INSA ROUEN NORMANDIE

Informatique et Technologie de l’Information 4
Année universitaire 2024-2025

Practical Work

Random Forest

simon.bernard@univ-rouen.fr

1 Spambase classification with Random Forest

In this practical session, we will use the scikit-learn's `RandomForestClassifier` implementation to train and test random forests. This implementation corresponds to the reference method presented in the course, which uses bagging and random feature selection.

1. Let's start by training a Random Forest classifier on the Spambase dataset you used in the previous practice sessions :

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import numpy as np

# Load the Spambase dataset
dataspam = np.loadtxt("spambase.data", delimiter=",")
...

# Split the dataset into training and testing sets
# (e.g. with 80% of the data for training and 20% for testing)
...

# Train a Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

   Fill in and test this python script.
2. Propose a small experiment to compare the performance of a random forest classifier to the performance of a SVM classifier on this dataset.
3. You have seen that the SVM learning method needs to be fed with normalized data. What about the Random Forest method? How does the performance of random forests with and without normalisation compare, and can you explain this result?

2 Tuning the `max_features` hyperparameter

The `max_features` hyperparameter of the Random Forest method is crucial for the performance of the classifier. This hyperparameter controls the number of features that are randomly selected to split each node of the decision trees. The default value in the Scikit-learn implementation is $\sqrt{d}$, where d is the number of features in the dataset, but one can set it to a fixed number or a fraction of the total number of features.

To understand the impact of this hyperparameter, let's consider the digits dataset from Scikit-learn and train several Random Forest classifiers with different values of `max_features` :

1. Load the digits dataset and analyse the nature of this data. In particular, what do the features of this dataset correspond to?

```python
X, y = load_digits(return_X_y=True)
n_instances, n_features = X.shape
```

2. Intuitively, would you say that the features of this dataset are mostly relevant?
3. Train several Random Forest classifiers with different values of `max_features` and compare their performance. For example, you can use the following code snippet :

```python
n_replications = 10
list_features = range(1, n_features+1, 2)
for k in list_features:
   clf = RandomForestClassifier(...)
   for i in range(n_replications):
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
      ...
```

4. Modify the previous code to measure the average performance (with standard deviations) of each classifier over the replications.
5. Plot these averaged performances with respect to the values of `max_features`. What can you conclude from this experiment?

3 Measuring the importance of features

This last experiments aims at measuring the importance of the features in the Random Forest classifier. The importance of a feature is computed as the average decrease in impurity (e.g. Gini impurity) that the feature causes in the decision trees of the forest. This information can be used to understand the relevance of the features and to perform feature selection.

1. Let's consider a synthetic dataset for which the number of informative feature is controlled :

   ```
   # Build a classification task using 5 informative features
   X, y = make_classification(n_samples=1000, n_features=20, n_informative=5,
       n_redundant=0)
   ```

   This dataset comprise 20 features amongst which 5 are informative.
2. Train a random forest classifier on the entire dataset, with 200 decision trees.
3. The resulting forest embed a `feature_importances_` attribute that directly gives a feature importance measure. This is also the case of each individual tree in the forest.

   ```
   importances = forest.feature_importances_
   tree_importances = [tree.feature_importances_ for tree in forest.estimators_]
   ```
4. Rank the features of the dataset from the most informative to the least informative.
5. Plot the feature importances in a bar plot, with the standard deviation calculated on the importances of the individual trees.
6. Analyze and comment the results.