Supervised Learning algorithms

https://github.com/nossa-y/supervised_learning

Nossa Iyamu
Telecom SudParis
Institut Polytechnique de Paris

Abstract

In this study, we embark on a comparative exploration of various supervised classification algorithms, focusing on decision trees and aggregation methods. Through a series of practical tests and analyses utilizing real-world datasets, we delve into the subtleties of classification performance and algorithmic efficacy. Leveraging datasets such as Fisher's Iris, alongside the MNIST digits dataset and the Breast Cancer dataset from scikit-learn, we meticulously analyze the performance nuances of decision trees, random forests, and ensemble methods such as bagging and boosting. By scrutinizing hyperparameters and algorithmic configurations, our investigation aims to provide insights into the comparative advantages and limitations of each approach in the realm of supervised classification.

Introduction

In the realm of machine learning, the quest for optimal supervised classification algorithms remains a pivotal endeavor, particularly in domains where accurate pattern recognition is paramount. Decision trees, alongside ensemble methods such as random forests, bagging and boosting, stand as stalwarts in the arsenal of classification methodologies. In this research, we embark on a comparative exploration of these algorithms, aiming to unravel their intricacies and discern their comparative advantages in discerning patterns within diverse datasets.

Grounded in practical tests utilizing real-world datasets such as Fisher's Iris, MNIST digits, and Breast Cancer dataset, our investigation traverses the landscape of multi-class and binary classification scenarios. Through meticulous experimentation and analysis, we scrutinize the performance nuances of decision trees and aggregation methods, shedding light on their efficacy in discerning patterns indicative of various phenomena, from iris species classification to tumor malignancy detection.

By delving into hyperparameters, algorithmic configurations, and parameter visualization, our exploration aims to provide insights into the comparative advantages and limitations of each algorithmic approach. Through this comparative analysis, we endeavor to equip practitioners and researchers with valuable insights to inform their choice of classification methodologies in real-world applications.

1 Decision Tree

1.1 The Model

A decision tree is a foundational concept in machine learning used for both classification and regression tasks. It is essentially a tree-like model where each internal node represents a feature or attribute, the branches represent the decision rules based on these features, and each leaf node represents the outcome or label. The construction of a decision tree involves recursively splitting the dataset into subsets based on the feature that provides the best split according to certain criteria, such as entropy or Gini impurity for classification tasks, or variance reduction for regression tasks. Decision trees offer interpretability and are easy to visualize, facilitating an understanding of how decisions are made.

One of the primary advantages of decision trees is their interpretability. The transparent nature of decision trees allows for easy visualization and comprehension of the decision-making process. By examining the path from the root node to the leaf nodes, one can gain insights into the factors influencing predictions. Additionally, decision trees can handle both numerical and categorical data, making them versatile for a wide range of applications.

1.2 The Iris dataset

The Iris dataset from scikit-learn serves as a fundamental benchmark for evaluating and comparing machine learning algorithms, particularly those designed for classification tasks. With its standardized structure and easily interpretable features, the Iris dataset provides researchers with a reliable foundation for assessing the efficacy of various classification approaches.

Comprising 150 samples, each characterized by four attributes—namely, 'sepal length (cm)', 'sepal

width (cm)', 'petal length (cm)', and 'petal width (cm)'—the dataset encapsulates numerical data in float format. The dataset's target variable for the classification of iris species consists in three distinct classes: Setosa (0), Versicolour (1), and Virginica (2). Notably, the dataset maintains a balanced distribution across classes, with precisely 50 samples allocated to each category. The balanced representation of classes within the dataset ensures equitable evaluation of classification algorithms, minimizing biases stemming from class imbalances.

To gain a comprehensive understanding of the dataset, we print every five rows, focusing on the dataset's features. This approach allows us to delve into the characteristics of the data and uncover any discernible patterns. Notably, we observe in Figure 1 a striking disparity among samples belonging to different classes. Such visual exploration provides valuable insights into the underlying structure of the data and highlights the potential efficacy of these features in classification tasks.

	petal width	petal length	sepal width	sepal length
Class 0	between 0 and 1	between 1 and 2	not relevant	between 4 and 5
Class 1	between 1 and 2	between 3 and 5	not relevant	between 5 and 7
Class 2	around 2	between 5 and 6	not relevant	between 6 and 7

Figure 1: Prompt analysis of every five rows of the Iris dataset

We can hypothesize that the data may be linearly separable based on our observations. To illustrate this hypothesis, Figure 2 presents representative plots that provide a visual perspective on the separability of the data. These graphs depict the distribution of samples across different classes in feature space, allowing us to discern any discernible patterns or separability between classes. Through these visualizations, we aim to provide empirical evidence supporting our hypothesis of potential linear separability in the dataset.

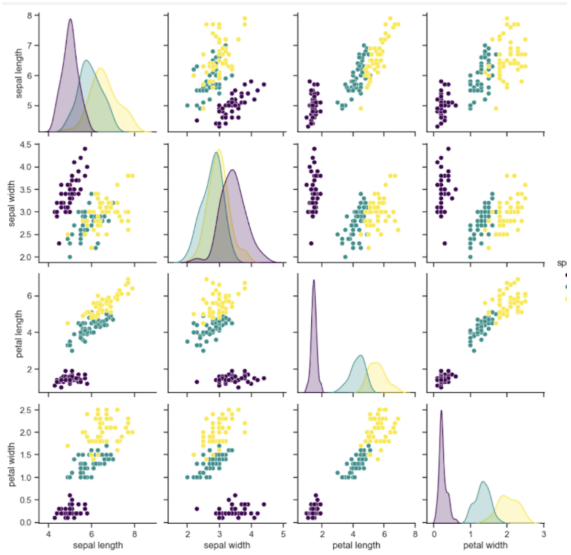

Figure 2: Representative plots illustrating potential relationship between combinations of features and the classification of samples.

The plots of petal width against petal length and petal width against petal width reveal distinct clusters of data points that can be separated by diagonal lines. Each split represents data points sharing similar characteristics, corresponding to samples from a particular class. This observation suggests that the data indeed exhibit linear separability. The hypothesis we formulated for a fifth of the data can be validated for the whole dataset.

In scenarios where linear separability is established, models such as Linear Regression or Support Vector Machines (SVM) may be suitable for classification tasks. These models leverage the linear relationships between features to accurately classify samples into different classes. By exploiting the inherent separability of the data, these models can effectively delineate boundaries between classes, facilitating robust classification performance.

1.3 Experiments

The Figure 3 shows how a decision tree behave on our dataset.

Figure 3: Diagram depicting the behavior of the decision tree model on the Iris dataset.

We briefly explain how the decision tree operates on our dataset:

1st Depth of the Tree:

- If the 4th attribute (petal width) is less than 0.75 cm, the class is Setosa.
- Otherwise, the class is Virginica.

2nd Depth of the Tree:

- If the petal width is above 0.75 cm, and the 3rd attribute (petal length) is less than 4.95 cm, the class is Versicolor.
- Otherwise, the class is Virginica.

3rd Depth of the Tree:

- If the petal width is above 0.75 cm and the petal length is under 4.95 cm:
  - If the petal width is less than 1.65 cm, the class is Versicolor.
  - Otherwise, it is Virginica.
- If the petal width is above 0.75 cm and the petal length is above 4.95 cm:
  - If the petal width is less than 1.75 cm, the class is Versicolor. (*Note: Mistake on the draw*)
  - Otherwise, it is Virginica.

In each frame, the tab $[x, y, z]$ indicates the number of samples that belong to each class, where x represents the number of samples for class 0, y for class 1, and z for class 2.

We can infer that the decision tree achieves 100% accuracy on the training data, accurately assigning the correct label to each sample. Upon evaluation of the test data, which constitutes 30% of the total dataset, corresponding to 45 data points, the overall accuracy is determined to be 97.8%, with a weighted accuracy of 98%. These results are highly promising, indicating excellent performance. Nonetheless, it is essential to analyze the confusion matrix (see Figure 4)provided below to identify any misclassified data points.

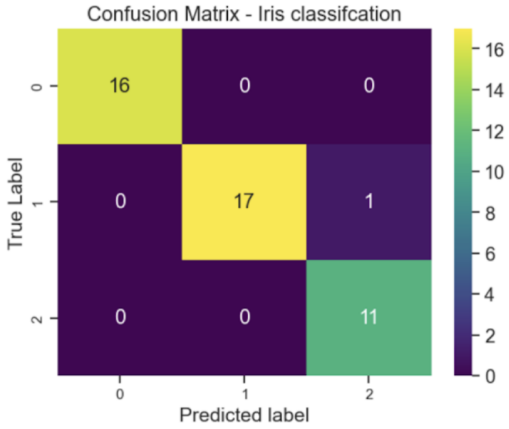

Figure 4: Confusion matrix displaying the performance of the decision tree model on the test data, highlighting any misclassified data points.

We observed that one data point labeled as versicolor was misclassified as virginica, while all other data points were correctly classified. To enhance the model's performance further, we can experiment with various hyperparameters. Some of the key hyperparameters that can be adjusted to potentially improve the model's accuracy include:

1. Criterion: This parameter determines the function used to measure the quality of a split. Options include "gini" for Gini impurity, "entropy" for Shannon information gain, and "log_loss" for logarithmic loss. 2. Splitter: This parameter specifies the strategy used to choose the split at each node. It can be set to "best" to select the best split or "random" to choose the best random split. 3. Max Depth: This parameter controls the maximum depth of the decision tree. Setting it to a specific integer restricts the depth of the tree, while setting it to "None" allows nodes to expand until all leaves are pure or contain fewer samples than the specified minimum samples for splitting. 4. Min Samples Split: This parameter determines the minimum number of samples required to split an internal node. It can be specified as an integer, where it represents the minimum number of samples, or as a float, where it represents a fraction of the total number of samples.

Some of the key hyperparameters that significantly impact the model in this problem are max_depth and min_samples_split. Using GridSearchCV, we can systematically search through different combinations of hyperparameters to find the optimal values that maximize the accuracy score. After conducting the grid

search, the best suggested values for these hyperparameters are: max_depth: 10 min_samples_split: 5 With these hyperparameters, the model achieved an accuracy score of 0.977.

Figure 5 shows the breakdown of how the decision tree model split the data.

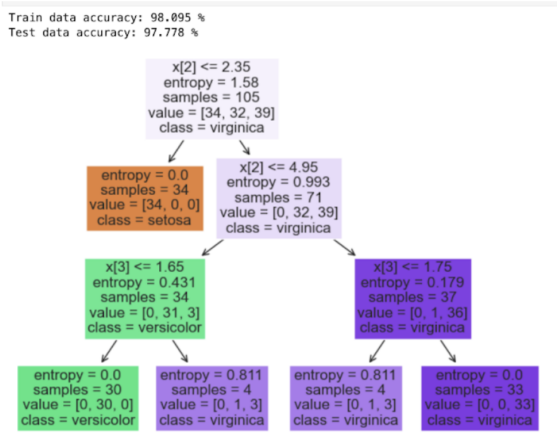

Figure 5: Visualization presenting the breakdown of how the decision tree model splits the data into different classes.

In comparison to our previous model, we observe that the test accuracy remains unchanged, indicating consistent performance in predicting unseen data. However, there is a reduction in train accuracy, suggesting a decrease in overfitting. Despite this improvement, overfitting is still evident in the model. Further optimization may be required to mitigate this issue and enhance the generalization capability of the model.

2 Random Forest

2.1 The Model

Our previous results show that decision trees can suffer from overfitting, especially when they are deep and complex. This is where ensemble methods like random forests come into play. Random Forest is an ensemble learning method built upon decision trees. It operates by creating a multitude of decision trees during training and then averaging their predictions for classification tasks or taking the average prediction for regression tasks. Each tree in the forest is trained on a random subset of the training data (sampling with replacement), and at each split, a random subset of features is considered. By constructing multiple trees and combining their predictions, random forests mitigate overfitting and enhance generalization performance compared to individual decision trees. They are highly adaptable and robust, suitable for various tasks, and capable of handling high-dimensional data. Additionally, random forests provide a measure of feature importance, highlighting which features are most influential in making predictions. They are computationally efficient and can effectively handle large datasets with high dimensionality.

2.2 Algorithm Benchmarking on Iris dataset

As anticipated, when using a single estimator in the Random Forest model and maintaining all other parameters at the same values as those used for the decision tree, we achieve comparable accuracy levels. Since only one Decision Tree is employed to make a decision, there is no aggregation, effectively resulting in a Decision Tree model. The input hyperparameters for this model are as follows:

n_estimators: 1 criterion: "entropy" min_samples_split: 5 max_depth: 10 random_state: 0 The accuracy achieved on the training data is 98.095%, while on the test data, it is 97.778%. On Figure 6 we proceed by increasing the number of estimators.

Estimators	Accuracy
1	Train data accuracy: 98.095 % Test data accuracy: 97.778 %
10	Train data accuracy: 97.143 % Test data accuracy: 95.556 %
100	Train data accuracy: 98.095 % Test data accuracy: 97.778 %
1 000	Train data accuracy: 98.095 % Test data accuracy: 97.778 %
10 000	Train data accuracy: 98.095 % Test data accuracy: 97.778 %
100 000	Train data accuracy: 98.095 % Test data accuracy: 97.778 %

Figure 6: Table displaying the benchmarking results of the Random Forest algorithm on the Iris dataset, showcasing the impact of varying the number of estimators on model accuracy.

We achieve maximum accuracy when n_estimators is set to 100, with the exclusion of a single estimator. However, beyond 100 estimators, increasing the number of estimators does not notably impact predictions. Surprisingly, selecting 10 estimators yields poorer results than using only one. This phenomenon can be attributed to the nature of random forests, where decision trees are based on random partitions of samples and attributes. In contrast, a single decision tree considers every sample and attribute when choosing which attribute to use for splitting the data. A sufficiently high number of trees is required to capture diversity and exploit the results effectively. Otherwise, the ensemble method may still be susceptible to capturing noise and variance present in the data. Consequently, by increasing the number of estimators, we observe a reduction in overfitting, as anticipated with this model.

2.3 Feature engineering

As elaborated previously, the Random Forest model offers the capability to identify the most influential features for prediction. Delving into this aspect provides an avenue for enhancing model performance. This section elucidates how manipulating attributes can affect accuracy. Employing an importance method allows us to discern which features are pivotal for predictive outcomes. By selectively choosing attributes for partitioning, we can mitigate the impact of irrelevant attributes and optimize computational efficiency.

During the prediction process, our Random Forest Model assesses the significance of various attributes in predicting the target variable. It quantifies how predictions fluctuate with changes in attributes and incorporates these insights into the prediction mechanism. The plot on Figure 7 illustrates the importance of features for the model, ordered in descending order of significance:

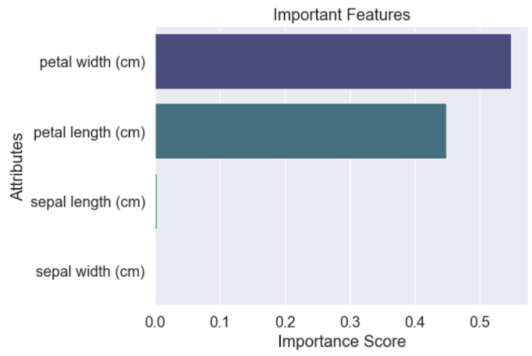

Figure 7: Importance of features for the Random Forest model, ordered in descending order of significance.

Indeed, our findings corroborate the initial hypothesis posited at the outset of the study: attributes such as petal width, petal length, and sepal length contain substantial discriminatory information regarding the data and its respective class, contingent upon the attribute's interval. Conversely, sepal width emerges as inconsequential, offering negligible predictive insight.

When solely utilizing 'sepal width' as the attribute for class prediction, the resulting accuracies are as follows: Train data accuracy: 60.000%
Test data accuracy: 51.111%. The table on Figure 8 illustrates the performance metrics when each attribute is considered independently.

Attribute	sepal width	sepal length	petal length	petal width
Importance	0.020535	0.086650	0.415814	0.477001
Train data accuracy	60.000 %	80.952 %	96.190 %	95.238 %
Test data accuracy	51.111 %	60.000 %	91.111 %	97.778 %

Figure 8: Performance metrics when each attribute is considered independently.

The analysis demonstrates that accuracy improves with the selection of features possessing higher importance. Although the achieved accuracy may not match that obtained when utilizing all attributes in the Random Forest model, there is a noticeable reduction in overfitting. Specifically, when 'petal width' is chosen as the sole attribute, overfitting is entirely absent. This approach effectively disregards any potential noise present in the data.

We deduce that it may be advantageous to exclusively consider the most pertinent features, namely petal width, petal length, and sepal length. Figure 9 evaluates the model's performance with these adjustments.

Attribute	'petal width' and 'petal length'	'petal width' and 'sepal length'	'petal length' and 'sepal length'	'petal width', 'petal length' and 'sepal length'
Train data accuracy	99.048 %	98.095 %	99.048 %	100.000 %
Test data accuracy	97.778 %	95.556 %	93.333 %	97.778 %

Figure 9: Evaluation of the model's performance with adjustments ignoring 'sepal width'.

Focusing solely on 'petal width' and 'petal length' yields the highest accuracy for the model. We have achieved comparable accuracy to that obtained when utilizing all four attributes but with only two, thus reducing computational requirements.

This methodology becomes particularly significant when dealing with large datasets containing numerous attributes. In such cases, considering only a subset of attributes often yields superior accuracy. For instance, if the dataset includes an irrelevant attribute such as "color of the flower petal," including this attribute would likely degrade the accuracy of the model. Flower petal color is not relevant for distinguishing between different species of iris, and including it would introduce noise into the dataset without providing meaningful information for classification. Therefore, it would be advisable to exclude such irrelevant attributes from the dataset.

Incorporating irrelevant attributes in the dataset can lead to overfitting, where the model learns noise as if it were meaningful patterns. Consequently, the model's generalization ability to new, unseen data diminishes, resulting in poorer predictive performance. Thus, careful feature selection is crucial for improving model accuracy and generalization.

3 Decision tree vs Random Forest

3.1 The Digit dataset

Now, we delve into the handwritten digit dataset from scikit-learn, commonly known as the digit database. This dataset serves as a standard benchmark for evaluating and comparing machine learning algorithms, particularly those designed for classification and image recognition tasks. The dataset comprises images of handwritten digits, each represented as an 8x8 pixel grayscale image. Due to the relatively small size of the images, they can be easily visualized and analyzed, making the dataset suitable for exploring visualization techniques and understanding the behavior of machine learning models.

The dataset consists of labeled samples, with each sample corresponding to a handwritten digit ranging from 0 to 9. There are approximately 180 samples for each digit class, resulting in a total of 1797 samples. The distribution of samples per class is as follows:

{ 0: 178, 1: 182, 2: 177, 3: 183, 4: 181, 5: 182, 6: 181, 7: 179, 8: 174, 9: 180 }

Each sample is represented by 64 features corresponding to the pixels of the 8x8 grayscale image. These features are labeled as 'pixel_0_0', 'pixel_0_1', ..., 'pixel_7_7', with each pixel value ranging from 0 to 16. It is important to note that these pixel values do not directly represent the intensity of the pixel but rather indicate a scale of grayscale shades specifically tailored for the images. This scaling or normalization process may have been applied during the creation of the dataset.

The small scale of the images and the limited range of grayscale levels have a significant impact on the quality of the data, as demonstrated in Figure 10.

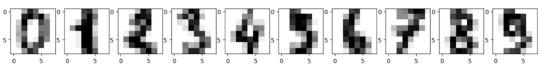

Figure 10: Visualization of a sample of the written digits dataset.

The Decision Tree model demonstrates correct performance in identifying each digit, achieving a perfect 100% accuracy on the training data, and a respectable 82.76% accuracy on the test data, these results suggest further examination.

3.2 Algorithm Benchmarking on Digit dataset

In order to compare the performance of the Decision Tree model with the Random Forest Model on this dataset, we conducted a comprehensive evaluation using the following steps:

1. For each value of N ranging from 0 to 100: - Randomly select train and test data. - Apply the Decision Tree model. - Record the performance metrics.

2. Average the performance metrics obtained from the 100 runs.

3. Repeat this process 10 times.

4. Output the average accuracy and standard deviation of the 10 runs.

This methodology allows us to assess the average performance of 1000 randomly sampled Decision Tree models. In Figure 11, the first value presented indicates the average accuracy percentage, while the second value represents the standard deviation.

1st pred	2nd pred	3rd pred	4th pred	5th pred	6th pred	7th pred	8th pred	9th pred	10th pred	AVG	Random Forest
82.467 0.008	82.464 0.008	82.574 0.009	82.454 0.007	82.476 0.008	82.489 0.008	82.537 0.008	82.667 0.008	82.648 0.007	82.538 0.009	82.538 0.008	95.572 0.005

Figure 11: Accuracy and standard deviation of 10 runs for Decision Tree, average of these accuracies and Random Forest model accuracy on the digit dataset.

Based on our analysis, we can conclude that a Random Forest with 300 estimators (i.e., 300 decision trees) outperforms the average accuracy obtained from 1000 randomly sampled decision trees. Several factors contribute to this observation:

1. Decision trees in the Random Forest model operate independently, with each tree providing an individual prediction. In contrast, the average of 1000 random decision trees does not consider any consensus among the trees.

2. Random Forests make predictions based on the average decision of all trees, where each tree is trained on a different subset of the dataset. This diversity helps in capturing a broader range of patterns and reducing overfitting.

3. While decision trees in both approaches are trained on random subsets of the data, every tree in the Random Forest model operates on randomly selected partitions of features, contributing to better generalization.

To determine the optimal model, it is essential to find the optimal number of trees for the Random Forest model. Plotting the accuracy as a function of the number of trees allows us to visualize this relationship. (see Figure 12)

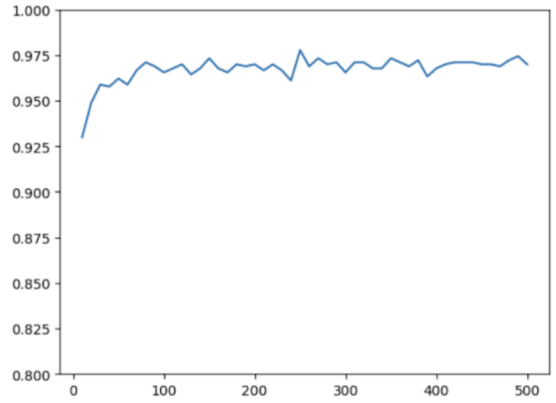

Figure 12: Accuracy as a function of the number of trees for the Random Forest model.

The analysis of the accuracy as a function of the number of trees reveals an interesting trend. Initially, as the number of trees in the Random Forest model increases, the precision also tends to increase. However, we observe a peak in precision when selecting 250 trees, with an accuracy of 96.89%. This accuracy is slightly higher than that obtained with 300 trees.

It is noteworthy that further increasing the number of trees beyond this point may not necessarily yield significant improvements in accuracy. Moreover, there is a risk of overfitting, where the model learns noise in the training data as if it were meaningful patterns, leading to reduced generalization performance on unseen data.

Therefore, selecting 250 trees strikes a balance between accuracy, avoiding overfitting, and reducing computing power with a high accuracy of 96.89% deemed satisfactory for our classification task.

4 Bagging versus Boosting

4.1 The Models

Bagging and Boosting are two ensemble learning techniques to improve accuracy of predictive models

Bagging, short for Bootstrap Aggregating, operates by generating multiple subsets of the original training dataset through bootstrap sampling. This process involves randomly selecting samples from the dataset with replacement to create diverse subsets. Each subset is then used to train a base model, such as a decision tree. The predictions from these models are aggregated, typically through majority voting for classification tasks or averaging for regression tasks, to produce the final prediction. Bagging effectively reduces overfitting by introducing diversity among the base models and stabilizing their predictions.

Boosting, on the other hand, follows a different strategy. Instead of training models independently, boosting trains a sequence of models sequentially. Each subsequent model focuses on correcting the errors made by its predecessors. During training, more emphasis is placed on examples that were misclassified by the previous models, thereby prioritizing difficult-to-classify instances. The final prediction is obtained by combining the predictions of all the models, with higher weights assigned to those models that perform better on the training data. Boosting aims to iteratively improve the overall performance of the model by focusing on challenging examples and reducing bias.

While bagging and boosting share the goal of improving model performance through ensemble learning, they differ in their approach. Bagging emphasizes variance reduction and overfitting prevention by training multiple models independently, while boosting focuses on bias reduction and accuracy improvement by sequentially refining the model's performance on difficult examples.

4.2 The Breast Cancer dataset

The Breast Cancer dataset from scikit-learn provides a standardized set of features and classes for binary classification tasks, particularly in diagnosing breast cancer. The dataset comprises 569 samples, each with 30 features of float datatype. These features encapsulate various characteristics of cell nuclei obtained from digitized images of breast cancer biopsies.

The classes are represented as follows: class 0 corresponds to 'malignant,' indicating cancerous cells, while class 1 corresponds to 'benign,' representing non-cancerous cells. Each feature provides crucial insights into the morphological and textural properties of cell nuclei, which are vital for accurate classification.

For instance, 'Mean Radius' signifies the average distance from the center of the nucleus to its perimeter, while 'Mean Texture' denotes the standard deviation of gray-scale values in the image. Similarly, 'Mean Perimeter,' 'Mean Area,' and other features capture essential aspects such as smoothness, compactness, concavity, and symmetry of the nucleus.

The dataset also includes measures of error and "worst" or largest mean values for each feature, offering comprehensive information about the characteristics of the cell nuclei. These features play a pivotal role in developing robust machine learning models for breast cancer diagnosis, enabling accurate classification of malignant and benign cases based on the extracted features from biopsy images.

4.3 Algorithm benchmarking on Breast Cancer dataset

Model	Decision Tree	Bagging	Random Forest	AdaBoost
Accuracy (%)	94.152	95.906	97.076	97.661

Figure 13: Performance comparison of Decision Tree, Bagging, Random Forest, and AdaBoost classifiers on the breast cancer dataset.

As demonstrated in the preceding analysis, Figure 12 validates our initial expectation regarding the superior performance of Random Forest over Decision Tree classifiers was validated. However, it is noteworthy that the Bagging classifier outperforms the Decision Tree classifier, as it aggregates the predictions of multiple decision trees trained on random partitions of the training dataset. This approach enhances the accuracy by reducing variance and mitigating overfitting.

Furthermore, the Random Forest classifier exhibits superior performance compared to the Bagging classifier due to the additional layer of diversity introduced during training. Notably, each decision tree in the Random Forest model utilizes a random subset of features, in addition to the random partitioning of the training dataset. This increased diversity enhances the robustness and generalization capability of the model.

Contrarily, the AdaBoost classifier surpasses the Random Forest classifier's performance in this scenario, attributed to its boosting technique. In AdaBoost, weak learners, typically decision trees, are sequentially trained, with each subsequent tree focusing on correcting the errors made by its predecessors. This iterative process enables AdaBoost to learn complex patterns and achieve superior accuracy.

While ensemble techniques like Bagging, Random Forest, and AdaBoost are widely employed in scientific research, other ensemble methods, such as Stacking, offer alternative approaches to improving model accuracy. Stacking involves combining the predictions of multiple base models, often employing diverse algorithms like Decision Trees, SVMs, or neural networks, and utilizing a meta-model, such as linear regression or neural networks, to learn optimal combinations of the base models' predictions. This comprehensive approach further enhances predictive performance by leveraging the strengths of various algorithms and mitigating their individual weaknesses.

4.4 Adaboost

In our previous experiment, AdaBoost demonstrated superior performance compared to other ensemble techniques. Now, our objective is to further enhance the performance of this model by optimizing its hyperparameters.

4.4.1 Depth

First, on Figure 14, we explore the impact of adjusting the depth of the Decision Tree Classifier, which serves as the weak learner in the AdaBoost ensemble method.

Depth	1	3	10	30	100
Accuracy (%)	97.661	97.076	94.152	93.567	93.567

Figure 14: Impact of adjusting the depth of the Decision Tree Classifier on AdaBoost performance.

The experimentation demonstrates that employing decision trees with lesser depths results in improved performance for the AdaBoost algorithm. This highlights a key characteristic of the model: AdaBoost tends to achieve higher accuracy when utilizing weaker models as estimators.

This phenomenon is attributed to the algorithm's practice of assigning greater weight to misclassified examples, thereby directing its focus towards challenging instances. Weak learners, such as shallow decision trees, prove effective in capturing these complex cases because they tend to make errors on different subsets of the data. Consequently, AdaBoost can iteratively correct these errors and refine its predictions. The efficacy of using weaker models lies in their ability to mitigate overfitting and reduce sensitivity to noisy data. This enables AdaBoost to achieve better generalization performance on unseen data.

Moreover, weak learners, characterized by their simplicity and limited expressiveness, contribute to the diversity within the ensemble by offering distinct perspectives on the data. This diversity often leads to improved overall performance when combined.

4.4.2 'Weak model' and number of estimators

Within the realm of "weak learners," SVM Classifier and Logistic Regression Classifier are notable contenders.

The SVM classifier operates by identifying the optimal hyperplane that delineates distinct classes within the feature space. Additionally, it possesses the capability to handle non-linear classification tasks by employing kernel functions such as linear or polynomial kernels.

On the other hand, the Logistic Regression Classifier models the probability, determined through maximum likelihood, that a given input belongs to a particular class. This is accomplished by employing a logistic function, typically the sigmoid function, to map inputs to class probabilities.

In the table provided on Figure 15, we evaluate the accuracy of AdaBoost models employing different weak learners, while also varying the number of estimators. As a point of clarification, the number of

estimators denotes the quantity of models aggregated to formulate a prediction.

model ╲ n estimators	5	50	100	500
Decision Tree accuracy (%)	95.906	97.661	98.246	98.246
SVM accuracy (%)	97.661	95.906	91.813	63.158
Logistic Regression accuracy (%)	98.830	98.246	97.661	97.076

Figure 15: Evaluation of AdaBoost models employing different weak learners, varying the number of estimators.

The experimentation reveals notable insights. In the case of the first model, a discernible improvement in accuracy is observed with an increase in the number of estimators. This enhancement can be attributed to the augmented consideration of misclassified classes facilitated by a higher number of models. Consequently, the overall model performance is enhanced. However, it becomes apparent that beyond a certain threshold, notably at around 100 estimators, the model's improvement plateaus. Moreover, training a large number of estimators can impose significant computational demands. Thus, in this scenario, an appropriate choice for the number of estimators would be 100.

Conversely, for the other models, the performances exhibit a decline as the number of estimators is increased. Notably, there is an optimal point reached at approximately n_estimators = 5, beyond which the accuracy diminishes. This phenomenon can be attributed to overfitting. With an increase in the number of estimators, AdaBoost introduces greater complexity, thereby elevating its capacity to fit the training data. Given that the dataset is relatively small and noisy, escalating the number of estimators exacerbates overfitting tendencies.

4.4.3 Learning Rate

The learning rate plays a crucial role in controlling the pace at which weights are adjusted during each iteration of the learning process.

In the Figure 16, we assess the accuracy of the Decision Tree model under various combinations of 'n_estimators' and 'learning_rate', while maintaining a fixed 'max_depth' value of 1.

learning rate ╲ n estimators	5	50	100	500
0.5	96.491	95.906	96.491	98.246
1	95.906	97.661	98.246	98.246
2	89.474	66.667	69.006	69.006

Figure 16: Assessment of AdaBoost performance under various combinations of 'n_estimators' and 'learning$_r ate'$.

We observe that the choice of hyperparameters significantly influences the accuracy of the algorithm. Notably, certain combinations stand out as particularly effective, as indicated by the overlined values in the experiment. This aligns with our expectations, as with a smaller number of estimators, it becomes crucial to carefully consider the impact of misclassifications. Conversely, with a larger number of estimators, the algorithm can tolerate some errors with confidence, given the redundancy of models available to discern their significance. Therefore, a higher learning rate paired with a smaller number of estimators can still produce satisfactory results, as evidenced by our analysis.

Thus far, our second-best model employs the Decision Tree classifier as the estimator, with n_estimators set to 100 and max_depth limited to 1, yielding an accuracy of 98.246%. Our top-performing model remains AdaBoost with 5 Logistic Regression estimators, utilizing a learning rate of 1.

Figure 17 delineates the contribution of each estimator to the final prediction:

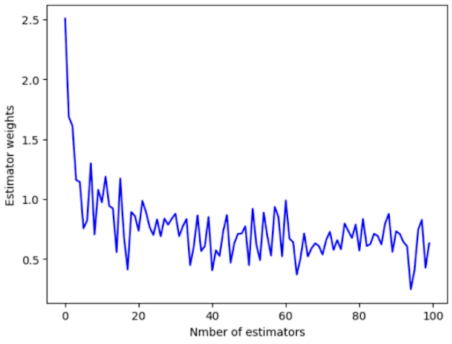

Figure 17: Contribution of each estimator to the final prediction in the AdaBoost model.

Figure 18 shows how well each estimator perform

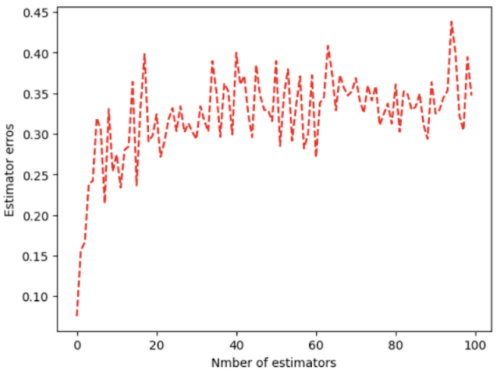

Figure 18: Performance of each estimator in the AdaBoost model.

As expected, the weakest estimators have a greater

contribution to the model. These weaker estimators, although they may have lower individual accuracies, play a vital role in capturing the misclassified instances and improving the model's overall accuracy.

4.5 Feature Engineering

In the realm of feature engineering, a strategic approach involves selecting a subset of features that are most relevant to the problem at hand. By doing so, we can enhance both the computational efficiency and the accuracy of the model. While considering a partition of the features can be effective, it's also important to explore the potential of linear or nonlinear combinations of features, although this aspect is not addressed in this particular study.

To provide insight into feature importance, Figure 19 presents a plot showcasing the features ordered by their importance according to our AdaBoost model.

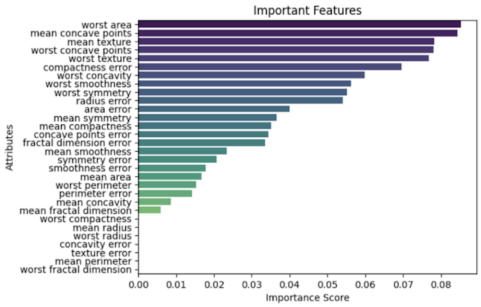

Figure 19: Feature importance plot showcasing the features ordered by their importance according to the AdaBoost model.

This visualization offers valuable insights on which features contribute the most to the predictive power of the model, The analysis of feature importance reveals that the most relevant features for our model are 'worst area', 'mean concave points', and 'mean texture'. This observation aligns well with the context of the case study.

To explore the impact of using different partitions of the features, we conducted experiments to assess the accuracy of the model when trained on subsets of features. Interestingly, our findings suggest that we can achieve the same level of accuracy by utilizing only a subset of the most important features. Specifically, removing the last seven features, which have negligible importance for our target, does not affect the accuracy of the model. This indicates that we can potentially reduce computational costs by focusing solely on the most informative features.

However, our experiments also revealed that further feature removal beyond this point leads to a decrease in accuracy. This underscores the importance of striking a balance between feature reduction for efficiency and preserving the critical predictive power of the model.

Conclusion

In conclusion, our research has provided valuable insights into the comparative performance of various supervised classification algorithms, specifically focusing on decision trees, random forests, and AdaBoost. Through rigorous experimentation and analysis, we have demonstrated the impact of different hyperparameters, such as the depth of decision trees and the number of estimators in ensemble methods, on model accuracy and overfitting.

Furthermore, our study has highlighted the importance of feature engineering in enhancing model performance and efficiency. By selecting the most relevant features and optimizing their combinations, we were able to achieve comparable or even better accuracy with reduced computational complexity.

Looking ahead, our exploration into supervised classification algorithms naturally leads us to consider the potential of unsupervised learning and neural networks. Unsupervised learning techniques, such as clustering and dimensionality reduction, offer promising avenues for uncovering hidden patterns and structures within data without the need for labeled examples. Additionally, the advancement of neural networks, particularly deep learning architectures, presents exciting opportunities for tackling complex classification tasks with unprecedented accuracy and scalability.

As we continue to push the boundaries of machine learning research, the integration of supervised, unsupervised, and neural network approaches promises to unlock new realms of understanding and innovation in data-driven decision-making. By embracing the diverse toolkit of modern machine learning techniques, we can pave the way for transformative advancements across a wide range of fields and applications.