Run the LOF Algorithm

Objective

- We conclude this project with another interesting algorithm. When investing a machine learning problem it important to approach it from multiple angles. So far, we have tried linear classifiers (SVMs), probabilistic approaches (elliptic envelope), and ensembles (isolation forest). We will conclude by using a nearest-neighbors approach, which is based upon principles that are popular in unsupervised learning: the local outlier factor (LOF).
- LOF is a very powerful anomaly detection algorithm, which is based on the same logic as the nearest-neighbors algorithm. In this milestone you will have the chance to experiment with it.
- This algorithm works based on a nearest-neighbor approach. In our case study, the algorithm aggregates groups of patients into clusters and tries to figure out whether some patients stand isolated on their own. These patients constitute anomalies in the data and require special attention. When we use a large number of neighbors, the algorithm will identify more patients as anomalies than if we used a smaller number. Using a small number of neighbors, then, makes the algorithm “stricter,” as it will be less predisposed to describe points as anomalies.
- This can also affect the risk of using such an approach. If a hospital has a lot of resources, then it is okay for an anomaly-detection model to trigger many warnings even if many of them are false positives. If a hospital does not have enough resources, then we might want to make it more difficult for the algorithm to trigger warnings.

Importance to project

- LOF is the last algorithm we will use to compare against the other options. In any machine learning problem, we have to compare and contrast multiple approaches to define the best choice. The LOF algorithm is complementary to the other approaches we’ve followed so far, and it will help us converge on the best approach.

Workflow

1. Instantiate four different local outlier factor objects by calling `sklearn.neighbors.LocalOutlierFactor` and setting `n_neighbors` to 3, 10, 20, and 50.
   - The number of neighbors has an effect on performance. The documentation recommends using `n_neighbors=20`. In practice, we'll try a mix of small and large values until we figure out the best range of parameters.
   - Make sure novelty is set to True so that we can use the predict, decision_function, and score_samples methods on new unseen data. Otherwise, you will be forced to use the `fit_predict()` method, in which case applying LOF will no longer be novelty detection.
2. Similarly 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. Fitting using the entire training set reduces 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_estimators` parameter on the F1-score, Precision, Recall, and Average Precision? Do your observations agree with the documentation that `n_neighbors=20` provides the best performance?
6. Copy the metrics to the cumulative metrics DataFrame, which can be used to compare performance of the algorithms across different milestones.

Notes

- LOF compares the local density of the sample with densities of its neighbors (as defined by the n_neighbors param when instantiating the LOF). Outliers are those that have substantially lower local density compared to their neighbors.
- The score gives an indication of how isolated the given sample is when compared

to the neighborhood.

Deliverable

The deliverable for this milestone is a Jupyter Notebook with a report on your results. Use the `custom_classification_metrics_function` to get the evaluation metrics.

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

Additional resources

- Local outlier factor by scikit-learn
- Outlier detection with Local Outlier Factor (LOF)
- Novelty Detection using LOF