A comparison of Random Forest (RF) and Naïve Bayes (NB) in the classification of breast mass as malignant or benign

Christina Pavlou, Student ID: 190041644

Breast cancer is one of the most frequent cancer types women face every year and is altogether a major public health issue [1]. Predicting if a lump is malignant or benign can be useful for the future in treating individual patients and cancer as a disease.
Aim of the project is to compare the performance of Random Forest and Naïve Bayes in classifying a lump as benign (non-cancerous, the lump does not spread or can infect other areas in the body) or malignant (which means that the lump is made of cancer cells). The dataset, originating from the University of Wisconsin, can be directly loaded from the sklearn python library, through the load_breast_cancer module. The original dataset holds information on 569 incidents and has a total of 30 features.

DATA DESCRIPTION – EXPLORATORY DATA ANALYSIS (EDA)

The dataset in total has 30 attributes, which are all of numerical type, All of the features were collected using Fine Needle Aspiration (FNA) technique [7]. The last column in the dataset is our target and has 569 different instances.
The target variable is binary, classifying whether the lump is cancerous or not, 357 of all the instances are classified as 1, which indicates the lump is benign (B) and the other 212 are classified as 0 meaning malignant (M). There is an imbalance between the classes, category 1 accounts for 62% of the data and category 0 accounts for 38% of the data.
The dataset holds information on the radius, texture, perimeter, smoothness, compactness, concavity, concave points, symmetry, fractal dimension of the mass. For each instance we have values for mean, standard error and worst, which were derived from images from the breast mass.
The dataset does not have any missing or null values. Observing the correlation heatmap on the right, the following variables show a strong correlation: mean radius - worst radius, mean texture - worst texture, mean perimeter - worst perimeter and mean area - worst area, compared to symmetry and fractal dimension that do not display the same level of correlation. Most of the attributes in the dataset are not normally distributed but the majority are positively skewed, as it can be observed from the plots conducted during pre-processing of the data.

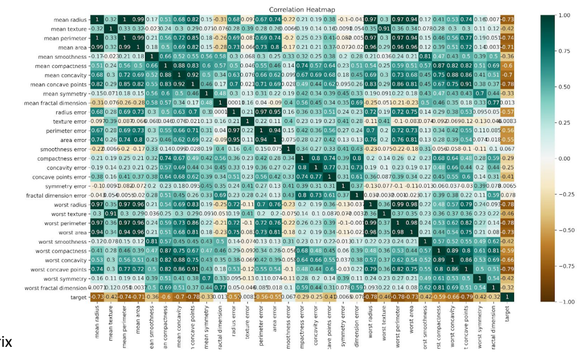

Fig1. Correlation Matrix

NAÏVE BAYES

Naïve Bayes is a supervised machine learning classifier based on Bayes' Theorem.
The method has this name because it makes the (naïve) assumption that all the attributes in the dataset are independent and one does not have an effect on the other attributes. The aforementioned assumption allows the algorithm to train faster than other algorithms. Even though in real world scenarios we do not expect that assumption to be true, Naive Bayes leads to accurate results in a variety of domains. [3] Naïve Bayes works well with natural language processing (NLP) problems, such as spam filtering [4].

Pros:

1. Good performance in multi-class classification.
2. Computes faster compared to other algorithms, for example Random Forests.
3. Adapts well to missing values.
4. Limited number of hyper-parameters to tune.

Cons:

1. Prone to the 'Zero frequency' problem. [5]
2. Assumes that all predictors are independent. In real life, this assumption is violated.

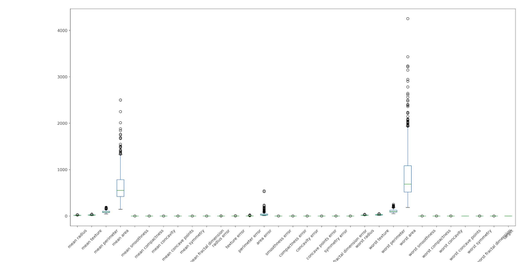

Fig2. Outliers

RANDOM FOREST

Random Forest is a supervised algorithm which can be used to solve both classification and regression problems. With the use of Bootstrapping & Bagging (Bootstrap aggregating), random forests create a number of trees. Random forests is an ensemble method, which employs a number of models that are then aggregated to acquire a single output. [6] Each model containing different data, which originated from our dataset and proceed to train the trees and make predictions. The prediction ranked highest will be the final output [2]. Random Forests are commonly used in health care to diagnose patients using their history.

Pros:

1.Robust to outliers.
2.Less prone to overfitting, due to the construction of multiple trees.
3.Works well on large datasets
4.Works well with non-linear data
5. Possible to have a good performance without tuning the hyper-parameters.

Cons:

1.Requires more computation power, more training time and memory because of the large number of trees it will train. Training time is significantly longer than simple decision trees.
2.As the random forests grow, it can be difficult to interpret.

Fig3. Target value

HYPOTHESIS

Considering the difference between the two algorithms in the treatment of dependency amongst features, Random Forest should perform better than Naive Bayes, since in the classification problem that we are dealing with the attributes are related (not independent of each other).
In terms of computation power and time needed to train the model it is expected that Random Forest will be needing more in comparison to Naive Bayes.
According to similar research, it is expected that a larger number of trees at Random Forest will have a better performance [8].

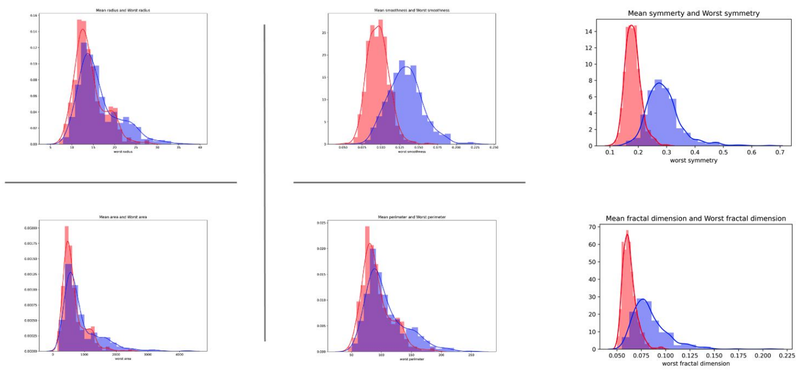

Fig4. Distribution Plots

GENERAL METHODOLOGY

As an initial step, we train the model using the 70% of the data as training data, and the rest as a test set. However, later we experiment with taking 50% and 90% as training data. The hyperparameters of both models are tuned using Bayes Optimization [10], and the models are later trained with the best selected ones.

NAÏVE BAYES PARAMETERS

The Naïve bayes model was firstly trained using the dataset which was split during EDA (Exploratory data analysis) using Python. Training and testing initially had a ratio of 70 % to 30 %.
In order to find the best training setting for the Naive Bayes a number of different trainings were tested. Starting with a simple train with no parameter turning we moved on to a slightly different version where we tuned the prior probability to be the percentage of the class imbalance we have observed in the dataset, however this did not seem to increase the classification results.
Bayesian optimisation was used to find the best hyper-parameters suggested to train the model. When training the model because the data was not standardised, the only optimisation which was shown was the distribution name. In order to get a suggesting for kernel as well we took a step back into pre-processing of the data using python to standardise the dataset and train the model.
Another consideration into training the model was the partition of the data into training and testing (cvpartion, 'HoldOut'). HoldOut partitions used were the following: 0.5, 0.3 and 0.1. As shown in the table Fig9 bellow, best performance was observed when the model was training using the 0.1 HoldOut and also it required slightly less time to train.
After the mentioned step the results from Bayesian Optimisation were the following: DistributionNames: kernel, Width : 0.25236

RANDOM FOREST PARAMETERS

Random Forest model was trained using the TreeBagger method and it was set to have out of bag prediction on, therefore some of the observations were kept separate to use for testing. In order to increase performance of the model the following hyper- parameters were tuned by setting ranges for each one: number of predictors, number of trees and the minimum leaf size, then using Bayesian Optimisation to find the optimal combination amongst the set ranges and save the best hyper-parameter to use for training. As it can be observed from the figure on the right (Fig5) as the number of tree increases, out of bag error decreases. Approximately 50 trees show a significant improvement which follows a gradual decrease in error as the trees increase.
Training using Bayesian optimisation indicated the following numbers to best train the model on.
Similarly to Naive Bayes, when we had the optimal hyper-parameters from our optimiser, we tested how splitting the data in different ratios will affect the model. Using cvpartition to create 3 different datasets at: 0.5, 0.3 and 0.1.(Fig.10)
Results from Bayesian Optimisation (bestHyperparameters = XAtMinObjective) for the optimal HoldOut ratio: tree_num : 444, min_leaf_size : 1 and num_predictors : 10

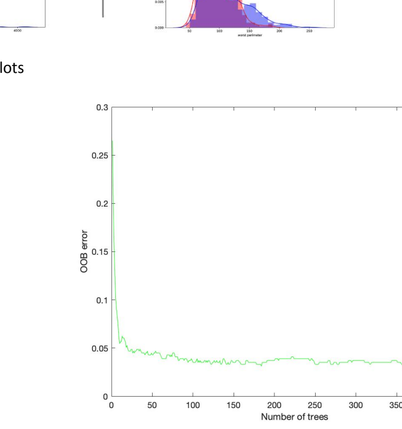

Fig5. Out of bag Error/ number of trees, Random Forest

ANALYSIS

Consistent with the hypothesis and literature review random forest took more time to train when compared to Naive Bayes and also had a better performance when compared. (approximately random forest needed a minute longer than naive bayes to train each time,hyper-parameter tuning is not included in training time for Random Forest). This is also highlighted in figures 9 and 10 were we can see the metrics for the different HoldOut training options.
On a side note both models were trained on an imbalanced dataset (Can be seen on the distribution of the target class plot, Fig3).

Bayesian Optimisation for Naive Bayes could be employed while training the model compared to Random Forest which had to be employed separate from the training of the model.
Both models used Bayesian Optimisation to optimise, Naive Bayes required from the dataset to first be standardise compared to Random Forest which did not require this step. Therefore for the final trained models, Naive Bayes uses a standardised version of the dataset, random forest uses the dataset as it was provided from the python library and exported using python.

After tuning the models to the best hyper-parameter indicated, we partitioned the data using the HoldOut method in MATLAB which partitions the data randomly into the indicated ratio to test how would that affect the performance.
For both of the models the best split of the data was the 0.1 options, ROC curves were plotted in figures 11 and 12, we can see the performance of both models for every HoldOut option used in training in the tables in figures 9 and 10.
We see that Random Forest has an accuracy of 1 which could be due to simplicity of the dataset.

Fig6. Confusion Matrix for Random Forest

Holdout	Training Time	AUC	Accuracy	Error
0.5	00:00:52	0.9410	0.9472	0.0528
0.3	00:00:50	0.9530	0.9529	0.0471
0.1	00:00:46	0.9619	0.9643	0.0357

Fig9.Naive Bayes performance metrics

Holdout	Training Time	AUC	Accuracy	Error
0.5	00:01:21	0.9652	0.9613	0.0387
0.3	00:01:30	0.9828	0.9824	0.0176
0.1	00:01:38	1	1	0

Fig10. Random Forest performance metrics

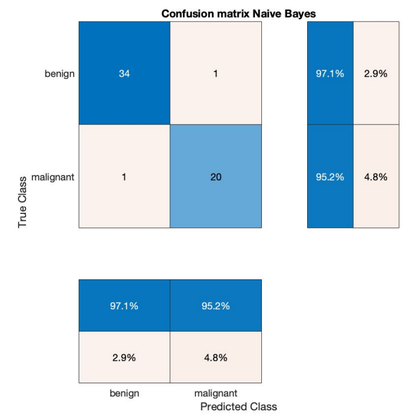

Fig7. Confusion Matrix for Naïve Bayes

LESSONS LEARNED

- Both algorithms performed well in classifying the instances
- Random Forests take more time to be tuned
- In this concept, both performed well but the hypothesis prior to training the models supporting Random Forest will perform better compared to Naive Bayes were true.

FUTURE WORK

• In terms of future work, feature selection can be applied ('OOBPredictorImportance' set to 'on'), have a selection of 5, 10 or 15 which comprise the most important features on making a prediction and train the models again. After training the models we can use a number of classification metrics to observe the changes in the ability of the algorithm to classify correctly the data. (Fig. 8)
• Another step which could improve the performance of the models would be to fix the imbalance present in the dataset, as we can observe from the distribution of the target variable, malignant class has less instances compared to benign. A way to overcome this is by applying the SMOTE technique on the dataset before training the models, these can be used to over-sample the data, not only to balance the dataset (target attribute) but also to increase by a few instance the dataset that we have since it is small to start with. Data augmentation will allow us to test the performance of Random Forest at a bigger dataset.
• Apply regularisation techniques on the models.
• Other algorithms could be used to train the model along the two used in the project, for example, Support Vector Machines (SVMs) are a popular choice in medical care problems. Some domains where they are used are the following: classification of proteins, image segregation and text categorisation.

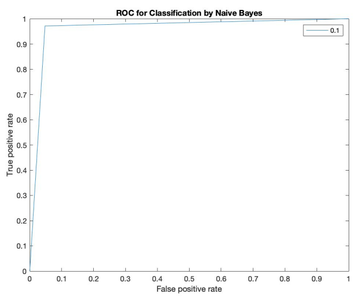

Fig11. Naive Bayes ROC curve, HoldOut: 0.1

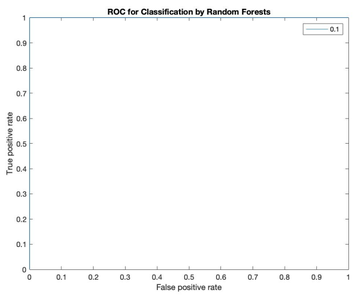

Fig12. Random Forest ROC curve, HoldOut: 0.1

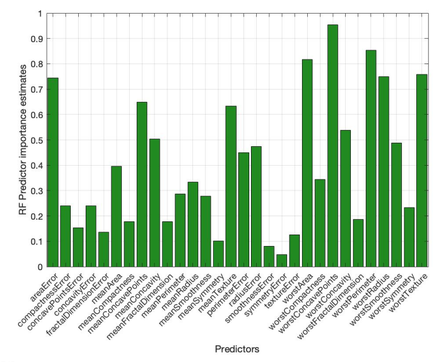

Fig8. Feature Selection

References:

[1] S. Rebecca, K. D. Miller and A. Jemal, "Cancer statistics," CA: a cancer journal for clinicians, vol. 66, no. 1, pp. 7 - 30, 1 Jan 2016.
[2] O. Mbaabu, "Introduction to Random Forest in Machine Learning," 11 12 2020. [Online]. Available: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/. [Accessed 29 11 2021].
[3] K. P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012, p. 1096.
[4] K. Gunasekara, "Spam Filtering Emails: An Approach with Natural Language Processing," 24 Oct 2018. [Online]. Available: https://kasumisanchika.medium.com/spam-filtering-emails-an-approach-with-natural-language-processing-15abb46dd7d5.
[5] S. Ray, "6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R," 11 Sep 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Accessed 29 Nov 2021].
[6] J. Kelleher, B. Mac Namee and A. D'Arcy, Fundamentals of machine learning for predictive data analytics: Algorithms Worked Examples, and Case Studies, The MIT Press, 2015.
[7] K. Lemons and I. Wickramasinghe Rathnathungalage, "A Comparison Between Naïve Bayes and Random Forest to Predict Breast Cancer," IJURCA: International Journal of Undergraduate Research & Creative Activities, vol. 12, no. 12, 13 Oct 2020.
[8] L. Vig, "Comparative Analysis of Different Classifiers for the Wisconsin Breast Cancer Dataset," Open Access Library Journal, vol. 1, 19 Sep 2014.
[9] R. Chatterjee, "Top 6 AI Algorithms In Healthcare," 18 April 2020. [Online]. Available: https://analyticsindiamag.com/top-6-ai-algorithms-in-healthcare/. [Accessed 2 Dec 2021].
[10] mathworks, "Tune Random Forest Using Quantile Error and Bayesian Optimization," [Online]. Available: https://uk.mathworks.com/help/stats/tune-random-forest-using-quantile-error-and-bayesian-optimization.html. [Accessed 10 Dec 2021].