National University
of computer and emerging sciences

Naïve Bayes Classification

Bayesian Approach:

Imagine a situation where you are trying to predict whether it will rain the forthcoming week, backed with information relating to weather conditions of the 7 days before the actual "rainy day", for the past 2 years. Yes, it's the historic data. You would consider the key factors, which affect the weather conditions, and try to give a prediction based on this past behavior. This approach is called the Bayesian approach.

Probability Theory:

It is not only important what happened in the past, but also how likely it is that it will be repeated in the future. Probability theory is all about randomness vs. likelihood. It basically quantifies the likelihood of an event occurring in a random space.

For example, if I flip a coin and expect a "heads", there is a 50%, or 1⁄2, chance that my expectation will be met, provided the "act of flipping", is unbiased (a fair, or unbiased coin has the same probability to get head or tail). This assumption of fairness is attributed to randomness and the chance of meeting the expectation is my probability.

Let's take another classic example of rolling a dice. If I roll a dice and expect to get a "4", what are my odds?

It is quantified by (expected outcome / total no. of outcomes), which is 1/6, i.e. out of a total possible 6 outcomes, we expect one specific outcome of a number 4. The sample space contains or holds all possible events. Probability is always quantified as a percentage or a number between 0 and 1. The probability can either be a discrete or a continuous variable.

The Bayes Theorem:

Before going to Bayes' theorem, we need to know about the basic concepts of probability. Firstly, in the above example, we are calculating the probability of the coin landing on heads AND the dice landing on 4. This is called a **joint probability**. There are two other types of probabilities. One is called **conditional probability**, which calculates the probability of heads GIVEN THAT the dice lands on 4. Lastly, if you want the probability of specific outcomes, i.e. probability of JUST the coin or JUST the dice, we call it the **marginal probability**. Bayes' theorem (named after Rev. Thomas Bayes 1702-1761) is based on this.

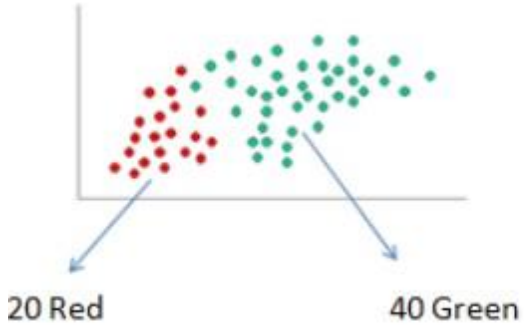
It states, for two events A & B, if we know the conditional probability of B given A and the probability of B, then it's possible to calculate the probability of B given A.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where A and B are events and $P(B) \neq 0$.

- $P(A \mid B)$ is a conditional probability: the likelihood of event A occurring given that B is true.
- $P(B \mid A)$ is also a conditional probability: the likelihood of event B occurring given that A is true.
- $P(A)$ and $P(B)$ are the probabilities of observing A and B independently of each other; this is known as the marginal probability.

Let's understand Bayes' rule by an example. The task is to identify the color of a newly observed dot.



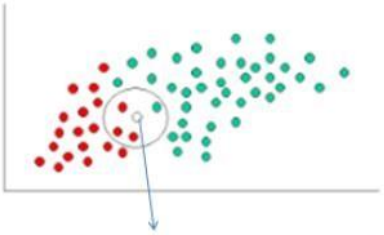
Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have a membership with GREEN rather than RED. In the Bayesian analysis, this belief is known as the **prior probability**. Prior probabilities are based on previous experience, in this case, the percentage of GREEN and RED objects, and are often used to predict outcomes before they actually happen.

Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership can be written as below:

$$Prior\ Probability\ of\ GREEN = \frac{number\ of\ GREEN\ objects}{total\ number\ of\ objects} = \frac{40}{60}$$

$$Prior\ Probability\ of\ RED = \frac{number\ of\ RED\ objects}{total\ number\ of\ objects} = \frac{20}{60}$$

Having formulated our prior probability, we are now ready to classify a new object (WHITE circle in the diagram below). Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this **likelihood**, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label. From this we calculate the likelihood:



LIKELIHOOD OF THIS BEING RED = (NO. OF RED IN THE VICINITY / TOTAL NO OF RED)

From the illustration above, it is clear that the likelihood of X given GREEN is smaller than the likelihood of X given RED, since the circle encompasses 1 GREEN object and 3 RED ones.

Although the **prior probabilities** indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the **likelihood** indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a **posterior probability** using Bayes' rule.

$$Posterior\ Probability\ of\ GREEN = Prior\ Probability\ of\ GREEN \times Likelihood\ of\ GREEN = \frac{40}{60} \times \frac{1}{40}$$

$$Posterior\ Probability\ of\ RED = Prior\ Probability\ of\ RED \times Likelihood\ of\ RED = \frac{20}{60} \times \frac{3}{20}$$

Finally, we classify X as RED since its class membership achieves the largest posterior probability.

The Naïve Bayes Classifier:

It is a classification technique based on Bayes' theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The algorithm is called Naïve because it assumes that features are conditionally independent given the class. In other words, they assume that for all instances of a given class, the features have little/no correlation with each other.

On the positive side, Naive Bayes classifiers are fast to train and use for prediction and thus are well suitable to high dimensional data including text. And the applications involving very large data sets where efficiency is critical and computational costs rule out other classification approaches.

On the negative side, when the conditional independence assumption about features doesn't hold. In other words, for a given class, there's significant covariance among features, as is the case with many real-world datasets. Other more sophisticated classification methods that can account for these dependencies are likely to outperform Naive Bayes.

On a side note, when getting confidence or probability estimates associated with predictions, Naive Bayes classifiers produce unreliable estimates, typically. Still, Naive Bayes Classifiers can perform very competitively on some tasks and are also often very useful as baseline models against which more sophisticated models can be compared.

Lab Task:

Use the Naïve Bayes classifier on the Tennis Play dataset, your code should ask the user to input some features and return the class of the input.

- Load the dataset into the pandas data frame.

```
data = pd.read_csv('play_tennis.csv')
data.head()
data.drop('day',axis = 1)
```

	outlook	temp	humidity	wind	play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

- Compute probabilities for the target class for both (yes and no)

```
class_counts = list(data['play'].value_counts())
class_counts
```

```
[9, 5]
```

```
prob = [x/len(data) for x in class_counts]
prob
```

```
[0.6428571428571429, 0.35714285714285715]
```

- Compute probabilities for all the features one by one for both cases (yes and no).
- Take a test example from user asking for the features and compute the target class for that example.