## **Machine Learning Lecture**

**Random Forests in Machine Learning: An In-Depth Tutorial**

**Introduction:**  
Random Forest is a very strong and yet flexible learning algorithm which, in order to perform both classification and regression tasks, will combine the predictions of many decision trees, in its quest for better performance and reducing overfitting. Using some techniques like Bootstrap Aggregation-also called Bagging-and Feature Randomness, Random Forest makes sure each tree in the model gets trained on a different subset of the total data, with different features at every split. The diversity helps in enhancing the performance of the model, especially when it deals with a complex dataset comprising noise or non-linear relationships.

**Theories of how Random Forest really works:** A deeper explanation of theoretical explanations will be given in the report. We are also going to look at some advanced concepts of Random Forest, such as OOB error-unique for this ensemble method and giving a very efficient way of model performance estimate. Further, the implementation is using a classic Iris dataset to show the capabilities of this algorithm. We will look at how Random Forest handles noise in the data, provides insight into feature importance, and why it is so effective at reducing overfitting.

A diagram of a forest

Description automatically generated

**Theoretical Foundation:**  
The random forest is a nonbiased and supervised machine learning algorithm that balances the prediction and, therefore, is never sensitive to overfitting. With basic steps in Decision Trees and an enhancement in their effectiveness by the incorporation of ensemble learning methods, the process works like coming up with a "forest" of Decision Trees trained over somewhat different data and using their predictions in an enhanced manner toward improving the overall prediction.

1. **Decision Trees: The Building Blocks:**  
   A decision tree is a model which, with the help of an input dataset, creates smaller subgroups of data based on feature values trying to classify or predict a target variable. The idea here is to split the branches so that each split results in the best segregation among the data points, and impurity can be minimized in each split.

**Common Split Criteria:**

* **Gini impurity:** A measure of impurity in a dataset for classification tasks.

**Formula:**

**Where:**

* p*i​* is the proportion of class
* *i* in the node
* C is the number of classes.

**Example:** Example: If a node contains 70% samples of class A and 30% of class B (pA​=0.7, pB​=0.3).

**G=1−(0.72+0.32) = 1−(0.49+0.09) =0.42**

This means the node has moderate impurity. A pure node with samples of only one class would have G = 0

**Entropy:** Entropy quantifies how uncertain or random the distribution of classes is. Lower entropy is found in nodes with homogeneous classes.

**Formula:**

**Where:**

* p*i​* is the proportion of class
* *i* in the node.

**Example:** For a node with pA = 0.7, pB = 0.3

**H= −(0.7log (0.7) +0.3log (0.3))**

**Computing:**

**H= −(0.7×−0.514) −(0.3×−1.737) ≈ 0.88**

Higher entropy indicates more diversity in classes.

* **Variance Reduction:** While considering regression, the target is to reduce the variance in the target variable during each split. The variance reduction formula calculates how many variances are lost/ reduced after each split.

**Formula:**

**Where:**

* + N: Number of samples in the parent node.
  + Nk: Number of samples in the K-th child node.
  + Varianceparent: Variance of target values in the parent node.
  + Variancechild: Variance of target values in the k-th child node.
* **Explanation:** It gives the reduction in variability owing to a split. Greater it is, better it is owing to it being an informative split.

1. **Random Forest: Aggregating Trees**

## Key Features:

1. **Bagging (Bootstrap Aggregation)**

* Random Forest trains each decision tree on a bootstrap sample a random subset of the data selected with replacement. This introduces variability among trees, improving overall model robustness.

1. **Feature Randomness:**

* At each node, a random subset of features is considered for splitting, reducing the correlation between trees.

**Prediction Aggregation:**

**Classification:** Random Forest makes predictions by majority voting across all trees.

​**Where:**

is the forecast of the i-th tree, and B is the number of Trees in a Forest.

**Regression:** The overall forecast is simply an average across all individual forest trees.

## Advantages of Random Forest

* **High Accuracy:** Random Forest can handle a large amount of data and complex relationships, providing high accuracy for both classification and regression tasks.
* **Robustness:** By averaging multiple trees, it reduces variance and handles noisy data effectively.
* **Overfitting Resistance:** Due to the randomness in both the training data and features, Random Forest is less likely to overfit compared to single decision trees.
* **Feature Importance:** Random Forest can evaluate the importance of each feature, helping in feature selection and model interpretation.
* **Handles Missing Data:** Random Forest is resilient to missing data and can handle missing values during training and prediction.

## Limitations:

* **Computational Cost:** Random Forest requires significant computational resources, especially with a large number of trees.
* **Interpretability:** While individual decision trees are easy to interpret, the overall Random Forest model is more complex and harder to explain.

## Dataset Overview: Iris Dataset

The Iris dataset is an example of a multiclass classification problem and comprises three species of iris flowers: Setosa, Versicolor, and Virginica. These are described by four features:

* Sepal length
* Sepal width
* Petal length
* Petal width

This dataset is the best to show the work of Random Forest because it is multiclass, and the importance of features is different.

# **Advanced Topic:**

**Out-of-Bag (OOB) Error Explanation:**

This is a peculiar way of validation in which, for its estimation of the prediction error, it does not require an additional validation dataset. This is made possible by using the points not falling in that bootstrap sample for the training of any given tree.

In Random Forest, each decision tree is trained on a random subset of the data, selected with replacement. This means that approximately 1/3 of the data is left out (not used) for each tree, and these data points are referred to as Out-of-Bag (OOB) samples.

Instead of requiring a separate test set, the OOB samples can be used to test each tree's performance, hence the performance of the overall Random Forest model. This allows it to provide a pretty good estimate of its performance on unseen data while at the same time fully utilizing the available training data.

# **How OOB Error Works?**

**Bootstrap Sampling:**

* Random Forest trains each tree on a random subset of the data, known as a bootstrap sample. This sample is obtained by drawing data points from the original training dataset with replacement.
* On average, about 2/3 of the data are used to train each tree, while the remaining 1/3 of the data points are left out and become the OOB samples for that tree.

**Making Predictions for OOB Samples:**

Once each of the trees is trained, it makes predictions using the OOB samples-the data that it did not see during training. These predictions are then compared against the actual true labels of the OOB samples to estimate the performance of the model on unseen data.

**Aggregating OOB Predictions:**

* After all trees have made their predictions on their respective OOB samples, the final OOB error is computed by aggregating the predictions of all trees for each OOB sample.
* In the case of a classification problem, this is usually done by a majority vote, whereby the class that the majority of the trees predict is taken as the final prediction.
* For regression, the aggregation is performed by averaging all the predictions made by all the trees.

**Computing the OOB Error:**

* Then, the OOB error is calculated by comparing these OOB predictions to the actual labels, and tabulating how often those predictions were wrong. This provides an estimate of the model's generalization error, how well it is likely to perform on unseen data.

**Formula:**

The formula for the OOB Error is given by

**Where:**

*N* is the total number of data samples in the dataset.

*y*i​ is the true label of the *i*-th sample (from the actual data).

is the predicted label for the *i*-th sample based on the OOB predictions from the trees.

1 is an indicator function:

1=1 when the prediction the OOB is different from the true label yi​ (i.e., misclassified).

1 = 0 if the prediction is correct.

The OOB error is the average number of misclassified across OOB samples for all the tress in the forest.

## Advantages of OOB:

1. **Error No Need for a Separate Validation Set:**

* The OOB error estimates the performance of a model without you needing to have a separate validation set, therefore able to utilize all the data available to training.
* In the case of limited data, it is very useful, letting your model train on the most possible data.

1. **Efficient and Reliable Estimate of Error:**

* OOB error is an unbiased estimate of the model's generalization error and provides an insight into how well the model will generalize to data that it hasn't seen.
* It acts like cross-validation but is computationally more efficient since it doesn't require training and testing on separate datasets.

1. **Built-in Cross-Validation:**

* Using the OOB samples, Random Forest can perform a sort of internal cross-validation automatically during the training process.
* It gives a good estimate of the model performance without requiring extra validation.

# Comparison: Standard Vs Enhanced Random Forest

|  |  |  |
| --- | --- | --- |
| **Aspects** | **Standard Random Forest** | **Enhanced Random Forest** |
| **Core Principle** | Combines predictions from multiple decision tree. | Incorporates advanced techniques like OOB error for validation. |
| **Split Criteria** | Gini Impurity, Entropy, or Variance Reduction. | Same Criteria but enhances by additional features randomness. |
| **Bagging (Bootstrap Aggregation)** | Trains tree on random subset of the data. | Efficient handles noisy or imbalanced dataset trough varied sampling. |
| **Prediction Mechanism** | Classification: Majority voting.  Regression: Mean of prediction | Same method but futures refined with OOB sample insights. |
| **Overfitting Prevention** | Reduces due to ensemble averaging and random sampling. | Greatly minimized due to OOB validation and feature randomness. |
| **Features Evolution** | Evaluates features contributions. | More robust insights, useful for feature selection and interpretation. |
| **Dealing with Missing Data** | Can process missing values during training and prediction. | Same capabilities but often with enhanced accuracy. |
| **Validation Approach** | Required a separate validation dataset. | No need for a separate validation sat due to OOB error estimation. |
| **OOB Error Application** | Not applicable | Provides and unbiased estimation of generalization error without extra data. |
| **Model Interpretability** | Easy for single decision tree but less for the ensemble. | Similar, through OOB offers clearer model performance metrics. |
| **Computational Demand** | Moderate; Scale with number of trees and features. | Higher due to advanced validation like OOB error calculation. |

# **Conclusion**

The Random Forest algorithm possesses one very useful feature: Out-of-Bag Error. It thus provides a means of model validation, not requiring the use of a validation set. Using, at each iteration, samples not involved in the composition of the particular bootstrap sample during the training stage, the OOB error will give an efficient and reliable estimate of the generalization ability of a model.

In order to get a quick estimate of the OOB error, predictions were made for the OOB samples of each tree. Later, these predictions needed to be aggregated across all the trees in the forest. Besides avoiding overfitting, such a mechanism saves time and computational resources since cross-validation or holding out a validation set is no longer required.

In practice, OOB error enables the facilities of internal cross-validation in the training of Random Forest. Therefore, it is a very robust and efficient tool dealing with big datasets while avoiding the overfitting problem by a very big margin. Further, the OOB error is very useful for those datasets where splitting into separate training and validation sets may not be possible due to small or limited data.

## References:

Random Forests, Breiman, L. (2001). Machine Learning, 45(1), 5-32.  
The foundational paper establishing Random Forests can be found at DOI: 10.1023/A:1010933404324.  
  
A. Géron (2017). Practical Machine Learning Using TensorFlow, Keras, and Scikit-Learn (1st Edition). Media by O'Reilly.  
(A useful manual for using Random Forests in Python.)  
  
Friedman, J., Tibshirani, R., and Hastie, T. (2009). Data Mining, Inference, and Prediction: The Foundations of Statistical Learning (2nd Edition). (In-depth statistical description of Random Forests and other machine learning techniques, Springer.)  
  
Hastie, T., Tibshirani, R., Witten, D., and James, G. (2013). R. Springer, "An Introduction to Statistical Learning: With Applications."  
(Comprehensible synopsis of Random Forests and further ensemble techniques.)  
  
Documentation for Scikit-learn, n.d. The Random Forest Classifier.  
#random-forests, taken from https://scikit-learn.org/stable/modules/ensemble.html  
(In-depth Scikit-learn implementation of Random Forests.)

G. Louppe (2014). Comprehending Random Forests: From Concept to Application. Preprint arXiv:1407.7502.  
This information was taken from https://arxiv.org/abs/1407.7502.  
(Detailed theoretical description and real-world applications of random forests.)  
  
T. K. Ho (1995). Decision Forests at Random. Document Analysis and Recognition: Proceedings of the Third International Conference, 1, 278-282.  
(Presented the notion of merging decision trees to form a "forest.")  
  
Scornet, E., and G. Biau (2016). A Guided Tour of a Random Forest. Test, 25(2), 197-227.  
10.1007/s11749-016-0481-7 is the doi  
(Offers insights into Random Forests' theory, operation, and application.)  
  
Cernadas, E., Barro, S., Fernández-Delgado, M., & Amorim, D. (2014). Can Real-World Classification Problems Be Solved with Hundreds of Classifiers? 15, 3133–3181; Journal of Machine Learning Research.  
(Compares Random Forests' performance to that of other classifiers.)