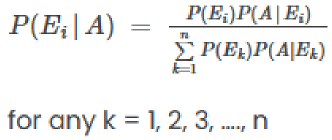
WEEK-07: NAÏVE BAYES CLASSIFIER

BAYES' THEOREM

Bayes' theorem describes the probability of occurrence of an event related to any condition. It is also considered for the case of conditional probability. Bayes theorem is also known as the formula for the probability of "causes". For example: if we have to calculate the probability of taking a blue ball from the second bag out of three different bags of balls, where each bag contains three different colour balls viz. red, blue, black. In this case, the probability of occurrence of an event is calculated depending on other conditions is known as conditional probability.

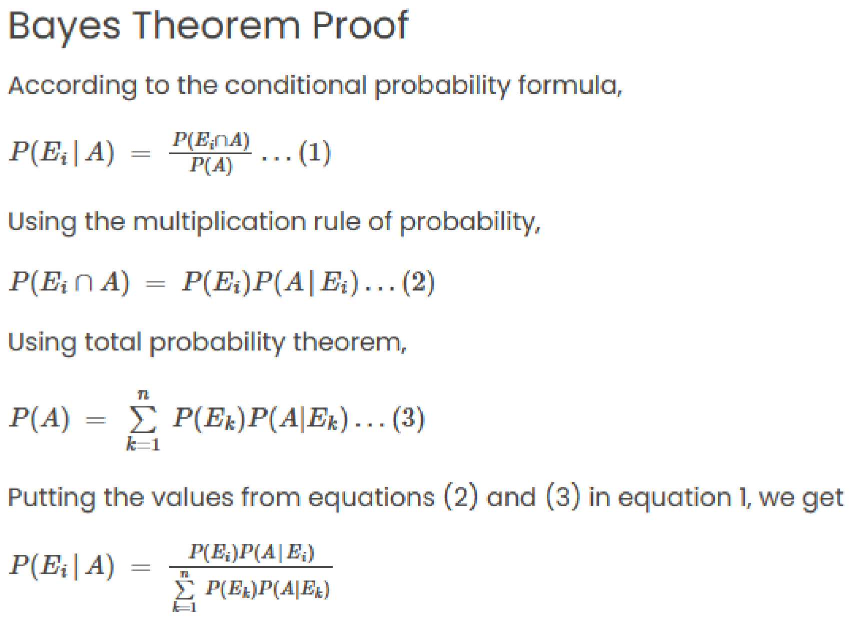
Bayes Theorem Statement

Let $E_1, E_2, \ldots, E_n$ be a set of events associated with a sample space S, where all the events $E_1, E_2, \ldots, E_n$ have nonzero probability of occurrence and they form a partition of S. Let A be any event associated with S, then according to Bayes theorem,



$$P(E_i \mid A) = \frac{P(E_i)P(A \mid E_i)}{\sum_{k=1}^{n} P(E_k)P(A|E_k)}$$

for any k = 1, 2, 3, …., n



Bayes Theorem Proof

According to the conditional probability formula,

$$P(E_i \mid A) = \frac{P(E_i \cap A)}{P(A)} \ldots (1)$$

Using the multiplication rule of probability,

$$P(E_i \cap A) = P(E_i)P(A \mid E_i) \ldots (2)$$

Using total probability theorem,

$$P(A) = \sum_{k=1}^{n} P(E_k)P(A|E_k) \ldots (3)$$

Putting the values from equations (2) and (3) in equation 1, we get

$$P(E_i \mid A) = \frac{P(E_i)P(A \mid E_i)}{\sum_{k=1}^{n} P(E_k)P(A|E_k)}$$

Note:

The following terminologies are also used when the Bayes theorem is applied:

Hypotheses: The events $E_1, E_2, \ldots E_n$ is called the hypotheses

Priori Probability: The probability $P(E_i)$ is considered as the priori probability of hypothesis E_i

Posteriori Probability: The probability $P(E_i|A)$ is considered as the posteriori probability of hypothesis E_i

Bayes' theorem is also called the formula for the probability of "causes". Since the E_i's are a partition of the sample space S, one and only one of the events E_i occurs (i.e. one of the events E_i must occur and the only one can occur). Hence, the above formula gives us the probability of a particular E_i (i.e. a "Cause"), given that the event A has occurred.

Bayes Theorem Derivation

Bayes Theorem can be derived for events and random variables separately using the definition of conditional probability and density.

From the definition of conditional probability, Bayes theorem can be derived for events as given below:

$P(A|B) = P(A \cap B)/ P(B)$, where $P(B) \neq 0$

$P(B|A) = P(B \cap A)/ P(A)$, where $P(A) \neq 0$

Here, the joint probability $P(A \cap B)$ of both events A and B being true such that,

$P(B \cap A) = P(A \cap B)$

$P(A \cap B) = P(A \mid B) P(B) = P(B \mid A) P(A)$

$P(A|B) = [P(B|A) P(A)]/ P(B)$, where $P(B) \neq 0$

Naming the Terms in the Theorem

The terms in the Bayes Theorem equation are given names depending on the context where the equation is used. It can be helpful to think about the calculation from these different perspectives and help to map your problem onto the equation.

Firstly, in general, the result P(A|B) is referred to as the posterior probability and P(A) is referred to as the prior probability.
P(A|B): Posterior probability.
P(A): Prior probability.
Sometimes P(B|A) is referred to as the likelihood and P(B) is referred to as the evidence.
P(B|A): Likelihood.
P(B): Evidence.
This allows Bayes Theorem to be restated as:
Posterior = Likelihood * Prior / Evidence

Implementation in python

```
# Define the prior probability P(A), likelihood P(B|A), and evidence P(B)
prior_probability = 0.01  # P(A) - Prior probability of event A
likelihood = 0.9  # P(B|A) - Likelihood of observing event B given A
evidence = 0.02  # P(B) - Total probability of observing event B

# Calculate the posterior probability P(A|B) using Bayes' theorem
posterior_probability = (prior_probability * likelihood) / evidence

# Print the result
print(f"Posterior Probability P(A|B): {posterior_probability:.4f}")
```

Example

Suppose the probability of the weather being cloudy is 40%. Also suppose the probability of rain on a given day is 20%. Also suppose the probability of clouds on a rainy day is 85%. If it’s cloudy outside on a given day, what is the probability that it will rain that day?

Solution:

- P(cloudy) = 0.40
- P(rain) = 0.20
- P(cloudy | rain) = 0.85

Thus, we can calculate:

- P(rain | cloudy) = P(rain) * P(cloudy | rain) / P(cloudy)
- P(rain | cloudy) = 0.20 * 0.85 / 0.40
- P(rain | cloudy) = 0.425

Implementation in python

```
#define function for Bayes' theorem
def bayesTheorem(pA, pB, pBA):
    return pA * pBA / pB
#define probabilities
pRain = 0.2
pCloudy = 0.4
pCloudyRain = 0.85
#use function to calculate conditional probability
bayesTheorem(pRain, pCloudy, pCloudyRain)
```

NAÏVE BAYES CLASSIFIER

The Naive Bayes classifier is a simple yet powerful machine learning algorithm that is particularly well-suited for tasks involving text classification and spam filtering. It is based on Bayes' theorem and makes the "naive" assumption of independence between the features used for classification. Despite this simplification, Naive Bayes often performs surprisingly well in practice. In this analysis, we will delve into the key concepts behind the Naive Bayes classifier, its applications, advantages, limitations, and practical implementation.

Key Concepts of Naive Bayes Classifier

1. **Bayes' Theorem**: At the heart of the Naive Bayes classifier is Bayes' theorem, a fundamental concept in probability theory. It allows us to calculate the probability of a particular event based on prior knowledge of conditions that might be related to the event.

Bayes' theorem can be expressed as:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Where:

- P(A|B) is the conditional probability of event A given event B.
- P(B|A) is the conditional probability of event B given event A.
- P(A) and P(B) are the probabilities of events A and B, respectively.

2. **Independence Assumption**: The "naive" part of Naive Bayes comes from assuming that the features used for classification are independent of each other. In practice, this assumption is often not entirely accurate, but the model can still perform well.
3. **Multinomial and Gaussian Naive Bayes**: There are different variants of Naive Bayes classifiers, including Multinomial and Gaussian Naive Bayes. Multinomial Naive Bayes is commonly used for text data, while Gaussian Naive Bayes is suitable for continuous data with a normal distribution.

Applications of Naive Bayes Classifier

The Naive Bayes classifier finds applications in various domains:

1. **Text Classification**: It is widely used for spam email detection, sentiment analysis, and topic classification of documents.
2. **Document Categorization**: Naive Bayes can categorize documents into predefined categories, making it useful for news article classification and content recommendation.
3. **Medical Diagnosis**: It can assist in medical diagnosis by classifying patient data into different disease categories.
4. **Customer Sentiment Analysis**: Businesses use it to analyze customer feedback, reviews, and social media comments to determine customer sentiment.
5. **Recommendation Systems**: Naive Bayes can be used to recommend products or content based on user behavior and preferences.

Advantages of Naive Bayes Classifier

1. **Simplicity**: Naive Bayes is easy to understand and implement, making it a good choice for quick prototyping and initial model development.
2. **Efficiency**: It is computationally efficient and can handle large datasets with a relatively small amount of memory.
3. **Good for Text Data**: Naive Bayes performs well on text data, making it a popular choice for tasks like spam detection and sentiment analysis.
4. **Works with Small Data**: Even with limited training data, Naive Bayes can produce meaningful results.

Limitations of Naive Bayes Classifier

1. **Independence Assumption**: The assumption of feature independence is often violated in real-world data, which can lead to suboptimal performance.
2. **Limited Expressiveness**: Naive Bayes is a linear classifier, which means it cannot capture complex relationships between features.
3. **Poor with Rare Events**: It may perform poorly when dealing with rare events or classes with limited examples.
4. **Sensitive to Feature Quality**: The quality of features used in the model can greatly impact its performance.

Practical Implementation of Naive Bayes Classifier

Implementing a Naive Bayes classifier typically involves the following steps:

1. **Data Preprocessing**: Collect and preprocess the data, including cleaning, tokenization, and feature extraction.
2. **Split Data**: Split the dataset into training and testing sets.
3. **Model Training**: Train the Naive Bayes classifier using the training data.
4. **Model Evaluation**: Evaluate the model's performance on the testing data using appropriate metrics such as accuracy, precision, recall, and F1-score.
5. **Model Tuning**: Experiment with different variants of Naive Bayes (e.g., Multinomial or Gaussian) and adjust hyperparameters as needed.
6. **Deployment**: Once satisfied with the model's performance, deploy it to make predictions on new, unseen data.

Conclusion

The Naive Bayes classifier is a simple yet effective machine learning algorithm with applications in text classification, spam detection, sentiment analysis, and more. While it has its limitations, its simplicity, efficiency, and suitability for text data make it a valuable tool in the machine learning toolkit. When properly implemented and tuned, Naive Bayes can provide meaningful insights and predictions for various real-world problems. However, it is essential to be mindful of the independence assumption and consider other algorithms when dealing with complex, interdependent features.

Implementation in python

Certainly, the Naïve Bayes Classifier is a popular choice for text classification and other machine learning tasks. Below is a simple implementation of the Naïve Bayes Classifier in Python using the scikit-learn library.

Step 1: Install scikit-learn

If you haven't already installed scikit-learn, you can do so using pip:

```
!pip install scikit-learn
```

Step 2: Import Necessary Libraries

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
```

Step 3: Prepare Your Data

Assuming you have a dataset with text samples and corresponding labels (e.g., positive or negative sentiment), you should split the data into a training set and a test set. Here's an example:

```
# Sample data
texts = ["This is a positive review.", "Negative sentiment detected.", "A very positive experience.", "I didn't like this at all."]
# Corresponding labels (1 for positive, 0 for negative)
labels = [1, 0, 1, 0]
# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2, random_state=42)
```

Step 4: Feature Extraction

You need to convert the text data into numerical features. One common approach is to use the CountVectorizer, which counts the frequency of words in the text. Here's how to do it:

```
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

Step 5: Train the Naïve Bayes Classifier

Next, create and train the Naïve Bayes classifier. For text classification, the Multinomial Naïve Bayes classifier is commonly used:

```
clf = MultinomialNB()
clf.fit(X_train_vec, y_train)
```

Step 6: Make Predictions

Once the classifier is trained, you can use it to make predictions on new data:

```
y_pred = clf.predict(X_test_vec)
```

Step 7: Evaluate the Model

Evaluate the model's performance using appropriate metrics:

```
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred) print(f"Accuracy: {accuracy}") print(report)
```

This code demonstrates a basic implementation of the Naïve Bayes Classifier in Python using scikit-learn. Depending on your specific task and dataset, you may need to fine-tune the pre-processing steps, hyper parameters, and model selection to achieve the best performance.

Questions

1. Implement in python program of the following problems using Bayes Theorem.

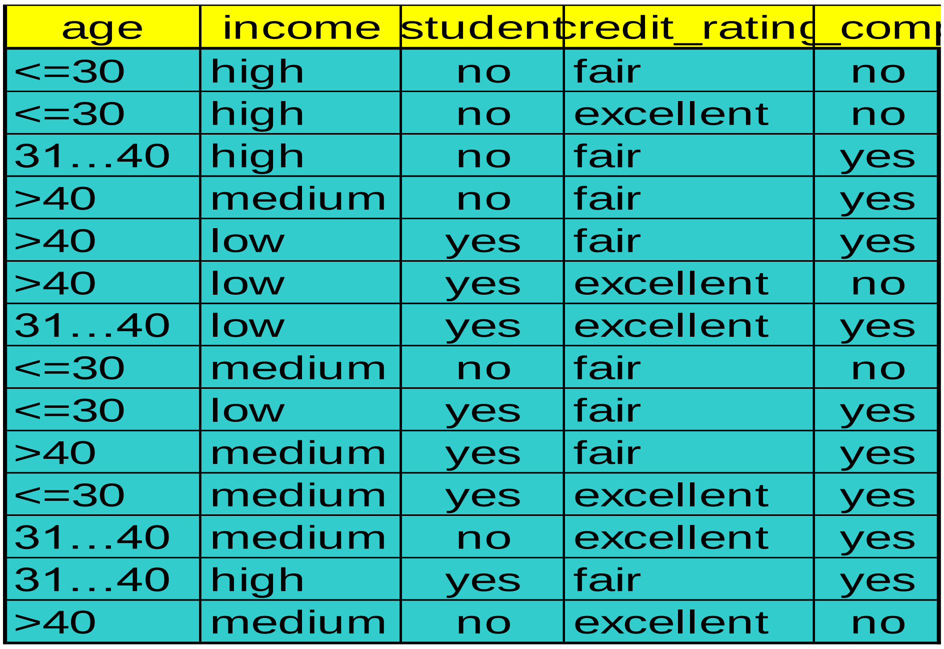
a) Of the students in the college, 60% of the students reside in the hostel and 40% of the students are day scholars. Previous year results report that 30% of all students who stay in the hostel scored A Grade and 20% of day scholars scored A grade. At the end of the year, one student is chosen at random and found that he/she has an A grade. What is the probability that the student is a hosteler?

b) Suppose you're testing for a rare disease, and you have the following information:

- The disease has a prevalence of 0.01 (1% of the population has the disease).
- The test is not perfect:
    - The test correctly identifies the disease (true positive) 99% of the time (sensitivity).
    - The test incorrectly indicates the disease (false positive) 2% of the time (1 - specificity).

Calculate the probability of having the disease given a positive test result using Bayes' theorem.

2. Develop a function python code for Naïve Bayes classifier from scratch without using scikit-learn library, to predict whether the buyer should buy computer or not. Consider a following sample training dataset stored in a CSV file containing information about following buyer conditions (such as “<=30,” “medium,” “Yes,” and “fair”) and whether the player played golf (“Yes” or “No”).



age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

3. Write a Python function to implement the Naive Bayes classifier without using the scikit-learn library for the following sample training dataset stored as a .CSV file. Calculate the accuracy, precision, and recall for your train/test dataset.

a. Build a classifier that determines whether a text is about sports or not.

b. Determine which tag the sentence "A very close game" belongs to.

Text	Tag
"A great game"	Sports
"The election was over"	Not sports

"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Additional Questions

1. Write a function python program to implement the naïve Bayesian classifier without using scikit-learn library for the following sample training data set stored as a .CSV file. Calculate the accuracy, precision, and recall for your train/test data set. To classify 'If the weather is sunny, then the Player should play or not'?

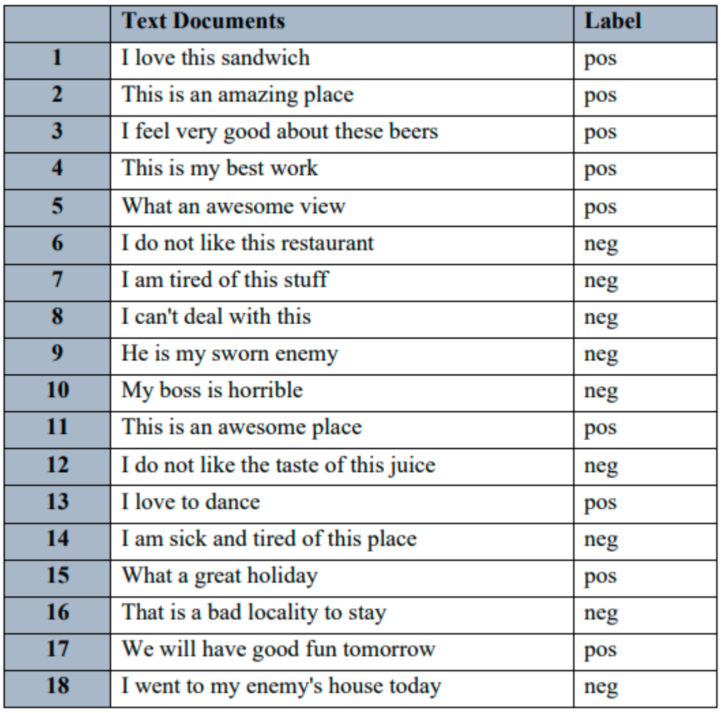
	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

2.Develop a function python code for Naïve Bayes classifier from scratch without using scikit-learn library, to predict whether the player should play golf based on weather conditions. Consider a following sample training dataset stored in a CSV file containing information about following weather conditions (such as "Sunny," "Hot," "High Humidity," and "Not Windy") and whether the player played golf ("Yes" or "No").

	WEATHER	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No

7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
11	Rainy	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Sunny	Mild	High	True	No

3. Write a function python program to implement the naïve Bayesian classifier without using scikit-learn library for a following sample training data set stored as a.CSV file. Calculate the accuracy, precision, and recall for your train/test data set.



	Text Documents	**Label**
1	I love this sandwich	pos
2	This is an amazing place	pos
3	I feel very good about these beers	pos
4	This is my best work	pos
5	What an awesome view	pos
6	I do not like this restaurant	neg
7	I am tired of this stuff	neg
8	I can't deal with this	neg
9	He is my sworn enemy	neg
10	My boss is horrible	neg
11	This is an awesome place	pos
12	I do not like the taste of this juice	neg
13	I love to dance	pos
14	I am sick and tired of this place	neg
15	What a great holiday	pos
16	That is a bad locality to stay	neg
17	We will have good fun tomorrow	pos
18	I went to my enemy's house today	neg