SIT307 Machine Learning

Van Nam Quang Nguyen - 223149341

1 Hyperparameter and Parameter

1.1 Hyperparameter

Hyperparameters are external to the model and set before the training process begins. They control aspects like the model's structure, the complexity, and the training procedure itself. Hyperparameters does mpt come from trained data, and they need to be specified before training process. In order to have a good machine learning model, the scientist have to tune the hyper-parameters. Some popular techniques like grid search or random search or halving successive search, to optimize the model's performance.

1.1.1 Linear Regression

In linear regression, we can tune the regualarizer which is add to the loss function to prevent overfitting. Since it applies to the model to affect the trade-off between the variance and bias to find the coefficients to avoid overfitting. There are two cases of regularization strengths which is Ridge (λ) and Lasso (α).

- **Lasso Regularization** (Least Absolute Shrinkage and Selection Operator) is a regression method that combines variable selection and regularization to improve model accuracy and interpretability. Lasso simplifies the model by shrinking some coefficients to zero, thereby selecting only the most important features and reducing multicollinearity.
- **Ridge Regularizaion:** penalizes the loss function by adding the squared values of the coefficients, which reduces overfitting by shrinking them. Unlike Lasso Regression, which can set some coefficients to zero and perform feature selection, Ridge Regression retains all features while reducing their influence.

1.1.2 Support Machine Vector

Unlike Linear Regression, SVM have more hyperparameters to tune in order to have the best SVM model:

- **Kernel type:** The kernel function is a hyperparameter that specifies the type of transformation used on the data to achieve linear separation in higher-dimensional space. Common kernels include linear, polynomial, sigmoid and RBF kernels.
- **C (Parameter of Regularization):** The trade-off between maximizing the margin and reducing the classification error on the training data is managed by the hyperparameter C. While a smaller C value concentrates on increasing the margin and may result in a softer decision boundary, a greater C value prioritizes reducing classification mistakes.

- **Gamma:** The hyperparameter γ regulates the impact of a single training sample on the RBF kernel. A larger influence is produced by a low γ, whereas a high γ gives a closer match to the training set.
- **Degree:** establishes how complicated the choice boundary is. A larger degree could result in overfitting but also enable the model to fit more intricate patterns.
- `coef0`: regulates the kernel function's effect between higher-order and lower-order terms. The influence of higher-order terms is amplified by a greater `coef0` value, which modifies the decision boundary's form.

1.2 Parameter

Parameters are internal to the model and which can be found throughout the training process. They are adjusted automatically during the training phase, and their values determine how the model makes predictions on new data.

1.2.1 Linear Regression

In linear regression, model's parameters are intercept and coefficients, often known as weights. These parameters are found by minimizing the error between the predicted and actual values A basic linear regression model provided as

$$y = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b,$$

where the weights $w_1, w_2, \ldots, w_n$ and the bias term b are the parameters.

1.2.2 Support Vector Machine

In SVM, we find the parameters, support vectors, which are the data points that lie closest to the decision boundary. These support vectors define the optimal hyperplane and play a critical role in the model's classification decisions. The decision boundary is a hyperplane determined by the weight vector $\mathbf{w}$ and the bias term b::

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

2 Elastic from Lasso and Ridge regularised

Elastic net is the general form of Lasso and Ridge regularization, it follows as:

$$\min_{\mathbf{w}} \left(\frac{1}{n} \sum_{i=1}^{n} L(y_i, \mathbf{x}_i^T\mathbf{w}) + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 \right)$$

- $||\mathbf{w}||_1$ is the LASSO (L1) penalty, which encourages sparsity by shrinking some coefficients to zero.
- $||\mathbf{w}||_2^2$ is the Ridge (L2) penalty, which penalizes large coefficients without shrinking any to zero.

If we set $\lambda_1 = 0$, to remove LASSO (L1) we can obtain Ridge regression, whereas $\lambda_2 = 0$ to remove Ridge(L2) we will have Lasso regression.

By applying both penalties, Elastic Net performs feature selection like LASSO, shrinking irrelevant coefficients to zero, while also stabilizing the selection process by spreading weight across correlated features, as Ridge does. This balance helps prevent overfitting by controlling the size of coefficients, improving generalization to unseen data.

3 Feature Importance

3.1 Gini Importance

Gini importance assesses feature relevance in tree-based models like Decision Trees and Random Forests by calculating how much each feature decreases the Gini impurity when utilized for splits. It computes the average reduction in impurity caused by a feature over all nodes and trees, indicating its efficacy in enhancing node purity.
Features that lead to significant reductions in impurity are considered more important. High Gini Importance indicates that a feature is crucial for making accurate predictions by improving the purity of the nodes where it is used.
This technique, which is special to tree-based models, naturally manages feature interactions and non-linear correlations. However, it may be biased towards features with more categories.

3.2 SHAP Importance

Permutation importance measures the decline in model performance when feature values are swapped. After training the model and measuring its baseline performance, the feature values in the test set are randomly mixed. The model is then re-evaluated, and the performance reduction is used to estimate the relevance of the feature.
The importance of the feature is the difference between the model's performance with original and shuffled feature values. A large drop in performance indicates high importance.
This technique is model-independent and offers a direct estimate of a feature's influence; nevertheless, it is computationally costly and sensitive to feature correlations.

3.3 Advantages vs Disdvantages

3.3.1 Gini Importance

Advantages:

- Built directly into tree-based models .
- The model implicitly captures feature interactions.
- Calculated during the model-training process.

Disadvantages:

- Bias towards features with several categories
- Only applies to tree-based models and may not be relevant to other types of models.
- Features that appear frequently in splits may have a high value even if their overall influence is minimal.

3.3.2 Permutation Importance

Advantages:

- Model-Agnostic: Can be applied to any supervised learning model.
- Direct Impact Measurement: Provides a clear measure of how much a feature contributes to model performance.

Disadvantages:

- Computational Cost: Requires multiple model evaluations, which can be time-consuming.
- Sensitive to Feature Correlation: May not accurately reflect importance if features are highly correlated.

3.4 Similarities and Differences

3.4.1 Similarities

Aspect	Gini Importance	Permutation Importance
Objective	Determines feature importance in predicting the target.	Assesses feature importance by measuring performance impact.
Feature Contribution	Measures how features reduce impurity in decision trees.	Measures how feature value shuffling affects model performance.
Interpretability	Provides insight into how features impact node purity and model decisions.	Shows how features influence model accuracy or error when shuffled.
Use in Feature Selection	Features with high importance are often selected for inclusion.	Features with significant performance drop are chosen for further analysis.
Model Understanding	Helps explain which features are crucial for model predictions.	Provides a clear view of feature relevance based on performance changes.

Table 1: Similarities between Gini Importance and Permutation Importance

Aspect	Gini Importance	Permutation Importance
Definition	Measures the contribution of a feature to the reduction in Gini impurity during tree splits.	Measures the change in model performance when a feature's values are randomly shuffled.
Model Specificity	Specific to tree-based models (e.g., Decision Trees, Random Forests).	Model-agnostic; can be applied to any supervised learning model.
Computation	Automatically computed during model training; less computational overhead.	Requires multiple model evaluations (one for each feature); computationally expensive.
Handling Interactions	Captures feature interactions implicitly as it considers how features affect node purity.	Can detect interactions indirectly if they affect model performance.
Bias	May favor features with more categories or higher variability.	Less biased towards features with more categories; directly reflects the feature's impact on performance.
Interpretability	Directly linked to how well features improve node purity in tree models.	Provides a clear measure of feature importance based on performance impact.
Sensitivity	Sensitive to feature correlations; may overemphasize features used frequently in splits.	Sensitive to correlations and noise; may be affected by random fluctuations.

Table 2: Comparison of Gini Importance and Permutation Importance

3.4.2 Differences

3.5 Outcomes

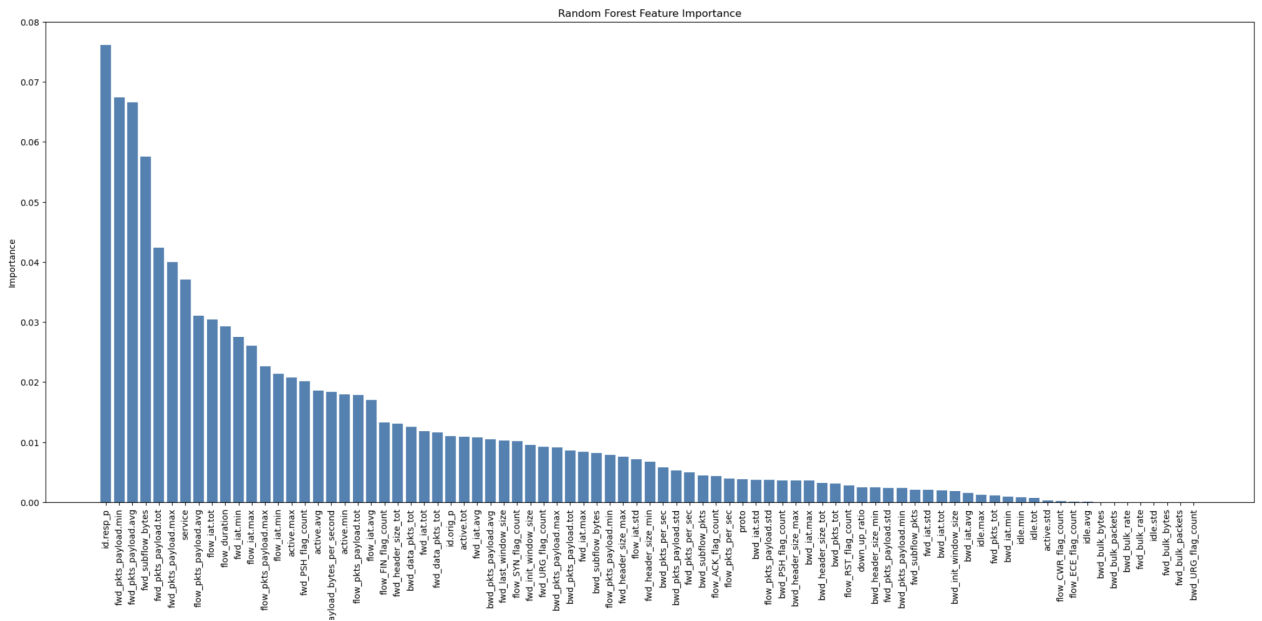

Figure 1: Gini Approach

These approaches yield different results, with one identifying only a few important features while the other reveals more significant ones. This disparity can make analysis challenging, particularly because permutation methods may struggle due to the noise present in the datasets.

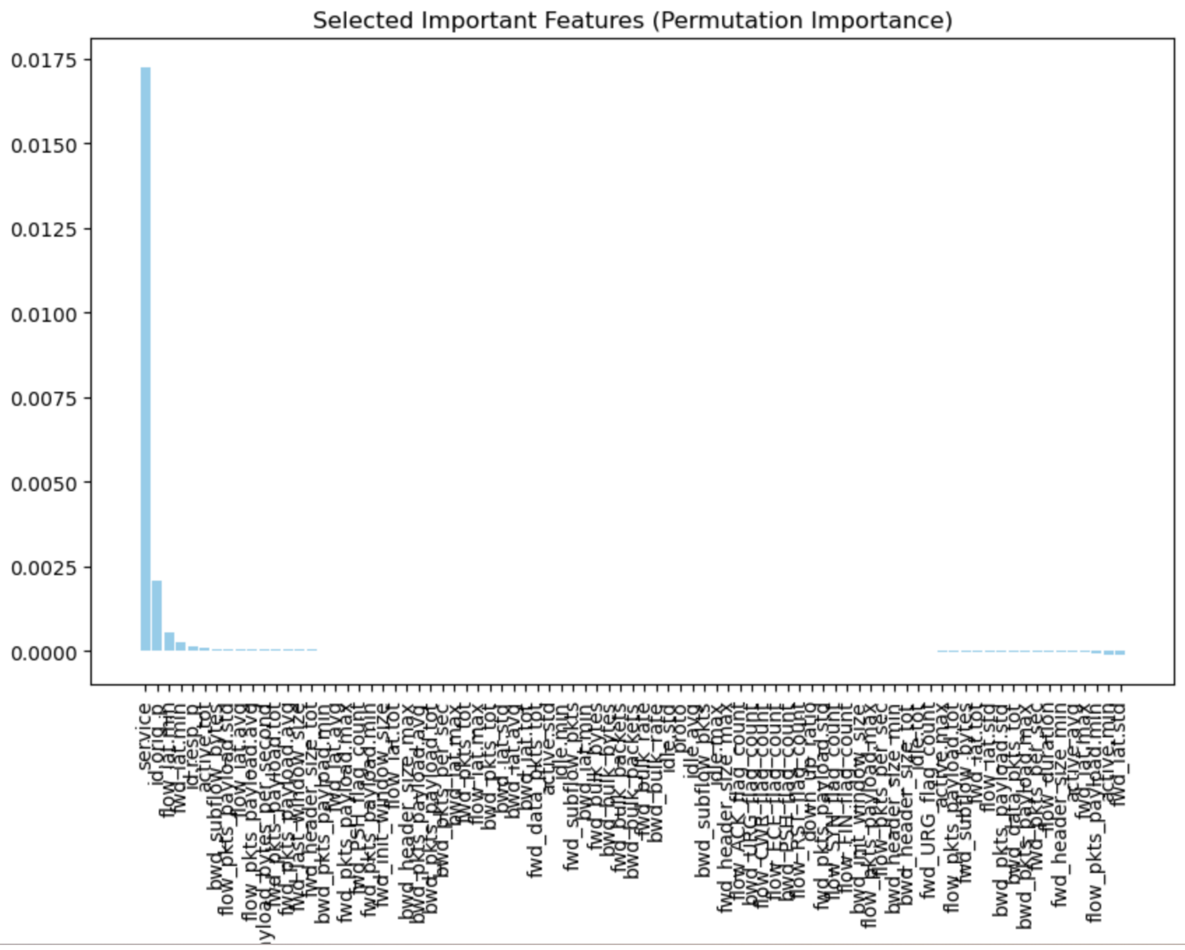

Figure 2: Permuation Approach

Consequently, variables that are clearly important can be easily identified, but those that are less significant and contain noise may be difficult for permutation methods to detect. This leads to substantial differences in outcomes.

Therefore in this case another approach cna be consider, SHAP. We can compare this one with these two to find the feature importance, the following figure will show the output of SHAP.

Now, we can conlude that Gini method work well on finding the feature importance.

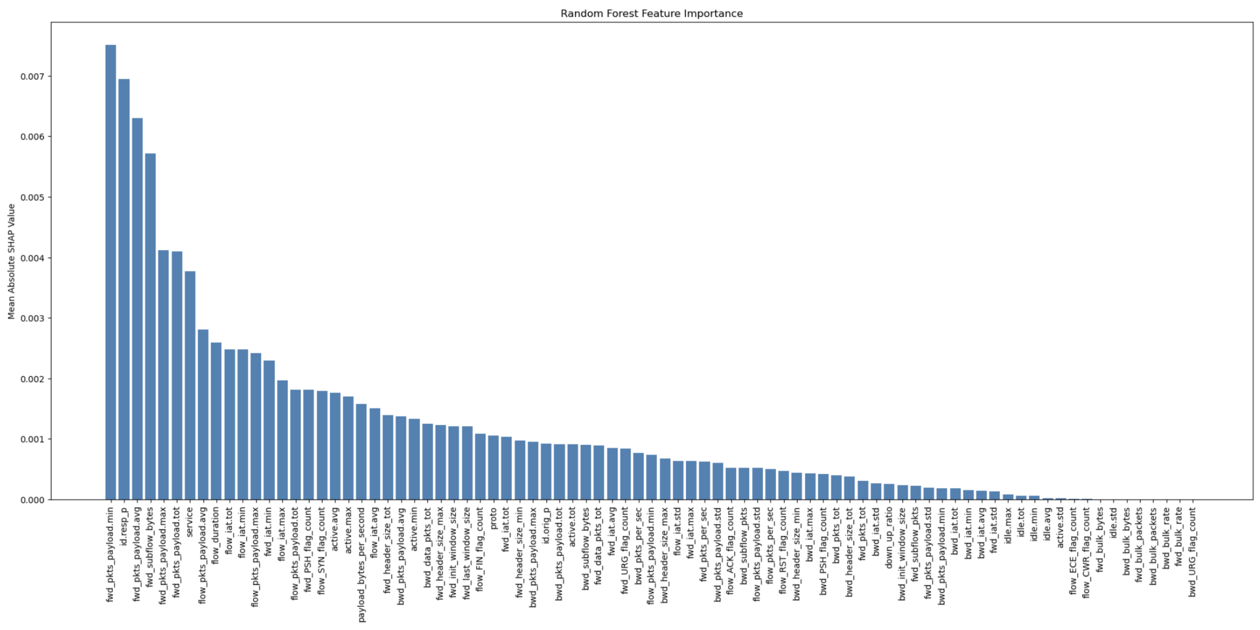

Figure 3: SHAP Approach

4 Supervised machine learning model

4.1 Performance report and overfitting check

4.1.1 Performance report

The following table give the performance of the KNN model:

Class	Precision	Recall	F1-score	Support
0.0	0.97	0.98	0.97	2311
1.0	0.99	0.99	0.99	158
2.0	1.00	1.00	1.00	28342
3.0	1.00	1.00	1.00	1266
4.0	0.85	0.92	0.88	12
5.0	0.08	0.20	0.11	5
6.0	0.89	0.88	0.89	627
7.0	0.96	0.93	0.94	300
8.0	0.97	0.97	0.97	741
9.0	0.99	0.98	0.99	503
10.0	0.98	0.98	0.98	2419
11.0	0.84	0.67	0.74	72
Accuracy			**0.9915**	36756

Table 3: Classification Report for KNN

The KNN model obtained an amazing accuracy of about 99%, with most classes displaying precision greater than 84%, with the exception of class 5. The recall values are also high, with class 5 having the lowest recall of 0.2, most likely due to its low presence in the sample. Overall, the result reveals that KNN did an excellent job categorizing the data.

Now let consider the performance of SVM model It looks like SVM give better performance

Class	**Precision**	**Recall**	**F1-score**	**Support**
0.0	0.96	0.98	0.97	2311
1.0	1.00	0.99	0.99	158
2.0	1.00	1.00	1.00	28342
3.0	1.00	0.99	1.00	1266
4.0	0.85	0.92	0.88	12
5.0	0.17	0.20	0.18	5
6.0	0.89	0.99	0.94	627
7.0	0.99	0.98	0.98	300
8.0	0.98	0.99	0.98	741
9.0	1.00	0.99	1.00	503
10.0	0.98	0.98	0.98	2419
11.0	0.95	0.51	0.67	72
Accuracy			**0.9933**	36756

Table 4: Classification Report for SVM

in classifying, this because the recall and precision of all class quite high. The lowest is class 5 which gives 0.17 and 0.2 for precision and recall respectively, which is quite higher than KNN. Moreover, accuracy of SVM is slightly higher than KNN (0.9933 ¿ 0.9915).

Now let consider Decision Tree: Decision tree gives the result is quite different from the

Class	**Precision**	**Recall**	**F1-score**	**Support**
0.0	0.96	0.96	0.96	2311
1.0	0.96	0.97	0.97	158
2.0	1.00	1.00	1.00	28342
3.0	0.99	1.00	1.00	1266
4.0	0.91	0.83	0.87	12
5.0	0.00	0.00	0.00	5
6.0	0.90	0.89	0.90	627
7.0	0.95	0.95	0.95	300
8.0	0.97	0.96	0.96	741
9.0	0.98	0.99	0.99	503
10.0	0.96	0.97	0.97	2419
11.0	0.70	0.60	0.65	72
Accuracy			**0.984**	36756

Table 5: Classification Report for Decision Tree

other model since the accuracy is lower than 2 models above (0.984). Moreover, class 5 have f1 score is 0, which mean that it even though the accuracy is quite high but it can not recognize all the classes, which show that the model can not classify all the classes.

4.1.2 Overfitting

As we all know that over-fitting is when a model performs significantly better on training data than on unseen test data. So that to check whether the result is overfitted we can compare Training and Testing set result to understand the model completely. If the result are high in training set but poor in testing set than it is over-fitted.
if the model is too complex which means, it have too many parameter then it can be reslut in over fitting.

4.2 Hyper-parameter tunning

Hyperparameter tuning is a crucial process in optimizing machine learning models. It entails modifying the settings (hyperparameters) of a model to improve its performance. The hyperparameters for each machine learning model are listed below, along with the reasons why they were tuned:

1. K-Nearest Neighbors (KNN)

- **'n_neighbors':** The number of neighbors to consider when making a prediction.
  - *Reason for Tuning:* A low number of 'n_neighbors' makes the model more sensitive to noise, whereas a high value smoothes it out but may distort decision boundaries. Tuning helps in finding the right balance for better generalization.
- **'distance_metric':** The metric used to calculate distances between data points (e.g., Euclidean, Manhattan).
  - *Reason for Tuning:* Different metrics affect the model's ability to capture data structure.

2. Support Vector Classifier (SVC) with RBF Kernel

- **'gamma':** Controls the influence of a single training example. Low values mean a broader influence, while high values mean a narrower influence.
  - *Reason for Tuning:* A low 'gamma' value results in a smoother decision boundary, whereas a high 'gamma' value leads to a more detailed boundary. Proper tuning balances complexity and generalization.
- **'C':** The regularization parameter that trades off between achieving a low training error and minimizing model complexity.
  - *Reason for Tuning:* Finding the optimal 'C' helps balance bias and variance to avoid underfitting and overfitting.

3. Logistic Regression

- **‘C‘:** The inverse of regularization strength; smaller values indicate stronger regularization.
  - *Reason for Tuning:* Adjusting ‘C‘ balances fitting the training data and keeping coefficients small to avoid overfitting.
- **‘solver‘:** Algorithm used to optimize the logistic regression function.
  - *Reason for Tuning:* Different solvers have varying convergence properties and computational efficiencies. Choosing the right solver impacts training speed and model convergence.
- **‘multi_class‘:** Strategy for handling multi-class classification.
  - *Reason for Tuning:* The choice of strategy affects multi-class handling and model performance.

4. Decision Tree

- **‘max_depth‘:** Maximum depth of the tree.
  - *Reason for Tuning:* Limiting tree depth prevents overfitting by avoiding excessively complex models.
- **‘min_samples_split‘:** Minimum number of samples required to split an internal node.
  - *Reason for Tuning:* Ensures that splits are made only when there are enough samples, reducing overfitting.
- **‘min_samples_leaf‘:** Minimum number of samples required to be at a leaf node.
  - *Reason for Tuning:* Ensures leaf nodes are sufficiently populated, reducing overfitting.
- **‘max_features‘:** Number of features to consider when looking for the best split.
  - *Reason for Tuning:* Restricting the number of features helps in reducing model variance and enhancing generalization.

Hyperparameter tweaking is used to improve model performance and guarantee that it generalizes well to new data, avoiding both underfitting and overfitting. Adjusting these hyperparameters improves the model’s capacity to generate accurate predictions. Because each dataset has a distinct distribution, there may be differences in hyperparameters between models.

5 Ensemble model

5.1 When to use ensemble model?

Ensemble models are frequently favored over individual models such as Decision Trees for a variety of reasons, most notably to improve performance and robustness. If a single Decision Tree is not properly constrained, it might show significant variance and overfit the training data. This significant variance leads to a model that may not generalize well to new data.
While a single Decision Tree has obvious interpretability and can handle a wide range of data types, it may be less stable and accurate than ensemble techniques. Ensemble models aggregate the predictions of several trees, which improves model stability and predictive power. Although less interpretable, approaches like feature significance ratings can nonetheless provide useful information. Overall, ensemble models are especially useful for complicated and huge datasets, where individual Decision Trees may struggle to perform well.
Random Forest creates numerous Decision Trees using bootstrapped data and averages their predictions to minimize variation and improve stability. AdaBoost, on the other hand, works progressively to rectify prior trees' faults, minimizing bias and increasing accuracy. XGBoost employs gradient boosting to repeatedly enhance the model by adding trees that correct earlier mistakes, as well as regularization to prevent overfitting.

5.2 Similarities and Differences

5.3 Use situation

5.3.1 Random Forest

Random Forest is frequently the recommended solution when we want a basic yet strong model that works well with minimum tuning, making it suited for large and high-dimensional datasets. Bagging (bootstrap aggregating) efficiently decreases overfitting, increasing its robustness in circumstances where variance is an issue. Furthermore, Random Forest gives obvious insights into feature importance, making it easy to interpret. This model performs well in jobs that need both stability and general robustness, particularly in situations involving large data sets with low noise.

5.3.2 AdaBoost

AdaBoost is used in circumstances where the aim is to enhance accuracy using difficult-to-classify instances. It operates by repairing faults from prior trees in a sequential order to reduce bias. However, it is noise-sensitive, and its sequential training procedure can be computationally expensive. AdaBoost is useful in tasks where accuracy on challenging data points is crucial, but it requires cleaner datasets and careful tuning to avoid overfitting.

Aspect	**Random Forest**	**AdaBoost**	**XGBoost**
Methodology	Bagging (Bootstrap Aggregating)	Boosting (Sequentially correcting previous errors)	Boosting (Gradient Boosting with advanced features)
Training Approach	Trains multiple trees independently	Trains trees sequentially, with each tree correcting errors from the previous ones	Trains trees sequentially, optimizing residuals of previous trees with regularization
Error Handling	Reduces variance by averaging predictions of multiple trees	Reduces bias by focusing on hard-to-classify examples	Reduces both bias and variance with regularization and optimizations
Model Complexity	Simpler; focuses on averaging predictions	Can be sensitive to noisy data due to sequential training	Complex, with many hyperparameters and optimizations
Handling Overfitting	Reduces overfitting via averaging and randomness	Can overfit if not carefully tuned, especially with noisy data	Regularization techniques help prevent overfitting
Computational Efficiency	Faster to train and predict due to parallel tree building	Slower due to sequential tree training	Can be slower, but with high performance optimization
Hyperparameters	Number of trees, depth of trees, features per split	Number of trees, learning rate, tree depth	Number of trees, learning rate, max depth, min child weight, gamma, etc.

Table 6: Comparison of Random Forest, AdaBoost, and XGBoost

5.3.3 XGBoost

need great precision and speed, particularly on large-scale datasets. With its built-in regularization and gradient boosting architecture, XGBoost handles overfitting better than AdaBoost and frequently outperforms other models on structured/tabular data. It also has the capability to handle missing data and complicated feature relationships. This makes it appropriate for issues that require high-performance models with precise optimizations.

5.4 Report performance

The following table show the accuracy of all used models:

Model	**KNN**	**SVM**	**Decision Tree**	**Random Forest**	**AdaBoost**	**XGBoost**
Accuracy	0.9915	0.9934	0.9898	0.9929	0.9284	0.9923

Table 7: Accuracy of Different Models

All ML models achieved over 90% accuracy, with SVM having the best score of 0.9934. Random Forest and XGBoost perform better than week learners, Decision Tree, and KNN (`n_neighbour = 1`). This allows the ensemble model to perform better than the weak learner. However, AdaBoost demonstrates that the ensemble model does not always perform well, and it performs worse than the Decision Tree. Now let's look at the AdaBoost Performance report:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0.0	0.66	0.49	0.56	2311
1.0	1.00	0.09	0.17	158
2.0	0.99	1.00	0.99	28342
3.0	0.95	0.99	0.97	1266
4.0	0.00	0.00	0.00	12
5.0	0.00	0.00	0.00	5
6.0	0.68	0.99	0.81	627
7.0	0.00	0.00	0.00	300
8.0	0.70	0.57	0.63	741
9.0	0.83	0.76	0.80	503
10.0	0.64	0.84	0.72	2419
11.0	0.17	0.01	0.03	72
Accuracy			0.93	36756

Table 8: Classification Report of AdaBoost

Looking deeper at the report, we can observe that AdaBoost does not perform well in categorizing the data. Class 4, 5, and 7 have Precision and Recall values of 0, indicating that the model is unable to categorize these classes and demonstrating the model's poor performance. Furthermore, when we look at Random Forest and XGBoost, we can see that the performance of these models is fairly good, even though the accuracy is not high, but the class as a whole has high precision and recall. This is also better than SVM.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0.0	0.96	0.99	0.97	2311
1.0	1.00	0.99	1.00	158
2.0	1.00	1.00	1.00	28342
3.0	1.00	1.00	1.00	1266
4.0	0.91	0.83	0.87	12
5.0	0.25	0.20	0.22	5
6.0	0.90	0.95	0.92	627
7.0	0.97	0.95	0.96	300
8.0	0.98	0.97	0.98	741
9.0	0.99	0.99	0.99	503
10.0	0.99	0.97	0.98	2419
11.0	0.92	0.62	0.74	72
Accuracy			0.99	36756

Table 9: Classification Report for Random Forest

Class	Precision	Recall	F1-Score	Support
0.0	0.97	0.98	0.98	2311
1.0	1.00	0.98	0.99	158
2.0	1.00	1.00	1.00	28342
3.0	1.00	1.00	1.00	1266
4.0	1.00	0.83	0.91	12
5.0	0.20	0.20	0.20	5
6.0	0.89	0.90	0.90	627
7.0	0.95	0.92	0.94	300
8.0	0.98	0.98	0.98	741
9.0	0.99	0.99	0.99	503
10.0	0.98	0.98	0.98	2419
11.0	0.93	0.74	0.82	72
Accuracy			0.99	36756

Table 10: Classification Report

Does ensemble model is always good?

Ensemble models have increasing complexity and processing costs, making them more difficult to train, deploy, and analyze. Ensembles can potentially lead to overfitting if not properly controlled, and they may not always deliver substantial performance increases after a point. As a result, while they provide significant benefits, ensemble models may not be the ideal fit for every case, particularly when simpler models suffice or resources are limited. The AdaBoost is an example of overfitting since it perform good at training which result the accuracy at 0.992 while the testing result is much lower.

5.5 Ensemble model using other classifers than Decision Tree

Yes, ensemble models can include classifiers beyond decision trees. For example, combining K-Nearest Neighbors (KNN) with $n\_neighbors = 1$ using a bagging approach involves training multiple KNN models on different data subsets and aggregating their predictions. Alternatively, stacking can be used, where KNN and Support Vector Machines (SVM) are trained separately, and their predictions feed into a meta-model like Logistic Regression to improve overall performance. Such ensembles leverage the strengths of different classifiers, offering enhanced accuracy and robustness. Additionally, combining weaker models like Decision Trees, Logistic Regression, and low $n\_neighbors$ KNN can also improve predictive performance.

References

[1] Mohammed Alhamid. *Ensemble Models: What Are They and When Should You Use Them? — Built In*. builtin.com, May 2022. URL: https://builtin.com/machine-learning/ensemble-model.

[2] Jason Brownlee. *How to Calculate Feature Importance With Python*. Machine Learning Mastery, Mar. 2020. URL: https://machinelearningmastery.com/calculate-feature-importance-with-python/.

[3] Srijani Chaudhury. *Tuning of Adaboost with Computational Complexity*. Medium, Aug. 2020. URL: https://medium.com/@chaudhurysrijani/tuning-of-adaboost-with-computational-complexity-8727d01a9d20.

[4] Tahera Firdose. *Fine-Tuning Your Random Forest Classifier: A Guide to Hyperparameter Tuning*. Medium, Aug. 2023. URL: https://tahera-firdose.medium.com/fine-tuning-your-random-forest-classifier-a-guide-to-hyperparameter-tuning-d5ceab0c4852.

[5] GeeksforGeeks. *Random Forest Hyperparameter Tuning in Python*. GeeksforGeeks, Dec. 2022. URL: https://www.geeksforgeeks.org/random-forest-hyperparameter-tuning-in-python/.

[6] Lari Giba. *Elastic Net Regression Explained, Step by Step*. Machine Learning Compass, June 2021. URL: https://machinelearningcompass.com/machine_learning_models/elastic_net_regression/.

[7] Satish Kumar. *Does Ensemble Models Always Improve Accuracy?* Analytics Vidhya, Oct. 2020. URL: https://medium.com/analytics-vidhya/does-ensemble-models-always-improve-accuracy-c114cdbdae77.

[8] RITHP. *Optimizing XGBoost: A Guide to Hyperparameter Tuning*. Medium, Jan. 2023. URL: https://medium.com/@rithpansanga/optimizing-xgboost-a-guide-to-hyperparameter-tuning-77b6e48e289d.

[9] *Understanding Feature Importance in Machine Learning*. Built In. URL: https://builtin.com/data-science/feature-importance#:~:text=Feature%20importance%20refers%20to%20techniques.

[10] *What is ensemble learning? — IBM*. www.ibm.com, Feb. 2024. URL: https://www.ibm.com/topics/ensemble-learning.