Feature Importance Calculation

Feature Importance methods:

1) Gini Importance or Mean Decrease in Impurity
2) Permutation Importance or Mean Decrease in Accuracy

Gini Importance / Mean Decrease in Impurity (MDI)

- **Explanation1:** MDI counts the times a feature is used to split a node, weighted by the number of samples it splits. Gini Importance or Mean Decrease in Impurity (MDI) calculates each feature importance as **the sum over the number of splits** (across all tress) that include the feature, proportionally to the number of samples it splits.

- However, Gilles Louppe gave a different version: Instead of counting splits, the actual decrease in node impurity is summed and averaged across all trees. (Weighted by the number of samples it splits).

So, "Gini importance" or "mean decrease impurity" is defined as the **total decrease in node impurity** (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)) averaged over all trees of the ensemble.

- **Explanation2:** It is based on the decrease of Gini impurity when a variable is chosen to split a node. When a tree is built, the decision about which variable to split at each node uses a calculation of the Gini impurity. For each variable, the sum of the Gini decrease across every tree of the forest is accumulated every time that variable is chosen to split a node. The sum is divided by the number of trees in the forest to give an average.

- **Explanation3:** Random forest consists of a number of decision trees. Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set. The measure based on which the (locally) optimal condition is chosen is called impurity. For classification, it is typically either Gini impurity or information gain/entropy and for regression trees it is variance. Thus when training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this measure.

Permutation Importance or Mean Decrease in Accuracy (MDA)

- **Explanation1:** It is based on how much the accuracy decreases when the variable is excluded. It is based on experiments on out-of-bag (OOB) samples, via destroying the predictive power of a feature without changing its marginal distribution.
- Random forests also use the OOB samples to construct a different variable-importance measure, apparently to measure the prediction strength of each variable. When the **b**th tree is grown, the OOB samples are passed down the tree, and the prediction accuracy is recorded. Then the values for the **j**th variable are randomly permuted in the OOB samples, and the accuracy is again computed. The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable **j** in the random forest.
- **Explanation2:** Each tree has its own out-of-bag sample of data that was not used during construction. This sample is used to calculate importance of a specific variable. First, the prediction accuracy on the out-of-bag sample is measured. Then, the values of the variable in the out-of-bag-sample are randomly shuffled, keeping all other variables the same. Finally, the decrease in prediction accuracy on the shuffled data is measured.

- The mean decrease in accuracy across all trees is reported. Intuitively, the random shuffling means that, on average, the shuffled variable has no predictive power. This importance is a measure of by how much removing a variable decreases accuracy, and vice versa — by how much including a variable increases accuracy.

Note that if a variable has very little predictive power, shuffling may lead to a slight increase in accuracy due to random noise. This in turn can give rise to small negative importance scores, which can be essentially regarded as equivalent to zero importance.

- **Explanation3:** This approach directly measures feature importance by observing how random re-shuffling (thus preserving the distribution of the variable) of each predictor influences model performance.

The approach can be described in the following steps:

1. Train the baseline model and record the score (accuracy/R^2/any metric of importance) by passing validation set (or OOB set in case of Random Forest). This can also be done on the training set, at the cost of sacrificing information about generalisation.
2. Re-shuffle (Interchange the positions/Put in a new order; rearrange.) values from one feature in the selected dataset, pass the dataset to the model again to obtain predictions and calculate the metric for this modified dataset. The feature importance is the difference between the benchmark score and the one from the modified (permuted) dataset.

3. Repeat 2 for all feature in the dataset.

- **Explanation4:** Another popular feature selection method is to directly measure the impact of each feature on accuracy of the model. The general idea is to permute the values of each feature and measure how much the permutation decreases the accuracy of the model. Clearly, for unimportant variables, the permutation should have little to no effect on model accuracy, while permuting important variables should significantly decrease it.

Importance for numeric outcomes

For a numeric outcome, there are two similar measures:

- Percentage increase in mean square error is analogous to accuracy-based importance, and is calculated by shuffling the values of the out-of-bag samples.
- Increase in node purity is analogous to Gini-based importance, and is calculated based on the reduction in sum of squared errors whenever a variable is chosen to split.

Node impurity / Impurity Criterion

For classification, they both use Gini impurity by default but offer Entropy as an alternative. For regression, both calculate variance reduction using Mean Square Error.

Impurity	Task	Formula	Description
Gini impurity	Classification	$\sum_{i=1}^{C} -f_i(1-f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Entropy	Classification	$\sum_{i=1}^{C} -f_i \log(f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Variance / Mean Square Error (MSE)	Regression	$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$
Variance / Mean Absolute Error (MAE) (Scikit-learn only)	Regression	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i-\mu\rvert$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$

Information Gain

Another term worth noting is “Information Gain” which is used with splitting the data using entropy. It is calculated as the decrease in entropy after the dataset is split on an attribute:

Gain(T,X) = Entropy(T) — Entropy(T,X)

- T = target variable
- X = Feature to be split on
- Entropy(T,X) = The entropy calculated after the data is split on feature X

Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature.

The Variables importance has 2 measures:

1. %IncMSE - It is computed from permuting test data: For each tree, the prediction error on test is recorded (Mean Squared Error - MSE). Then the same is done after permuting each predictor variable. The difference between the two are then averaged over all trees, and normalized by the standard deviation of the differences. If the standard deviation of the differences is equal to 0 for a variable, the division is not done (but the average is almost always equal to 0 in that case). Higher the difference is, more important the variable. MSE = mean((actual_y - predicted_y)^2)

2. IncNodePurity - Total decrease in node impurities from splitting on the variable, averaged over all trees. Impurity is measured by residual sum of squares. Impurity is calculated only at node at which that variable is used for that split. Impurity before that node, and impurity after the split has occurred.

- You compute variable importance by computing out-of-bag error in the normal way. Then, you randomly mix the values of one feature across all the test set examples -- basically scrambling the values so that they should be no more meaningful than random values (although retaining the distribution of the values since it's just a permutation). Then you compute out-of-bag error again. If error increases a lot, that feature was important; the real values in the right place lead to better predictions than if they were meaningless. The change is the measure of feature importance.

- Scikit-learn does something simpler and more efficient, although I'm not aware of the theoretical basis for it. It certainly has intuitive appeal. It pushes the entire data set down the trees and counts the number of times a data set passes through a node whose decision is based on a given feature. Features that appear often and high up the tree end up with high counts. And that suggests the features were frequently chosen as providing the best split. At the least, it tells you that, empirically, the trees you generated consult that feature frequently for better or worse.

- Two most popular ways to measure variable importance with random forests. Let me try to recap those two and add a few more. **First**, node impurity is the simplest method. It is calculated by looking at the total decrease in node impurities from splitting the variables averaged over all the trees, so for regression it is measured by the residual sum of squares (Gini for a classification). However, this method is biased towards preferring variables with more categories. Second, if there are correlated variables, node impurity will favor one over the other.

- **Second**, is the permutation method, which Sean described so well. For each tree, the error is on the OOB portion of the data is recorded (error rate for classification, MSE for regression). Then the variables are permuted (so keeping the original distribution). The difference between the two are then averaged over all the trees and normalized. The randomization is akin to setting a coefficient to zero in a linear model. So, for unimportant variables, the permutation should have little to no effect on model accuracy, while permuting important variables should significantly decrease it. This method is widely recognized as an improvement over node purity and widely implemented (R/Python).

AUC ROC Curve

In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two diagnostic groups (diseased/normal).

Sensitivity and specificity versus criterion value

When you select a higher criterion value, the false positive fraction will decrease with increased specificity but on the other hand the true positive fraction and sensitivity will decrease:

When you select a lower threshold value, then the true positive fraction and sensitivity will increase. On the other hand the false positive fraction will also increase, and therefore the true negative fraction and specificity will decrease.

A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test.

An ROC curve demonstrates several things:

1. It shows the trade-off between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
2. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
3. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.