UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

Data Science and Business Informatics

DM2 Project

Advanced Data Mining on "Powering the nation" dataset

Pietro Argento

ANNO ACCADEMICO 2023/2024

Contents

1 Task 1. Data Understanding & Preparation

1.1 Dataset description

The dataset analyzed in this report is a collection of time series recorded during a study in the UK called "Powering the Nation". The intention was to collect behavioural data about how consumers use electricity within the home to help reduce the nation's carbon footprint. The study reports that data contains readings from 251 households, sampled in two-minute intervals over a month. However, in the dataset analyzed there are only 96 time points for each record. It is likely that they represent aggregated data for a single day, where each time point is 15 minutes. This hypothesis is supported by the visual inspection of random time series plots, where the interval of time supposed to be between 0:00 and 5:00 is often flat.

The following problems are given in the dataset description:

Type	Category	Appliance
Dissimilar Patterns	Small kitchen	Kettle, Microwave, Toaster
Dissimilar Patterns	Large Kitchen	Dishwasher, Tumble Dryer, Washing Machine
Similar Patterns	Refrigeration	Fridge/Freezer, Refrigerator, Upright Freezer
Similar Patterns	Computers	Desktop, Laptop
Similar Patterns	Screen	CRT TV, LCD TV, Computer Monitor

Table 1: Problems

In other terms, there are:

- 2 distinct types of categories, based on usage pattern
- 5 distinct categories of appliances
- 14 distinct appliances

In the dataset, there are 7 class labels, but it is not clear what is correspondence with the types, categories and appliances. In the next pages of the report, hypothesis will be formulated and tested.

1.2 Data Partition

There are two datasets provided: `train.csv` and `test.csv`. All the models presented in this report are trained on the `train.csv` dataset, appropriately split each time in train and validation. Furthermore, the performance of models presented in Task 2 and Task 4 are related to the use of these model on the external dataset `test.csv`, every time pre-processed with the appropriate pipeline. However, the evaluation of pre-processing approaches reported in Task 3 are obtained from `train.csv` only.

Starting from `train.csv`, which is a collection of time series, some basics data cleaning has been performed, described in the next section, to obtain a dataset for model training, called `ts_clean`.

Then, a tabular version of the dataset has been generated using `TSFresh` and, again, data cleaning has lead to a tabular dataset, called `df_clean`. Later, advanced data pre-processing has been done to obtain more polished datasets, more details in 3. All will be used to train the models and the differences will be highlighted.

1.3 Data Cleaning

Before exploring the data, some basic data cleaning is performed on `train.csv` to avoid biased results in the analysis and the models. As mentioned, two datasets are obtained from the original dataset: `ts_clean` and `df_clean`.

With regard to `ts_clean`, the cleaned version of the time series dataset, the steps performed are the following:

class	count
1	726
2	2231
3	747
4	1470
5	2393
6	509
7	723

Table 2: Frequency of classes

1. Duplicates drop, (not applied in `test.csv`);
2. Check for problematic records, such as NaN or flat time series, but none was found.

On the other hand, `df_clean`, the tabular version, is obtained with the following pipeline, starting from `ts_clean`:

1. Extract features with `TSFresh`;
2. Feature selection, dropping features with more than 90% of NaN values;
3. NaN and Inf imputation, using edited versions of `TSFresh` imputer, where NaN are replaced with the median and Inf with the max of the column.

Note that for `test.csv` duplicates drop was not applied and the selected columns are those chosen for the training set.

Few additional observations can be mentioned, regarding the features extracted, to justify the choices. First, the function provided by `TSFresh` to select the features based on the groundtruth was not used, feature selection is discussed in Task 3. Then, the columns dropped (210) are probably computations that are not suitable for this kind of time series, as a validation, the test set confirmed this hypothesis, generating approximately the same amount of NaN values. Finally, the remaining columns that needed imputation of NaN (11), contained very few problematic records, from 2 to 12 in the training and from 59 to 110 in the test set, for each column. Only one column contained Inf values (11). Imputation was chosen instead of dropping the records to be consistent and to be able to test all the instances in `test.csv`.

The shapes of the cleaned dataset, respectively for training and test, are (8799, 574) and (7711, 574).

1.4 Data Exploration

First of all, exploring `ts_clean`, we should mention that the range of values in the time series is [-9.6958969, 9.6958969], meaning that the columns are normalized using a MinMax scaler. On the other hand, each record has a mean of zero and a standard deviation of one, suggesting a Standardization.

Now, an overview of the classes. It is useful to extract some aggregated variables for each record and the group the records, taking the mean for of the variable for each class. The result is reported in Figure 1, showing some clear distinctions for Class 2 and Class 5.

Following the initial hypothesis, the time points can be aggregated using PAA to display the mean's shape of each class along the 24 hours. This validates the hypothesis, since the night hours between 0:00 and 5:00 show the lowest values of consumption. See Figure 2

class	min	max	0.1	0.25	0.5	0.75	0.9
1	-0.211283	6.555165	-0.211283	-0.211283	-0.211283	-0.211110	-0.098891
2	-1.043881	1.691159	-0.998937	-0.838508	-0.278029	0.879265	1.381744
3	-0.207617	6.659856	-0.207617	-0.207617	-0.207617	-0.196680	-0.064572
4	-0.435081	6.204052	-0.372594	-0.317601	-0.206697	-0.076783	0.247228
5	-0.743588	3.181053	-0.655383	-0.503505	-0.225641	0.128938	0.987229
6	-0.202933	7.364749	-0.202933	-0.202933	-0.202710	-0.199860	-0.068419
7	-0.440922	5.249118	-0.433696	-0.426783	-0.246952	0.050104	0.402613

Figure 1: Summary aggregated by class taking the mean.

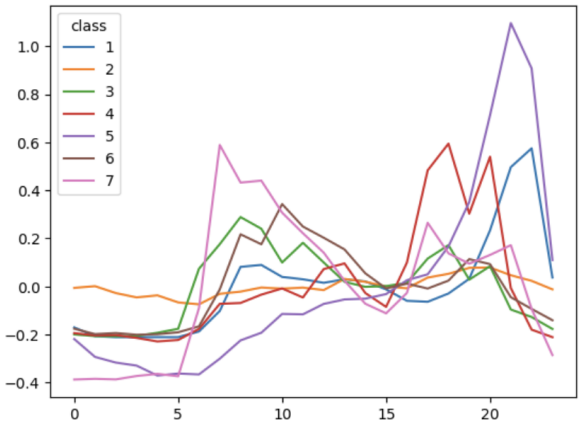


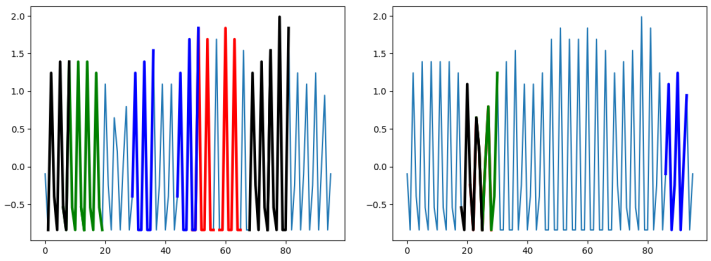
Figure 2: Classes means after PAA with n=24

Figure 3: Random Time Series samples divided by class

2 Task 2. Time Series Analysis

2.1 Motifs and Discords

An example of motifs and one of discords on a random time series from the dataset is showed in Figure 4a and 4b.



(a) Example of Motifs (b) Example of discords

2.2 Time Series Clustering

Clustering, as an unsupervised task, might be a good approach to solve the problem of identifying the different type (Similar or Dissimilar pattern) and category of each time series. The identification of the appliance within the category appears much harder.

The first experiment is to distinguish the type of pattern. TimeSeriesKMeans with k=2, metric=dtw and SakoeChiba=0.3 delivers a good result, with the best silhouette score among the tested configurations of 0.283. Some samples, together with the centroids, can be visualized in Figure 5.

Regarding the identification of the 5 categories, the configuration of TimeSeriesKMeans without sakoe-chiba looks better from both the silhouette score (0.212 against 0.186) and from the visual inspection of random samples, reported in Figure 6.

We can speculate on the shapes presented in Figure 6. For example, Cluster 1 and Cluster 5 have very similar consumption patterns, constant along all day, they might be part of the Refrigeration Category. Most samples showed in Cluster 3 have sharp peaks in the hours of lunch and dinner, compatible with the Small Kitchen Category, with Kettles, Microwaves and Toasters.

TimeSeriesKMeans with euclidean distance is not reported because of the very poor results compared to the dtw configuration. As an example, the silhouette score with k=5 is 0.048, compared to the mentioned 0.212 of dtw.

An additional experiment has been done with DBSCAN on the dataset of extracted features. However, the configuration of parameters determined using the elbow method is not satisfactory, even

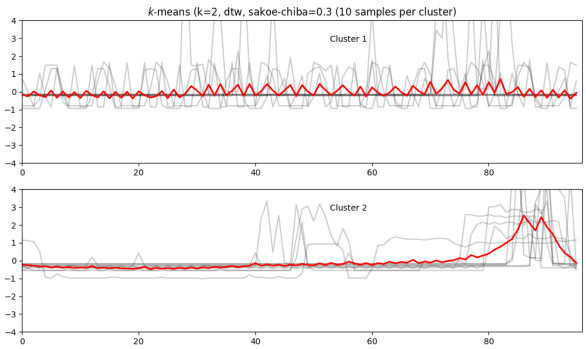


Figure 5: TimeSeriesKMeans with k=2, metric=dtw and sakoe-chiba=0.3, samples and centroids

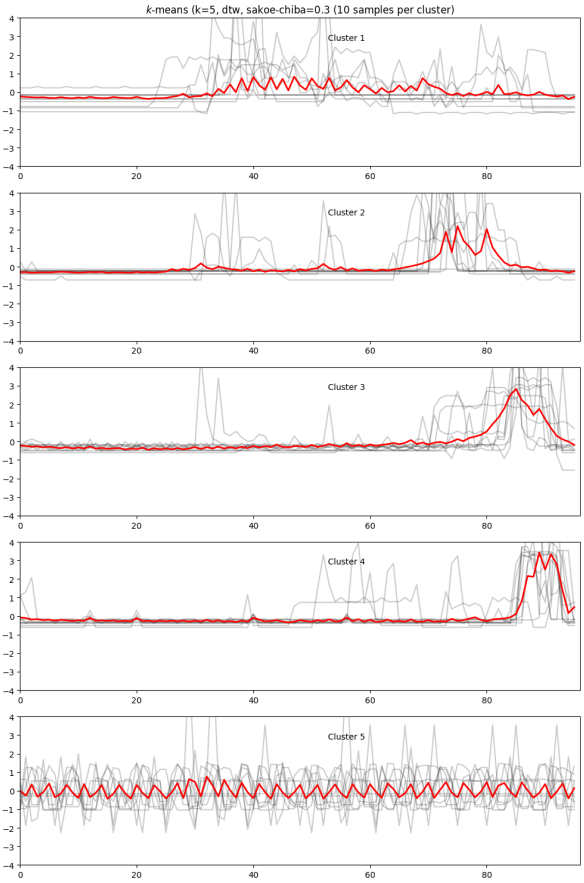


Figure 6: TimeSeriesKMeans with k=5, metric=dtw, samples and centroids

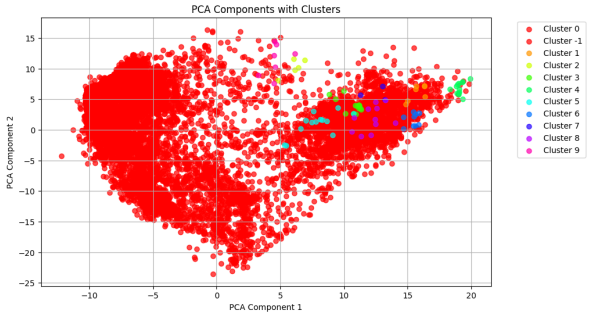


Figure 7: PCA on DBSCAN Clustering

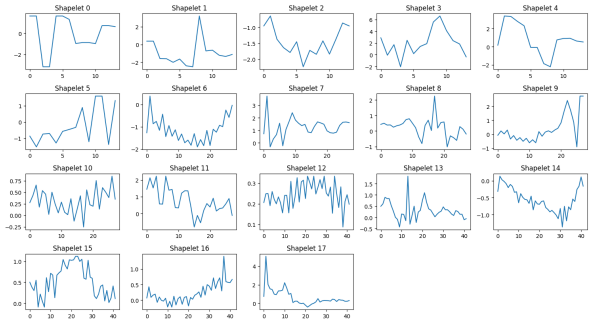


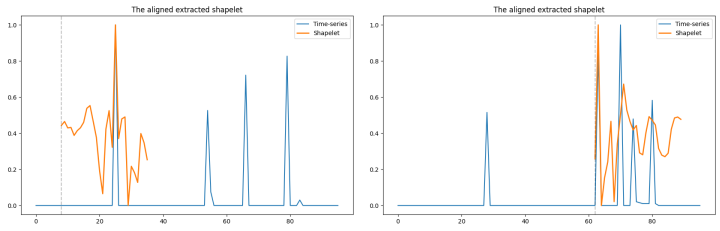
Figure 8: Shapelets extracted

though more configurations could be tested. The silhouette score is close to -1, revealing that there is no clear distinction between clusters. A visualization using PCA is reported in Figure **??**.

2.3 Time Series Classification

With the aim of generalizing as much as possible, the KNeighborsTimeSeriesClassifier (k=5), was tested on the PAA version of the dataset (n=24). The choice of 24 segments reflects the hours of the day, as already mentioned. Both Euclidean Distance and DTW were tested. The result of the first metric on the external test is a low accuracy of 0.49, given a validation score of 0.67 that was already showing a bad performance. The DTW configuration, on the other hand, delivers 0.29 of accuracy, which is close to random guessing. Even though more configurations could be tested, this result suggest that the approach might not be ideal for this kind of time series.
Another approach is the one with Shapelets. The shapelets obtained (17) from the training dataset are reported in Figure 8. Two examples of aligned shapelets are showed in Figure 9a and 9b. A DecisionTreeClassifier was trained on the dataset built using the distances from the shapelets and the accuracy of the model on the test is 0.62, while the AUC is 0.74. Given the higher accuracy (0.80) on the validation set, the shapelets might be too specific for the training set, failing in helping the prediction of unseen data. Also, more advanced classifiers could be tested on the dataset generated from the shapelets' distances.

A more advanced classifier has been tested as well: Rocket, included the 'minirocket' configuration. Surprisingly, the simpler technique of 'minirocket' delivers a better score than 'rocket', 0.735 against 0.729. However, this suggests, together with the much higher scores of validation, the overfitting problem. This is the reason why the simpler model performs better on unseen data, because it is more general.



(a) An example of aligned shapelet

(b) Another example of aligned shapelet

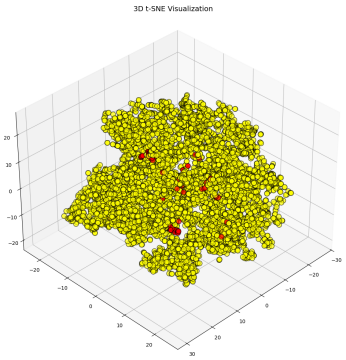


Figure 10: T-SNE Visualization of ABOD Outliers

3 Task 3. Advanced Data-Preprocessing

3.1 Outliers

From now on, only the tabular dataset will be considered. In particular, the goal of the "Task 3" is to derive two improved versions of `df_clean`.

In outliers detection, the techniques that will be used are the following. The reason behind this choice is the high-dimensionality of the dataset, other types of algorithms, like distance-based, might fail due to the curse of dimensionality.

1. ABOD (Angle-based Outlier Degree), because angles are more stable than distances in high-dimensional space
2. LODA (Lightweight On-line Detector of Anomalies), even though one-dimensional histograms are not ideal, the ensemble might result in a strong approach
3. Isolation Forest, because it appears to be the state of the art.

	ABOD	LODA	Isolation Forest
1	0.83%	1.10%	0.14%
2	1.43%	0.40%	0.63%
3	0.27%	0.40%	nan%
4	1.02%	0.68%	0.61%
5	1.30%	2.17%	2.59%
6	0.39%	0.59%	0.20%
7	nan%	0.41%	0.14%

Table 3: Ratios of classes in outlier detection

Evaluating the result is not straightforward, since there is no groundtruth. Note that the 1% of outliers was extracted for all the techniques, resulting in 88 records. The chosen technique for `df_refined` is ABOD. There are few reasons that lead to this choice. First, the high-dimensional

friendliness by-design. Then, the distribution of outliers among classes (see Table **??**), which is more balanced compared with the results of LODA and Isolation Forest. In fact, they detected respectively 59% and 70% of outliers exclusively from class 5, suggesting that those types of records where treated as very different from the others, but probably the models failed in identifying the reasons for being outliers. However, visualizing the outliers using feature projection techniques is not helpful, in fact it casts doubt on the effectiveness of all the techniques used, since they outliers lie in the middle of the cloud of points (see Figure 10).

In the set up of `df_refined`, the outliers identified are imputed, not removed with the intention of keeping as many records as possible. This effort is necessary, given the small dimension of the dataset and the willingness to test all the records. Given the high-dimensionality, the KNN Imputer was discarded and the more advanced IterativeImputer was used.

3.2 Feature Selection

The features in `df_clean` are 573. This fact raises some questions on the usefulness of the features for classification. Different approaches will be discussed for reducing the number of features without losing accuracy.

First, a look at the variance. Using the `VarianceThreshold` set to 0.8, it would be possible to exclude 207 features. But additional techniques can be used.

Using a simple `DecisionTreeClassifier`, it is worth mentioning that there are 356 features with zero importance. The two most relevant features are calculations of `agg_linear_trend`, meaning linear least-squares regression for values of the time series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one. However, impurity-based feature importances, like those of Decision Trees, can be misleading for high cardinality features. More on this fact in the Explainability Section.

Other techniques, like RFE (Recursive Features Elimination) were tested, but the final decision is to use Select From Model with a DecisionTreeClassifier. This approach leads to the best classification accuracy of the standard DecisionTreeClassifier, used as a benchmark for evaluating the different approaches. See Table 4

	All features	Variance Threshold	Select From Model
1	0.851	0.804	0.852
2	0.972	0.969	0.967
3	0.640	0.611	0.703
4	0.804	0.787	0.805
5	0.893	0.887	0.900
6	0.743	0.728	0.741
7	0.632	0.568	0.689
accuracy	0.844	0.826	0.854

Table 4: f1 scores of DecisionTreeClassifier with different set of features

3.3 Imbalance Learning

In this section, there are two different parts that will be described.

1. the final step for obtaining the `df_refined` dataset, that will be used soon;
2. an experiment on an extremely imbalanced dataset to verify the efficacy of undersampling and oversampling approaches.

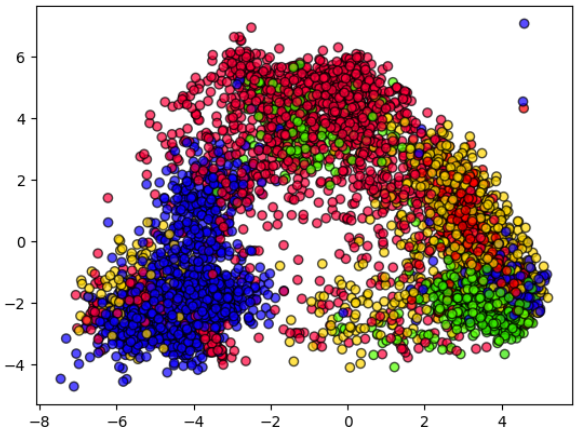


Figure 11: PCA of `df_refined`

	Imbalanced	SMOTE	ADASYN
1	0.84	0.88	0.72
2	0.97	0.97	0.96
3	0.67	0.70	0.55
4	0.81	0.83	0.75
5	0.90	0.90	0.87
6	0.79	0.84	0.69
7	0.62	0.66	0.57
accuracy	0.847	0.860	0.796

Table 5: DecisionTreeClassifier f1-score with or without SMOTE

The original dataset is not dramatically imbalanced, as already showed in Table 2. However, as it will be clear in a moment, the classifiers struggle to predict correctly the classes with less values available, in particular Class 6 with only 509 values, compared to the 2393 of Class 5. Hence, `df_refined` will be balanced, with the hope that advanced classifiers will benefit for this choice.

Since undersampling will drastically reduce the size of the dataset because of the smallest class, oversampling was chosen. In particular, the creation of synthetic data relies on the SMOTE Technique. As it is visible in Table 5, the DecisionTreeClassifier has slightly improved its performance, in particular for the less represented classes.

After balancing, `df_refined` is ready for more advanced classification in the next section. Meanwhile, an experiment for further testing of balancing techniques can be done on an extremely imbalanced set. Recalling the sample of time series in Figure 3, it might be interesting to verify if, given a very small set of instances from the least represented class, it can be classified correctly against the rest.

To set up this experiment, the target label is modified to turn the task into an extremely imbalanced binary classification. The records belonging to class 6, i.e. the smallest, are labelled as 1, while the others as 0. The result is a dataset with 509 ones and 8290 zeros, meaning 5.8% against 94.2%.

Again, using the DecisionTreeClassifier as benchmark. Both undersampling and oversampling approaches were tested. The results are reported in the Table 6. It is clear that ADASYN overperforms all the other techniques, which failed also compared to the original imbalanced dataset. A PCA visualization of ADASYN is reported in Figure 12

	Imbalanced	TomekLinks	ClusterCentroids	SMOTE	ADASYN
precision-0	0.98	0.98	1.00	0.99	1.00
recall-0	0.99	0.99	0.54	0.98	1.00
precision-1	0.75	0.75	0.12	0.66	0.96
recall-1	0.75	0.72	0.98	0.81	0.97

Table 6: DecisionTreeClassifier results of Class 6 detection with different balancing approaches

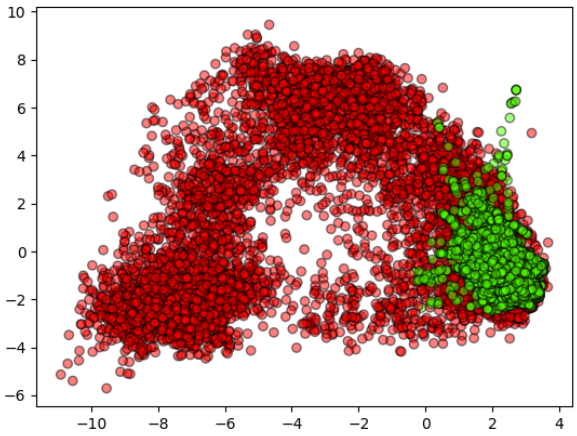


Figure 12: PCA after resampling with ADASYN

4 Task 4. Advanced ML & XAI

4.1 Multi-class Classification

In this section, advanced classification will be experimented on the multi-class problem of identifying the correct label in the original dataset.

The models trained for the multi-class are the following:

1. Neural Networks
2. Random Forest
3. Bagging with RandomForest
4. XGBoost

Neural Networks has been tested with many different configurations. Three models in particular, a simple one with no regularizations, one with L2 regularization and one with Dropout, each one in slightly different variants. The final layer was always using the 'softmax' function, to be able to address the multi-class problem. Additionally, different settings of epochs, ranging from 10 to 1000, and early stopping, with values from 10 to 50, were experimented as well.

As already seen during the Time Series classification, specifically with Rocket, the results in cross validation are very good, with accuracy around 0.95, but in the external test, the models struggles to obtain a value of accuracy higher than 0.75. To simplify as much as possible the model and investigate the reason for this result, the neural networks were tested with very low values for the epochs and the early stopping, together with higher values for the regularization parameters. Emblematic is the model trained for only 10 epochs, which delivers values of accuracy respectively for the validation and the external test of 0.94 and 0.74. One of the best configurations was a network with two Dropout(0.1), trained for 100 epochs, that reached an accuracy of 0.76 on the external test. The failure of the efforts to get a similar value in the validation and test could be related to different distributions of data given the class label. However, further tests should be done.

The default version of Random Forest delivers the accuracy values reported in Table 7. This suggest that the advanced pre-processing has improved that predictive power, even if slightly. However,

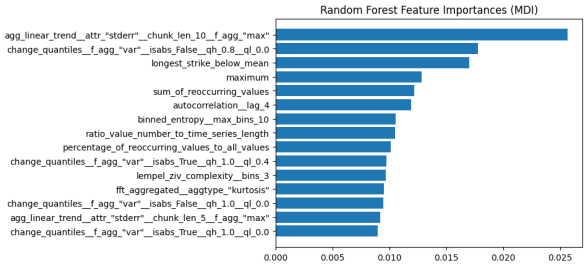
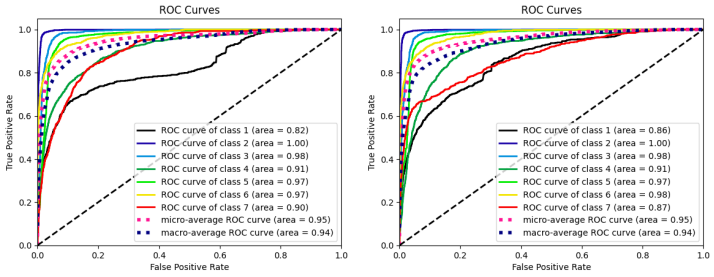


Figure 13: Feature Importance of best Random Forest



(a) XGB with df_clean (b) XGB with df_refined

Figure 14: XGB with best parameters

some concerns arises given the difference between the results in the training compared to the real-world data, which clearly indicates over-fitting. A GridSearchCV on `min_samples_split`, `min_samples_leaf` and `max_depth` is performed, but the accuracy of the model on the external test doesn't improve as expected. This fact might be due to a different data distribution in the external test, as well as problems with pre-processing, but further testing of parameters could be made to improve the generalization power of the model.

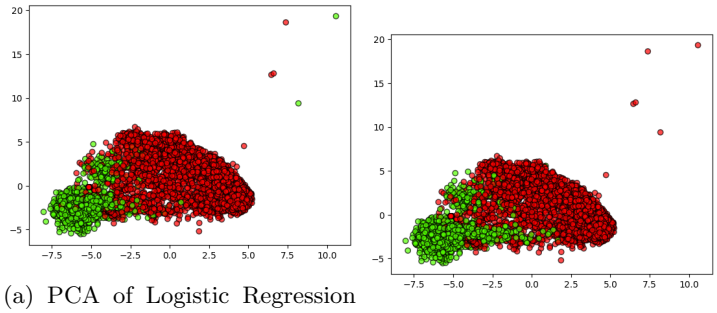
The feature importance of the Forest is reported in Figure 13, confirming the significance of `agg_linear_trend`, already seen in the section **??** about feature selection.

	`df_clean`	`df_refined`
test_split=0.3	0.903	0.948
cross_val	0.821	0.939
external test	0.762	0.786

Table 7: Accuracy scores of default RandomForest

Using Bagging with 100 Random Forests delivers an accuracy of 0.784, showing that there is no improvement with respect to Random Forest alone. Furthermore, the f1-score for the Class 1, the hardest to be identified correctly, went from 0.43 to 0.42.

Given `df_clean` in input, a RandomizedSearch on XGBoost has discovered the following best parameters: {objective: `multi:softmax`, `subsample`: 0.9, `n_estimators`: 100, `min_child_weight`: 3, `max_depth`: 5, `learning_rate`: 0.2, `gamma`: 0, `colsample_bytree`: 1.0}. The accuracy on the test without the advanced pre-processing is 0.781. An XGBoost trained with the same parameters on `df_refined` shows a very modest increase in the accuracy, reaching 0.793. In Figure 14 it is visible the difficulty in identifying Class 1 and Class 7.



(a) PCA of Logistic Regression predictions

(b) PCA of the groundtruth

Figure 15: PCA of Logistic Regression

4.2 Binary Classification

Advanced classification will be now experimented on the binary distinction between Class 2 and the rest. The choice of Class 2 for this task is motivated by the visual inspection in Figure 3 and the sufficiently high frequency of values, allowing experiments also on a slightly imbalanced dataset. Given the shape of those time series, it is reasonable to think of the Class 2 as the Refrigeration category.

The models tested for the binary problem are

1. Logistic Regression
2. Support Vector Machines

As expected, the score of the Logistic Regression is almost perfect in identifying the Class 2, despite the imbalance, i.e. 22% in the train and 25% in the external test. The precision and recall are 0.98/0.99 and 0.96/0.93 respectively for all the other classes and Class 2. The few mistakes done by the the model can be explained with the presence of outliers in the external test, which are imputed as explained in Task 3, but not removed. A visualization of these outliers can be obtained using PCA, see Figure 15.

An additional experiment can be done using the Support Vector Classifier. Given the shape in Figure 15b, the expectation is that 'linear', 'rbf' and 'poly' all will work quite good. The results confirm this hypothesis, in fact the f1-score of the target class is respectively 0.95, 0.96, 0.95 in the external test. Interestingly, a different value for C has been identified by GridSearchCV for each kernel of the SVC, respectively 0.1, 10, 1. These values are reasonable, since the lower C for the linear kernel allows some misclassification, while the higher C of RBF might be required to penalize more the misclassification given the higher complexity of the model.

4.3 Advanced Regression

The regression task chosen is the identification of the maximum of the time series, that is one the important features selected, appeared also in Figure 13. The regressors that will be used are:

1. RandomForestRegressor
2. BagginRegressor with Support Vector Regressor

The RandomForestRegressor shows already very good results with the default parameters also in the external test: R2 of 0.975 and MSE of 0.175. However, the visualization of the feature importance reveals that this result can be almost completely explained by the feature `'agg_linear_trend__attr_"stderr"__chunk_len_10__f_agg_"max"'`. This feature is probably calculated on the maximum during the feature extraction process, together with other features, showed in Figure 16. The features with a Spearman Correlation above 0.5 are removed to make the task harder.

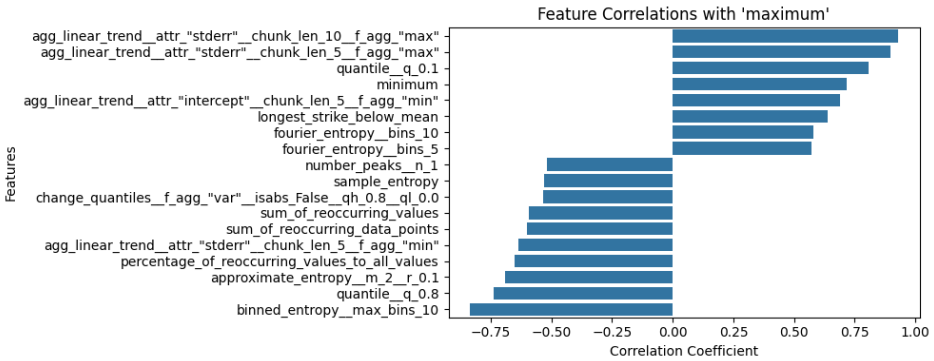


Figure 16: Features with above 0.5 Spearman Correlation with 'maximum'

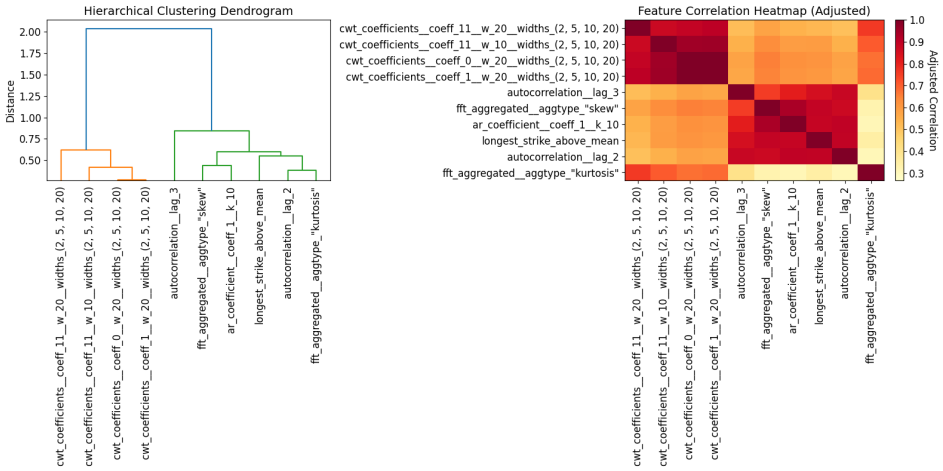


Figure 17: RandomForestRegressor Feature Permutation

Considering the new set of features adjusted deleting the high-collinearity features, the result of the RandomForestRegressor is still very good with values of R2=0.944 and MSE=0.389. The effect of this selection of features on importance is visible in Figure 17.

Given the adjusted dataset, another approach worth mentioning is BaggingRegressor with Support Vector Regressor. Selected 10 estimators with an RBF Kernel, the R2 is 0.893 and the MSE is 0.752. An additional test scaling also the target feature 'maximum', and then inverting the scale for the evaluation, reveals that this step is not crucial, but helps, showing an R2 of 0.896 and an MSE of 0.731 on the external test.

4.4 Explainability

From the test of a RandomForest in the multi-class setting, an explanation of the prediction has been analyzed using the LORE explainer. The result is reported in Figure 18. On the left, the rules for the prediction of class 3 has been printed, suggesting that the sum of reoccuring data points is very relevant and, recalling the Figure 3, this fact seems reasonable. In the counterfactual rules for Class 6, the sample entropy plays an important role as well as the maximum.

(a) Rules of LORE (b) Counterfactual Rules of LORE

Figure 18: LORE on Random Forest