Exam Project in Machine Learning

The chosen dataset

For this project I have chosen the dataset called Wine Quality, which contains two datasets for red and white vinho verde wine, respectively. Data can be found here:
http://archive.ics.uci.edu/ml/datasets/Wine+Quality

Data of the red wines (reds) and white wines (whites) are stored in the same dataframe. The wine quality is specified using a score between 0-10.

Description of the problem

The goal is to model the wine quality based on physicochemical tests. Since quality is a categorical variable, I approach this as a classification task. The data is imbalanced, meaning that the number of quality scores varies greatly depending on which quality score we are considering. We must account for this when fitting our models.

Chosen model architecture

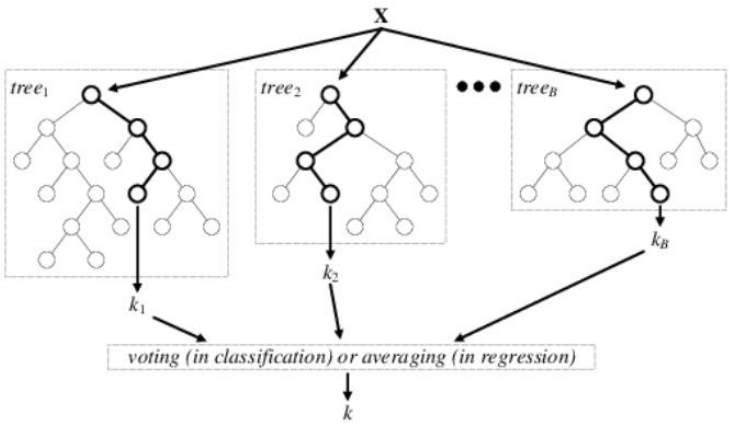
I attempt to model wine quality using the two models

- Random Forest
- Gradient Boosting.

Model architecture – Random Forest Classification

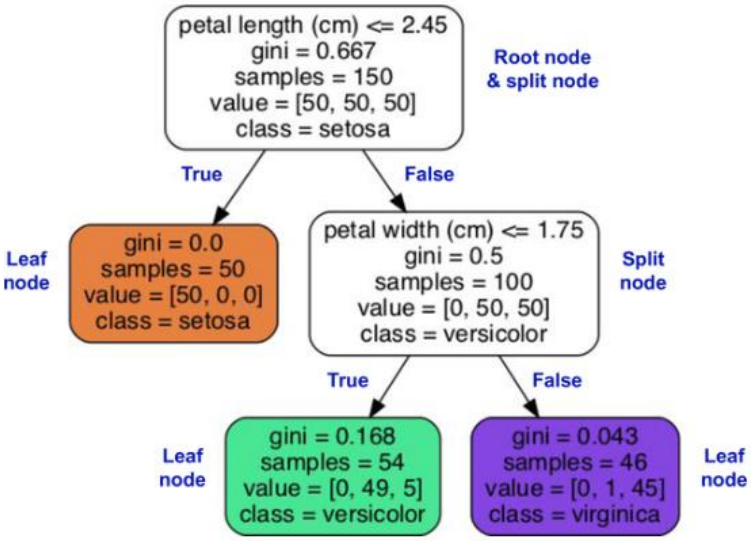
This is an ensemble method consisting of multiple decision trees. When used for classification the output of the model corresponds to the majority vote of all the decision trees. The vote of each tree, i.e. the optimization of the fitted tree is done with respect to an inputted score which determines what is emphasized in the optimization.

Per default in the model, then in each tree at each node only a subset of features is considered (corresponding to the square root of the number of features) and used for determining how to split the node. Again, per default, the model selects the feature and threshold that minimizes the Gini impurity at the node. A node is considered pure (Gini=0) if all observations from the training data located in this node all belong to the same class. As example see figure below.



Starting from the root node the optimization of Gini impurity does not take into account how this will affect Gini Impurity in any other nodes further down the decision tree.

Below figure shows a decision tree fitted to a dataset called Iris. It has 4 features and a categorical target feature equating either setosa, versicolor or virginica. The tree has a max depth of 2, meaning only 2 splits of the training data. The tree is optimized as described above using a random selection of features at each node and optimizing Gini impurity. In each split note we see an if-else condition. Considering an instance in the Iris data, if the condition is true/false we move 1 level down to the left/right. This procedure is repeated at all split nodes. Leaf nodes cannot be split. In each node the Gini impurity of the node is shown, the number of samples show the number of training instances allocated to the node, and value corresponds to the number from the samples allocated to each of the classifications of the target variable. Diving value by samples we get the probabilities of each classification within the note, e.g. at the bottom right leaf node 45/46 is the probability of a training instance belonging to the class virginica. The Gini impurity at node i is calculated using these probabilities: $G_i = 1 - \sum_{k=0}^{n} p_{i,k}^2$



where n=3 is the number of classes, and $p_{i,k}$ is the probability of class k at node i given the training instances at node i.

Model architecture – Gradient Boosting Classification

This is also an ensemble method. In our case, it combines predictions of multiple weak decision trees, called 'tree stumps'. It starts by fitting one decision tree to the training data. Then when fitting the next decision tree, it does so by fitting a decision tree onto the previous tree's residual errors, the next decision tree is fitted only these residual errors, etc. When fitting a decision tree on residual errors it tries to 'get the next' residual errors closer to zero. In other words, when a decision tree is optimized, it calculates a loss function which is a function of the residual errors. The gradient of the loss function is used to 'guide' the model in the direction that minimizes the residual errors. The speed at which the model learns depends on the gradient and a learning rate. Similarly to a random forest, the optimization takes a score as input which determines what is emphasized in the optimization.

Link to public repository

https://github.com/leaand01/MachineLearningEksamensProjekt