# Lecture 11 – Naive Bayes Classification

Mr. Yousif G. Arshak
University of Zakho
Computer Science Department
yousif.arshak@uoz.edu.krd

# OUTLINES

- Naïve Bayes
  - Introduction
  - Bayes Theory
  - Implementation of Naïve bayes in Python

# Introduction

- **Naive Bayes** is one of the simplest machine learning algorithms. It is supervised algorithm.
- Naive Bayes is a classification algorithm and is extremely fast. It uses Bayes theory of probability.
- **Why Naive?**
- It is called ‘naive’ because the algorithm assumes that all attributes are independent of each other.
- Naive Bayes algorithm is commonly used in text classification with multiple classes.
- To understand how Naive Bayes algorithm works, it is important to understand Bayes theory of probability. Let’s work through an example to derive Bayes theory.

# Bayes Theory

1. Let’s assume there is a type of cancer that affects 1% of a population. The test for the cancer, detects the presence of cancer correctly 90% of time. So it gets the remaining 10% wrong. The test also gives a correct negative result 90% of the time. The remaining 10% of time it detects a cancer when there is none. With these probabilities in place, what are the chances that a person actually has cancer when they get a positive result from the test.

# A simple way to work through this question is to take some nice round numbers and calculate values.

**Bayes Theorem**

| | | |
|---|---|---|
| R1 | People with Cancer = 10000 * 1% | 100 |
| R2 | People with no cancer = 10000 * 99% | 9,900 |
| R3 | True Pos = 100 * 90% | 90 |
| R4 | False Neg = 100 * 10% | 10 |
| R5 | False Pos = 9900 * 10% | 990 |
| R6 | P = 90 / (990+90) | 0.08333333333 |

- When a person gets a positive result from the test, the probability that the person actually has cancer =
- Probability of a true positive / (Probability of true positive + Probability of false positive)
- Now let's convert this into Bayes theorem.

$$P(c|x) = \frac{P(x|c)\,P(c)}{P(x|c)\,P(c) + P(x|\text{not } c)\,P(\text{not } c)}$$

- $P(c|x)$ = Probability of having cancer (c) given the test (x) is positive = 8.33% in our example
- $P(x|c)$ = Probability of getting positive test (x) given you had a cancer (c) = True positive = 90%

- P( c ) = Chances of having a cancer = 1%
- P( x|not c) = Probability of getting a positive test (x) given you did not have a cancer (c) = False positive = 10%
- P( not c) = Chances of not having a cancer = 99%
- In a simpler form, the denominator can be called P( x ). The probability of test being positive, false or true.
- Rewriting the equation,

$$P(c|x) = \frac{P(x|c)\,P(c)}{P(x)}$$

# Example

- A pizza chain wants to open its delivery centres across a city. What do you think would be the possible challenges?
  - They need to analyze the areas from where the pizza is being ordered frequently.
  - They need to understand as to how many pizza stores has to be opened to cover delivery in the area.
  - They need to figure out the locations for the pizza stores within all these areas in order to keep the distance between the store and delivery points minimum

# Here is some data for when a person, say Joe, plays tennis.

| Data | |
|---|---|
| **Temperature** | **Play Tennis** |
| Hot | No |
| Hot | No |
| Hot | Yes |
| Mild | Yes |
| Cool | Yes |
| Cool | No |
| Cool | Yes |
| Mild | No |
| Cool | Yes |
| Mild | Yes |
| Mild | Yes |
| Mild | Yes |
| Hot | Yes |
| Mild | No |

| Probability Table | | | |
|---|---|---|---|
| **Temperature** | **Play Tennis : Yes** | **Play Tennis : No** | **Probability** |
| **Hot** | 2 | 2 | 4/14 = 0.29 |
| **Cool** | 3 | 1 | 4/14 = 0.29 |
| **Mild** | 4 | 2 | 6/14 = 0.43 |
| **All** | 9 | 5 | |
| **Probability** | 9/14 = 0.64 | 5/14 = 0.36 | |

- Now let’s validate the statement: when the temperature is mild, Joe will play tennis. Is this statement true?
- What we need is the probability that Joe will play tennis given the temperature is mild, i.e., P(Joe Plays | Mild Temperature)
- Which is P(Mild Temperature | Joe plays) P(Joe Plays) / P(Mild Temperature)
- (4/9) * (0.64) / (0.43) = 0.65
- When the temperature is mild, there is a good probability that Joe will play tennis.

# Implementation Naive Bayes in Python in Python

```
#First, start with importing necessary python packages

import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_sp
lit
from sklearn.metrics import accuracy_score
```

- # read Iris dataset from csv file from your PC
- `play_tennis = pd.read_csv("PlayTennis.csv")`

- Our data contains details about the weather outlook, temperature, humidity and wind conditions. The last column is the target variable that suggests the possibility of playing tennis.

# we need to convert the categorical information in our data into numbers.Because SKLearn library requires the features to be numerical arrays

```
number = LabelEncoder()
play_tennis['Outlook'] = number.fit_transform(play_tennis['Outlook'])
play_tennis['Temperature'] = number.fit_transform(play_tennis['Temperature'])
play_tennis['Humidity'] = number.fit_transform(play_tennis['Humidity'])
play_tennis['Wind'] = number.fit_transform(play_tennis['Wind'])
play_tennis['Play Tennis'] = number.fit_transform(play_tennis['Play Tennis'])
```

define the features and the target variables.

```
features = ["Outlook", "Temperature", "Humidity", "Wind"]
target = "Play Tennis“
```

To validate the performance of our model, we create a train, test split. We build the model using the train dataset and we will validate the model on the test dataset. We use SKLearn’s train_test_split to do this.

```
features_train, features_test, target_train, target_test =
train_test_split(play_tennis[features],
play_tennis[target],
test_size = 0.33,
random_state = 54)
```

Let’s create the model now.

```
model = GaussianNB()
model.fit(features_train, target_train)
```

Now we are ready to make predictions on the test features. We will also measure the performance of the model using accuracy score. Accuracy score measure the number of right predictions.

```
pred = model.predict(features_test)
accuracy = accuracy_score(target_test, pred)
```

| Outlook | Temperature | Humidity | Wind |
|---|---|---|---|
| Rain | Mild | High | Weak |

```
1 | print model.predict([[1,2,0,1]])
```

which gives a prediction 1 (Yes)