# Classifiers Comparison: Decision Tree vs. Naïve Bayes for Predicting the Length of Hospital Stays

Alessandro Alviani, *Department of Computer Science, INM431 Machine Learning Coursework*

CITY UNIVERSITY OF LONDON EST 1894

## 1. Motivation and Problem Description

The length of stay (LOS) is an important indicator of the efficiency of hospital management [1] and can be helpful to monitor the quality of inpatient care [2] and provide optimized treatment plans to the patients.
In this study a LOS dataset was used to build two supervised machine learning classification models, with the aim to compare them and critically evaluate their results in this domain.

## 2. Initial Analysis of the Data

In this study a diminished version of the "AV : Healthcare Analytics II" datasets were used. The original datasets were first proposed in a Hackathon in 2020 and then published on Kaggle. These comprise two datasets: a training set of 318,483 rows and a test set of 13,704. Both sets had 17 predictors, mixed categorical and numerical and a target variable with 11 classes. In this project only the "AV" training set was used (split in 80-20, training-testing) and the target variable was transformed from 11 classes to binary: 0 = short, 1 = long.

- The dataset was presented cleaned and without missing values.
- Minor presence of outliers. These were considered informative and left as such.
- The relevant predictors were distributed as shown in figure 1.
- The dataset was largely biased towards the negative class (80.4%: "Short stay") (figure 2).
- Some multicollinearity between features. This was dealt with by keeping the features most correlated to the target and dropping the others. Correlations between the final predictors and the target variable are shown in figure 3.
- After the exploratory data analysis the original dataset was split into 80% train set and 20% test set.
- In the training set the minority class was augmented with a Synthetic Minority Oversampling Technique (SMOTE) to reach a 50/50 split.
- The categorical numerical variables were label encoded.
- Feature standardization was tested but led to worse model performance hence abandoned.
- The test set was left unbalanced and only pre-processed to be compatible with the train set.

**Figure 1 – Feature distributions**

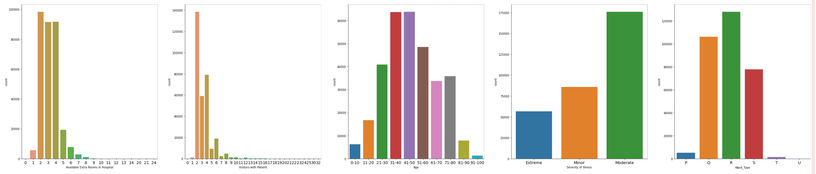


**Figure 2 – Target variable distribution**

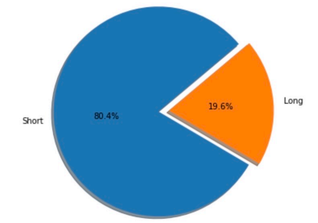
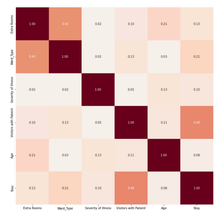



**Figure 3 – Correlation matrix**



## 3. Model Summaries

**Decision tree (DT)** is a supervised machine learning model used for both classification and regression tasks. Being a non-parametric nonlinear model, its function can take as many parameters as needed and generate a nonlinear mapping of X to y. DT is a simple flowchart like tree structure, where the topmost node is the root node, each leaf (or terminal node) holds a class label, each internal node denotes a test on an attribute and each branch represents an outcome of the test [4]. DT grows by splitting on the feature that achieves the maximum Information Gain at each stage.

**Pros**
- Highly tunable and adaptable.
- Works with mixed categorical and numerical data.
- Intrinsic feature selection.
- Doesn't require feature standardization.
- Handles missing data.
- Easy to interpret and explain.
- Fast to train and make predictions.
- Non-parametric (makes no assumptions regarding the shape of data).

**Cons**
- Tendency to overfit the training set if not properly tuned.
- Fails to predict on previously unseen data.
- Can become computationally expensive.

**Naïve Bayes (NB)** is a supervised machine learning method used mostly for classification tasks and with some adaptations for regression as well [3]. The algorithm uses Bayes' theorem to assign each observation to a class. Such classification is based on a maximum at posteriori probability, calculated by multiplying each observation's prior probability by the likelihood of that observation to belong to that class. NB naïvely assumes that all the features are independent.

**Pros**
- Highly tunable, particularly on numerical data.
- Works with mixed categorical and numerical data.
- Handles missing data.
- Computationally efficient.
- Can be optimized by prior knowledge.
- Resilient to noise in the data.
- Highly scalable (linear time complexity).
- Can be updated as new data comes in.

**Cons**
- Based on the assumption of independence between features (often violated).
- The algorithm would return 0 for any feature absent in the observations. (Sum of logs and smoothing techniques can prevent this).
- Kernel distributions can make the model computationally expensive.

## 4. Hypothesis Statement

Both literature and previous studies have shown that Naive Bayes (NB) and Decision trees (DT) can be two effective classifiers and perform similarly on the same preprocessed data [4]. In this project these models were trained for early prediction of the length of stay (LOS) of hospital admissions. This study hypothesizes that the two models will be able to predict the positive class better than a random guess. In line with previous research [6] it's also hypothesized that, given the small number of relevant features in the dataset, the DT will be more accurate in its predictions as well as faster at training and predicting.

## 5. Choice of Training and Model Evaluation Methodology

**Choice of training methodology**
One of the main concerns in machine learning is the bias-variance trade-off of a model. Models need to fit the training data and learn its patterns but at the same time must be able to generalize well on new, unseen data. In this study much attention was paid to avoid overfitting to the training set and to build models that would be accurate whilst simultaneously as simple as possible and capable to perform on new data. The dataset was initially split into 80% training set and 20% test set and, to minimize the chance of overfitting, the models were trained and validated on a 5-fold cross-validated set generated from the initial 80% training set. The test set was only used after training and exclusively to score the models' performance and analyze their behavior on its biased classes.

**Evaluation methodology**
When there is a strong imbalance in the target class the accuracy of a classifier, or misclassification rate, tends to become a non informative metric and can potentially provide misleading insights in the model performance. Measures of performance between the classes are to be preferred, both for training and scoring [5]. In this study, both models were mostly challenged by detecting the minority class (long stay) meanwhile keeping an overall good accuracy, hence the focus on the precision (hit rate), f1 score, balanced accuracy and sensitivity of the model. Attention was also paid to those performance metrics that tended to move in opposite directions (i.e. sensitivity and specificity). Models in training were cross-validated and scored against the validation set and, finally, against the test set.

## 6. Choice of Parameters and Experimental Results

**Classification Tree (CT)**

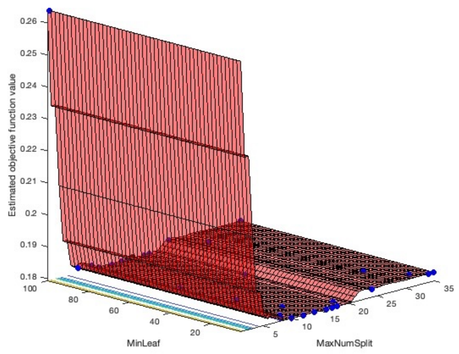
In building the CT model, the effect of tuning four hyperparameters were studied:

1. **Depth parameters**:
   - Maximum number of splits (MNS): tested in a range from 1 to 50. The chosen value was 10.
   - Minimum leaf size (MLS): tested in a range from 1 to 300. The chosen values was 60.
   - Minimum number of parent size (MPS) – this value was left to the default of 10.

The tree algorithm, if let ran, would split the training data space into as many subspaces as needed to progressively minimizes its loss. In other words, a tree could potentially grow exponentially large until each observation is assigned to a leaf. In most cases this would result in a severe form of overfitting and consequently in a model that is incapable of generalizing.
The above depth parameters work on stopping this behavior and consequently are the main hyperparameters to tune in a decision tree.

**Experimental Results**: The MNS was the parameter that helped control the model variance most effectively and some of its accuracy indicators. The MLS produced a similar effect but not as pronounced. Simultaneously tuning MNS and MLS marginally improved some performance metrics.

**Figure 4 – F1 Score optimization by Max Number of Splits and Min Leaf Size**




2. **Splitting criterion:** Gini diversity index (gdi), Twoing and Deviance were tested. Gdi (default) was chosen.

3. **Algorithm for categorical**: Exact, PullLeft, PCA, OVAbyClass. Exact was the algorithm of choice.

**Experimental Results**: Different choices of splitting criterion and algorithm for categorical predictors made no significant difference to the model.

4. **Prior probability of each class**.
Prior assigns a prior probability for each class. This was known and different to the one the model was defaulting to. This was due to SMOTE artificially creating a 50/50 split when the actual class split was 81/19.

**Experimental Results**: Setting different priors was the strategy that overall changed the model performance the most and could be used to tailor the classifier to specific needs. Figure 5 shows this phenomenon in the NB classifier, with False Positive Rate and False Negative Rate responding oppositely to a same prior.

**Naïve Bayes (NB)**

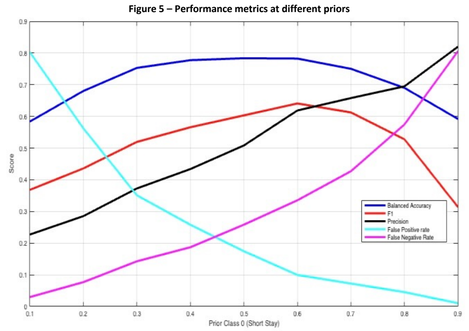
The hyperparameters tuned in the NB classifier were:

1. Prior probability of each class.

This would be the first term in the Bayes theorem and, as expected, is a very influential hyperparameter to tune.

**Experimental Results:** Choosing the right priors was one of the most delicate operations in tuning the NB algorithms and, similarly to the decision tree (which was Bayesian optimized), priors could help improve the model performance and tailor it to one's expectations or beliefs. Figure 5 shows how the different performance metrics move as the priors change.

**Figure 5 – Performance metrics at different priors**




2. **Predictor distributions**: normal, multinomial, kernel.

The Naïve Bayes classifier predicts a membership probability by multiplying an instance prior probability by the likelihood of that instance of belonging to that specific class. Different predictor distributions lead to different likelihoods, thus this is one of the most important parameters used to tune this model.

**Experimental Results**: predictor distributions had a great impact on the model performance both in terms of accuracy and in training and predicting times. Given the data, categorical predictors worked much better than any other numerical distribution on 4 out of 5 features. A kernel distribution, despite being the most appropriate for one predictor, significantly slowed the fitting and predicting times (from seconds to several minutes) and was abandoned in favor of a normal distribution that generated equivalent performance at a fraction of the time.

3. **Misclassification cost**. This is the cost of misclassification assigned to each class. It is possible to "penalize" missed labels of a specific class by an arbitrary factor of the operator's choice.

**Experimental Results:** Tuning this parameter was very similar to tuning the priors. As the misclassification cost of one class is increased, the model started predicting that class preferentially, thereby increasing positive predictive power at the cost of false positives. This parameter was left untouched in the end but could be useful to tune in case of requiring better predictive power on one specific class.

The prevalence of categorical features tended to limit the tuning options when working on the NB model, but any attempt to use different distributions, even after feature normalization, caused a severe decrease in model performance.

### Performance Metrics

- TN = CM(1,1) – FP = CM(1,2) – FN = CM(2,1) – TP = CM(2,2)
- Accuracy = 1 – Loss : Correct predictions (%).
- Sensitivity = TP / (TP + FN) : Probability of detection.
- Specificity = TN / (TN + FP) : Selectivity.
- Precision = TP / (TP + FP) : Positive predictive value.
- F1 = 2 x (Precision ∗ Sensitivity) / (Precision + Sensitivity)
- Balanced Accuracy (BA) = (Sensitivity + Specificity) / 2

## 7. Analysis and Critical Evaluation of Results

In accordance with the literature [4] and supporting this study's hypothesis, the experimental tests show how CT and NB can perform very similarly on preprocessed data. Data preprocessing and feature selection (discussed in section 1) were essential preparatory steps for the models to gain any predictive power and become responsive to tuning. It's also important to note that both models were tuned to obtain the highest accuracy without sacrificing specificity between classes. Allowing models to overpredict one class resulted in >90% accuracy, but at the severe expense of the models' power (sensitivity) and F1 score, due to a steep increase in the false negative rate (FNR).

The optimal performance for the two models was found at different prior probabilities: 70/30 for the CT and 60/40 for the NB and, due to the important role that priors pay in the models' behavior, this should be taken in account when observing the following comparative results:

- The CT performed better than the NB overall when comparing accuracy (84.29% vs 80.23%), F1 (67.34% vs 64.31%) and balanced accuracy (82.33% vs 77.83%). It particularly outperformed the NB on sensitivity, the model's power or probability of detection, with a score of 77.46% vs 65.83% (figure 6, 7).
- The NB had a slightly higher specificity (89.83% vs 87.21%) and precision (61.15% vs 59.57%) than CT (figure 6, 7).
- Both models had a significantly higher specificity than sensitivity indicating better predictive power for the majority class ("short stay") than the minority ("long stay") (figure 8), likely due to the models being trained on the unbiased training set and tested on the significantly positively-biased test set.
- Interestingly, the Receiver Operating Characteristic (ROC) curve of the two models showed very similar areas under the curve (AUC), CT: 83,56%, NB: 84.71%, but different trajectory lines. The NB model did better at lower false positive rates (FPR), peaking at around 10% FPR whereas the CT reached its best point at about 12.3% FPR (figure 9).
- False detection rate (CT: 40.4%, NB: 38.8%), as shown in the confusion matrices (figure 8), was the biggest challenge for both models and, without any additional work on the features, could be further improved only by changing classification costs or priors, consequently biasing the model towards a class.

Both optimized models were very quick at training and predicting (< 1 second). The times to train and predict for the NB classifier drastically increased with the use of kernel distributions, and so were not adopted.
The DT overall appeared to be the best model in this study, which could be explained by its "divide and conquer" algorithm that tends to perform well when a few highly relevant attributes exist [6], as in our dataset.

**Figure 6 – Metrics comparison (table)**

| | Decision Tree (DT) | Naïve Bayes (NB) |
|---|---|---|
| Accuracy | **0.8429** | 0.8023 |
| F1 Score | **0.6734** | 0.6341 |
| BA | **0.8233** | 0.7783 |
| Sensitivity | **0.7746** | 0.6583 |
| Specificity | 0.8721 | **0.8983** |
| Precision | 0.5957 | **0.6115** |

**Figure 7 – Performance metrics comparison (histograms)**

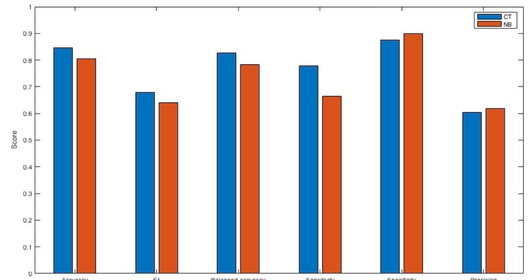



**Figure 8 – Confusion Matrices**

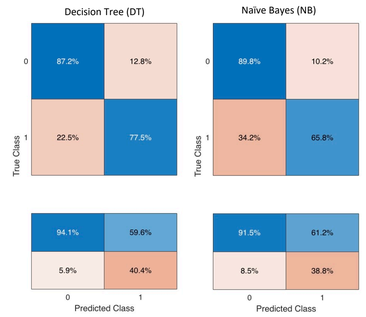
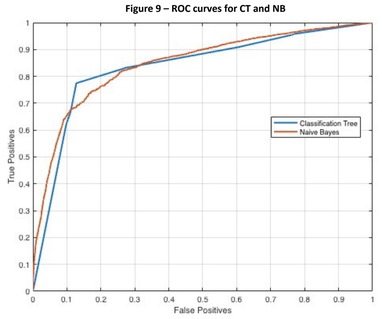



**Figure 9 – ROC curves for CT and NB**




## 8. Lessons Learned and Future Work

Data preprocessing played a key role in building the models. Both CT and NB had no predictive power on the preprocessed dataset and, during their tuning, some challenges could only be overcome by more extensive data cleaning and processing. Subsequently, tuning the algorithms' hyperparameters led to significant performance improvements but these plateaued quite rapidly. Allocating more time to feature engineering could have resulted in better outcomes and in general this is a point that should be kept in mind [7]. On the same note, whenever possible, the Data Scientist may consider suggesting good data sampling practices to reduce the need for synthetic techniques and obtain better quality data on which to train the models.
It was important to use different performance metrics and visualize their movements on validation and test set. An extensive use of plotting throughout the modelling stage seemed a very effective way to understand the algorithms and these figures could also be used to present and explain the models, effectively serving two purposes.
Further work may include deriving new predictive features from the data and optimizing the existing predictors by adopting a different binning . Allowing some degree of interdependencies between the features, employing the principle of local learning and adjusting the probabilities by attribute weight might represent effective strategies to explore [8].
Additionally, the CT model could be enhanced into a Random Forest and that would likely yield better results [9] .

### References

[1] Baek, H., Cho M., et al. "Analysis of length of hospital stay using electronic health records: A statistical and data mining approach". *PloS one, 13(4)*, e0195901. (2018).
[2] Pei-Fang (Jennifer) Tsai, P. Chen, et al. "Length of Hospital Stay Prediction at the Admission Stage for Cardiology Patients Using Artificial Neural Network", Journal of Healthcare Engineering, vol. 2016, Article ID 7035463, 11 pages, (2016).
[3] Frank, E., Trigg, L., Holmes, G. et al. "Technical Note: Naive Bayes for Regression". Machine Learning 41, (2000), 5–25.
[4] Xhemali, Hinde D., et al. " Naive Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages." IJCSI International Journal of Computer Science Issues, Vol. 4, No.1, (2009).
[5] K. J. D'souza and Z. Ansari, "Big Data Science in Building Medical Data Classifier Using Naïve Bayes Model," IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (2018), pp. 76-80.
[6] A. Ashari , I. Paryudi , A Toja, "Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool" *International Journal of Advanced Computer Science and Applications, Vol. 4, N.11,(2013)*.
[7] P. Chandrasekar, K. Qian, et al. "Improving the Prediction Accuracy of Decision Tree Mining with Data Preprocessing," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), (2017), pp. 481-484.
[8] K. M. Al-Aidaroos, A. A. Bakar and Z. Othman, "Naïve bayes variants in classification learning," 2010 International Conference on Information Retrieval & Knowledge Management (CAMP), (2010), pp. 276-281.
[9] Jehad Ali, R Khan, et al. "Random Forests and Decision Trees" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3, (2012).