Run Robust Covariance

Objective

- The elliptic envelope method is a simple outlier-detection method. This method can work well on simple problems, and it is very easy to implement. Therefore, it makes sense to also try and see what performance we can get using it on the thyroid disease dataset. In the domain of medicine, a small improvement in accuracy can make a huge difference, given that we are talking about human lives. Therefore, we want to be absolutely certain that we are close to the best performance we can get.
- The elliptic envelope method assumes that the data points follow a Gaussian distribution. In practice, while this assumption might not be true, there are many use cases where it applies. Since no dataset will have a perfectly Gaussian distribution, we will just have to try out the elliptic envelope technique and see what the evaluation metrics (F1 score, Precision/Recall, Average Precision) turn out to be.
- Your goal is to see whether the performance of this algorithm on the task of predicting thyroid problems is better than other families of algorithms by tuning the contamination parameter, which tells the algorithm how many outliers to expect.

Importance to project

- In this project we will compare different approaches for anomaly detection. Robust covariance is one of them.

Workflow

1. Instantiate three different Elliptic Envelope objects with the contamination parameter set to 1, 2, and 4x the outliers fraction quantified in the previous milestone. Refer to the documentation given here to find out how to change the *contamination* parameter.
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 model from step 1 on the train data
4. Evaluate the fitted models on the test features using the function `custom_classification_metrics_function` used in the previous Milestone and store the classification metrics in the DataFrame initialized previously
5. Inspect the results.
    - What do you observe based on the F1-score, Precision, Recall and Average Precision when `outlier_fraction` changes?
6. Copy the metrics to another DataFrame, which can be used to compare performance of these across different milestones.