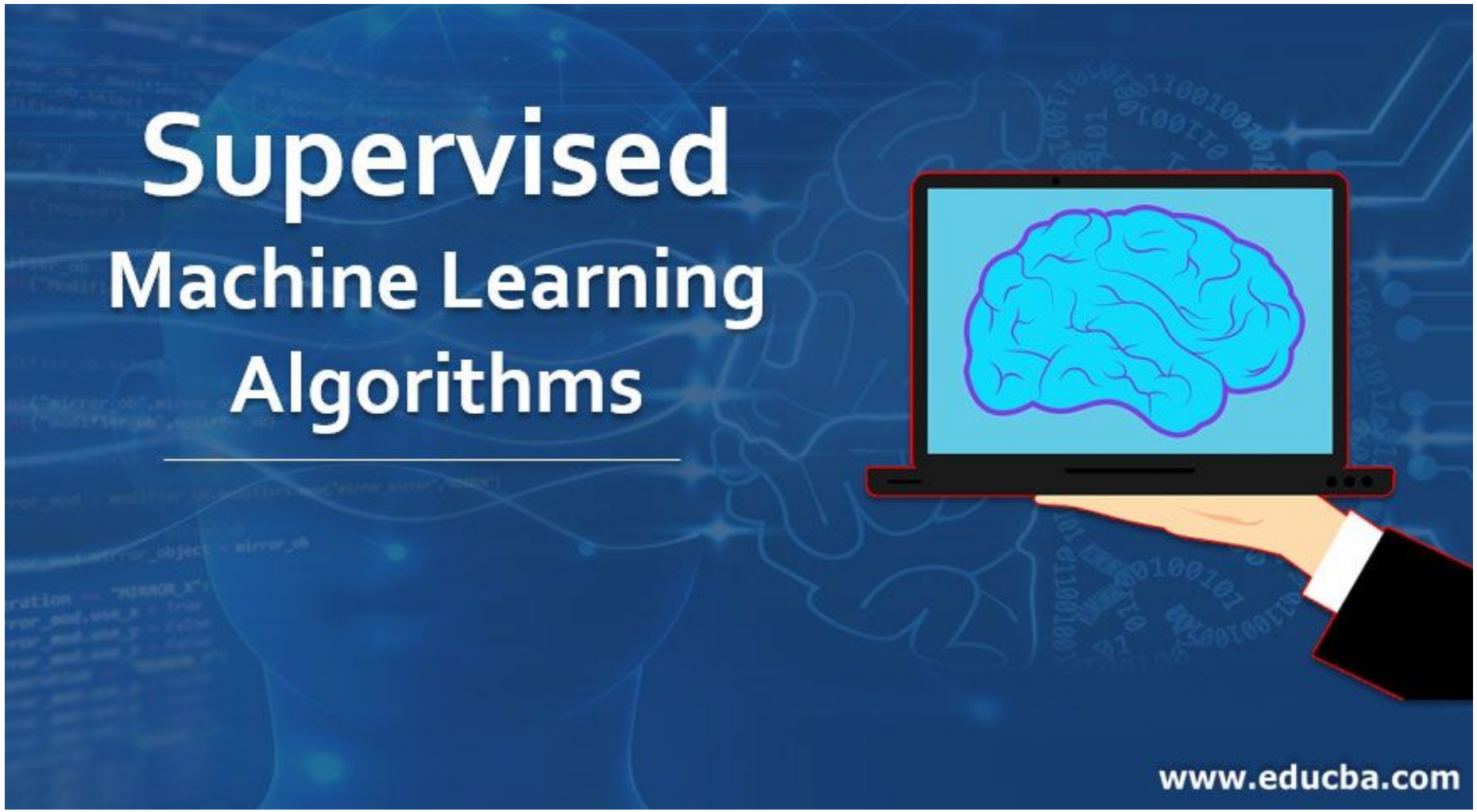

Supervised
Machine Learning
Algorithms
www.educba.com

Last lecture reminder

We learned about:

- Decision Tree - Gini Impurity usage
- Decision Tree - Gini Impurity with multiple features
- Decision Tree - Simple Python example
- Decision Tree model with custom parameters
- Decision Tree Parameters - Python example

Random Forest - Introduction

Random Forest → A Random Forest is a machine learning model that is used for both regression and classification tasks. It is an ensemble learning method, where a group of weak models combine to form a strong model. In the case of a random forest, the weak models are decision trees.

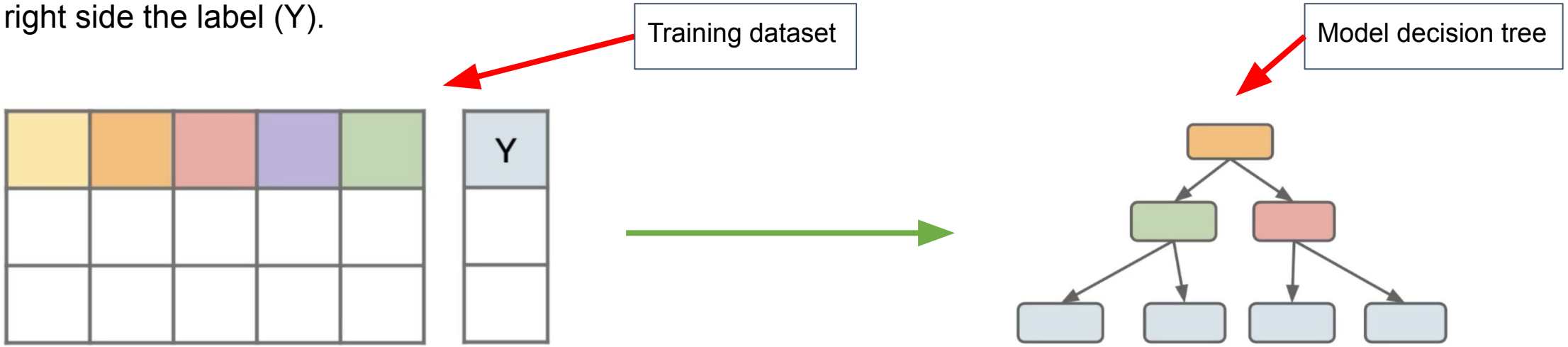
The idea behind a Random Forest is simple but powerful. When we construct many decision trees, each on a different fraction of the original data, and each having a different error, we can reduce this error by averaging the results of all trees (in regression) or by taking a vote for the most popular outcome (in classification).

Overall, the Random Forest model generates multiple decision trees and merges them to get a more accurate prediction, enhancing the model's ability to generalise over different datasets.

Random Forest - Introduction

Let’s try to visualize how random forest is working and generating the different decision trees.

Let’s say we have the following dataset while on the left side we have the different features and on the right side the label (Y).



As expected, some of the features in the original model was not included in the decision tree itself making them redundant to the model prediction process.

Random Forest - Introduction

A Random Forest model will perform the following process:

1. Randomly select a subset of features and construct the first split according to Gini Impurity value from this feature group only.
2. In case the model need additional split, a new group of features been randomly selected.
3. The algorithm then split the tree again according to Gini Impurity value from this feature group only.
4. The algorithm keep repeating the process until the entire tree completed.
5. Once the tree was completed the algorithm start creating a new decision tree in the same way.
6. Once the selected number of trees been created the algorithm will predict according to the following condition:

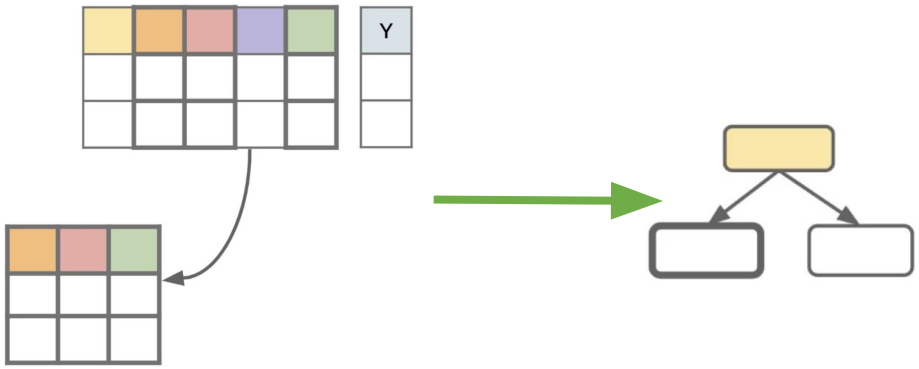
   Regression problem → The prediction value will be the average value from all the decision trees.

   Classification problem → The prediction value will be the most predicted class from all decision trees.

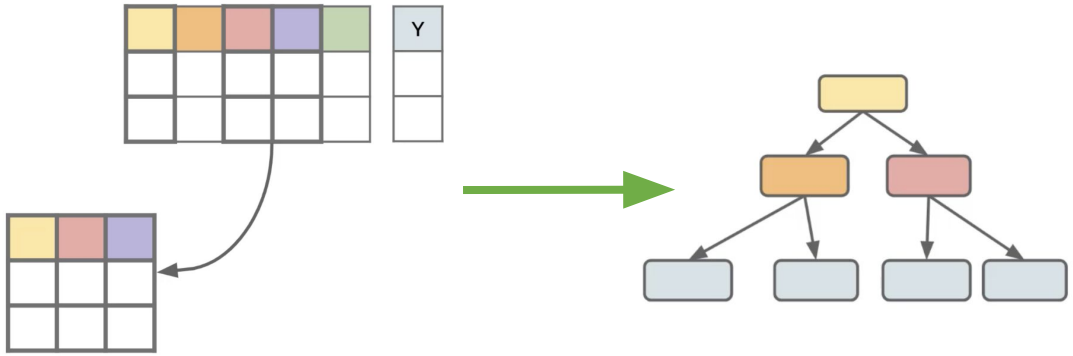
Random Forest - Introduction

Let’s continue our previous visualization and now run a Random Forest model.

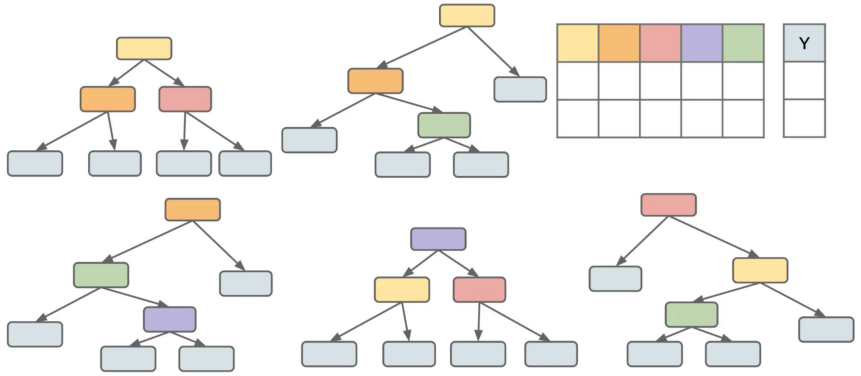
First iteration:



Second iteration:



The final Random Forest result will be different decision trees build from different features:



Random Forest Vs Single Tree

Main disadvantage with single decision tree models:

- **Overfitting** → Single decision trees tend to overfit the training data, particularly if the tree is allowed to grow deep with many layers. This is because they keep splitting until they have perfectly classified all samples in the training data, often at the expense of being able to generalize well to new, unseen data. This will eventually cause overfitting to the training data.

- **Removing features** → Single decision tree may not take into account all the features. Certain features may be disregarded if they do not improve the purity of the nodes (Gini Impurity value) during the splitting process. Not considering some data features in the model prediction process may also cause to misclassification errors.

Random Forest Vs Single Tree

How Random Forest model deal with those disadvantages?

- **Handle overfitting** → With Random Forest, the model process many decision trees that different from each other and based the model prediction on aggregation values like average prediction value in regression problems and the most predicted category in classification problems. This reduce the risk for overfitting to the training dataset because each tree been constructed from partial group of features from the entire data.

- **Handle removing features** → The way Random Forest is generate it's trees is by randomly selecting different group of features each time and construct the decision tree according to those selected features.This process make sure that almost every feature in the dataset will be a part of at least 1 decision tree in the forest. In addition, this feature randomness can prevent one or few strong features from dominating across all trees, and thus enable the algorithm to explore a diversity of feature interactions and improve model robustness.

Random Forest - Parameters

Most of the Random Forest algorithm parameters are similar to the basic single decision tree parameters that we learned in the previous lectures.
For example, min_samples_split, max_leaf_nodes, etc…)
Once we configured specific decision tree parameters in the Random Forest model it will effect all decision trees that been generated.

In Random Forest we still have unique parameters to choose from:

- **n_estimators** → Determine how many different decision trees our Random Forest model will generate.
- **max_features** → Determine the number of features to include in each split decision.
- **bootstrap** → Determine if the algorithm will use bootstrapping to resample the original dataset.
- **oob_score** → Determine if the model will include also ‘Out-Of-Bag’ scoring.

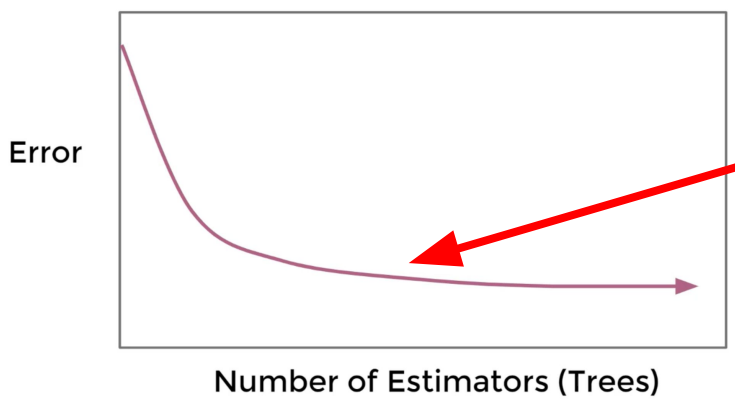
Random Forest - Parameters

Let’s now understand how each parameter can affect our model performance.

Number of generated decision trees (n_estimators) → Intuitively, the more decision trees our Random Forest algorithm will generate the more the model will learn about the provided data, and the more features will combined in the prediction process.

Is if there is a limit to adding more trees? Is adding more trees can cause overfitting?

The best way to answer this question is to plot a line chart that shows for each number of trees the corresponding error.



We can see that when passing some threshold, increasing the number of trees not reducing the error value but also not making it spike.

Random Forest - Parameters

In conclusion, we can say on the ‘n_estimators’ parameter the following:

- Increasing the number of trees will increase the model accuracy to some point
- Increasing the number of trees above a specific threshold will no longer improve the model accuracy but it will also not caused overfitting.

Why increasing the number of trees won’t cause overfitting?

When increasing the number of trees to some point our model will keep generating similar and similar trees. Because there is no correlation between decision tree to another (each decision tree stand by it’s own) new similar decision trees will not add any new information to the model and therefor won’t affect the model accuracy at all.

Note: A reasonable ‘n_estimators’ default value can be 100 trees and after that adjust the number according to our need (by using a grid search for example). Adding more trees won’t cause an overfitting but it will increase the model generalization cost.

Random Forest - Parameters

Number of features (max_features) → The default number of features we should choose is **sqrt(N)** when N is the number of total features in our dataset.

Same with the ‘n_estimators’ parameter, we could use a grid search to adjust the number of features to include in each split decision.

Bootstrapping (bootstrap) → The bootstrap boolean parameter determine if we allowing bootstrap sampling of each training subset of features.

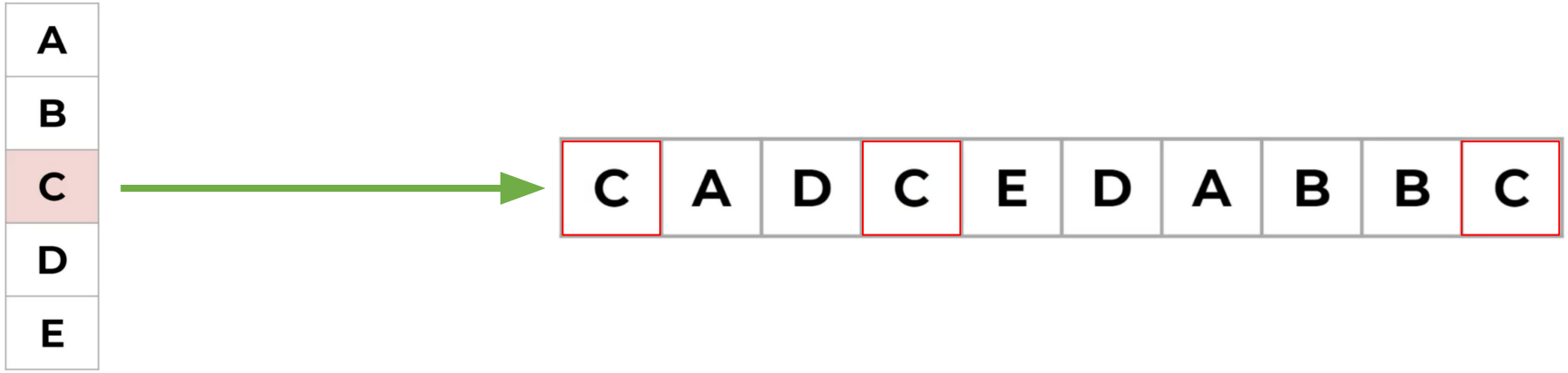
A bootstrap sampling meaning that for each new decision tree in our Random Forest we won’t use the entire dataset given but will randomly generated a subset from the original dataset and construct the decision tree according to this subset.

In addition the bootstrapping process also involve ‘replacement’, meaning that we can repeat the same data more than once.

Random Forest - Bootstrapping

Let’s take a look at a simple bootstrapping example:

We want to randomly select 10 rows from the following set of 5 letters.



Because we need to choose 10 letters from a dataset of only 5 letters we don’t have any other option but to choose some letters more than once.

This is the idea of bootstrapping, the algorithm will randomly choose rows from the original dataset but it will also allow to choose rows more than once.

Random Forest - Bootstrapping

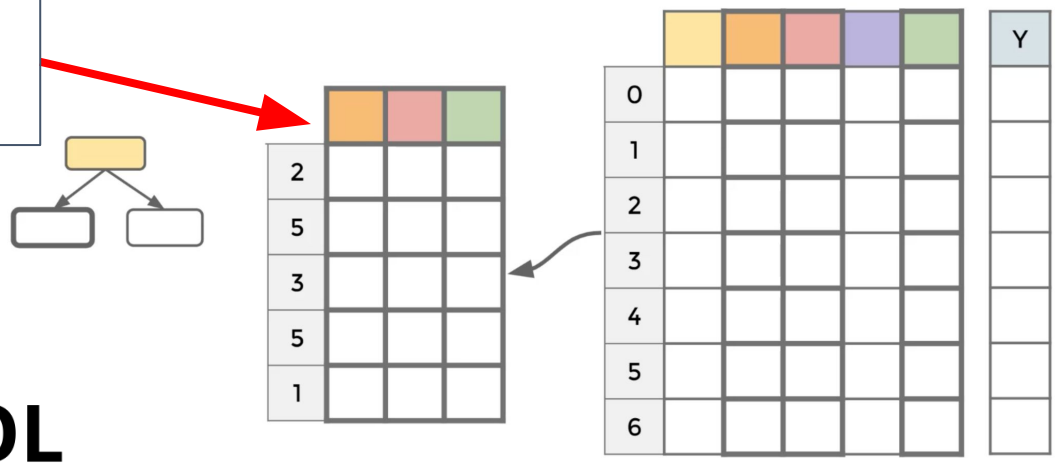
Why to use bootstrapping?

By applying bootstrapping we allowing our model to generate more different trees because each tree not only contain different features but also been constructed from a different dataset.

This action will reduce the correlation between different trees and allowing our model to gain more new insights from each decision tree it generated.

If all trees were identical or very similar, they would all make the same errors. With tree diversity, errors made by one tree are likely to be corrected by others.

With bootstrapping, each tree constructed with different features subset and different rows subset.



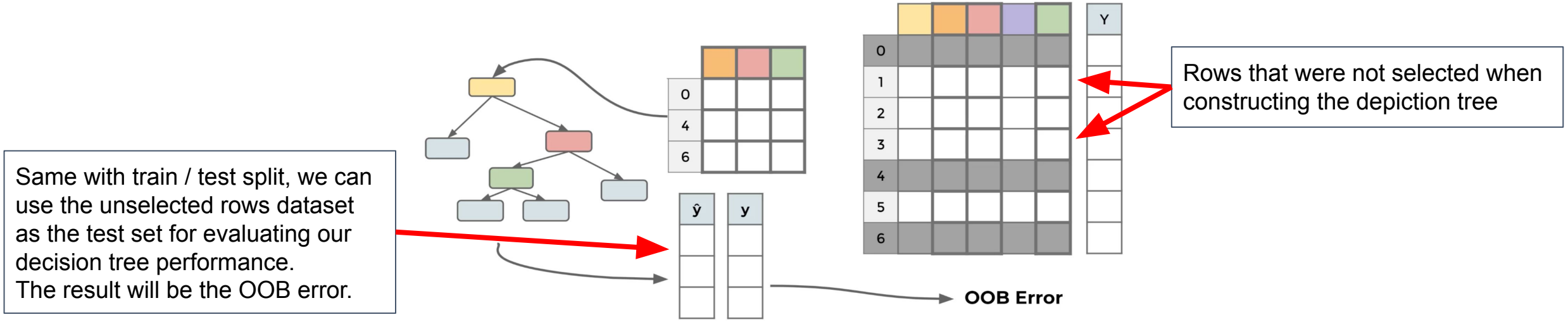
Random Forest - Out-Of-Bag Error

Bagging → Bagging is a term related to bootstrapping meaning that our model is using a bootstrapped data and then calculate a prediction based on the aggregated prediction of all the depiction trees (for classification - most voted class, for regression - average predicted value).
This construct the term **'Bag'** - **b**ootstrapped **ag**gregated data.

Out-Of-Bag Error (oob_score) → This boolean parameter determine if our model will calculate the OOB error during the training process. This error been calculated by evaluating the decision tree accuracy based on the unselected rows.
When applying bootstrapping we randomly select subset of rows to construct the decision tree. This mean that some rows will not be selected.
Calculating the OOB error means to use those unselected rows as test set in order to evaluate the accuracy of the decision tree we created.

Random Forest - Out-Of-Bag Error

To better understand this, let's take a look at the following chart:



OOB error will not affect the training process but will provide additional measurement regarding our algorithm performance.

In addition, this error is much less useful because it only been calculated separately on each decision tree and not on the entire model like in a classic train / test split. This is why the default value for this parameter is set to False.

Random Forest - Out-Of-Bag Error

Why bootstrapping allowing replacement?

As we mentioned, when performing bootstrapping we also allowing the model to select the same row more than once (replacement). Selecting the same row more than once in bootstrapping does influence the results somewhat, but in a positive way for Random Forests.

The key benefit from replacement is that it adds an extra layer of variation to the training process of each decision tree in the forest since each tree is trained on a slightly different set of data. This increases the diversity among the trees in the forest and thereby helps to improve the robustness and generalization ability of the overall model.

So, even though selecting the same row more than once may create some bias in a single bootstrap sample, when considered across all the trees and across all the different bootstrap samples in a Random Forest, it actually contributes to reducing the variance of the model's predictions without increasing the bias too much.

Random Forest - Python Example

For this example we will use the **‘data_banknote_authentication.csv’** file.

The "data_banknote_authentication" dataset is a commonly used dataset for binary classification problems in machine learning. It's used to distinguish genuine money bills from forgeries. The data was extracted from images that were taken from both genuine and forged money bills.

Dataset columns explanation:

Variance_Wavelet → The variance of the wavelet (mathematical function) of the bill image.

Skewness_wavelet → The skewness (amount of asymmetry) of the wavelet of the bill image.

Curtosis_Wavelet → The Curtosis (statistical measure describing the distribution around the mean) of the wavelet of the bill image.

Image_entropy → Measure how much randomness or noise was in the bill image

Class → The output variable which specifies whether the bill was genuine (0) or not (1).

Random Forest - Python Example

Let's explore the dataset and see the correlation between the different features and the label classes.

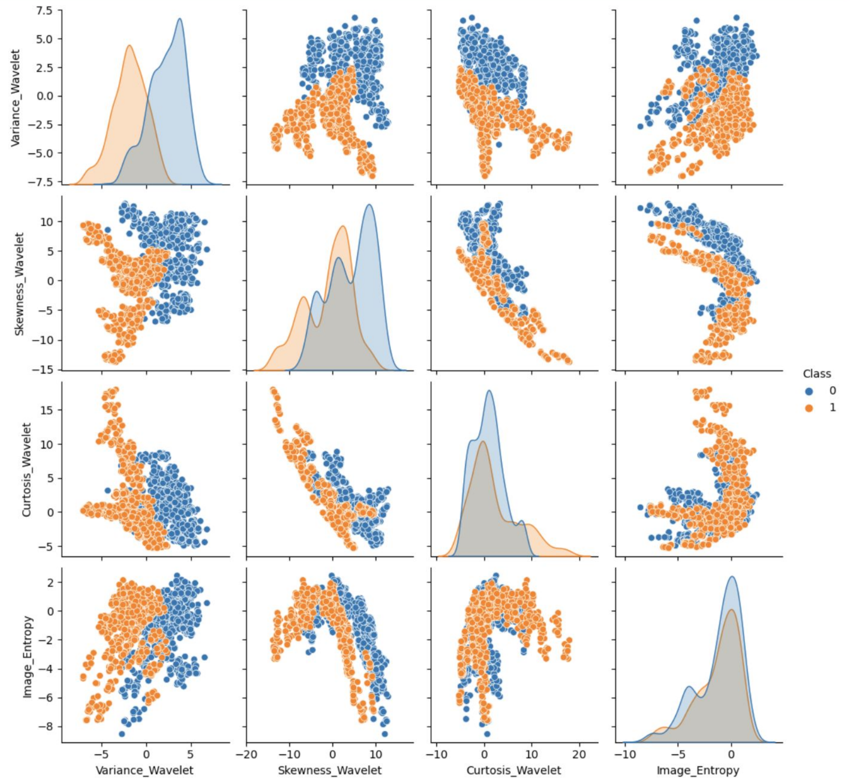
```
In [241]: ## data set that provide data about fake bill 0 - not fake 1 - fake
          df = pd.read_csv('/Users/ben.meir/Downloads/data_banknote_authentication.csv')
          df.head()
```

Out[241]:

	Variance_Wavelet	Skewness_Wavelet	Curtosis_Wavelet	Image_Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

```
In [226]: sns.pairplot(df, hue='Class')
          plt.show()
```

```
/Users/ben.meir/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/ben.meir/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/ben.meir/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/ben.meir/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



We can see that for some features there is no different between the classes and for some there is a big difference.

Random Forest - Python Example

Now, what we want is to use a Grid Search to find the best Random Forest parameters.

We will also examine if using bootstrapping will improve our model or not.

```
In [244]: df = pd.read_csv('/Users/ben.meir/Downloads/data_banknote_authentication.csv')
          X = df.drop('Class', axis=1)
          y = df['Class']
```

```
In [245]: from sklearn.model_selection import train_test_split
          from sklearn.model_selection import GridSearchCV
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.metrics import classification_report, confusion_matrix

          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=101)

          n_estimators = [64,100,128,200]
          max_features = [2,3,4]
          bootstrap = [True, False]
          oob_score = [True, False]

          param_grid = {'n_estimators': n_estimators,
                        'max_features': max_features,
                        'bootstrap': bootstrap,
                        'oob_score': oob_score}

          rfc = RandomForestClassifier()
          grid = GridSearchCV(rfc, param_grid)
```

Our Grid Search will check for the best n_estimators and max_features between the provided values.
In addition to rather to also enable bootstrap and oob_score

Random Forest - Python Example

Keep in mind that because oob_score can only be enabled when bootstrap parameter is enabled we will get some warnings and errors when the Grid Search will try to perform combinations of oob_score is True while bootstrap is False. The Grid Search will still execute successfully even with those errors.

```
In [243]: grid.fit(X_train, y_train)

/Users/ben.meir/anaconda3/lib/python3.10/site-packages/sklearn/model_selection/_validation.py:425: FitFailedWarnin
g:
60 fits failed out of a total of 240.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/ben.meir/anaconda3/lib/python3.10/site-packages/sklearn/model_selection/_validation.py", line 732, i
n _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/ben.meir/anaconda3/lib/python3.10/site-packages/sklearn/base.py", line 1151, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/Users/ben.meir/anaconda3/lib/python3.10/site-packages/sklearn/ensemble/_forest.py", line 417, in fit
    raise ValueError("Out of bag estimation only available if bootstrap=True")
ValueError: Out of bag estimation only available if bootstrap=True

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/Users/ben.meir/anaconda3/lib/python3.10/site-packages/sklearn/model_selection/_search.py:976: UserWarning: One or
more of the test scores are non-finite: [0.99571549 0.99228201 0.99228201 0.99399875 0.99314038 0.99399875
 0.99314038 0.99399875 0.99314038 0.99142365 0.99314038 0.99228201
 0.99399875 0.99314038 0.99399875 0.99399875 0.98541506 0.98884854
 0.98884854 0.98799017 0.98799017 0.98799017 0.98799017 0.98884854
        nan 0.99399142        nan 0.99484979        nan 0.99484979
        nan 0.99313305        nan 0.99056528        nan 0.99313672
        nan 0.99056528        nan 0.99141998        nan 0.97770441
        nan 0.97770441        nan 0.98112688        nan 0.97684604]
  warnings.warn(

Out[243]:
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

Here we got an error that we are trying to train a model with oob_score = True while bootstrap parameter is not enabled.

Random Forest - Python Example

We can also check for our Grid Search best parameters values:

```
In [248]: grid.best_params_
Out[248]: {'bootstrap': False, 'max_features': 2, 'n_estimators': 64, 'oob_score': False}
```

In case we also want to include the oob_score, we can generate a separate Random Forest model with the best params values and enable the oob_score as well:

```
In [252]: rfc = RandomForestClassifier(n_estimators=64, max_features=2, oob_score=True)
          rfc.fit(X_train, y_train)
Out[252]: RandomForestClassifier
          RandomForestClassifier(max_features=2, n_estimators=64, oob_score=True)
```

```
In [257]: rfc.oob_score_
Out[257]: 0.9939965694682675
```

Random Forest - Python Example

Finally, because this is a classification problem we can generate the classification report and confusion matrix to evaluate our model accuracy.

```
In [258]: from sklearn.metrics import classification_report, confusion_matrix

          predictions = rfc.predict(X_test)

          print(classification_report(predictions, y_test))
          print(confusion_matrix(predictions, y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       122
           1       1.00      0.98      0.99        84

    accuracy                           0.99       206
   macro avg       0.99      0.99      0.99       206
weighted avg       0.99      0.99      0.99       206

[[122   0]
 [  2  82]]
```

Our model has 99% accuracy and misclassified only 2 money bills out of 206 in total.

Class Exercise - Random Forest

Instructions:

Use the **‘data_banknote_authentication.csv’** file from our previous example with the same best parameters that we found and show that increasing the number of trees in the Random Forest model won’t improve the model accuracy.
Base your answer on a plot of your choice.

Hint: Use a for loop to calculate for each ‘n_estimations’ value it’s corresponding accuracy and create a plot according to your observations

Class Exercise Solution - Random Forest