
Anomaly Detection models for financial market data
Giulia Bergonzoli, Michele Cosi, Carl Jonas Gustafsson,
Jon Anders Conrad Jonsson, Martina Del Basso

TABLE OF CONTENTS

01

DATA STAZIONARIZATION

Making historical data stable over time

```
dataset.loc[1:, Indices] = np.diff(np.log(dataset[Indices]), axis=0)
dataset.loc[1:, InterestRates] = np.diff(dataset[InterestRates], axis=0)
```

Dealing with historical data to infer something about the structure of systemic risk

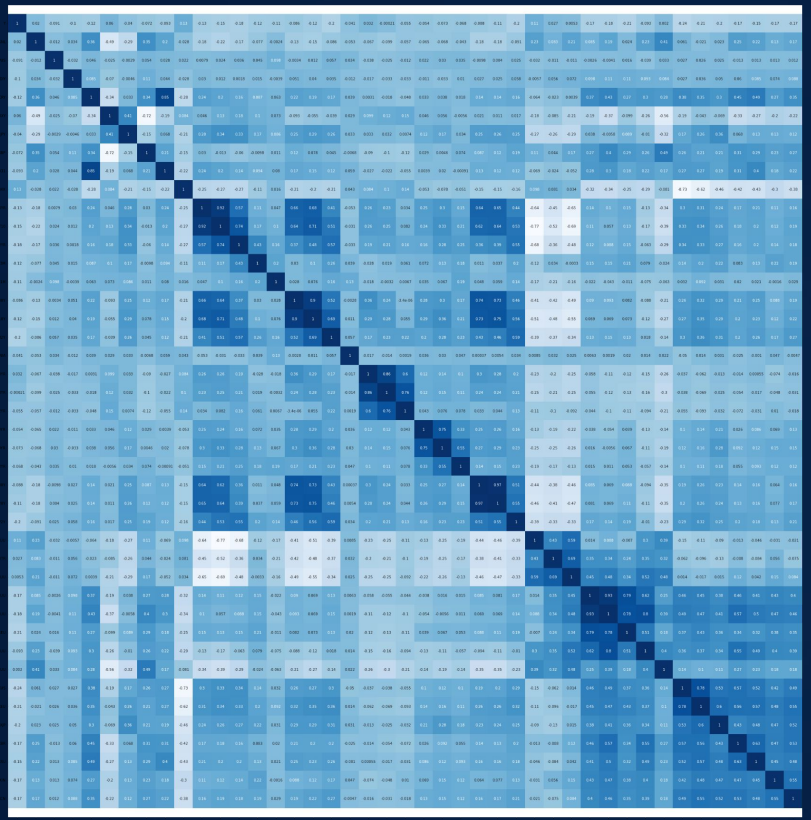
need for stationary data (the joint distribution of the relevant variables must be stable over time) in order to obtain accurate and reliable inspections

differentiating the features

02

DATA CLEANING

Correlation analysis



COLLINEARITY

We noticed that just 4 variables were with a correlation greater than 90%, so we have decided to keep them all

- These are the one mentioned:

```
{'GT10', 'LG30TRUU', 'GTDEM10Y', 'GTGBP20Y'}
```

03

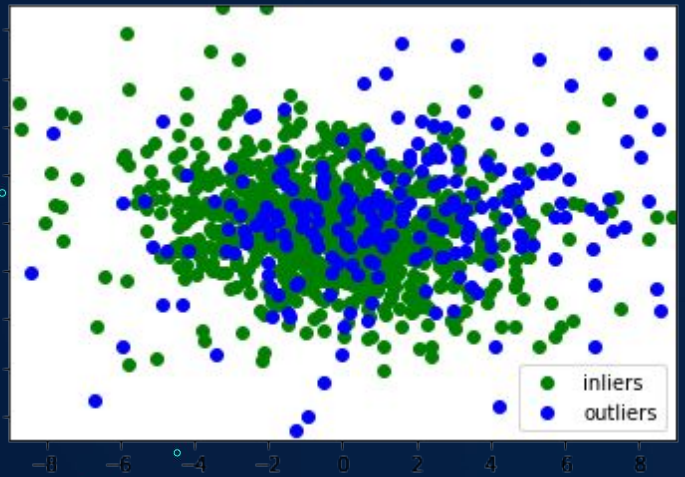
DIMENSIONALITY REDUCTION

Principal Component Analysis & K-means Clustering

PCA

- explained only 50% of the variability with 5 components so we did not obtain sufficiently satisfactory results to consider applying these reductions to our dataset

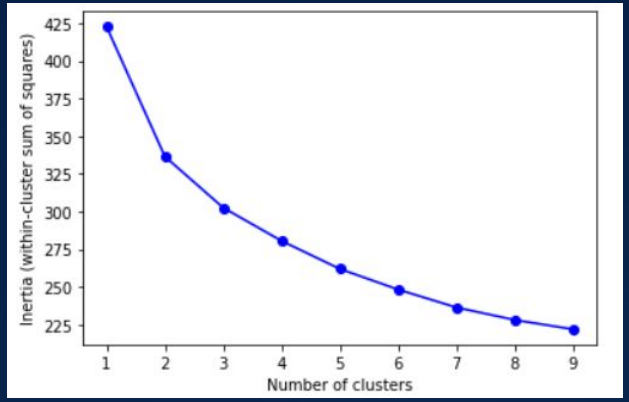
```
Explained Variance
   Component 1 0.22
   Component 2 0.17
   Component 3 0.06
   Component 4 0.05
   Component 5 0.05
   Component 6 0.04
   Component 7 0.04
   Total Explained Variance 0.63
```

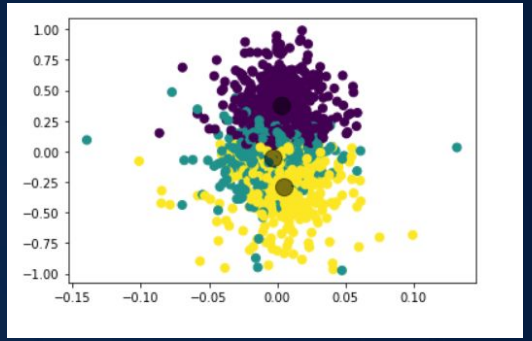


K-MEANS CLUSTERING

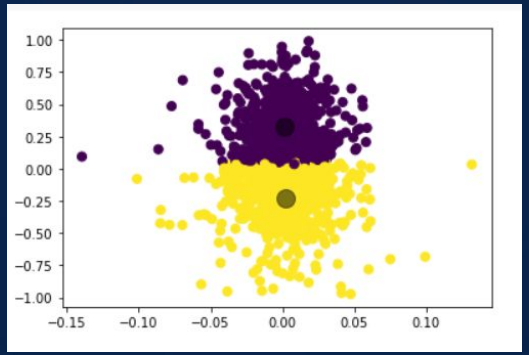
ELBOW METHOD

The marginal gain of adding one cluster dropped quite a bit from k=3 => Computing K-means with k=3





```
For n_clusters= 2 The average silhouette_score is : 0.19708403402114116
For n_clusters= 3 The average silhouette_score is : 0.15500134894680473
For n_clusters= 4 The average silhouette_score is : 0.14291430108385306
For n_clusters= 5 The average silhouette_score is : 0.1460346098615955
For n_clusters= 6 The average silhouette_score is : 0.15001144038375236
For n_clusters= 7 The average silhouette_score is : 0.12490247100506371
For n_clusters= 8 The average silhouette_score is : 0.11332831002297651
For n_clusters= 9 The average silhouette_score is : 0.10319897632545463
For n_clusters= 10 The average silhouette_score is : 0.09315691583387307
```



SILHOUETTE ANALYSIS

maximum average silhouette score across all observations for k=2 => Computing K-means with k=2

04

ANOMALY DETECTION

One-class Support Vector Machine
Isolation Forest
Trees models

ONE-CLASS SUPPORT VECTOR MACHINES

One-class SVMs detect anomalies by fitting an SVM using one single target class. The data, which is assumed to contain no anomalies (indeed SVMs are sensitive to outliers), is used for training a model. The output returns boundaries that can be used to classify future incoming data points.

```
Outlier Ratio 0.200     Number of errors: 921
Outlier Ratio 0.100     Number of errors: 898
Outlier Ratio 0.050     Number of errors: 888
```

```
Outlier Ratio 0.200     Number of errors: 921
Outlier Ratio 0.100     Number of errors: 897
Outlier Ratio 0.050     Number of errors: 885
Outlier Ratio 0.010     Number of errors: 876
```

ISOLATION FOREST

Isolation forest detects anomalies using isolation, calculating how far a data point is to the rest of the data. Indeed it computes the number of splits required to isolate a single data points: how many times we need to perform splits on features in the dataset before we end up with a region that contains only the single target sample.

TREES MODELS

Decision Tree
Bagging Tree
Adaboost Tree
Random Forest

DECISION TREE

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Confusion Matrix (Train) - DecisionTree
        0       1
0        658     40
1        70      120

Accuracy (CV)   0.804 0.038
Precision (Train) 0.575 0.126
Recall (Train)    0.432 0.084
F1 (Train)        0.485 0.083

Confusion Matrix (Test) - DecisionTree
        0       1
0        150     25
1        21      26

Accuracy (Test) 0.793
Precision (Test)  0.510
Recall (Test)     0.553
F1 (Test)         0.531
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.3s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.3s finished
```

BAGGING TREE

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Confusion Matrix (Train) - Bagging
        0       1
0        698     0
1        0       190

Accuracy (CV)   0.823 0.017
Precision (Train) 0.632 0.062
Recall (Train)    0.416 0.063
F1 (Train)        0.500 0.060

Confusion Matrix (Test) - Bagging
        0       1
0        160     15
1        19      28

Accuracy (Test) 0.847
Precision (Test)  0.651
Recall (Test)     0.596
F1 (Test)         0.622
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    7.5s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    7.5s finished
```

ADABOOST TREE

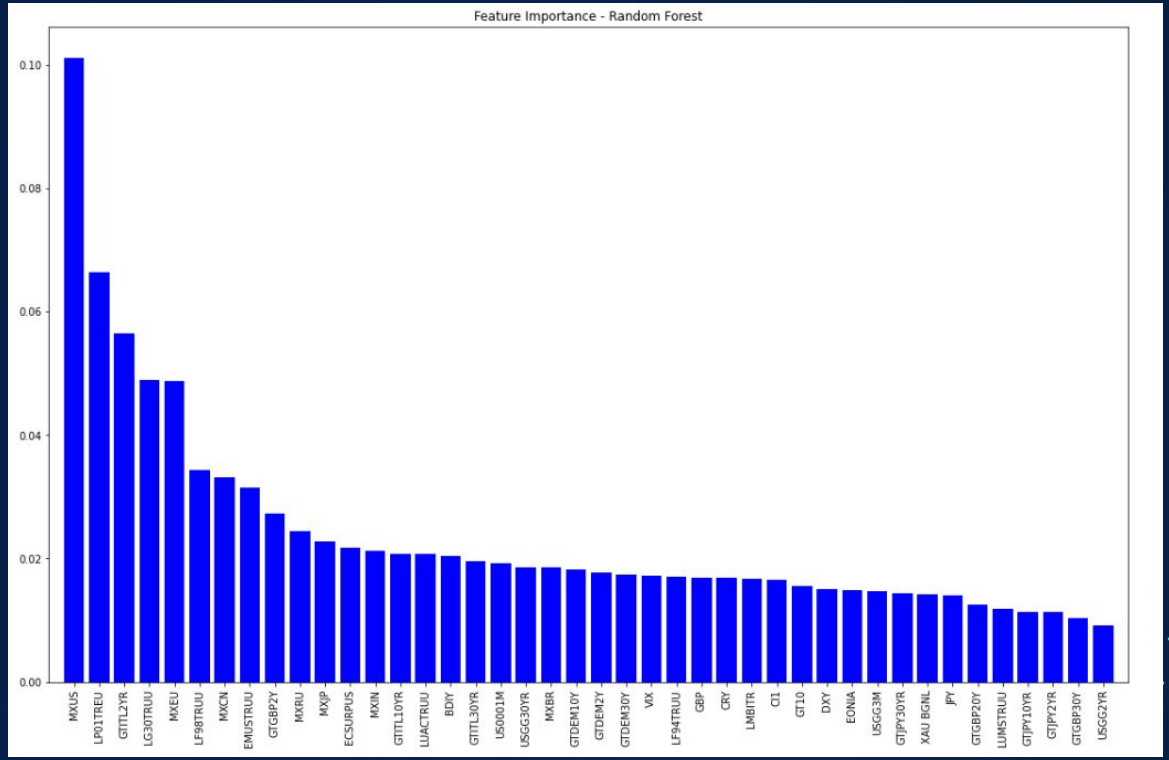
```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Confusion Matrix (Train) - AdaBoost (Tree)
        0       1
0        698     0
1        0       190

Accuracy (CV)   0.753 0.035
Precision (Train) 0.415 0.089
Recall (Train)    0.379 0.087
F1 (Train)        0.396 0.088

Confusion Matrix (Test) - AdaBoost (Tree)
        0       1
0        157     18
1        19      28

Accuracy (Test) 0.833
Precision (Test)  0.609
Recall (Test)     0.596
F1 (Test)         0.602
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.3s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.3s finished
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Confusion Matrix (Train) - Random Forest
        0       1
0        698     0
1        0       190

Accuracy (CV)   0.811 0.029
Precision (Train) 0.620 0.120
Recall (Train)    0.358 0.102
F1 (Train)        0.442 0.090

Confusion Matrix (Test) - Random Forest
        0       1
0        163     12
1        20      27

Accuracy (Test) 0.856
Precision (Test)  0.692
Recall (Test)     0.574
F1 (Test)         0.628
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    2.2s finished
```

RANDOM FOREST

- overall best result

FEATURES IMPORTANCE
of random forest



05

NEURAL NETWORKS

Neural networks with time series
Long Short-Term Memory network

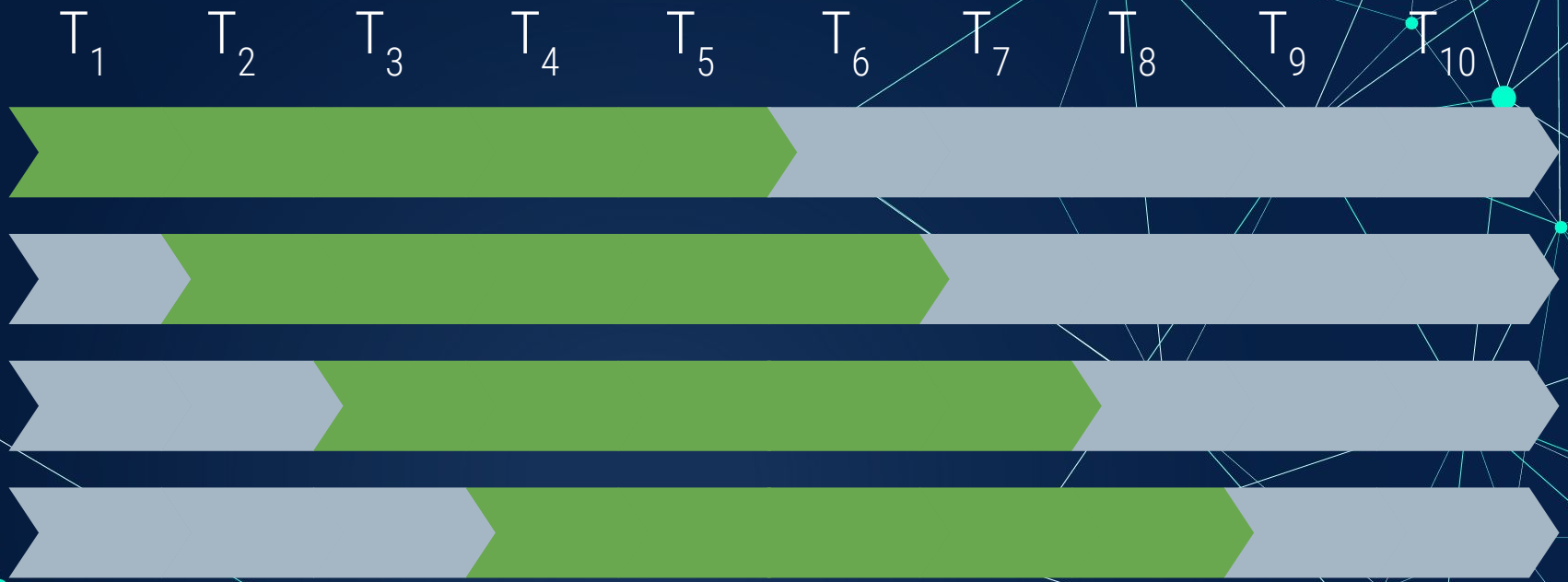
Approaches

Two approaches

1. Use the standardized version of the dataset to train a neural network

2. Create a sequenced version of the dataset and train a normal neural network as well as a Long-term short-term (LSTM) network on the sequenced dataset

Creating sequences



This example resulting in 4 sequences of 5 time units in length.

Three models/networks

Standard neural network

- Idea was deep neural network, started with 3 layers and 24 neurons in each (~2200 parameters)
- Final network was only 133 parameters (without inducing overfitting)

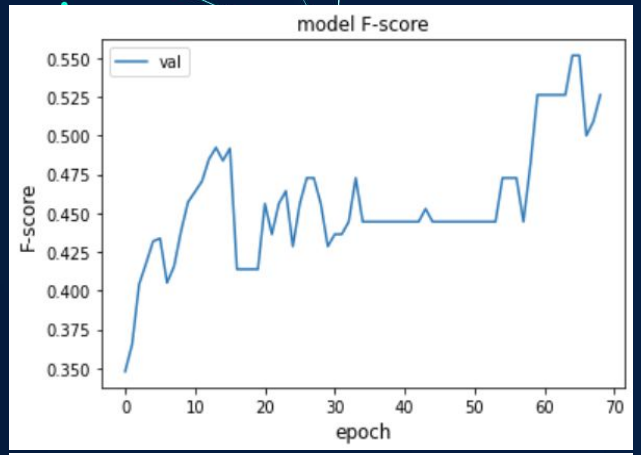
Final network has 193 parameters

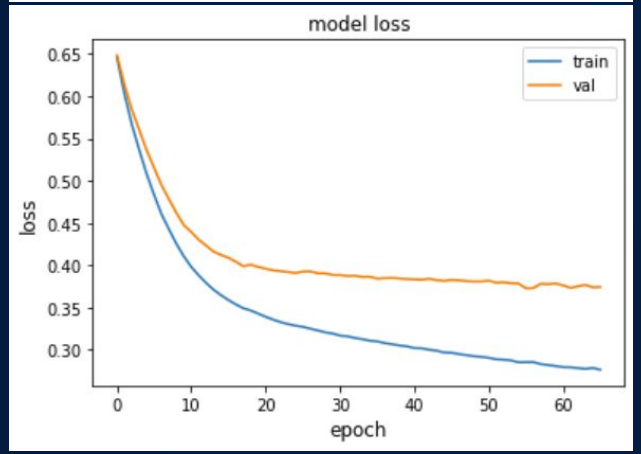
Standard neural network with sequenced data

Long short-term memory (LSTM) - network

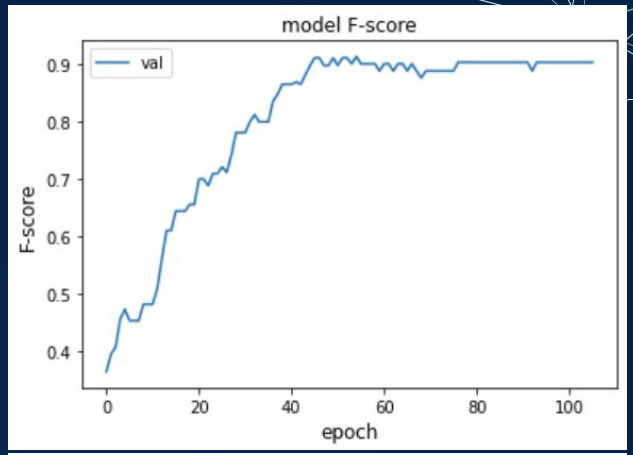
- LSTM networks are well-suited to classifying and making predictions based on time series data
- 4 LSTM cells ⟶ 757 parameters

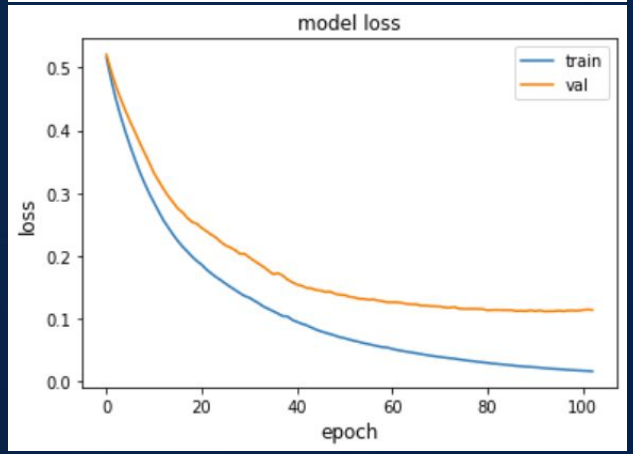
Neural Networks - Results





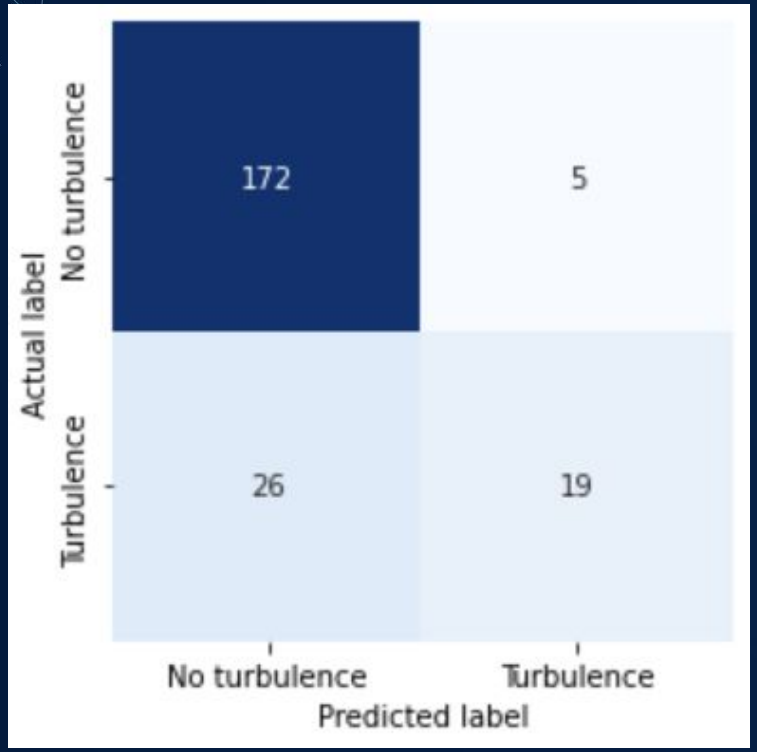
LSTM →
← Normal network
F-score calculated on the validation during training and model loss



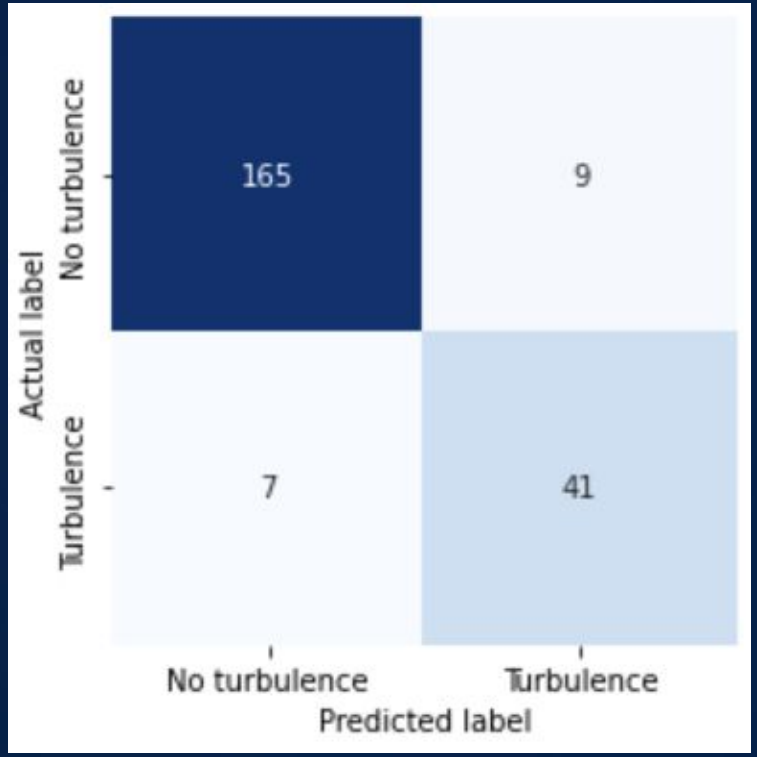


Neural Networks - Results (Sequenced data)

Normal network



LSTM network



Comments on Stationarity for neural networks

- Using non-stationary data for the networks yields better results and more stable models.
    - Black-box problem ⟶ We don't know why or how it would affect a “real world application”
- LSTM-Networks can handle non-stationary data.
- For better comparability with other models, data used in the presented results is stationary.

06

CONLCUSIONS

Final Results

- The final results are compared on F-score due to the nature of anomaly detection and unbalanced datasets.
- In general we have noticed that between the trees models the the Random Forest has the best performance (F-score of around 60%).
- Overall, the LSTM network is the best model for this business case, with a F-score, on the test set, of around 83%.
- All the reported results were obtained after trying different models and architectures (specially for NNs)