POLITECNICO
MILANO 1863

Assignment 1 EPLF - Group 10

Financial Engineering - A.A. 2023-2024

Group Components:

Spokeswoman: Marchetto Erica - CP: 10700150 - MAT: 232637

Corti Stefano - CP: 10730685 - MAT: 245554
Manfredi Giacomo - CP: 10776946 - MAT: 247438
Maspes Marco - CP: 10677441 - MAT: 216843

Contents

1 Introduction

The objective of this assignment is to develop probabilistic time series forecasting systems for predicting electricity price and load. Specifically, we will focus on the **Autoregressive with Exogenous Inputs (ARX) model** in a multi-step forecasting context.

2 ARX - L1 Experiments

In the first point we were asked to exploit the L_1 regularization in our predictions variyng the regularization coefficient. The L_1 regularization, also called Lasso regularization, is a procedure used to reduce the number of relevant features in a dataset. More precisely the Lasso process penalizes those models which maintain a high number of parameters in order to predict the target variable. The penalization is exploited through the shrinkage, toward zero, of the weights associated to the i^{th} feature.

The Lasso loss function can be computed theoretically as:

$$L(\underline{w}) = \min_{\Omega} \sum_{n=1}^{N} (f_\Omega(x_n) - y_n)^2 + \lambda \overbrace{\#\Omega}^{\text{L1 norm}} \sum_{j=1} |w_j|$$

Training and Validation loss: During each recalibration step, we computed both training and validation loss for the entire period of epochs. It's important to use the regularization techniques in the validation set in order to let the loss converge to the training one, otherwise it will diverge and overfit.

We analyzed three different cases of Lasso regularization changing the parameter λ; we studied $\lambda = 10^{-1}$, $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$. Through the use of the *plot history* attribute in the fitting of the model during the recalibration, we obtained the training and validation losses for each iteration of the process:

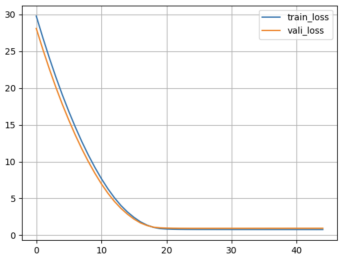

Figure 1: Case $\lambda = 10^{-1}$

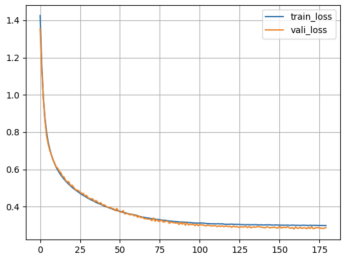

Figure 2: Case $\lambda = 10^{-3}$

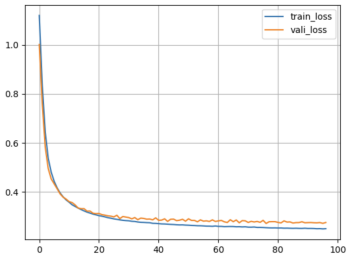

Figure 3: Case $\lambda = 10^{-5}$

Looking at the previous plots, we could infer different observations:

- **lambda = 0.1 :** the values of the losses are much higher than the other cases but it will take only a few epochs in order to decrease near to zero. This behaviour is a consequence of the high Lasso penalization that rapidly shrinks the weights toward zero. The behaviour is similar for all the recalibrations and could lead to underfitting of the test predicted values.
- **lambda = 0.001 :** the values of the losses are lower than the previous ones but they have a slow convergence ($\sim$ 160 epochs). The behaviour is similar for all the recalibrations.
- **lambda = 0.00001 :** the values of the losses are comparable to the previous ones but, even if they reduce to zero in a few epochs, they will converge in a long time. The convergence speed changes through the recalibrations and it could lead to a bit of overfitting in the test predicted values.

Prediction convergence: The previously trained model needed to be tested on an independent test set; consequently we showed the predicted Energy Market hourly prices on a batch of 10 days after the training set.

As previously supposed the case **lambda = 0.1** is highly under fitted; the predicted values are much lower than the true ones even if the seasonality is maintained. Instead it's not possible to strongly infer the over fitting on

the case **lambda = 0.00001** since it's similar to the **lambda = 0.001** one; hence we're required to compute some metrics to choose the best regularization between them.

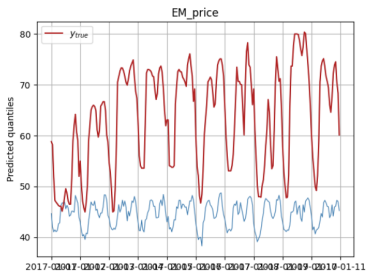

Figure 4: Case $\lambda = 10^{-1}$

Figure 5: Case $\lambda = 10^{-3}$

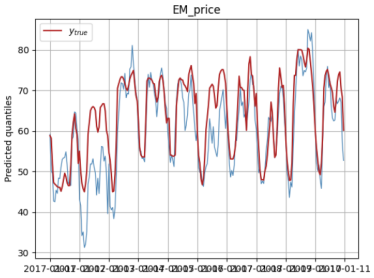

Figure 6: Case $\lambda = 10^{-5}$

In order to compare all the final plots together, we also introduce a few metrics. We calculated the average - ie. over the whole period - RMSE, MAE and sMAPE indicators:

	$\lambda = 10^{-1}$	$\lambda = 10^{-3}$	$\lambda = 10^{-5}$
RMSE	20.71	5.92	6.49
MAE	18.90	4.69	5.04
sMAPE	33.72%	7.99%	8.58%

Table 1: Error Metrics on L_1 Values

As we've previously stated, the case **lambda = 0.001** is the better choice for the hyper-parameters since the error metrics are the lowest between them all.

3 Weights Analysis

The aim of this section is to plot and analyze the weights of the layers' units computed during the recalibration of the model. In particular, we are dealing with the single hidden dense layer of the model.

The function *plot_ weights* used in *run_ recalibration* shows for each iteration of the recalibration process an image that represents the learned patterns in our model's weights. Every image is built with the hours of the day on y-axis and with the layer's unit on x-axis, we also added a *colorbar* in order to understand the value assigned to each color in the image.

Looking through the dataset, we found out that the features considered for the weights computation describe different frameworks: energy price of two days before the computation point, future prediction of wind and sun, load prevision and the seasonality terms. In this main interpretation we're considering a test set over January.

The images below represents the last update of the model calibration, which is the one used for the predictions computation, for each value of λ.

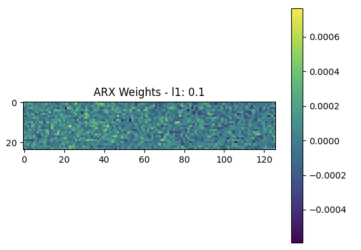

Figure 7: Case $\lambda = 10^{-1}$

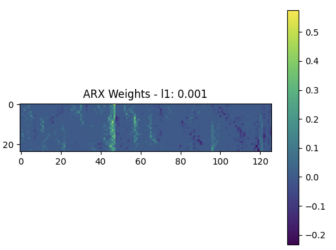

Figure 8: Case $\lambda = 10^{-3}$

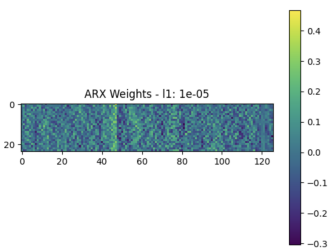

Figure 9: Case $\lambda = 10^{-5}$

The plots above are highly descriptive of the model that they're creating. The most correct model is the one with **lambda = 0.001** since maintain the weights similar for each hour of the day; in this way it's possible to interpret the features in a more direct way. Moreover (thanks to this regularity between different hours) it's possible to save computational power in the creation of models for the load forecast; we can create only 3 macro models - 0/8h, 8/16h and 16/24h - that forecast the entire day long.

The **lambda = 0.1** case highly reduce the weights towards zero but introduce too much bias in the framework; it's not possible anymore to get a good prediction from it.

The **lambda = 0.00001** case instead is similar in prediction to the best one but change the weights for each hour of the day. This can lead to computational difficulty in the explanation and interpretability of results to the users of the model.

4 Ridge and Elastic Net Regularization

We tried to compare the impact of using different types of regularization; more precisely we used the Ridge regularization - also called L_2 norm regularization - and the Elastic Net regularization.

In the following paragraphs, we will compare only the final prediction in order to lighten the report. The remarks on the train and validation losses are the same as the L_1 case.

Ridge Regularization The Ridge regularization can be written through the following formula:

$$L(\underline{w}) = \min_{\Omega} \sum_{n=1}^{N} (f_\Omega(x_n) - y_n)^2 + \gamma \overbrace{\sum_{j=1}^{\#\Omega} w_j^2}^{\text{L2 norm}}$$

Inside the Ridge regularization there is no great difference changing the hyper-parameter γ which tries to penalizes the use of multiple weights different from zeros. The absence of difference is a consequence on the L_2 norm which shrink the weights to 0 but not identically equal to it; thus it's possible to have inconsistent weights for all the features without penalizing them too much.

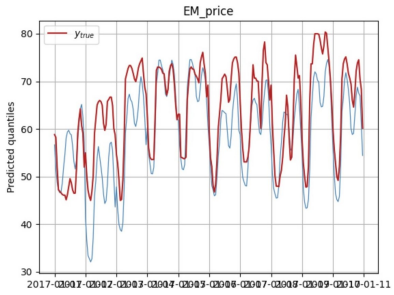

Figure 10: Case $\lambda = 10^{-1}$

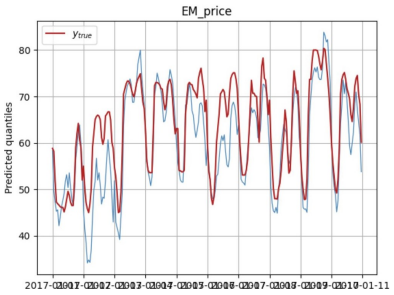

Figure 11: Case $\lambda = 10^{-3}$

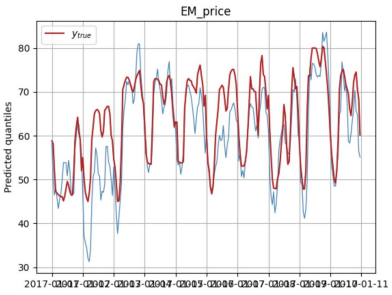

Figure 12: Case $\lambda = 10^{-5}$

The error metrics are described in the following table:

	$\gamma = 10^{-1}$	$\gamma = 10^{-3}$	$\gamma = 10^{-5}$
RMSE	7.23	6.11	6.64
MAE	6.01	4.76	5.20
sMAPE	10.27%	8.14%	8.83%

Table 2: Error Metrics on L_2 Values

The plots are consistent with the related error metrics, also in this framework the case **lambda = 0.001** is the best one since its metrics are the lowest among them all.

Elastic Net Regularization: The Elastic Net regularization can be written through the following formula:

$$L(\underline{w}) = \min_{\Omega} \sum_{n=1}^{N} (f_\Omega(x_n) - y_n)^2 + \overbrace{\sum_{j=1}^{\#\Omega} |w_j|}^{\text{L1 norm}} + \gamma \overbrace{\sum_{j=1}^{\#\Omega} w_j^2}^{\text{L2 norm}}$$

The Elastic Net regularization is a middle ground between Ridge and Lasso since combine both of them in order to penalize the number of features in the model. As shown in the formula, there are two distinct penalization coefficients that can modify in different ways the choice of the weights.

However, the Lasso regularization is more prevalent with respect to the Ridge one. It can be verified in the **lambda = 0.1** case's plot where the underfitting of Lasso is, once again, found inside the picture.

Both the plots and the metrics below agrees in the identification of the best model as the **lambda = 0.001** case. It's slightly overfitted but in comparison of the other models, it's the best one.

Figure 13: Case $\lambda = 10^{-1}$

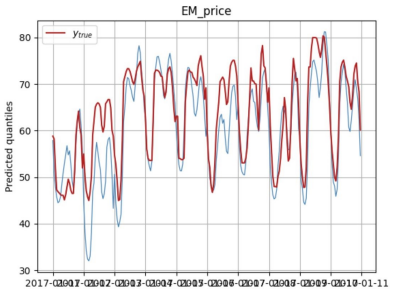

Figure 14: Case $\lambda = 10^{-3}$

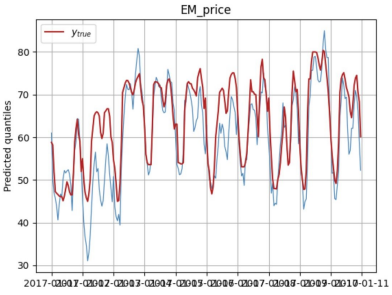

Figure 15: Case $\lambda = 10^{-5}$

	$\lambda, \gamma = 10^{-1}$	$\lambda, \gamma = 10^{-3}$	$\lambda, \gamma = 10^{-5}$
RMSE	20.77	5.96	6.28
MAE	18.92	4.71	4.84
sMAPE	33.74	8.04	8.29

Table 3: Error Metrics on $L_1 L_2$ Values

5 Impact of Metrics

In the assignment we were asked to implement evaluation metrics. These indicators are crucial to quantify the prediction accuracy and for this reason they can give us insights on the hyperparameters setting.

We report the formulas of the implemented indicators, where y_n are the values observed and $\hat{y}_n$ are the predictions:

$$RMSE = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2)} \qquad MAE = \frac{1}{N} \sum_{n=1}^{N} |y_n - \hat{y}_n|$$

$$sMAPE = \frac{1}{N} \sum_{n=1}^{N} \frac{|y_n - \hat{y}_n|}{0.5(|y_n| + |\hat{y}_n|)}$$

In the previous sections of the report we use the average metrics, which are computed using the whole prediction vector, however we were also asked to compute the hourly metrics. These are computed using the estimators above considering separately each hour along the 10 days.

Since we would have 24 errors for 3 metrics for each choice of hyperparameters, we have decided to report only the hourly RMSE for the choice of $\lambda = 0.001$ and $\gamma = 0.001$. For the other indicators we provide the Python code.

Time	00-01	01-02	02-03	03-04	04-05	05-06	06-07	07-08	08-09	09-10	10-11	11-12
L_1	4.30	6.32	5.31	4.88	4.63	5.81	6.34	5.88	5.55	5.38	5.41	5.62
L_2	5.12	6.20	5.36	4.27	4.55	5.81	7.05	6.72	5.40	5.81	5.20	5.27
L_1L_2	4.58	6.47	5.66	4.74	4.77	5.64	6.53	6.10	5.68	5.38	5.75	5.13
Time	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21	21-22	22-23	23-24
L_1	5.69	6.28	7.36	7.22	6.45	7.06	5.99	5.09	4.67	4.75	5.06	7.32
L_2	6.73	7.08	7.55	6.25	5.39	6.43	6.60	5.41	5.28	5.42	5.77	8.01
L_1L_2	5.92	6.24	7.58	7.23	6.52	7.31	6.07	5.31	4.37	4.33	4.97	7.46

Table 4: Hourly evaluation RMSE

Below, it's possible to have a visual representation of the data presented above:

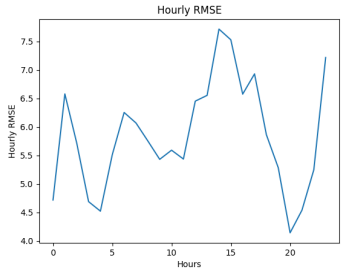

Figure 16: RMSE Lasso

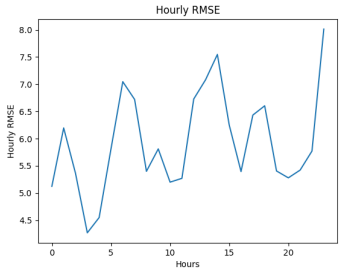

Figure 17: RMSE Ridge

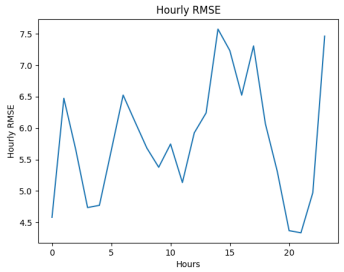

Figure 18: RMSE Elastic Net

As we can notice from the tables and graphs above, prediction accuracy seems to be different from hour to hour. Indeed the RMSE in all the three cases is significantly higher in the afternoon and has a peak around midnight, too. There could be different reasons which lead to less informative data during these hours, for example unforeseen spikes in energy demand. The computation of the metrics in the previous section was done through the implementation of the following python codes.

5.1 Python codes:

```
def calculate_rmse(y_true, y_pred):
    """
    Calculate Root Mean Squared Error (RMSE)
    :param y_true: Array of true values
    :param y_pred: Array of predicted values
    :return: RMSE value
    """
    assert len(y_true) == len(y_pred), "Lengths must be equal."

    squared_diff = (y_true - y_pred) ** 2
    mean_squared_diff = numpy.mean(squared_diff)
    rmse = numpy.sqrt(mean_squared_diff)
    return rmse
```

```
def calculate_mae(y_true, y_pred):
    """
    Calculate Mean Absolute Error (MAE)
    :param y_true: Array of true values
    :param y_pred: Array of predicted values
    :return: MAE value
    """
    assert len(y_true) == len(y_pred), "Lengths must be equal."

    abs_diff = numpy.abs(y_true - y_pred)
    mae = numpy.mean(abs_diff)
    return mae
```

```
def calculate_smape(y_true, y_pred):
    """
    Calculate Symmetric Mean Absolute Percentage Error (sMAPE)
    :param y_true: Array of true values
    :param y_pred: Array of predicted values
    :return: sMAPE value
    """
    assert len(y_true) == len(y_pred), "Lengths must be equal."

    numerator = numpy.abs(y_true - y_pred)
    denominator = (numpy.abs(y_true) + numpy.abs(y_pred)) / 2

    smape = 100 * numpy.mean(numerator / denominator)
    return smape
```

Estimators code:

```
#Implementation hourly evalution
predictions = numpy.array(test_predictions.iloc[:, 0])
predictions = predictions.reshape(10, 24)
targets = numpy.array(test_predictions.iloc[:, 1])
targets = targets.reshape(10, 24)

RMSE = numpy.zeros((1, 24))
MAE = numpy.zeros((1, 24))
sMAPE = numpy.zeros((1, 24))

for i in range(24):
    y_true = targets[:, i].transpose()
    y_pred = predictions[:, i].transpose()
    RMSE[0, i] = calculate_rmse(y_true, y_pred)
    MAE[0, i] = calculate_mae(y_true, y_pred)
    sMAPE[0, i] = calculate_smape(y_true, y_pred)

RMSE = RMSE.squeeze()
MAE = MAE.squeeze()
sMAPE = sMAPE.squeeze()
```

6 Change of testing period:

In order to understand the behaviour of our estimate we have performed our computation in different seasons of the year. We report below the hourly RMSE, the comparison of prices and test prediction and the average RMSE. We've decided to maintain the best choice of parameters from the January settings, thus L_1 regularization with $\lambda = 10^{-3}$.

Looking at the following plots it's possible to notice that in July and September the model is quite accurate, reaching good approximations of the true values of the energy prices. Regarding the month of April, we could detect a bit of bias due to the underfitting of the model; consequently we should need to recalibrate the hyper-parameter λ in order to reduce the bias.

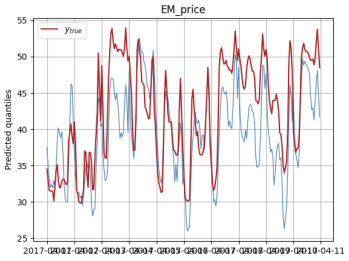

Figure 19: Price April

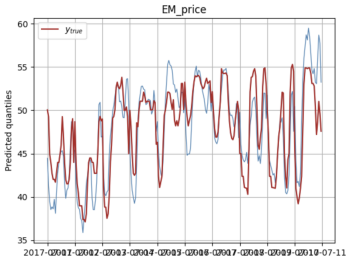

Figure 20: Price July

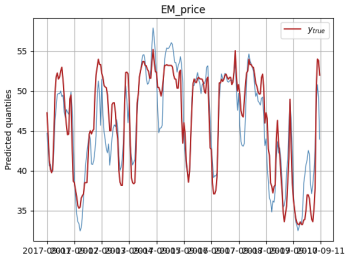

Figure 21: Price September

The previous interpretation of the prices is supported from the average metrics prediction in the table. Also in this case the April underfitting is showed by the high values of the error metrics.

	April	July	September
RMSE	5.06	2.80	2.89
MAE	4.25	2.22	2.24
sMAPE	10.30%	4.70%	5.03%

Table 5: Error Metrics changing test set

From the hourly RMSE plot we notice that even the prediction error changes from hour to hour, as mentioned in section 5, the hours during the day which have the biggest errors could vary in different season. However, the most difficult part to be predicted are generally midday, late evening and night.

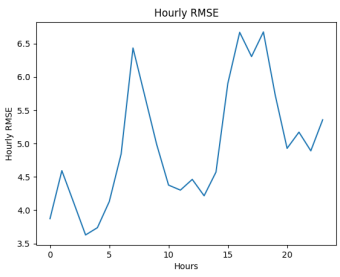

Figure 22: RMSE April

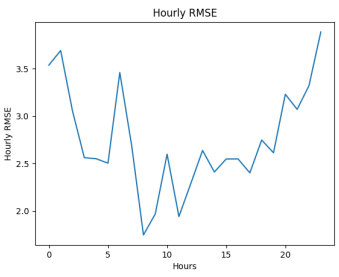

Figure 23: RMSE July

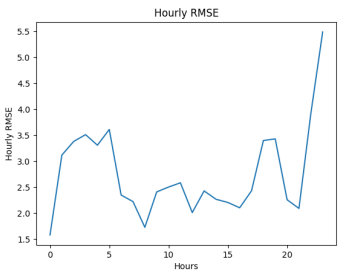

Figure 24: RMSE September