

## Introduction

Artificial intelligence (AI) is a rapidly growing field that has the potential to transform many industries and revolutionize the way we live and work. In this assignment, we will explore the use of AI in a specific application: a movie recommendation engine. Movie recommendation systems in the market use data about users' past movie ratings and preferences to suggest new movies that they might like. These systems rely on machine learning algorithms to analyze and learn from the data and can provide personalized recommendations to users based on their individual tastes. In this assignment, we will implement a movie recommendation system using the Naive Bayes classifier, a popular machine learning algorithm that is well-suited for this task. We will analyze the performance of our system and explore ways to improve its accuracy and effectiveness.

## Why Naïve Bayes?

To be able to create the engine, we've been given some AI algorithms to use. They were Graphs, Linear Regression, Min-Max, Naïve Bayes, k Nearest Neighbors, and Neural Networks. After careful consideration, we choose to go with the Naïve Bayes classifier.

In comparison to other machine learning models, the Naive Bayes classifier would be a good choice for our assignment due to its simplicity. In addition to that, our dataset will not be adequate for the others but NB classifier. However, it is important to evaluate the performance of multiple models and choose the one that performs best for a given dataset and application. Unfortunately, due to the limited time, we couldn't perform other models to make a comparison.

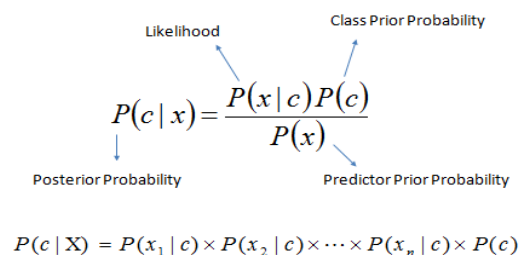
## Naïve Bayes

The Naive Bayes classifier is a probabilistic model that is based on the idea of using the Bayes theorem to make predictions. The Bayes theorem states that the probability of an event occurring is equal to the prior probability of the event occurring multiplied by the likelihood of the event occurring given some evidence.

In the case of the Naive Bayes classifier, the goal is to predict the class label (e.g. "positive" or "negative") of a given sample based on the features of the sample. The classifier first estimates the prior probability of each class, which is the probability of the class occurring in the overall dataset. It then estimates the likelihood of each feature belonging to each class and combines these likelihoods with the prior probabilities to estimate the posterior probability of each class for a given sample. The class with the highest posterior probability is chosen as the prediction for the sample.

One of the key assumptions of the Naive Bayes classifier is that all of the features are independent of each other, which is why it is called "naive."

Mathematic behind the Naïve Bayes classifier:<sup>1</sup>



The diagram shows the Naive Bayes formula with labels for its components. The formula is 
$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$
 where  $P(c | x)$  is labeled 'Posterior Probability',  $P(x | c)$  is labeled 'Likelihood',  $P(c)$  is labeled 'Class Prior Probability', and  $P(x)$  is labeled 'Predictor Prior Probability'. Below the formula, the expanded version is given: 
$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

<sup>1</sup> [https://www.intel.com/content/www/us/en/developer/articles/technical/mathematical-concepts-and-principles-of-naive-bayes.html#:~:text=Mathematics%20Behind%20Naive%20Bayes&text=P\(c%7Cx\)%3A%20posterior,out%20of%20all%20the%20observations.](https://www.intel.com/content/www/us/en/developer/articles/technical/mathematical-concepts-and-principles-of-naive-bayes.html#:~:text=Mathematics%20Behind%20Naive%20Bayes&text=P(c%7Cx)%3A%20posterior,out%20of%20all%20the%20observations.)

## Implementation, Central Flow, and Optimization

For the given input, the program calculates the posterior probability for each movie and saves the score to a dictionary. After doing this for all available movies in the dataset, it prints the movie as the best match which has the highest score.

### Central Flow:

1. Creating Dataset:
  - a. We used different sources to generate the final dataset:
    - i. General movie data (title, year, genre, IMDB score, etc.): IMDB<sup>2</sup> (CSV)
    - ii. Keywords for each movie: Cinemagoer (API)<sup>3</sup>
    - iii. Movies in the streaming services: Netflix, Amazon Prime, Disney Plus: [www.reelgood.com](http://www.reelgood.com)<sup>4</sup> (web scrapping)
    - iv. Movies in the streaming service HBO-Max: [www.justwatch.com](http://www.justwatch.com)<sup>5</sup> (web scrapping)
  - b. We did web scrapping and API connection in python then used python libraries to import CSV data and merged all different data frames into the final dataset and exported it to a CSV file.
2. Movie Library Class
  - a. Imports the movie dataset from a CSV file
  - b. Creates and keeps all the movie objects
  - c. Keeps all the keywords and their occurrences (for the AI algorithm)
3. Movie Class
  - a. A movie object with the attributes (title, year, keywords, etc.)
4. Input Class
  - a. It checks for crazy input (null/empty input, only digits)
  - b. Transforms input (removes special characters and punctuation)
  - c. Creates keywords from the given input for the search engine
5. Search Engine Class (AI Algorithm)
  - a. Uses Movie Library Class as a source to make a prediction for the given input
  - b. Calculates posterior probability for each movie and saves the score to a dictionary (key: movie, value: score).
  - c. Prints the best-scored movie.

### Optimization:

Working with a big dataset of keywords during the posterior probability calculation resulted in significantly small values, which start approaching zero. To overcome this problem, we used log-probabilities.

During the test period, it's been observed that the efficiency of the recommended movie depends on the input in terms of the number of words, the specificity of the words, etc. Therefore, a tip has been added to the introduction part to encourage the user to enter more keywords.

---

<sup>2</sup> <https://datasets.imdbws.com/>

<sup>3</sup> <https://imdbpy.readthedocs.io/en/latest/index.html>

<sup>4</sup> <http://www.reelgood.com/>

<sup>5</sup> <https://www.justwatch.com/>

## Ethic

The dataset which was used for the AI model has been generated from the top 10000 movies by the number of votes (source: IMDB). The less popular movies have already been ignored and have never been included in the dataset. This causes the fact that while the movies included in the dataset will become more popular by recommending them, even though some of the excluded movies might be a better recommendation for the input, they will become less and less popular over time, and the probability to become a popular movie will decrease. To solve this problem, the dataset might be replaced by a dataset that contains larger amounts of movies.

## Conclusion

In conclusion, the use of artificial intelligence in movie recommendation systems has the potential to transform the way we discover and enjoy new movies. In this assignment, we implemented a movie recommendation engine using the Naive Bayes classifier. We analyzed the performance of our system and explored ways to improve its accuracy and effectiveness. While the Naive Bayes classifier is a simple and effective model, it is important to consider the ethical concerns that can arise with movie recommendation systems and to take steps to address these issues in the design and implementation of these systems.

## Sources:

- [www.kaggle.com](https://www.kaggle.com/)
- [www.stackoverflow.com](https://www.stackoverflow.com)
- [www.w3schools.com](https://www.w3schools.com)
- <https://towardsdatascience.com/>
- <https://reelgood.com/>
- <https://www.justwatch.com/>
- <https://www.intel.com/>
- [www.imdb.com](https://www.imdb.com)
- <https://imdbpy.readthedocs.io/en/latest/index.html>