**To:** Instructor Nikhil Bhardwaj

**From:** Yeison Montoya - ID: 300375916

Link to my [Jupyter Notebook](https://github.com/yeisonmontoya1815/Special-Topics-in-Data-Analytics-CSIS-4260-002/blob/main/Seminar%201%20-%20Random%20Forest%20Algorithm.ipynb)

# Data mining techniques for structured data

## Executive Summary

Data mining involves extracting insights from extensive databases, focusing on uncovering patterns, and gaining additional knowledge from existing datasets rather than introducing entirely new information. Classification, a supervised learning technique within data mining, categorizes data into predefined classes by learning from labelled training data to predict class labels for new instances. The training phase constructs a model based on relationships between features and class labels in the labelled dataset, aiming to capture inherent patterns and decision boundaries. In the testing phase, the established model is applied to predict class labels for new, unlabeled instances.

This report delves into the Random Forest Algorithm, a popular model for classification within data mining. They construct multiple decision trees during training outputs the class by aggregating the mode of classes (for classification) or mean predictions (for regression) from individual trees. Numerous data mining tools and open-source libraries, such as "Orange," are widely utilized, exemplifying the versatility and accessibility of these techniques. The report includes a demonstration of applying the Random Forest Algorithm using the "Orange" software, showcasing its practical application in the field of data mining.

# Random Forest Algorithm

The Random Forest algorithm is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Each tree is built using a subset of the training data and a subset of the features. The basic mathematical model for a Random Forest can be broken down into several components.

* **Decision Trees:** a flowchart-like structure where each internal node represents a decision based on the value of a particular feature. This model can be represented by a set of rules based on the features of the data.
* **Bagging (Bootstrap Aggregating):** building multiple decision trees and preparing a random subset of the training data. The subsets are created by bootstrapping, which involves random sampling with replacement from the original training dataset.
* **Feature Randomness:** At each split in the decision tree, a random subset of features is considered. This adds a layer of randomness to the model, preventing it from relying too heavily on any feature.
* **Voting (Classification) or Averaging (Regression):** In the case of classification, each tree "votes" for a class, and the class with the most votes becomes the predicted class for the Random Forest.

In the case of regression, the predicted output is the average of the outputs predicted by all individual trees. Mathematically, suppose *T* represents the number of trees in the forest. In that case, *t* represents a particular tree, and *f (x, t)* represents the prediction of the *t-*th tree for input *x*, then the prediction for classification is given by:

mode({f(x, t)∣ t=1,…,T})

And for regression:

Where the mode is the value that appears most frequently in the set of predictions, this ensemble approach often results in a more robust and accurate model compared to individual decision trees, as it helps mitigate overfitting and captures a broader range of patterns in the data.

## How it works

Random forest algorithm involves setting hyperparameters (node size, number of trees, and sampled features) before training. It can solve regression or classification problems. It uses a collection of decision trees from a bootstrapped sample and reserves one-third as test data (OOB sample). Feature bagging is used to increase dataset diversity and reduce correlation among decision trees. Predictions are determined by averaging decisions for regression and majority vote for classification. The OOB sample is used for cross-validation to finalize the prediction.

Figure 1. Decoding Random Forest: Hyperparameter Tuning and Ensemble Dynamics

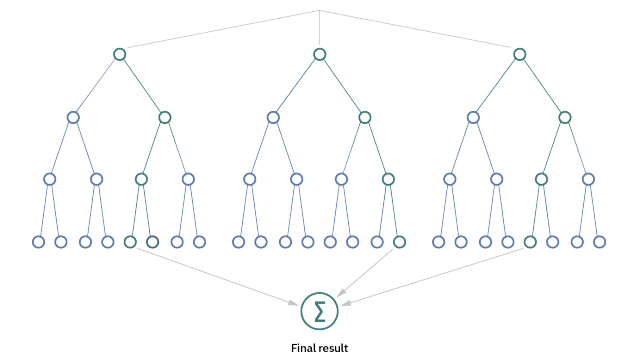


Image retrieved by [IBM web page](https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems)

Consider a simple example of a Random Forest for a binary classification task with three decision trees.

Table 1. Dataset for binary classification task

|  |  |  |
| --- | --- | --- |
| Feature 1 | Feature 2 | Target |
| 2 | 3 | 0 |
| 3 | 5 | 1 |
| 5 | 1 | 0 |
| 6 | 4 | 1 |
| 4 | 2 | 0 |
| 7 | 3 | 1 |

Proceed with building a Random Forest consisting of three decision trees. Here, each tree will be trained on a random subset of the given data. During each split, a random subset of features will be considered. After the training process, suppose the individual tree predictions for a new data point (X\_new) are as follows:

* Tree 1 predicts class 1
* Tree 2 predicts class 0
* Tree 3 predicts class 1

For a classification task, the Random Forest combines predictions through a majority vote. In this case, two out of three trees predict class 1. Therefore, the **Random Forest's final prediction for X\_new is class 1**.

## Random Forest Algorithm (an example in Python environment)

In this example, the Random Forest Algorithm is applied to the Iris dataset using Python's sci-kit-learn library. The dataset is divided into features and target labels and split into training and testing sets. A Random Forest Classifier with 100 trees is created and trained on the training set. The model's performance is evaluated on the test set using metrics and the feature importance of each variable. All the code and other simulations are available in my [Jupyter Notebook](https://github.com/yeisonmontoya1815/Special-Topics-in-Data-Analytics-CSIS-4260-002/blob/main/Seminar%201%20-%20Random%20Forest%20Algorithm.ipynb).

## Results

Accuracy: 1.0

Confusion Matrix:

[[10 0 0]

[ 0 9 0]

[ 0 0 11]]

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 10

1 1.00 1.00 1.00 9

2 1.00 1.00 1.00 11

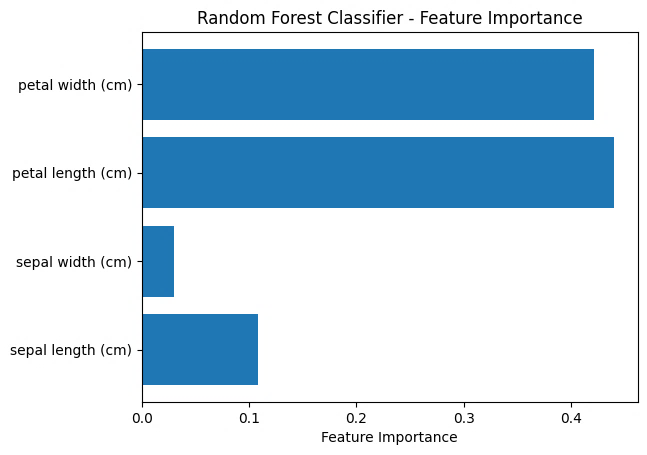
Accuracy 1.00 30

macro av 1.00 1.00 1.00 30

weighted avg 1.00 1.00 1.00 30

* The results of the Random Forest Classifier applied to the Iris dataset indicated an accuracy score of 1.0, implying a perfect classification of the test set.
* The confusion matrix further supports this with all diagonal elements being non-zero, indicating correct predictions for each class (setosa, versicolor, and virginica).
* The classification report provides additional details on the precision, recall, and F1-score for each class, all of which are at the maximum value of 1.0, emphasizing the model's ability to correctly identify instances from each class.
* The macro and weighted averages also highlight the overall excellent performance across all classes.
* The accompanying feature importance graph further contributes to the interpretability of the model, showing the contribution of each feature to the classification task. In summary, the Random Forest Algorithm demonstrates outstanding accuracy and robustness in classifying the Iris dataset, making it a reliable choice for similar classification tasks.

Figure 2. Bar chart of random forest model

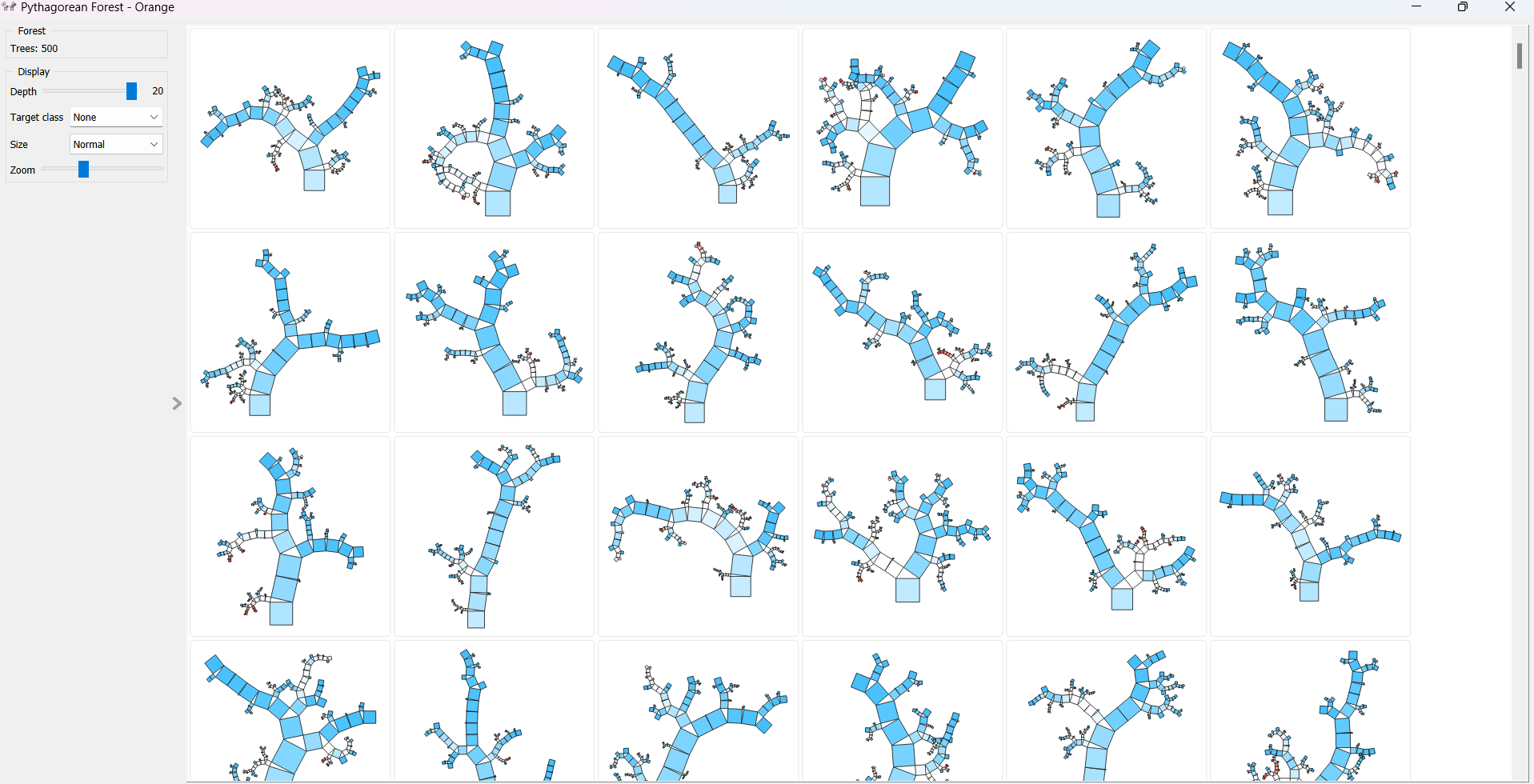
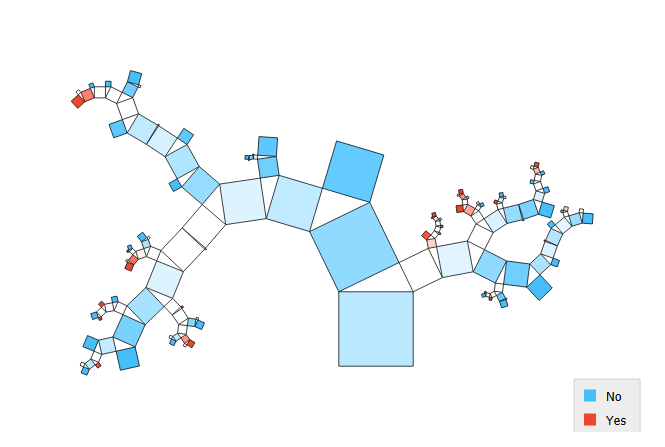


This example serves as a comprehensive illustration of the implementation and evaluation of the Random Forest Algorithm, showcasing its versatility in handling real-world datasets and providing insights into feature importance for better interpretability.

## Random Forest Algorithm, a practical example using a software tool (Orange)

The following figure explores interpretability in classification trees using the Pythagorean Tree viewer for larger trees. The spotlight then shifts to random forests, where we introduce and compare their performance to single trees via cross-validation. Additionally, this example provides a concise and comprehensive exploration of interpretability within classification trees and the considerations involved in employing ensemble methods like random forests through the use of “Orange” software. . Typically, Random Forests are visualized by aggregating the information from multiple decision trees. More detailed information about the ORANGE tool for the random forest algorithm can be found in my [Jupyter Notebook](https://github.com/yeisonmontoya1815/Special-Topics-in-Data-Analytics-CSIS-4260-002/blob/main/Seminar%201%20-%20Random%20Forest%20Algorithm.ipynb).

Figure 4. Decision tree model Pythagorean view(left) Vs Random Forest Pythagorean view (right)



## When to use Random Forest Algorithm

* **Classification and Regression Tasks:** Random Forests are versatile and can be applied to both classification and regression problems. They excel in scenarios where the relationship between features and the target variable is complex and non-linear.
* **Large Feature Sets:** Random Forests can handle datasets with a large number of features efficiently. They are capable of automatically selecting relevant features and providing insights into feature importance.
* **Ensemble Learning:** When the goal is to leverage the benefits of ensemble learning, Random Forests provides a robust solution. They combine the predictions of multiple decision trees to enhance overall model performance and reduce overfitting.

## Where NOT to use it

* **Interpretability is Critical:** If interpretability is crucial and a highly interpretable model is required, Random Forests might not be the best choice. The complexity of the ensemble makes it challenging to interpret individual decision trees comprehensively.
* **Real-Time Applications:** In real-time applications requiring low-latency predictions, Random Forests might not be the most suitable choice. The ensemble nature and computational intensity can slow down real-time predictions.
* **Linear Relationships:** If the underlying relationship between features and the target variable is predominantly linear, simpler linear models may offer a more interpretable and efficient solution.

## Summary

Classification trees are valuable for interpretability, but on employee attrition data, they become too large and intricate for practical understanding, revealing their instability with slight variations in samples. To address this, the concept of a random forest, or classification forest, is introduced. Instead of a single tree, a forest of trees is built through data sampling, providing diverse perspectives and voting on predicted classes, resulting in improved accuracy.

The text showcases the performance gain of random forests, highlighting an accuracy increase from 0.81 to 0.85 compared to individual trees. The impact of the number of trees in the forest is discussed, revealing a slight accuracy improvement with more trees. While excelling in accuracy, the challenge lies in interpretation, acknowledging the difficulty of understanding complex machine learning models despite their high predictive accuracy. Future videos are promised to explore additional classifiers and interpretation methods.

# References

##### ELKI Data Mining Toolkit. Retrieved from <https://elki-project.github.io/>

##### James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning: With Applications in R*. Springer.

##### Keras. Deep Learning for humans. Retrieved from <https://keras.io/>

##### KNIME. (n.d.). KNIME | Open for Innovation. Retrieved from <https://www.knime.com/>

##### Outlier Detection Toolbox (ODTBX). (2020). Retrieved from <https://www.sciencedirect.com/science/article/pii/S2352340920301316>

##### PyOD. A Python Toolbox for Scalable Outlier Detection. Retrieved from <https://pyod.readthedocs.io/>

##### PyTorch. Retrieved from <https://pytorch.org/>

##### RapidMiner - Your Competitive Edge in Data Science. Retrieved from <https://rapidminer.com/>

##### Raschka, S. MLxtend: Extensions and Helper Tools for Python's Data Science Ecosystem. GitHub. Retrieved from <https://github.com/rasbt/mlxtend>

##### Scikit-learn. Machine Learning in Python — scikit-learn 0.24.2 documentation. Retrieved from <https://scikit-learn.org/stable/>

##### Scikit-learn.ensemble.IsolationForest — scikit-learn 0.24.2 documentation. Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html>

##### Statsmodels. Retrieved from <https://www.statsmodels.org/>

##### TensorFlow. (n.d.). Retrieved from <https://www.tensorflow.org/>

##### Zupan, B. (Writer). (2023, August 21). Random Forests [Video]. Presented by N. Novšak. Produced and edited by L. Zupan. Biolab. Retrieved from <https://www.youtube.com/watch?v=gSQsFIMcA8A>