Run Isolation Forest

Objective

- Now we are moving to a more advanced family of methods: ensembles of trees. In practice, these are some of the best algorithms that we have at our disposal for any kind of machine learning problem. The benefit of ensembles of trees is that they can deal with many noisy features while also providing very good performance.
- The Isolation Forest is a novelty-detection technique that simply extends the principles of random forest (the most popular ensembling technique for trees) to anomaly detection. Isolation forests are decision trees, so the main parameter is the total number of trees. Your goal is to see whether the performance of this algorithm on the task of predicting thyroid disease is better than other families of algorithms.

Importance to project

- This will be the first ensemble method for novelty detection.

Workflow

1. Instantiate three different IsolationForest objects by calling `sklearn.ensemble.IsolationForest` and setting the `n_estimators` parameters to 50, 100, and 200 trees, respectively.
   - **Note**: The number of trees in an isolation forest is a kind of complexity parameter. More trees make for a more powerful model, but at the cost of a slower runtime. Hence finding a good performance with fewer trees is the goal.

2. Similar to the previous milestone, re-initialize the metrics DataFrame to hold the classification metrics of interest (Precision, Recall, F1 Score, Average Precision, TN, TP, FN, FP).

3. Fit each of the previously instantiated models from Step 1 on *only* the inliers training data because isolation forest tries to build a model from only the majority samples. Fitting using the entire training set reduces its performance.
4. Evaluate the fitted models on the test features using the `custom_classification_metrics_function` used in Milestone 2 and store the classification metrics in the DataFrame you initialized previously.
5. Inspect the results: what is the effect of changing the `n_estimator` parameter on the F1 score, Precision, Recall, and Average Precision?
6. Copy the metrics to the cumulative metrics DataFrame, which can be used to compare the performance of the algorithms across different milestones.

Author insights

- There are several advantages to using an isolation forest, including:

  1. linear time complexity with a low constant and a low memory requirement
  2. the ability to handle high-dimensional problems that have a large number of irrelevant attributes, and
  3. good performance in situations where the training set does not contain any anomalies

Deliverable

The deliverable for this milestone is a Jupyter Notebook with a report on your results. Create this code as an extension to the solution from the previous milestone. Do isolation forests outperform the other methods? What does this tell us? If a more complicated method can’t outperform simpler ones, then the added complexity of this method is not really justified. Is this the case here?

Upload a link to your deliverable in the Submit Your Work section and click submit. After submitting, the author’s solution and peer solutions will appear on the page for you to examine.

Help

Feeling stuck? Use as little or as much help as you need to reach the solution!

resources

Ensemble Methods for Machine Learning by Gautam Kunapuli

Chapter 2, "Homogeneous Parallel Ensembles: Bagging and Random Forests", explains random forests which is an algorithm very similar to isolation forest.

Additional resources

- Isolation forests
- Scikit-learn example of using Isolation Forest
- Original Research paper on Isolation Forest
- Blog post describing the Isolation Forest algorithm