1 Executive Summary

This report analyzes the data collected in the data challenge files for signal being ‘Noisy’ or ‘non-Noisy’. We find that a Logistic Regression classifier is able to obtain 0.87 AUC with 8 features. Tuning the classifier’s decision threshold may be used to operate in different regimes, in this example we have chosen to operate at maximum balanced accuracy by using a decision threshold of 0.46.

1. Data sources: CSV files given as part of challenge, with labeled time series data.

2 Data exploration

To begin, the data exploration component consists largely of visual inspection of the datasets time series results to generate ideas for feature generation. One finding here to flag is that we find that the labeling appears visually to be not very different between the positive and negative labels. That is, time series information is often qualitatively the same between the two channels when viewed as a normalized time series. In addition, the nature of the positive labels appears to be different from that demonstrated in the initial problem statement, see 1. In Figure 2 we see a visual comparison of the two classes, after min-max normalization. The vertical offset is simply to better visualize independent charts.

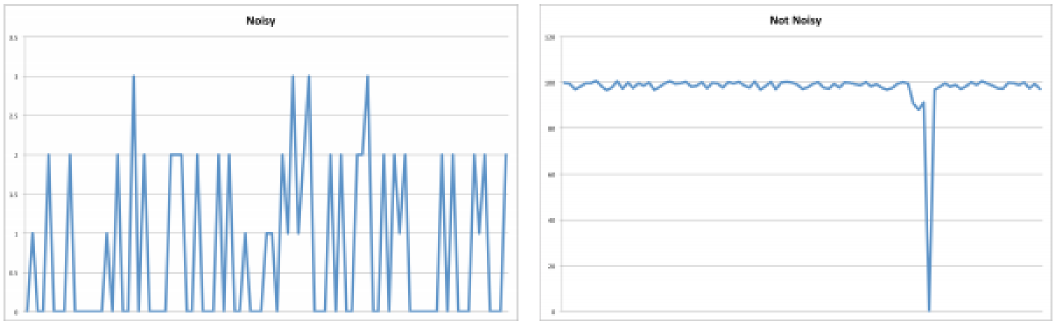


Figure 1: Description of noisy and non-noisy samples in the explainer document.

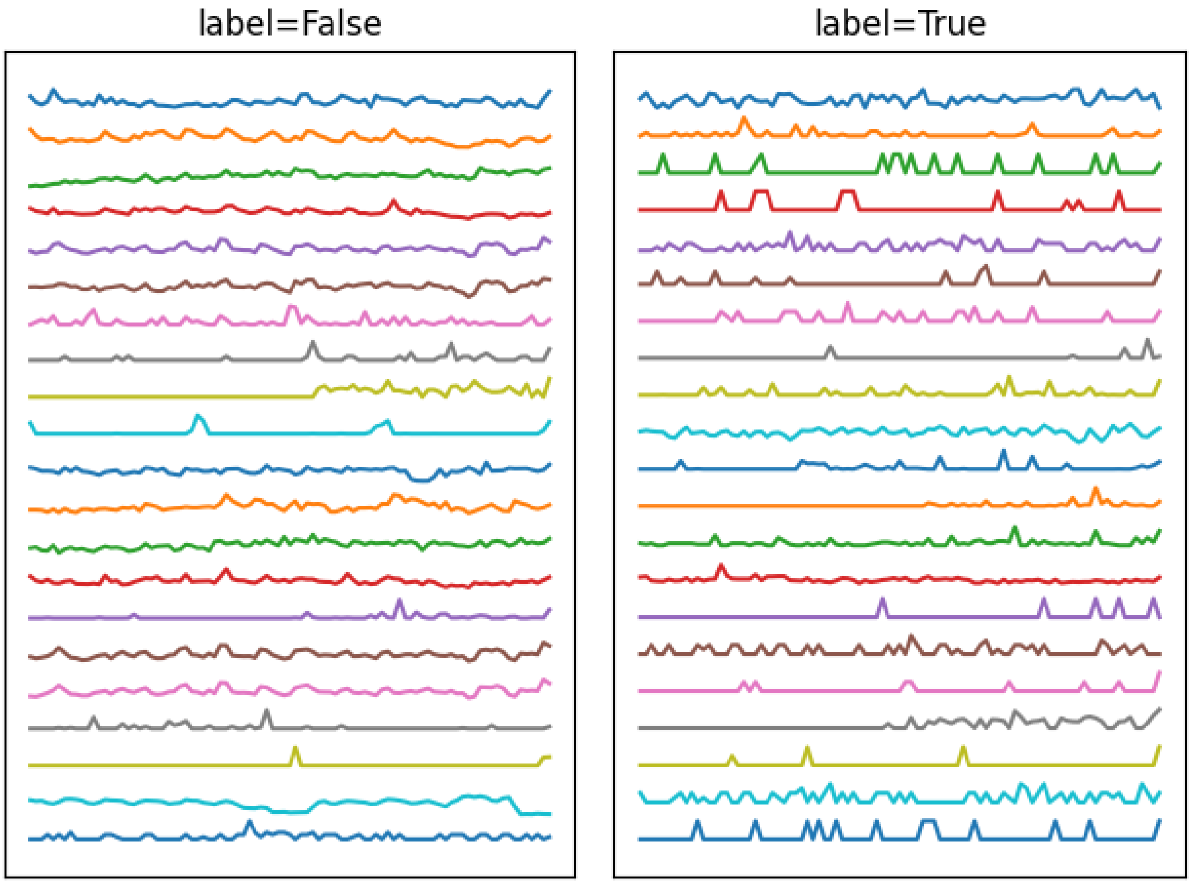


Figure 2: Raw time series plots from the two classes. The data is range normalized for better visual comparison. We see many qualitative similarities between the two labels.

3 Features

For features, based on the time series in Figures 1 and 2, we consider the following major categories:

1. FFT based features - decomposes time series signal into discrete frequency components. This transformation is often sensitive to certain types of time series information such as spikes, different periodicity
2. Regression based features - fits a simple linear line to the data, either standard Linear Regression or regressions less sensitive to outliers such as the Theil-Sen regressor. This transformation is expected to be useful for signals with a bias, or fluctuations which are biased to positive or negative trend, or extreme fluctuations in baseline behavior (spikes, etc).

These basic remappings of the data are then collapsed further by a variety of methods appropriate for statistical distributions such as basic mean, median, range, quartile concentrations, and standard deviation, skew, and kurtosis. A large number of features are generated programmatically by the code in `exploration.py`, with the generated features recorded appropriately in the column name of the resulting dataframe.

3.1 Feature selection

By producing combinations of features, such as algebraic combinations of features, more features are generated than we have data for. Therefore, feature selection will be an important aspect of the project. All things being equal, we prefer to use fewer features in the final model. In the final findings, we present a combination of individual and product features, then use Sequential Backward Selection to map out the behavior of the model to the test data. This is done by executing the file `ml_sbs.py`. We use the Logistic Regression Classifier in the SBS algorithm. The final performance point is shown in the right hand side of figure 3. The resulting features list is the following, which gives an average ROC AUC score of 0.87.

1. Signal concentration in the upper tail > 0.25, RANSAC Regressor Residual
2. Signal concentration in the upper tail > 0.01, Linear Regressor Residual
3. Ratio(Signal concentration in frequency weighted FFT < 0.05, Signal concentration in frequency weighted FFT < 0.2)
4. Ratio(Signal concentration in lower tail < 0.05, Theil-Sen Regressor Residual, Signal concentration in lower tail < 0.25 Theil-Sen Regressor Residual)
5. Ratio(Signal concentration in upper tail > 0.1, Theil-Sen Regressor Residual, Signal concentration in lower tail < 0.01, Linear Regressor Residual)

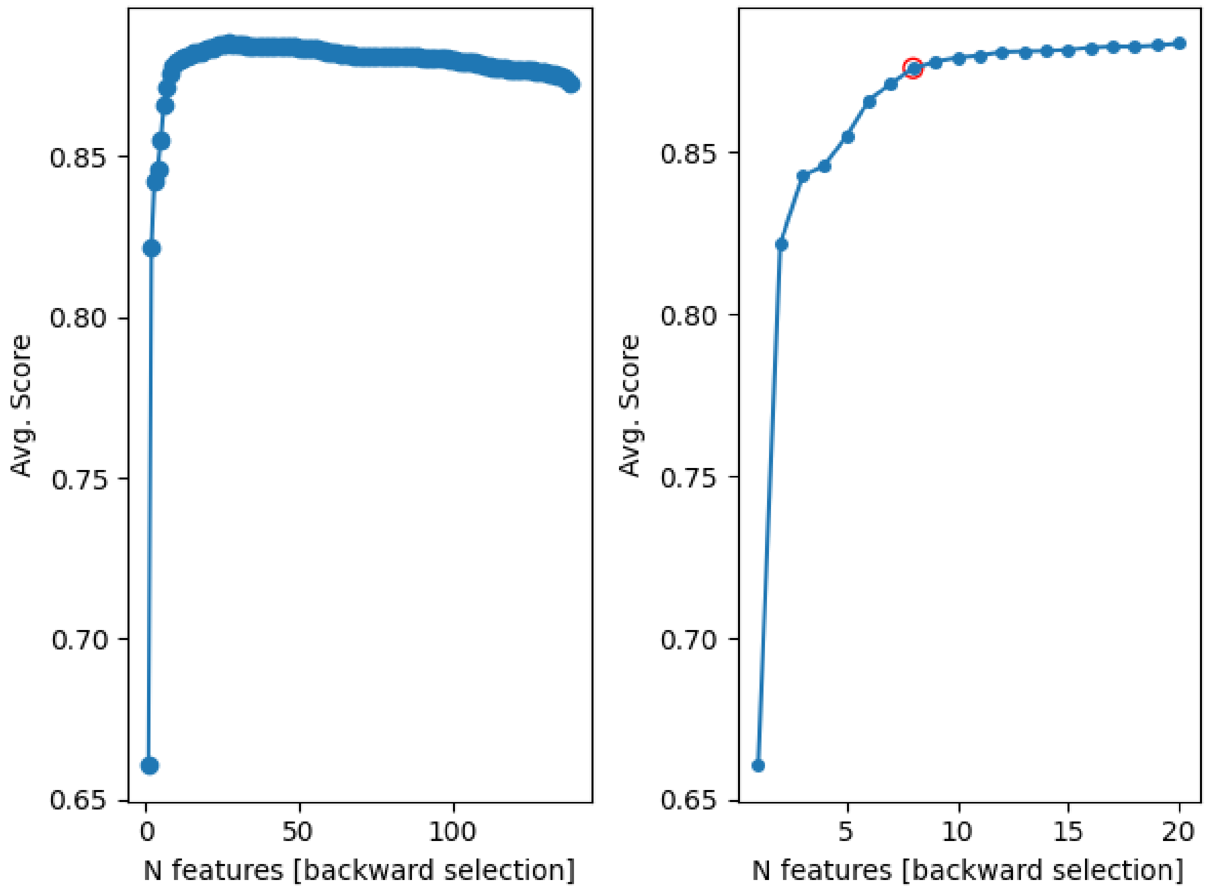


Figure 3: Sequential Backward Selection curves from the large list (132 features) down to 1 feature. We pick the point with 8 features as the point to operate the classifier.

6. Ratio(Signal concentration in lower tail < 0.01, Linear Regressor Residual, Signal concentration in upper tail > 0.01, Theil-Sen Regressor Residual)

7. Ratio(Signal concentration in lower tail < 0.2, Weighted FFT, Signal concentration in lower tail < 0.05, Theil-Sen Regressor residual)

8. Ratio(Signal concentration in lower tail < 0.2, Weighted FFT, Signal concentration in lower tail < 0.01, Linear Regressor residual)

Comment When the backward selection algorithm is taken all the way to one feature, the following single feature has a cross validation scores (ROC-AUC) of 0.66.

1. Ratio(Signal concentration in lower tail < 0.2, Weighted FFT, Signal concentration in lower tail < 0.05, Theil-Sen Regressor residual)

4 Model performance

In this section we look at final tuning and detailed performance of the Logistic Regression classifier used in the SBS feature selection algorithm. Using the raw feature crunching from `exploration.py` $\rightarrow$ `features.csv` and the feature list from the SBS algorithm `ml_sbs.py` $\rightarrow$ `final_columns.csv`, we execute the final training and evaluate performance on the test set.

The learning curve is presented in Figure 4. We see a continued improvement in the classifier as the number of training samples increases. However, the presence of a gap between the training and validation curves suggests that some hyperparameter tuning via regularization may be beneficial.

The out of the box performance in terms of confusion matrix is shown in Figure 5 at decision threshold of 0.5. The model performance on the test set is show in Figure 6a, with variation in the decision threshold from 0 to 1. Using this, we select the best model based on the balanced accuracy metric, though this could be adjusted in practice based on the application and preference for false positives vs false negatives. The resulting ROC curve is shown in Figure 6b, again with decision point highlighted.

Finally, we look at the confusion matrix of this classifier on the test set, compared to a random classifier (weighted by the labelled class distribution) in Figures 7a and 7b. We see a modest improvement over the random classifier, doubling the number of positive samples correctly detected. Further improvements may be obtained by looking at other classifiers, a wider range of features (we considered a subset of possible features here due to computational requirements), and tuning hyperparameters such as the regularization strength.

5 Conclusions

In this report we have taken the data challenge time series data set through a basic machine learning pipeline. We have produced infrastructure for generating a large number of features based on the time series, which have shown to give modest improvement over a random classifier. Further efforts would be spent on (1) understanding the nature of the features used in the model in order to hypothesize more features (2) further efforts to identify improved classifiers and (3) hyperparameter tuning. Finally, the business context of the classifier performance needs to be established to be able to better suggest a model to deploy to production (e.g. in order to maximize business value).

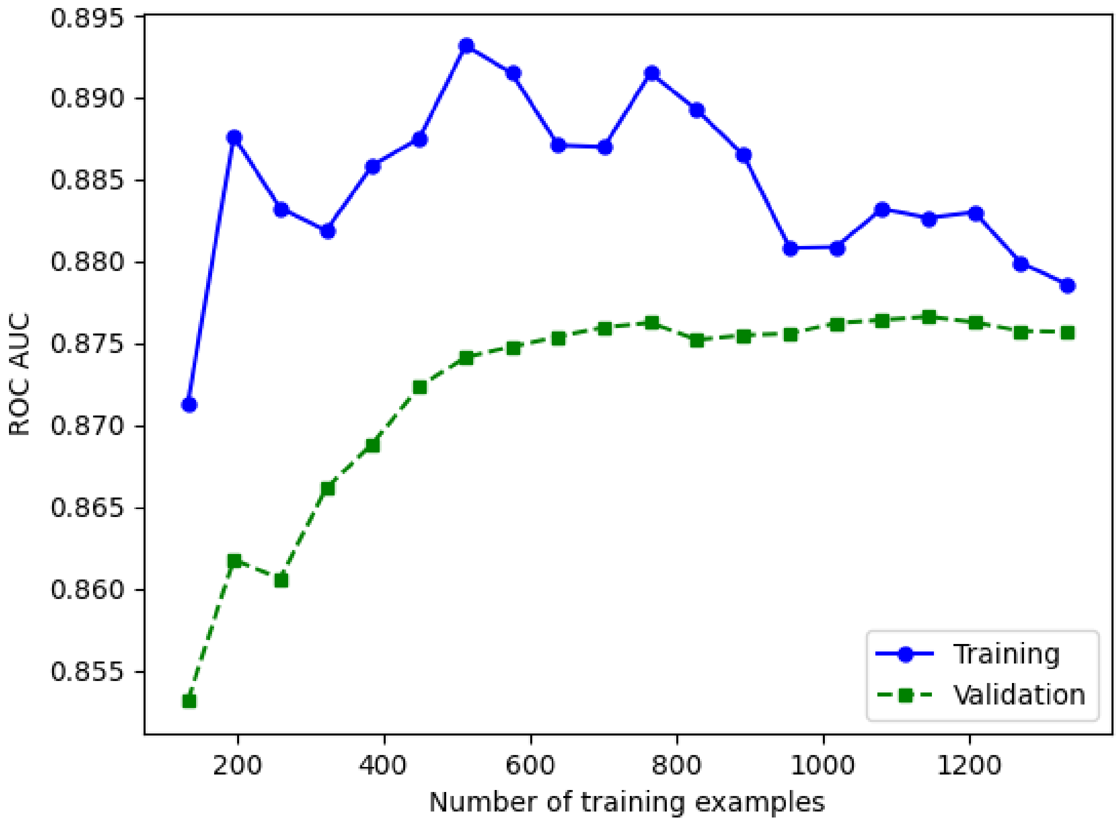


Figure 4: Learning curve for Logisic Regression Classifier

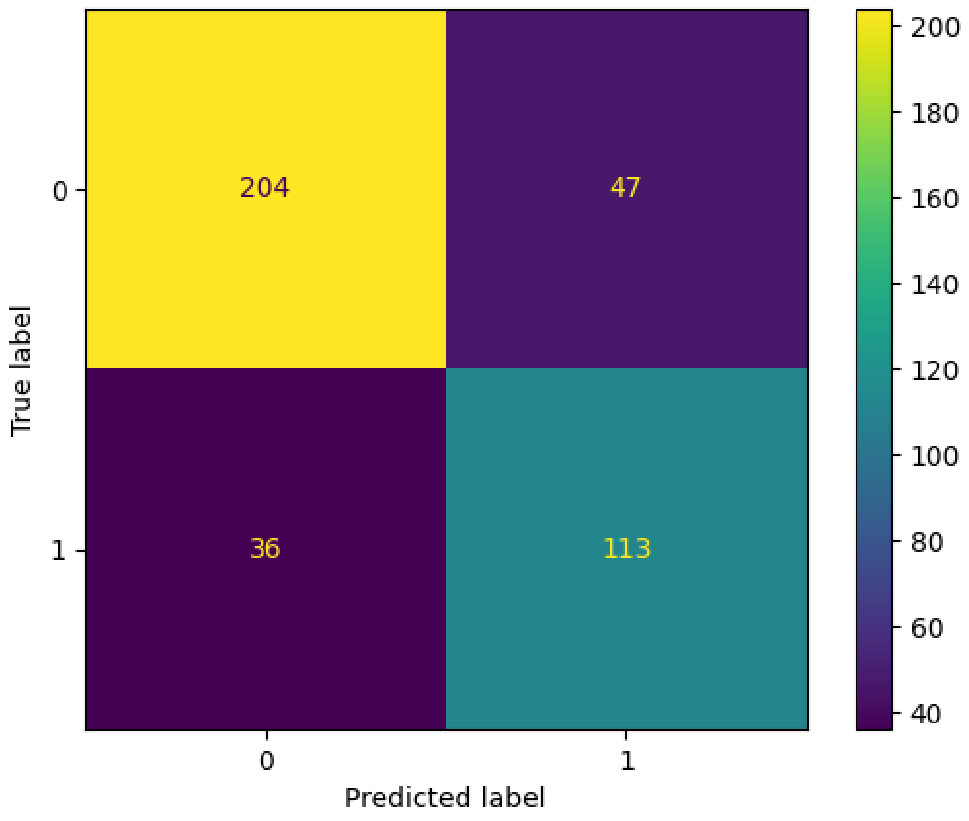
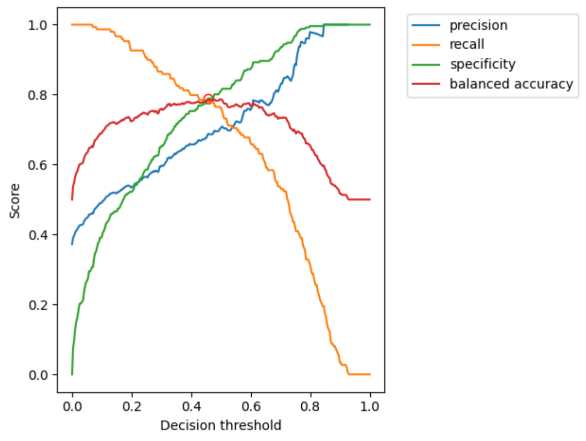
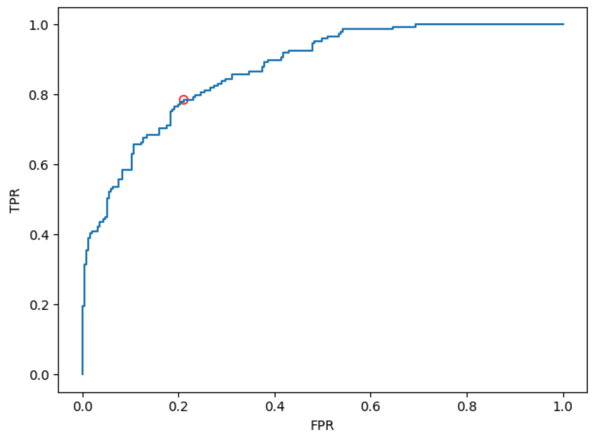


Figure 5: Confusion matrix for out of the box classifier

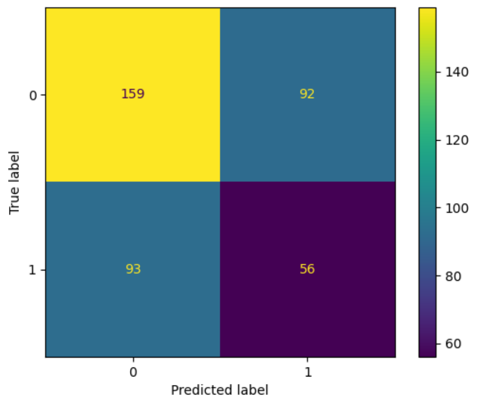


(a) Performance curves for Logisic Regression Classifier and the operating point at decision threshold 0.46 is highlighted.

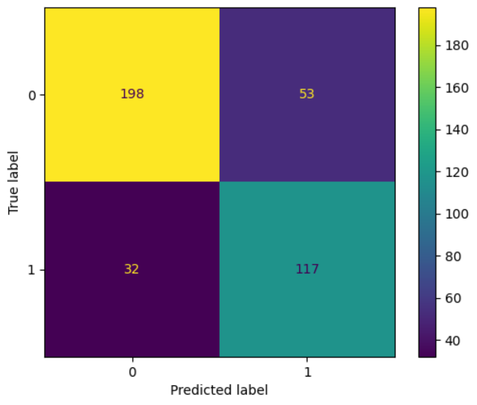


(b) ROC curve for Logistic Regression Classifier, and the operating point at decision threshold 0.46 is highlighted.

Figure 6: Performance comparison charts vs decision threshold. The operating point is taken at the point of maximum balanced accuracy



(a) Random classifier (weighed by class proportion)



(b) Logistic regression classifier operating at decision threshold 0.46

Figure 7: ROC comparison