Classification models: Naive Bayes and Decision trees

Anushka Hada

11-5-2025

1 Introduction

For this project, classification models such as Naive Bayes and Decision trees were implemented from scratch using tools like numpy and pandas library. The main idea was to understand the mechanics behind these models and to test them against the sklearns version of these models.
Two datasets from the UCI machine learning repository were used for the project:

- **Mushroom Dataset**: A dataset which classifies whether a mushroom is **edible (e)** or **poisonous (p)** based on several properties (22 features to be exact) of that mushroom.
- **Congressional Voting Records Dataset**: This dataset is used to predict a representative's affiliation, **Democrat or Republican**, based on their voting patterns (16 features). This dataset's name was shortened to **House Dataset** for easier reading.

2 Pre-Training Preparation

2.0.1 Tools used

Several libraries were used for implementing the models. **Pandas** and **NumPy** were used for data handling and model creation, while **Scikit-learn** tools were used for splitting data into training, development, and testing sets. **Scikit-learn** also provides classification models such as GaussianNB, BernoulliNB, and DecisionTreeClassifier. Each models performance was evaluated using metrics such as accuracy score and confusion matrix. Also, **OneHotEncoder** was used to encode strings into numbers.

2.0.2 Datasets preparation

For the Mushroom and House data set 80% went to training, 10% to testing, and 10% to development. Since some categories had unknown or missing values, it was decided that they would be treated as a separate category for analysis. One hot encoding was applied to both of the datasets, excluding the class, turning them into a numerical representation of categorical data. The labels were hard coded to prevent confusion. For the house dataset, democrat is 1 and republican is 0.

For the mushroom dataset, poison is 1 and edible is 0. Furthermore, the sets were turned into numpy arrays so they could be processed by the model.

3 Naive Bayes

The NaiveBayes class implements a Gaussian Naive Bayes classifier from scratch using pandas and numpy. During training, the fit method finds the parameters for each class including the mean, variance, and prior probability of each feature. When making predictions, the model calculates the posterior probability of each class when given a set of features using the Gaussian probability density function. The class with the highest posterior probability is selected as the predicted class. It's important to note that to prevent division by 0 clipping was used on the values of the Probability Density Function (PDF). Overall, this classifier assumes that the features are independent and has a normal distribution.

4 Decision Tree

The DecisionTree class implements a recursive approach to split the data based on the feature that provides the highest **information gain** at each node. This means that the model computes all the possible **thresholds** for every feature and selects the split that maximizes either **Gini impurity** reduction or **entropy** reduction, depending on which one was selected during initialization of model. The tree continues expanding until it reaches either the maximum depth or the minimum number of samples needed for continuing to split further. When it can no longer split the nodes, a **leaf node** is made and given the majority class of the samples that reach that node. During analysis, predictions are made by going through the tree starting from the root or the very top. It then follows the decision made using the **threshold** until a leaf node (ending node) is reached. This implementation is based off of a standard decision tree classifier.

5 Results

5.0.1 Naïve Bayes models for Mushroom Dataset

As shown below, Table 1 gives the performance of the three Naïve Bayes models on the Mushroom dataset. The results show that all models achieved a high accuracy for training, development, and test sets, demonstrating good generalization. Furthermore, Sklearn Gaussian Naïve Bayes performed the best, with a test accuracy of 96.06%. This is slightly higher then the one implemented from scratch and the Bernoulli version.

Table 2 confirms these trends by showing the confusion matrices for each model for the test set. The Gaussian model shows the fewest misclassifications with only one false negative, suggesting that it captures the feature distributions more effectively than other models. In contrast, the Bernoulli Naïve Bayes model shows higher false negatives (42), implying that it may be oversimplify the binary representation of features. Overall, the Gaussian Naïve Bayes offered the most balanced and accurate classification for this dataset.

Table 1: Accuracy comparison of Naïve Bayes models on Mushroom dataset

Model	**Test Accuracy**	**Dev Accuracy**	**Train Accuracy**
Custom Naïve Bayes	0.9483	0.9434	0.9412
Sklearn Bernoulli Naïve Bayes	0.9433	0.9225	0.9421
Sklearn Guassian Naïve Bayes	0.9606	0.9631	0.9569

Table 2: Confusion Matrices of Naïve Bayes models for the Mushroom Dataset

Model	**Confusion Matrix**	**Interpretation**
Custom Naïve Bayes	$\begin{bmatrix} 387 & 34 \\ 8 & 383 \end{bmatrix}$	TP=387, FP=34, FN=8, TN=383
Sklearn Bernoulli Naïve Bayes	$\begin{bmatrix} 417 & 4 \\ 42 & 349 \end{bmatrix}$	TP=417, FP=4, FN=42, TN=349
Sklearn Guassian Naïve Bayes	$\begin{bmatrix} 390 & 31 \\ 1 & 390 \end{bmatrix}$	TP=390, FP=31, FN=1, TN=390

5.0.2 Naïve Bayes models for House Dataset

Table 3 shows the performance of the three Naïve Bayes models on the House dataset. The results clearly show that all models achieved a reasonably high accuracy. The from-scratch Naïve Bayes and Gaussian Naïve Bayes models both obtained a test accuracy of 86.05%, outperforming the Bernoulli version. The development and training accuracies were also high across the models, though Bernoulli Naïve Bayes had a slightly lower dev accuracy of 93.18%. This suggests that Bernoulli had less stable generalizations.

The confusion matrices in Table 4 further validates these trends. Both the from-scratch and the other models correctly classify most samples with a balanced amount of true positives and true negatives. Though the Bernoulli model shows a higher number of false negatives at 5. This indicates that it sometimes misclassifed positive cases. Overall, the results suggest that the Gaussian and standard Naïve Bayes models provide the most consistent performance for the House dataset.

Table 3: Accuracy comparison of Naïve Bayes models on House dataset

Model	**Test Accuracy**	**Dev Accuracy**	**Train Accuracy**
Custom Naïve Bayes	0.8605	0.9773	0.9569
Sklearn Bernoulli Naïve Bayes	0.8372	0.9318	0.9109
Sklearn Guassian Naïve Bayes	0.8605	0.9545	0.9540

Table 4: Confusion Matrices for the Naïve Bayes models for the House Dataset

Model	Confusion Matrix	Interpretation
Custom Naïve Bayes	$\begin{bmatrix} 14 & 3 \\ 3 & 23 \end{bmatrix}$	TP=14, FP=3, FN=3, TN=23
Sklearn Bernoulli Naïve Bayes	$\begin{bmatrix} 15 & 2 \\ 5 & 21 \end{bmatrix}$	TP=15, FP=2, FN=5, TN=21
Sklearn Guassian Naïve Bayes	$\begin{bmatrix} 14 & 3 \\ 3 & 23 \end{bmatrix}$	TP=14, FP=3, FN=3, TN=23

5.0.3 Decision Tree models for Mushroom Dataset

The Decision Tree models trained on the Mushroom dataset got perfect performance regardless of the splitting criteria or the different implementations. All three models reached a test accuracy of 99.88%, a development accuracy of 99.88%, and a training accuracy of 99.98% .

The confusion matrices at Table 6 validate the accuracies. All three models correctly classified all samples, with only 1 false negative remaining. These results show that the Mushroom dataset is highly separable, meaning it allows the models to create splits that perfectly distinguish between the two classes.

Table 5: Accuracy comparison of Decision Tree models on Mushroom dataset

Model	Test Accuracy	Dev Accuracy	Train Accuracy
Custom Decision Tree (Gini)	0.9988	0.9988	0.9998
Custom Decision Tree (Entropy)	0.9988	0.9988	0.9998
Sklearn Decision Tree	0.9988	0.9988	0.9998

Table 6: Confusion Matrices for Decision Tree models on Mushroom dataset

Model	Confusion Matrix	Interpretation
Custom Decision Tree (Gini)	$\begin{bmatrix} 421 & 0 \\ 1 & 390 \end{bmatrix}$	TP=421, FP=0, FN=1, TN=390
Custom Decision Tree (Entropy)	$\begin{bmatrix} 421 & 0 \\ 1 & 390 \end{bmatrix}$	TP=421, FP=0, FN=1, TN=390
Sklearn Decision Tree	$\begin{bmatrix} 421 & 0 \\ 1 & 390 \end{bmatrix}$	TP=421, FP=0, FN=1, TN=390

5.0.4 Decision Tree models for House Dataset

As shown in Table 7, the models trained on the House dataset got a good performance, but slightly lower test accuracies compared to the Mushroom dataset. This may be due to the amount of features present in the two datasets. The from-scratch Gini tree got the highest test accuracy at 88.37%. Both the from-scratch Entropy and scikit-learn Decision Tree models reached 86.05%. All models got 100% development accuracy and high training accuracy of 99.14% to 99.71%. This suggests that the models effectively captured patterns in the training data but may have overfit,

especially on the Gini-based model.

The confusion matrices shown on Table 8 reveal small differences in misclassifications. The Gini model misclassified 2 false positives and 3 false negatives, the entropy model misclassified 3 false positives and 3 false negatives, and the scikit-learn model misclassified 2 false positives and 4 false negatives. Overall, these results show that while all three models perform well the choice of the splitting criteria influenced the values of the confusion matrices, where Gini slightly outperformed Entropy on the test set.

Table 7: Accuracy comparison of Decision Tree models on House dataset

Model	**Test Accuracy**	**Dev Accuracy**	**Train Accuracy**
Custom Decision Tree (Gini)	0.8837	1.0000	0.9971
Custom Decision Tree (Entropy)	0.8605	1.0000	0.9971
Sklearn Decision Tree	0.8605	1.0000	0.9914

Table 8: Confusion Matrices for Decision Tree models on House dataset

Model	**Confusion Matrix**	**Interpretation**
Custom Decision Tree (Gini)	$\begin{bmatrix} 15 & 2 \\ 3 & 23 \end{bmatrix}$	TP=15, FP=2, FN=3, TN=23
Custom Decision Tree (Entropy)	$\begin{bmatrix} 14 & 3 \\ 3 & 23 \end{bmatrix}$	TP=14, FP=3, FN=3, TN=23
Sklearn Decision Tree	$\begin{bmatrix} 15 & 2 \\ 4 & 22 \end{bmatrix}$	TP=15, FP=2, FN=4, TN=22

6 Conclusion

In conclusion, the experiments conducted on the models using the two datasets, House and Mushroom, revealed several notable trends. For the House dataset, the Custom Decision Tree (Gini) got the highest test accuracy at 88.37%. Overall, the House dataset had consistently lower accuracies on the same models compared to the Mushroom dataset. One theory as to why this may be could be due to the number of features. The Mushroom dataset is more complex containing around 22 features, leading to better generalizations.

Furthermore, for the Mushroom dataset, all three versions of the Decision Tree gave the highest accuracy for the test set at 99.88%. This reveals that the mushroom dataset is more "separable" then the House Dataset. As for the Gini and Entropy models, neither showed a big difference in performance, indicating that the way information gain is calculated does not have a big effect on accuracy. In the future, the hyper parameters shall be tuned further to account for overfitting and the lower accuracies of the House dataset.