunch", "together", "after", "work"}

2. **Calculate Prior Probabilities:** Calculate the probability of each class:

- P(Spam) = 3/6 = 0.5 (3 spam emails out of 6 total)
- P(Not Spam) = 3/6 = 0.5

3. **Calculate Likelihood Probabilities**: Calculate the probability of each word given each class. Use Laplace smoothing (add-1 smoothing) to avoid probabilities of zero. This means we add 1 to the count of each word in each class and add the size of the vocabulary to the denominator.

Spam:

- P(free|Spam) = (1+1)/(12+22) = 2/34 (Word "free" appears once in spam emails +1 / total words in spam emails + vocabulary size)
- P(money|Spam) = 2/34
- P(win|Spam) = 2/34
- P(prize|Spam) = 2/34
- P(urgent|Spam) = 2/34
- P(claim|Spam) = 2/34
- P(your|Spam) = 2/34
- P(reward|Spam) = 2/34
- P(discount|Spam) = 2/34
- P(offer|Spam) = 2/34
- P(limited|Spam) = 2/34
- P(time|Spam) = 2/34
- ... (words not in spam emails have probability 1/34)

Not Spam:

- P(meeting|Not Spam) = 2/34
- P(tomorrow|Not Spam) = 2/34
- P(at|Not Spam) = 2/34
- P(10am|Not Spam) = 2/34
- P(project|Not Spam) = 2/34
- P(update|Not Spam) = 2/34
- P(discussion|Not Spam) = 2/34
- P(lunch|Not Spam) = 2/34
- P(together|Not Spam) = 2/34
- P(after|Not Spam) = 2/34
- P(work|Not Spam) = 2/34
- ... (words not in not spam emails have probability 1/34)

4. Classify a New Email: Example :classify the email: **"free lunch offer"**

Calculate Posterior Probabilities:

- P(Spam|Email) $\propto$ P(Email|Spam) * P(Spam) ($\propto$ means proportional to; we can ignore the denominator P(Email) as it's the same for both classes) = P(free|Spam) * P(lunch|Spam) * P(offer|Spam) * P(Spam) = (2/34) * (1/34) * (2/34) * 0.5 ≈ 0.000086
- P(Not Spam|Email) $\propto$ P(Email|Not Spam) * P(Not Spam) = P(free|Not Spam) * P(lunch|Not Spam) * P(offer|Not Spam) * P(Not Spam) = (1/34) * (2/34) * (1/34) * 0.5 ≈ 0.000026

Prediction: Since P(Spam|Email) is **greater** than P(Not Spam|Email), we classify the email as **spam**.