TIMESERIES RNN PREDICTIVE MODEL

After having accomplished the main goal of our project, that was to study how much energy and money the UAB will save by installing more solar panels; we thought that we could do something more to really apply Artificial Intelligence to the data that you gave us. If you remember the data analysis that we did, as we had data from the Covid period, we studied if during this period we saved energy or not. Finally, we demonstrated that even if it is a simple solution, if we want to save energy, one option is to do more activities online. One of the given datasets was the electric consumption which was ordered by a timeseries with all the consumptions of each faculty at every hour of the year. So, we thought that we could construct a new dataset with the energy consumption as target to predict, with features extracted from other datasets (occupation, classroom information…) and with new features created by us. Finally, once the prediction is done the idea is to set a reasonable and high threshold and if the predicted consumption exceeds the threshold, the activities will be done at the corresponding hour (or day, because we will probably do a groupby to have the consumption per days) online.

DATASET CREATION (dataset_creation.py)

From each faculty we were given some data distributed in different datasets. If you remember we had datasets of energy consumption by hours, occupation, classroom information and gas consumption by days. The objective is to construct a new dataset with the same shape as the energy consumption by hours as it will be our target, and adding some features that can help to predict the target. The timeseries energy consumption has all the consumptions by hours from 2018 to 2022, so we have 5 years of data. It is important to say that we only generated these new datasets for the faculties of science and bioscience, communication and Sabadell as were from the only ones that we had the complete data we needed. The datasets we used to build the new one are the hourly energy consumption, the occupation and the classroom information. To merge these datasets and obtain features corresponding to each hour of consumption, what we have done is looking at the occupation dataset (also distributed by hours) which activities are being done at each hour. Then from these activities we keep the classroom where they are being done, we go to the classroom info dataset and we extract important features as the ‘Superficie’, ‘Metres per gestor’ and ‘Metres’. By this way at each hour, we will have the consumption and the space (‘Metres’) that are being used to do the corresponding activities at the give hour.

Furthermore, we have manually created some new temporal features that we considered that can be helpful for the consumption prediction. These new features are ‘Weekday’ which gives a value from 1 to 7 depending on the day of the week we are, ‘Month’ which gives a value from 1 to 12 depending on the month we are, ‘Hour_2’ which gives a value from 0 to 23 depending on the hour we are and ‘Covid’ which gives value 1 if we are inside a covid period (we considered as covid periods the state of alarm periods in Spain) and 0 if not.

Why we need lagged variables?

THIS IS SOMETHING THAT YOU (TEACHERS) ASKED IN THE PRESENTATION AND I TRIED TO EXPLAIN IT, BUT WITH THE NERVOUS AND THE DIFFICULTY OF THE TOPIC, WASN’T CLEARLY EXPLAINED. SO, I HOPE THAT WITH THE FOLLOWING EXPLANATION YOU CAN UNDERSTAND IT BETTER.

By the way we did our model we need to create some lagged variables in order to give to the model information about the past. **But you will probably ask, why we need lagged variables if**

the RNN by its own take's information about past observations (past occurrences in time)? This is a very special problem because as you know we are thinking on how to do this model to be used by the university. So, we think that the model is not useful if to predict the next hour consumption we need the information about the previous hour consumption because the university will not probably have this information immediately and we will not have any margin of time (the same hour) to communicate to the students in case we need to do online class. So, what we really need is for example with the information of consumption of 2, 3, 4 or whatever days before, to be able to run the model and decide which hours or days we will have online lectures. For example, we are on Monday and we want be able to predict the consumption of Tuesday (if we let a margin of 1 day), of Wednesday (if we let a margin of 2 days) … What we really don't want is a model that needs the consumption of the hour before to calculate the consumption of the next hour, because the model will not be useful for the university as it needs some margin of time to tell the students if they have to do online lectures. So, the problem comes down to that in the creation of our tensors we cannot set this margin of x days to start looking back. We can decide how many observations we want to look back but not FROM WHEN we want to start looking back. For example, we can define hour tensor in the dataset as `self.X = torch.reshape(X, (X.shape[0], 2, X.shape[1]))` and because we have defined '2' in the middle of the tensors as observations to lookback, it will ONLY extract info from the previous observation (in our case will be the previous hour). But as we have said, we can not set a range to decide when to start looking back (the tensor do not give this option). So, the solution consists of adding the lagged variables. So, what we will do is looking only to the observation we want to predict, by setting 1 in the tensor instead of 2 as in the last example, `self.X = torch.reshape(X, (X.shape[0], 1, X.shape[1]))` and in the same observation we will have the lagged features that will give us the information about the past trying to simulate what the classical RNN does, but with this margin of x days. To clarify, what this margin of x days really does, is that sets the initial observation in the timeseries data from which we can start creating these lagged features (from the hour/day we can start to extract past information). And it is important to say that by looking to the same observation (1 in the tensor) someone could think that we are doing kind of cheating by looking to the current value we want to predict, but NOT, because we have the data divided in X_train and y_train and the lagged variables will be added to the X_train where the target feature is not present.

Creation of the lagged variables:

The idea of the lagged variables is to give the model the information about the target from x days before. In the function 'lagg_data()' contained in the 'utils.py' we can define from how many days before we want to create these lagged variables. In our case we have set 2 days before, what means that for example if I am on Monday, I can predict the consumption of Wednesday (2 days of difference). We know that the model will work better by setting only 1 day before, but we need at least one day to communicate to the students that for example on Wednesday we will have online class. Then we have defined some lags values (24x7 and 24x30) for once we are 2 days back extract some important statistics (mean, max, min and std) from 168 (7 days, 1 week) and 720 (30 days, 1 month) observations before (hours before). These is done to give to the model information about the tendency of the data during the last hours/days/weeks/months. For example, if we go back to the situation we have described before about the prediction of 'Wednesday' being at 'Monday', the statistic lagged variables will give information about the tendency of the consumption in terms of statistics (mean, std, min and max) from 1 week before Monday (168 lag) and 1 month before Monday (720 lag). Then we also have a lagged variable called 'last' that takes the exact value 7 days before (1

week before) about the ‘consumption’, ‘metres’ and ‘Superficie gestor’ to have a measure of what happened at the same hour, the same day but one week before. For example, if we are at Monday at 8:00 hour, we will have a lagged variable last that tell us what was the ‘consumption’, ‘metres’ and ‘Superficie gestor’ the last Monday at 8:00h. Once we have the dataset created, we can start implementing the model

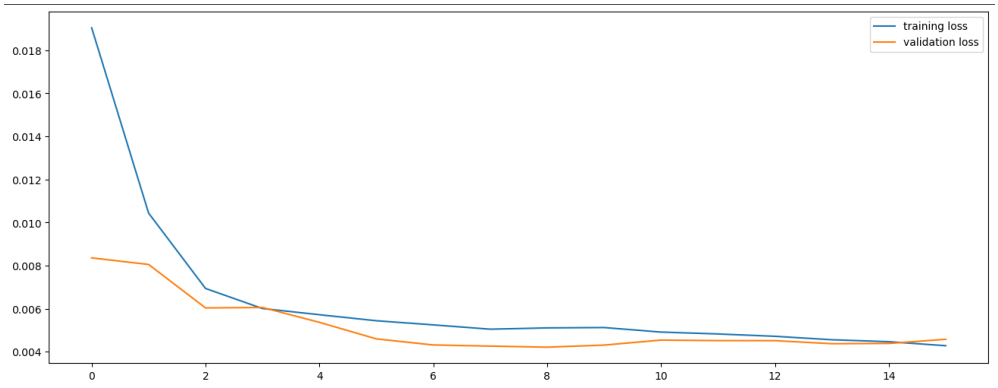
Model

First it is important to say that we only have trained the model for the faculty of science and bio science, but the same model could be applied for all the faculties. The first we did was splitting the data in train, validation and test sets and removing some non-needed features like the ‘Date’. In the split we have kept from 2018 to 2020 as the train set (60%), 2021 as the validation set (20%) and 2022 as the test set (20%). Moreover, the next step we did was to standardise the data. Then we created the data loaders, and we defined the model where we used an LSTM with two fully connected networks after to add complexity. We also applied dropout to the model, and these are the hyperparameters used:

```
# Hyperparameters
input_size = X_train.shape[1] # 18
hidden_size = 186
num_layers = 3
output_size = 1
num_epochs = 100
batch_size_train = 256
learning_rate = 0.001
batch_size_val_test = 256
drop_out=0.25
```

Then we also have proved with different optimizers and regularization techniques (weight decay, schedulers) and we finally used an Adam optimizer without any regularization technique. As we are in a regression problem, we used MSELoss as loss function.

So, we trained the model, and we reached a val_loss of 0.0042. It is important to say that we apply early stopping in the train_val() function to avoid overfitting and keeping the best loss validation result. Plot of validation and train losses:



	Predictions	Targets
2022-01-01 00:00:00	186.325592	222.900009
2022-01-01 01:00:00	186.431686	230.360016
2022-01-01 02:00:00	197.039825	231.690002
2022-01-01 03:00:00	196.497940	229.649994
2022-01-01 04:00:00	202.506729	232.099991
...	...	...
2022-12-31 19:00:00	235.742264	212.000015
2022-12-31 20:00:00	231.930725	210.000000
2022-12-31 21:00:00	228.676865	208.000015
2022-12-31 22:00:00	222.508301	206.000000
2022-12-31 23:00:00	220.532639	208.000015

8760 rows × 2 columns

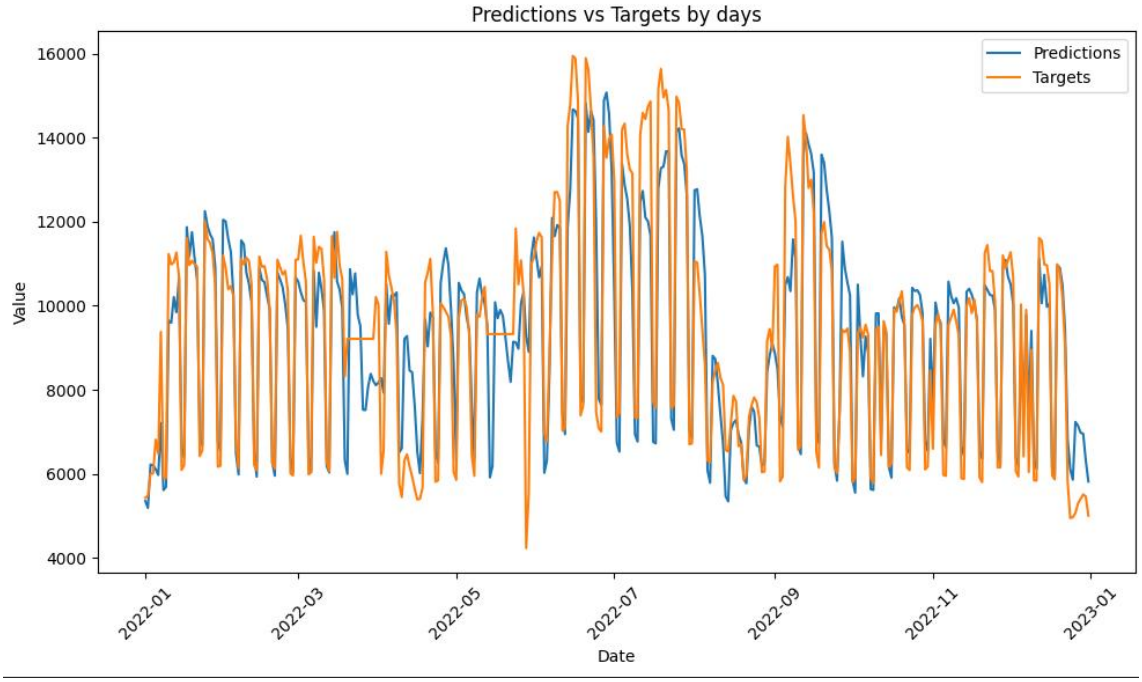
Afterwards, we calculated the test loss, and we had a 0.0037. At this point they seem to be quite good predictions but remember that the data is standardised, and we need to descale it to really evaluate if the model has done it well or not. So, once we denormalized the data, we calculated the descaled MSE loss, and we obtained a loss of 80.1788 kWh in each prediction. At first it seems to be a lot but if we look to some predictions, it seems that is because there are some outliers because in some cases it seems to work well.

	Predictions	Targets
2022-01-01	4762.638672	5424.800293
2022-01-02	4552.405273	5479.760254
2022-01-03	5656.481445	6003.550293
2022-01-04	5654.471680	6005.900391
2022-01-05	5563.232910	6809.430176
...	...	...
2022-12-27	7587.180176	5290.120117
2022-12-28	7392.260742	5397.340332
2022-12-29	7362.270508	5507.720215
2022-12-30	6599.865234	5456.890137
2022-12-31	5960.523438	4999.990234

365 rows × 2 columns

As you can see here, we have a data frame of our predictions and the real values by each hour. We thought that to use our model and to better visualize it we should group the consumption by days. So, we did the same dataset but grouped by days.

Here you have a plot by days with the comparison between our predictions and the true values:



In the plot we can see how even if there some values that are not well predicted most of them seem to be at least okey. We can see how months of June, July and August are the ones with higher predicted and true consumption probably because of the air conditioning. Let's erase these months as during these months we will not be able to apply the model to do online activities as there aren't lectures during these months. During June it is true that there are exams but we do not consider the possibility of doing online evaluation so we can also erase June from the distribution as it is not useful. Also, we will remove September as the course starts at the middle of September and we think it is not appropriate to do online classes the first days of the course as it is interesting for students to let new people meet each other. The same happens with January, we will also erase it as we start lectures at the middle of the

Now that the dataset was changed, we also decided to calculate the loss again to see if by dropping these unnecessary months we have better loss. The Descaled MSE Loss of the

reduced dataset is 74.2049 kWh in each prediction, if we compared with the original loss (80.1788 kWh), we can say that it has been improved.

But really in our model we don't really care about the losses, we care about how good it does on classifying if day has a high consumption and we need to do online classes or not. So, let's evaluate this. The first we need to do is to set a threshold to classify as 1 (online class) if the predicted consumption of a day exceeds the threshold. To define our threshold, we have selected the distribution of consumptions of the validation set (2021) and from this set we have selected the percentile 0.73 as the threshold.

Evaluation

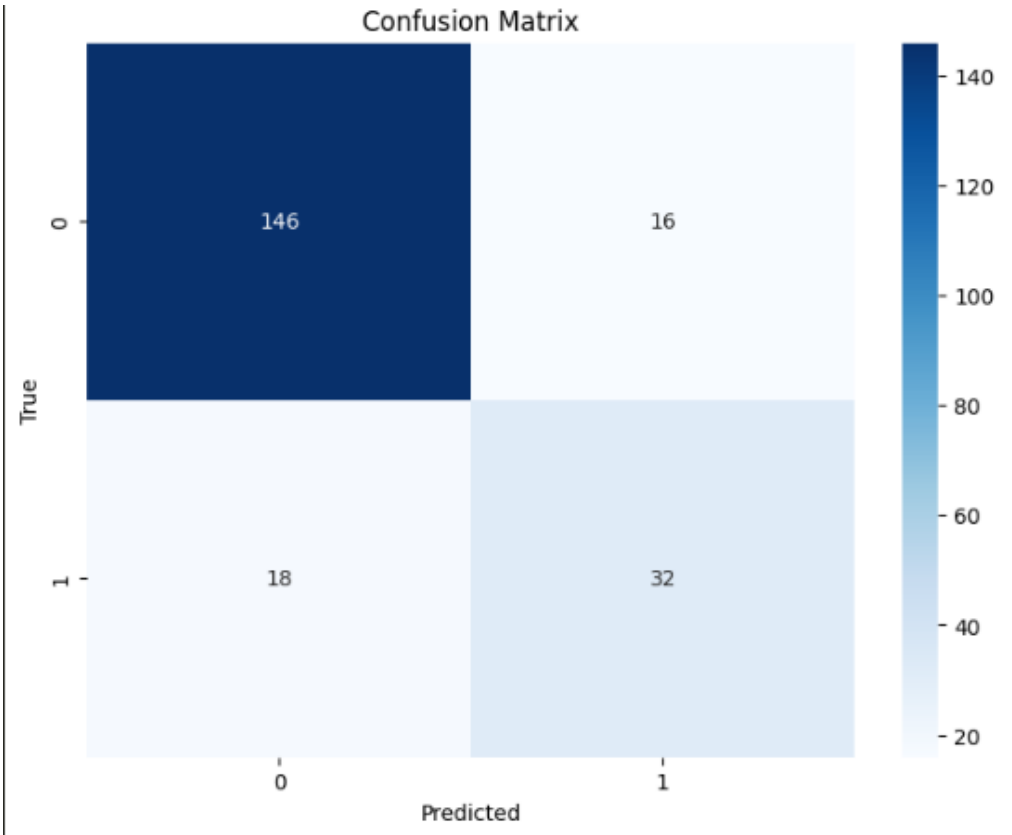
Finally, we converted it into a binary classification problem as we will give value 1 if the predictions and the targets have exceeded the threshold, and 0 otherwise. Now with the creation of this new variables we are ready to evaluate the model with some significative metrics.

	Predictions	Targets	prediction_class	target_class
2022-02-01	12341.981445	10908.580078	1	1
2022-02-02	11882.795898	10394.139648	1	0
2022-02-03	11530.743164	10469.269531	1	0
2022-02-04	10422.448242	10227.219727	0	0
2022-02-05	6530.919922	6761.990234	0	0
...	...	...	...	...
2022-12-27	7587.180176	5290.120117	0	0
2022-12-28	7392.260742	5397.340332	0	0
2022-12-29	7362.270508	5507.720215	0	0
2022-12-30	6599.865234	5456.890137	0	0
2022-12-31	5960.523438	4999.990234	0	0

212 rows × 4 columns

Metrics and Confusion Matrix:

```
Accuracy: 0.839622641509434
Recall: 0.64
Precision: 0.6666666666666666
F1 Score: 0.6530612244897959
```



Metrics for Each Class:

	precision	recall	f1-score	support
0	0.89	0.90	0.90	162
1	0.67	0.64	0.65	50
accuracy			0.84	212
macro avg	0.78	0.77	0.77	212
weighted avg	0.84	0.84	0.84	212