# Naive Bayes Classifier

Sukanya Harshvardhan
PES University
CSE Department
PES1201700214
mithalishashidhar8@gmail.com

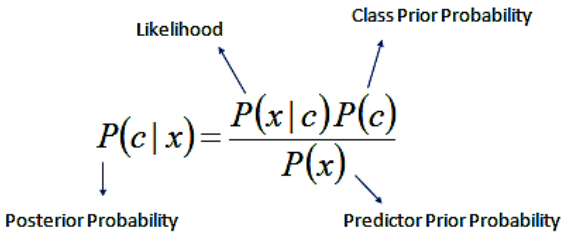
Prajna Girish
PES University
CSE Department
PES1201701261
prajna2310@gmail.com

Kevin Arulraj
PES University
CSE Department
PES1201700659
kevinarulraj@gmail.com

***Abstract*—The goal of the project is to use a Naive Bayes classifier to classify whether a person is a democrat or a republican based on their votes, using 16 binary attributes and 2 classes. This will help to estimate the accuracy of Naive Bayes algorithm using 5-fold cross validation on the dataset given. A decision tree has also been used to classify data and compare the performances of the two classifiers.**

## I. INTRODUCTION

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is a family of algorithms where all of them share a common assumption-every pair of features being classified is independent of each other. In machine learning, we are often interested in selecting the best hypothesis (h) given data (d). In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d) ie. a classification problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge. We have created a decision tree to classify the given data and compare it with the naive Bayes classifier that we constructed.




$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

*Fig: Bayes Theorem*

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. We picked this classifier because the target function was discrete.

## II. ML TECHNOLOGIES USED

We have used the concept of Bayesian learning and abided by the assumption that all the 16 binary attributes are independent of each other. The classification is done based on the formula given below.

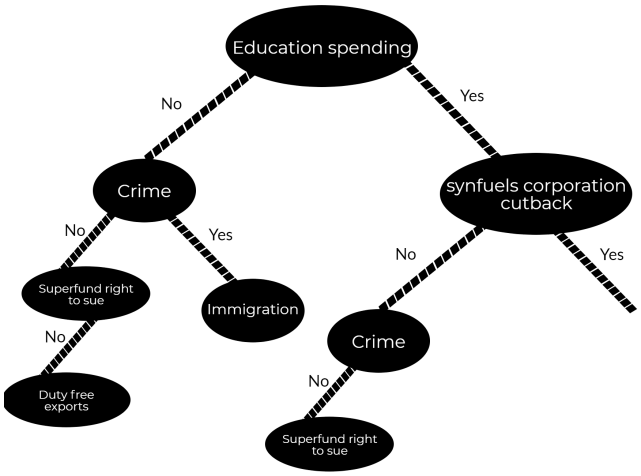
$$\text{argmax } P(V_i)\, \pi^{n}_{i=1} P(a_i|V_i)$$

All attributes have equal priority and are all independent of each other and if the probability of obtaining one class is greater than the probability of obtaining the other class, the model classifies the data as the class with the higher probability. To implement our decision tree, we have made use of the concept of entropy and information gain. To construct the tree, we made use of the ID3 (Iterative Deepening) algorithm.The bias used here is that we want the shortest tree, and hence a tree that classifies better and decreases chaos. Entropy is a measure of impurity in the model. Based on the value of entropy for a particular attribute, we find the associated information gain using the given formula. Then, we pick the attributes with the highest gain as the next node, and so on, so that we obtain the shortest tree.

$$Gain(S, D) = H(S) - \sum_{V \in D} \frac{|V|}{|S|} H(V)$$

where S is the dataset and D is a particular attribute, V is a particular value in that attribute.

*Fig: Obtained decision tree*




## III. IMPLEMENTATION

We have constructed a Naive Bayes classifier using a 5 fold cross validation. If it returns a 0, then the person is a democrat, and a republican otherwise. We built a Naive

Bayes classifier and a decision tree to classify the data. We have used python 3 for both the classifiers and the python modules sklearn and pandas for data pre-processing. First, we clean the data and fill in the missing values by replacing them with the moe of the column. For the Naive Bayes model, we first split the data into 5 folds and then run the Bayesian algorithm on all the 5 testing and training sets. Following this, we do the actual data classification. On the other hand, for the decision tree, we split the data into a testing and training data using a 1:9 ratio. We then calculate the entropy and hence information gain for each attribute and pick the one with the highest information gain to form the next node, and so on, till all the attributes are present in the decision tree, and then we classify the data. We then compare the actual and obtained output values obtained from both the classifiers and plot a confusion matrix to further obtain the accuracy, precision, recall and F measure.

## IV. GRAPHICAL COMPARISON

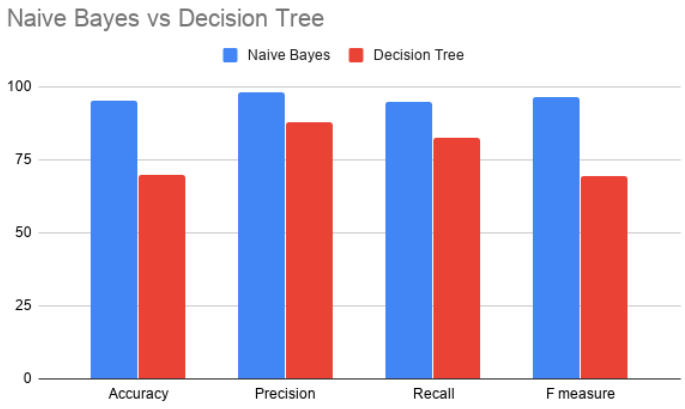



*Fig: Comparing Naive Bayes and Decision Tree outcomes*

## V. CONCLUSION

This aim of this project was to achieve a high accuracy in classifying the instances. After running an approbate number of epochs and various optimisation methods, we have obtained the accuracy of the Naive Bayes model to be at 95.4%. Contrary to this, the accuracy obtained from the decision tree was approximately 70%.

The value of the precision, recall and F measure for the Naive Bayes model was 98.22%, 94.83%, 96.49% respectively. Overall, all these parameters were better than the ones obtained from the decision tree. We also observed that the decision tree takes extremely long to build and train, as compared to the Naive Bayes. From the graph it can be concluded that the Naive Bayes classier was most suitable for this dataset, considering that our dataset was small, discrete and binary.

## VI. ACKNOWLEDGMENT

We would like to thank our mentor, N S Kumar for guiding us through the entire project and helping us learn and understand the required concepts. We would also like to thank our peers for constant support and encouragement. In conclusion, this entire project was a great learning experience where we were able to practically implement some of the algorithms we have only seen on paper.

## VI. REFERENCES

1. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid-*R Kohavi-Kdd, 1996, Citeseer*

*2.* An empirical study of the Naive Bayes classifier-*I Rish-IJCAI 2001 workshop on empirical methods in artificial intelligence, 2001- cc.gatech.edu*

3. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

4. https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

5. https://machinelearningmastery.com/k-fold-cross-validation/

6. https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

7. https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1

8. https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

9. https://www.smartdraw.com/decision-tree/