# Implementation of a Naive Bayes Classifier

## Introduction

The purpose of this report is to detail the implementation of a Naive Bayes classifier from scratch in Python. The Naive Bayes algorithm is a probabilistic classifier based on Bayes' theorem, which is used to predict the likelihood of a class label given a set of features. In this assignment, we apply the Naive Bayes classifier to the 'Play Tennis' dataset, a small toy dataset commonly used in machine learning examples. The classifier is implemented without the use of pre-built machine learning libraries, allowing a deeper understanding of the underlying calculations.

## Methodology

The Naive Bayes classifier assumes that each feature is conditionally independent given the class label. This 'naive' assumption simplifies the calculation of conditional probabilities. The classifier works by computing prior probabilities for each class, followed by likelihoods of feature values given each class. During prediction, these probabilities are combined to determine the most likely class for a new instance.

### Data Preparation

The 'Play Tennis' dataset includes weather conditions as features and a target variable indicating whether tennis was played. It contains four features: Outlook (Sunny, Overcast, Rain), Temperature (Hot, Mild, Cool), Humidity (High, Normal), and Wind (Weak, Strong). The target variable is PlayTennis (Yes, No). The data was prepared by manually entering it into a JSON or CSV format. The features are categorical, making them suitable for Naive Bayes without further encoding. A summary of the dataset is presented below.

## Implementation of Naive Bayes Classifier

### Calculating Priors and Likelihoods

In the training stage, the Naive Bayes algorithm calculates the prior probability of each class, i.e., the likelihood that a randomly selected instance belongs to a particular class. For each feature, the likelihood of each value given a class label is also computed. The likelihoods for categorical data are simply the proportions of each feature value within each class, with Laplace smoothing applied to avoid zero probabilities.

### Prediction Process

During classification, the model reads the stored probabilities and applies Bayes' theorem to calculate the posterior probability for each class. The predicted class is the one with the highest posterior probability. If a test set is provided, the model compares predicted labels with actual labels to calculate accuracy.

## Results and Analysis

After training and testing the classifier on the 'Play Tennis' dataset, accuracy was calculated by comparing predicted labels against actual labels. A confusion matrix was generated to show true positives, true negatives, false positives, and false negatives. Overall, the classifier achieved reasonable accuracy, with most instances correctly classified. Misclassifications were analyzed to identify potential patterns, such as overlapping feature values.

## Discussion and Conclusion

The Naive Bayes classifier performed well on the 'Play Tennis' dataset, a small and simple dataset where the assumption of conditional independence may largely hold. The classifier's simplicity and speed make it suitable for small datasets, though it may struggle with large datasets where features are not independent. Implementing the classifier from scratch deepened our understanding of the algorithm and the influence of each feature on predictions. Future improvements could involve experimenting with different datasets or exploring other probability-based models.

## References

- Alpaydin, E. (2010). Introduction to Machine Learning. MIT Press.  
- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.  
- Class lecture notes on Naive Bayes classifiers.