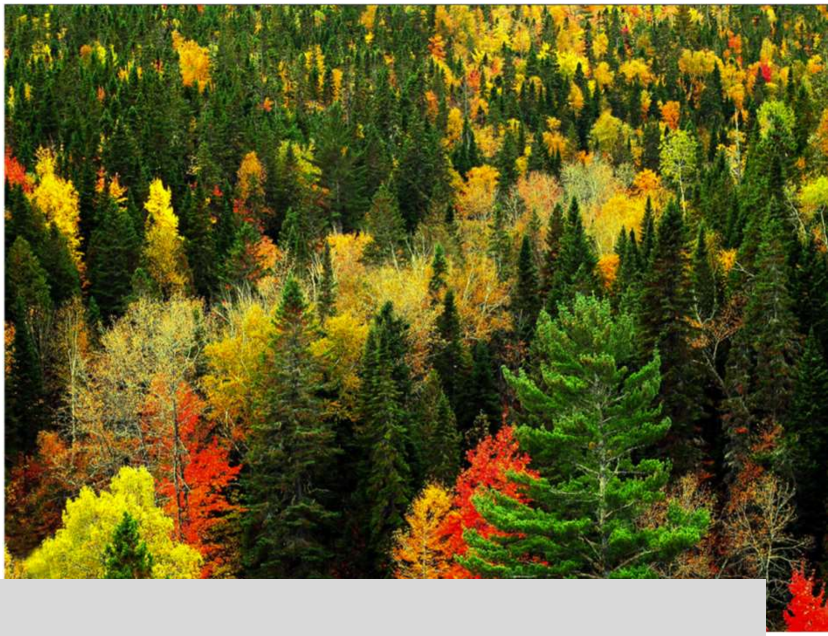
What is a Random Forest ? How does it work?

- **Random Forest** is considered to be a ***panacea*** of all data science problems and consists of many decision trees.
  - On a funny note, when you can’t think of any algorithm, use random forest!
- **Random Forests** are a way of averaging multiple decision trees,
  - trained on different parts of the same training set
  - with the goal of overcoming the over-fitting problem of individual decision tree.



What is Overfitting?

Explaining your training data instead of finding patterns that generalize is what overfitting is.

In other words, your model learns the training data by heart instead of learning the patterns which prevent it from being able to be generalized to the test data.

It means your model fits well on the training dataset but fails on the validation dataset.

How does it work?

- In **Random Forest,** we grow multiple trees as opposed to a single tree in CART model

- To classify a new object based on attributes,
  - each tree gives a classification
  - we say the tree “votes” for that class.
- The forest chooses the classification having the most votes
- In case of regression, it takes the average of outputs by different trees.

How does it work?

- In **Random Forest,** we grow multiple trees as opposed to a single tree in CART model

- To classify a new object based on attributes,
  - each tree gives a classification
  - we say the tree "votes" for that class.
- The forest chooses the classification having the most votes
- In case of regression, it takes the average of outputs by different trees.

Decision Tree vs. Random Forest

Decision tree is encountered with over-fitting problem and ignorance of a variable in case of small sample size and large p-value.

Random forests are a type of recursive partitioning method particularly well-suited to small sample size and large p-value problems.

Random forest comes at the expense of a some loss of interpretability, but generally greatly boosts the performance of the final model.

Popularity of Random Forest Algorithm

- **Random Forest** is one of the most widely used machine learning algorithm for classification.
- It can also be used for regression model (i.e. continuous target variable) but it mainly performs well on classification model (i.e. categorical target variable).
- It has become a lethal weapon of modern data scientists to refine the predictive model.
- The best part of the algorithm is that there are a very few assumptions attached to it so data preparation is less challenging and results in time saving.

Popularity of Random Forest Algorithm

- **Random Forest** is one of the most widely used machine learning algorithm for classification.
- It can also be used for regression model (i.e. continuous target variable) but it mainly performs well on classification model (i.e. categorical target variable).
- It has become a lethal weapon of modern data scientists to refine the predictive model.
- The best part of the algorithm is that there are a very few assumptions attached to it so data preparation is less challenging and results in time saving.

Popularity of Random Forest Algorithm

- **Random Forest** is one of the most widely used machine learning algorithm for classification.
- It can also be used for regression model (i.e. continuous target variable) but it mainly performs well on classification model (i.e. categorical target variable).
- It has become a lethal weapon of modern data scientists to refine the predictive model.
- The best part of the algorithm is that there are a very few assumptions attached to it so data preparation is less challenging and results in time saving.

Popularity of Random Forest Algorithm

- **Random Forest** is one of the most widely used machine learning algorithm for classification.
- It can also be used for regression model (i.e. continuous target variable) but it mainly performs well on classification model (i.e. categorical target variable).
- It has become a lethal weapon of modern data scientists to refine the predictive model.
- The best part of the algorithm is that there are a very few assumptions attached to it so data preparation is less challenging and results in time saving.

How random forest works

Each tree is grown as follows:

1. **Random Record Selection :** Each tree is trained on roughly 2/3rd of the total training data.
    - Cases are drawn at **random with replacement** from the original data. This sample will be the training set for growing the tree (**bootstrapping**).

2. **Random Variable Selection :** Some predictor variables (say, *m*) are selected at **random** out of all the predictor variables and the best split on these *m* is used to split the node.

Note : In a standard tree, each split is created after examining every variable and picking the best split from all the variables.

How random forest works

Each tree is grown as follows:

1. **Random Record Selection :** Each tree is trained on roughly 2/3rd of the total training data.
    - Cases are drawn at **random with replacement** from the original data. This sample will be the training set for growing the tree (**bootstraping**).

2. **Random Variable Selection :** Some predictor variables (e.g.: ***m***) are selected at **random** out of all the predictor variables and the best split on these *m* is used to split the node.

Note : In a standard tree, each split is created after examining every variable and picking the best split from all the variables.

How random forest works

Each tree is grown as follows:

3. **For each tree**, using the leftover (36.8%) data, calculate the misclassification rate - **out of bag (OOB) error rate.**
    - Aggregate error from all trees to determine **overall OOB error rate** for the classification.

4. **Each tree gives a classification** on leftover data (OOB), and we say the tree "votes" for that class.
    - The forest chooses the classification having the most votes over all the trees in the forest.
    - **For a binary dependent variable,** the vote will be **YES** or **NO**, count up the YES votes. This is the **RF score** and the percent **YES** votes received is the predicted probability.
    - **In regression case**, it is the average of the dependent variable**.**

How random forest works

Each tree is grown as follows:

3. **For each tree**, using the leftover (36.8%) data, calculate the misclassification rate - **out of bag (OOB) error rate.**

  - Aggregate error from all trees to determine **overall OOB error rate** for the classification.

4. **Each tree gives a classification** on leftover data (OOB), and we say the tree "votes" for that class.

  - The forest chooses the classification having the most votes over all the trees in the forest.
  - **For a binary dependent variable,** the vote will be **YES** or **NO**, count up the YES votes. This is the **RF score** and the percent **YES** votes received is the predicted probability.
  - **In regression case**, it is the average of the dependent variable**.**

What is random in 'Random Forest'?

'Random' refers to mainly two process

1. random observations to grow each tree
2. random variables selected for splitting at each node**.**

Pruning

In random forest, each tree is fully grown and not pruned. In other words, it is recommended not to prune while growing trees for random forest.

Methods to find best Split

The best split is chosen based on Gini Impurity or Information Gain methods.

What is random in 'Random Forest'?

'Random' refers to mainly two process

1. random observations to grow each tree
2. random variables selected for splitting at each node**.**

Pruning

In random forest, each tree is fully grown and not pruned. In other words, it is recommended not to prune while growing trees for random forest.

Methods to find best Split

The best split is chosen based on Gini Impurity or Information Gain methods.

What is random in 'Random Forest'?

'Random' refers to mainly two process

1. random observations to grow each tree
2. random variables selected for splitting at each node**.**

Pruning

In random forest, each tree is fully grown and not pruned. In other words, it is recommended not to prune while growing trees for random forest.

Methods to find best Split

The best split is chosen based on Gini Impurity or Information Gain methods.

The forest error rate:

It depends on two things:

1. The correlation between any two trees in the forest.
2. The strength of each individual tree in the forest.
   - A tree with a low error rate is a strong classifier.
   - Increasing the strength of the individual trees decreases the forest error rate.

The number of random variables used in each tree (mtry)

- Reducing **mtry** reduces both the correlation and the strength.
- Increasing it increases both.

➔ Somewhere in between is an "**optimal**" range of **mtry** - usually quite wide.

➔ Using the **oob error rate** a value of **mtry** in the range can be quickly found.

The forest error rate:

It depends on two things:

1. The correlation between any two trees in the forest.
2. The strength of each individual tree in the forest.
   - A tree with a low error rate is a strong classifier.
   - Increasing the strength of the individual trees decreases the forest error rate.

The number of random variables used in each tree (mtry)

- Reducing **mtry** reduces both the correlation and the strength.
- Increasing it increases both.

➔ Somewhere in between is an "**optimal**" range of **mtry** - usually quite wide.
➔ Using the **oob error rate** a value of **mtry** in the range can be quickly found.

Random Forest

Example:

Import the data file called **“breast-cancer_shuffled.csv”** into the variable **mydata**

> mydata= read.table(“breast-cancer_shuffled.csv”, header=TRUE, sep=“,”, stringsAsFactors = TRUE)
 # Check the dimensions of the data
 # View statistical summary of dataset
 # View the complete data

Process the dataset

- Divide **mydata** in two data frames as **dfTraining** (70%) and **dfTest** (30%)
- Create a **random forest** for classifying the data based on the type with **dfTraining** (function in R: **randomForest**)
- Check the created forest and its performance on the training data
- Create a **confusion matrix** for the predicted and true type on **dfTest**

Random Forest

Example:

Process the dataset

- Divide **mydata** in two data frames as **dfTraining** (70%) and **dfTest** (30%)

```
>countTraining = round(nrow(mydata)*0.7)
>randomRows=sample(1:nrow(mydata), size= countTraining, replace=F)
>dfTraining = mydata[randomRows,]
>dfTest = mydata[- randomRows,]
```

- Create a **random forest** for classifying the data based on the type with **dfTraining** (function in R: **randomForest**)

```
>library(randomForest) #library containing the randomForest function
>myForest = randomForest(type~., dfTraining)
#Create a random forest with type as class attribute and all other attributes as variables
```

- Check the created forest and its performance on the training data
- Create a **confusion matrix** for the predicted and true type on **dfTest**

Random Forest

Example:

Process the dataset

- Check the created forest and its performance on the training data

>myForest

- Create a **confusion matrix** for the predicted and true type on **dfTest**

Random Forest

Example:

Process the dataset

- Create a **confusion matrix** for the predicted and true type on **dfTest**

```
>myPred = predict(myForest, dfTest, type = "class")
>table(myPred, dfTest$type)
```

Random Forest

Exercise:

Import the data file called **“forestTypeTraining.csv**” into the variable **myTraining**
Import the data file called **“forestTypeTesting.csv**” into the variable **myTesting**

Process the dataset

- Create a **random forest** for classifying the data based on the class with **myTraining** (function in R: **randomForest**)
- Check the created forest and its performance on the training data
- Create a **confusion matrix** for the predicted and true class on **myTesting**

Random Forest

GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Exercise:

Import the data file called **“forestTypeTraining.csv**” into the variable **myTraining**
Import the data file called **“forestTypeTesting.csv**” into the variable **myTesting**

```
> myTraining = read.table("forestTypeTraining.csv", header = TRUE, sep =",", stringsAsFactors = TRUE)
> myTesting = read.table("forestTypeTesting.csv", header = TRUE, sep =",", stringsAsFactors = TRUE)
```

Process the dataset

- Create a **random forest** for classifying the data based on the class with **myTraining** (function in R: **randomForest**)
- Check the created forest and its performance on the training data
- Create a **confusion matrix** for the predicted and true class on **myTesting**

Random Forest

Exercise:

Process the dataset

- Create a **random forest** for classifying the data based on the class with **myTraining** (function in R: **randomForest**)

>myForest = randomForest(class~., myTraining)

- Check the created forest and its performance on the training data

>myForest

- Create a **confusion matrix** for the predicted and true class on **myTesting**

Random Forest

Exercise:

Process the dataset

- Create a **random forest** for classifying the data based on the class with **myTraining** (function in R: **randomForest**)

>myForest = randomForest(class~., myTraining)

- Check the created forest and its performance on the training data

>myForest

- Create a **confusion matrix** for the predicted and true class on **myTesting**

>myPred = predict(myForest, myTest, type = "class")
>table(myPred,myTest$class)

Random Forest

Exercise:

Import the data file called **"breast-cancer_shuffled.csv**" into the variable **mydata**
Process the dataset

Step 1:

- Shuffle and divide **mydata** in two data frames as **dfTraining** (70%) and **dfTest** (30%)
- Perform a Random Forest (RF) classifier and save the model in a variable called **myForest**
- Examine **myForest** and access the information for the error and the variable importance
- Plot the error rates
- Plot the variable importance
- Using **dfTest** create the confusion matrix and interpret the results
- Change the **type** parameter in predict function to "**vote**"

Random Forest

Exercise:

Step 2: ROC plot

- Consider the success probability of the predictions for **no-recurrence-events** (default: 0.5)
- Starting with 0 increase the probability by an increment of 0.01 to a maximum of 1 (100 steps)
- For each incremental step:
  - calculate the TPR and FPR
  - Save the results
- Create a plot with FPR on the X axis and TPR on the Y axis

$$TPR = \frac{TP}{TP + FN} \qquad TP + FN\!: real\ number\ of\ all\ positive\ cases$$

$$FPR = \frac{FP}{FP + TN} \qquad FP + TN\!: real\ number\ of\ all\ negative\ cases$$

Random Forest

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Exercise:

accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Step 3: Parameter optimization in RF

- Find the optimal number of mtry (number of variables) and **ntree** (number of trees) which maximize the ACC

- Write a **function** that:
  - Given parameters **mtry** (number of variables) and **ntree** (number of trees)
  - Create a **random forest** with above parameters on **dfTraining**
  - Calculate the accuracy of the prediction on **dfTest** and return it
- Write a **function** that:
  - Given parameter **mtry**
  - Loops over all values in the vector seq(100,1000,10) as **ntree**
  - Call the function above with **mtry** and **ntree** as parameters and return the **mtry** value and accuracy which achieve the highest accuracy (Hint: Use a **data.frame** to return multiple values)
- Write a **function** that:
  - Loops over all values in the vector 1:9 as **mtry**
  - Call the function above with **mtry** as parameter and return the **mtry** and **ntree** values which achieve the highest accuracy as well as the accuracy
- Call the function above