Deep learning workshop - assignment 2

Doron Laadan

Tal ben-senior

Part 1: Time series analysis

We chose to work on the Individual household electric power consumption from UCI.

which you can read more about here:
https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption

EDA:

The Household Power Consumption dataset is a multivariate time series dataset that describes the electricity consumption for a single household over four years. the date set has 2,075,259 samples, each of them are measurements gathered in a house located in Sceaux (7km of Paris, France) between December 2006 and November 2010 (47 months) each minute.

It is a multivariate series comprised of 7 variables (besides the date and time) as we can see

they are:

* global_active_power: The total active power consumed by the household (kilowatts).

* global_reactive_power: The total reactive power consumed by the household (kilowatts).

* voltage: Average voltage (volts).

* global_intensity: Average current intensity (amps).

* sub_metering_1: Active energy for kitchen (watt-hours of active energy).

* sub_metering_2: Active energy for laundry (watt-hours of active energy).

* sub_metering_3: Active energy for climate control systems (watt-hours of active energy).

the dataset is said to have missing values and from a quick look at it we can see they are sometime 'na' and sometime '?'. All the categories are of the same type.

Problem definition:

There are many ways to explore the household power consumption dataset.

Here we decided to use the data to explore a very specific question which is:

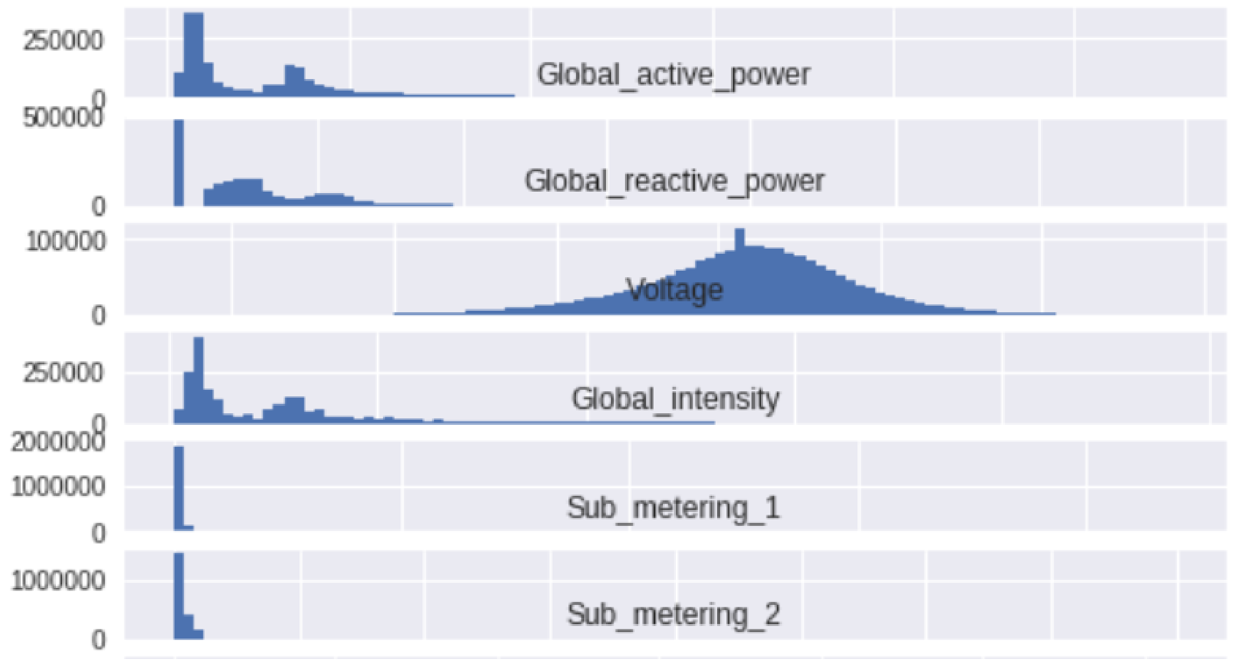
"Given recent power consumption, what is the expected power consumption for the week ahead?"

This requires that a predictive model forecast the total active power for each day over the next seven days.

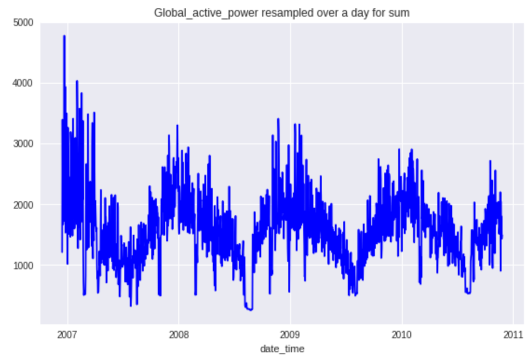
as we learned from the all mighty internet this framing of the problem is referred to as a multi-step time series forecasting problem, given the multiple forecast steps. So we are solving regression problem over several days.

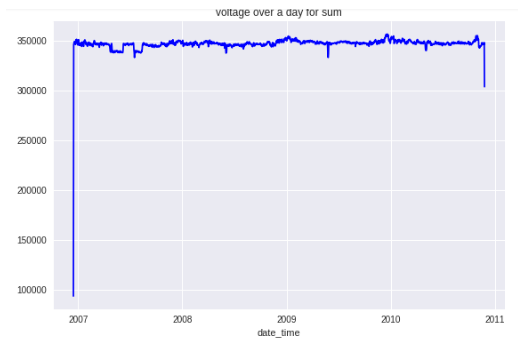
Some visualization to understand the data better:

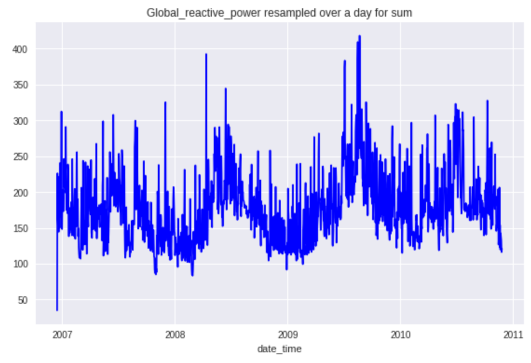
distributions of the data by histograms.

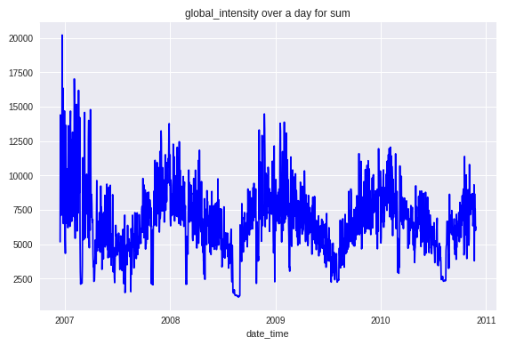


4 main features resample over a day

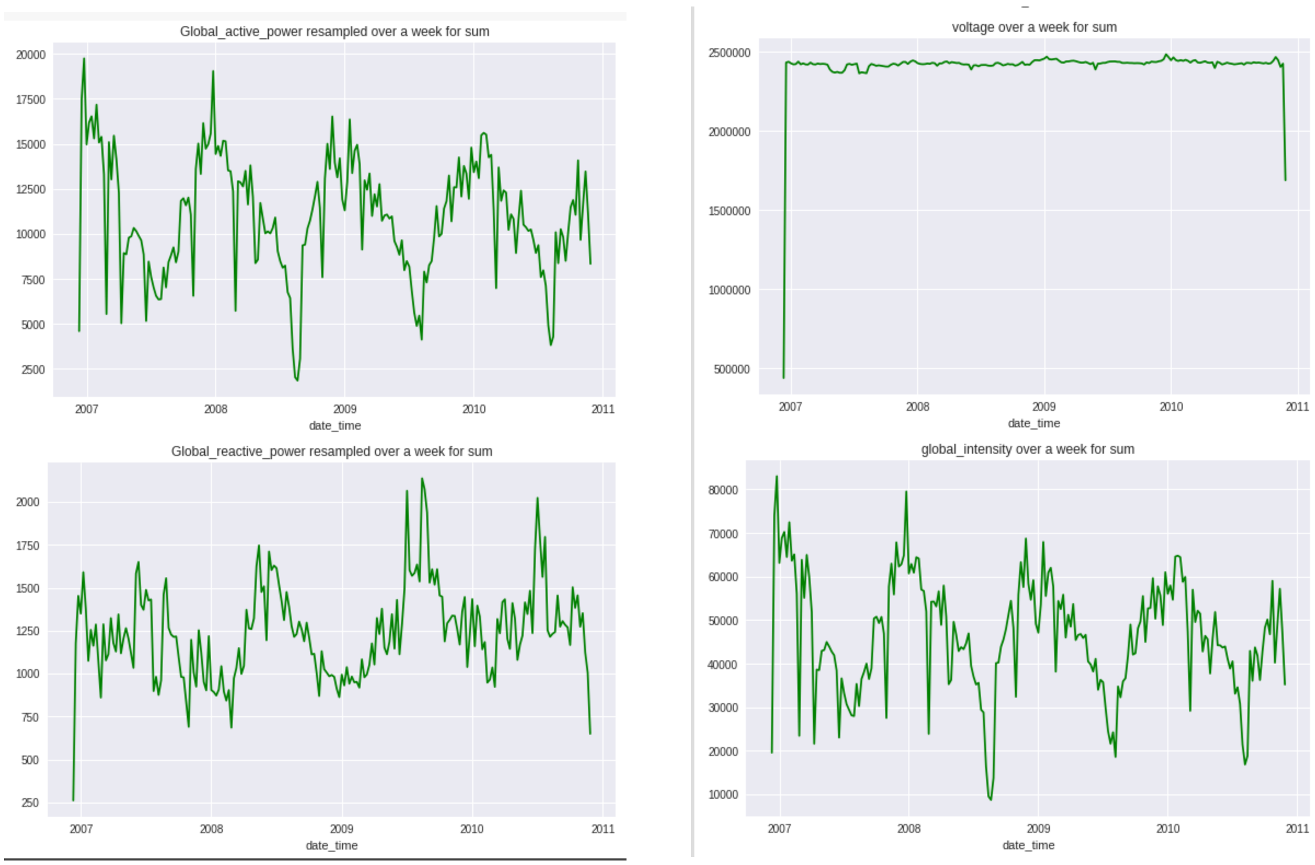






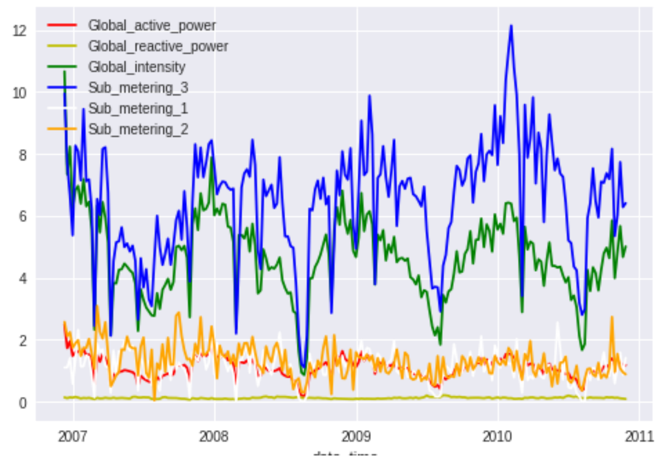


And over a week

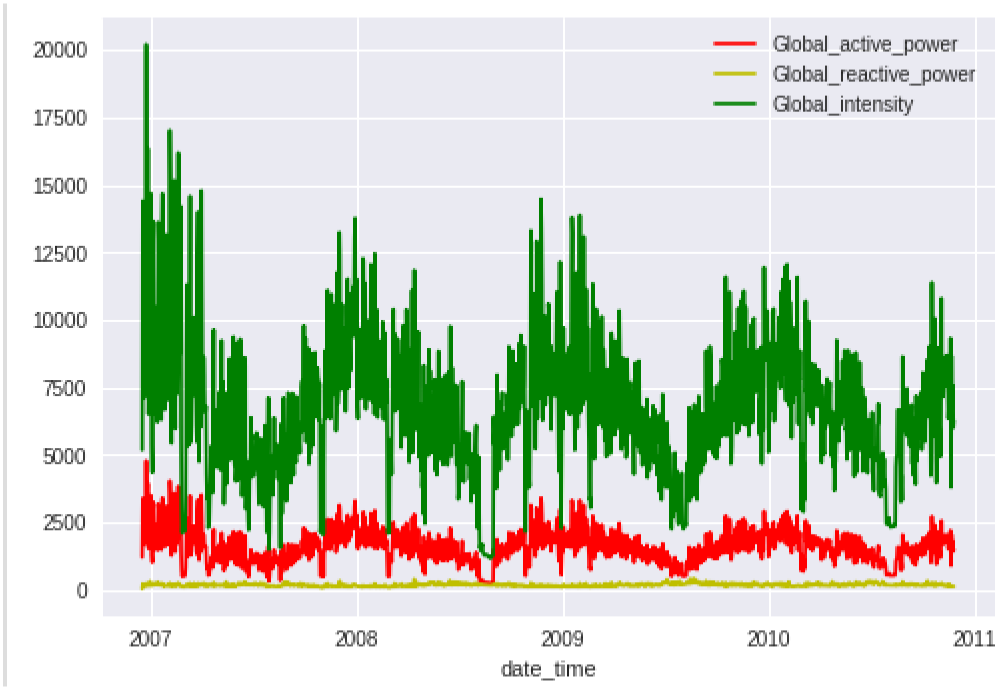


We can clearly see the Voltage acts different than the others. Also, it is measure in a different scale, so that might be a factor.

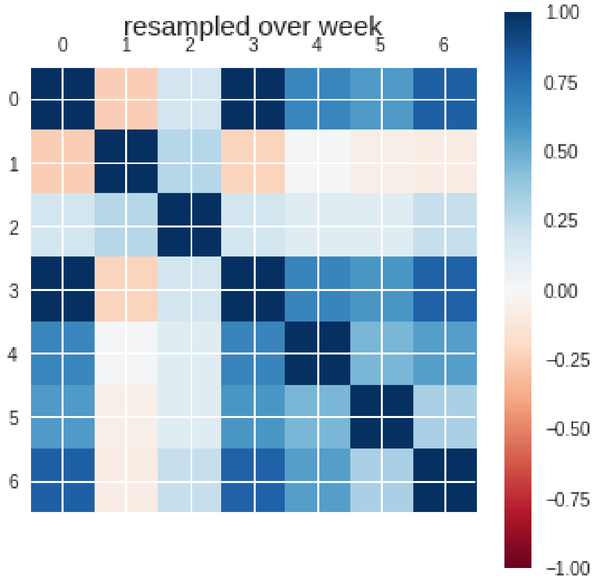
Now we want to see some of the features together on the same graph over a week

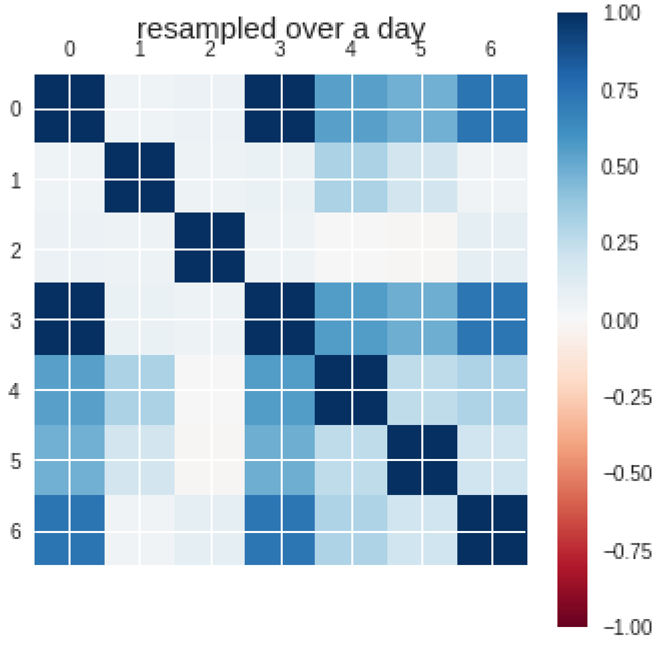


And the 3 main features over a day.



It's seems there is some correlation between the features. Let's check this out using a correlation matrix. First over a week and then over a day





As we can see, Voltage is indeed not correlating to any other feature on a week or daily bases And the sub-metric feature does correlate to one another and the active power which sounds reasonable.

Part 2:

Validation strategy:

Models will be evaluated using a scheme called walk-forward validation,
i.e. where a model is required to make a one-week prediction, then the actual data for that week is made available to the model so that it can be used as the basis for making a prediction on the subsequent week. This is both realistic for how the model may be used in practice and beneficial to the models allowing them to make use of the best available data. So to predict week 2 we use week 1, to predict week 3 we use week 1+2 and so on.

We will train on the first 3 years of data and use the last year as test.

Navie baseline:

Our naive solution: naive forecast, to forecast the week ahead, each day prediction value is the value of the same day at the previous week.

It is based on the idea that next week will be very similar to this week.

A forecast will be comprised of seven values, one for each day of the week ahead.

The units of the total power are kilowatts and it would be useful to have an error metric that was also in the same unit. we will use Root Mean Squared Error (RMSE).

Using the naive baseline we got a RMSE score of 480 for the entire week which is not bad for this kind of problem definition and is consistent with other result we saw online.

Machine learning:

Most predictive modeling algorithms will take some number of observations as input and predict a single output value. so, we can't use them directly to make a multi-step time series forecast like our problem.

What we will do is use a recursive approach for that:

This involves making a prediction for one-time step, taking the prediction, and feeding it into the model as an input in order to predict the subsequent time step.

To do so the first step is to convert the prepared training data in window format into a single univariate series. Next, the sequence of daily power needs to be transformed into inputs and outputs suitable for fitting a supervised learning problem.

The prediction will be some function of the total power consumed on prior days. There will always be a single output: the total power consumed on the next day.

The model will be fit on the true observations from prior time steps. We need to iterate through the sequence of daily features and split it into inputs and outputs. it's basically a sliding window.

To create predictions, we used the Lasso ml algorithm, and did a very similar process to what we did in the naive approach, that is walk forward validation. When the sliding window is the size of the days to look back at times the number of features we want to see in each day.

We ran the model several times with different features:

1. the active_power feature, RMSE: 397
2. the active and reactive features: RMSE: 2329
3. the active and reactive and intensity features RMSE: 2037

We can see that when using only the active_power feature to predict as we did in the naive model we get a better result than the naive model.

We can also see that when introducing more features to try and predict with them, the RMSE loss increases, which means, adding more features except 'active power' harm the predictions loss.

ANN model:

In this part we will use LSTM to solve the same problem as before. Recurrent neural networks, or RNNs, are specifically designed to work, learn, and predict sequence data.

As we saw in class, LSTM model expects data to have the shape:

[samples, timesteps, features]

Sample will be comprised of seven-time steps with number of features for the seven days of total daily power consumed.

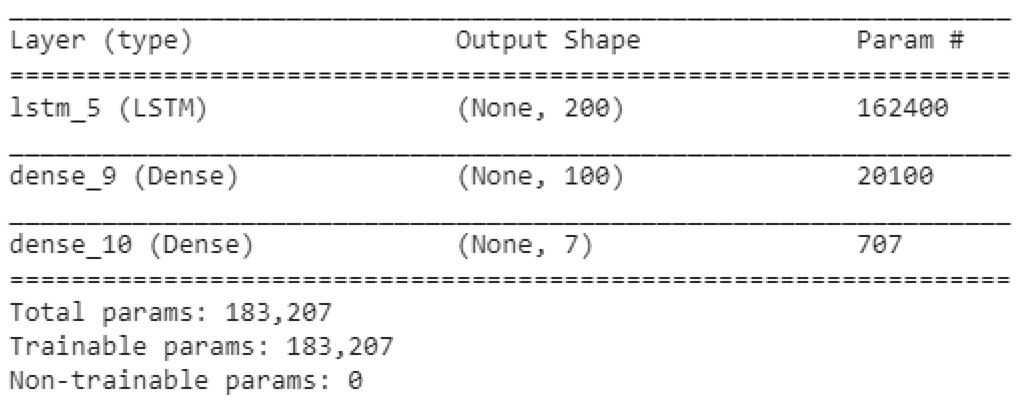
The training dataset has 159 weeks of data, so the shape of the training dataset would be:

[159, 7, number of features]

The data in this format would use the prior standard week to predict the next standard week. A problem is that 159 instances is not a lot. A way to create a lot more training data is to change the problem during training to predict the next seven days given the prior seven days, regardless of the standard week. This only impacts the training data, and the test problem remains the same: predict the daily power consumption for the next standard week given the prior standard week.

This will require flatting of the data, next we train and fit our model using n_samples - the days to look back each time and a list of features to use. using the above function to transform the problem into a supervised robust problem

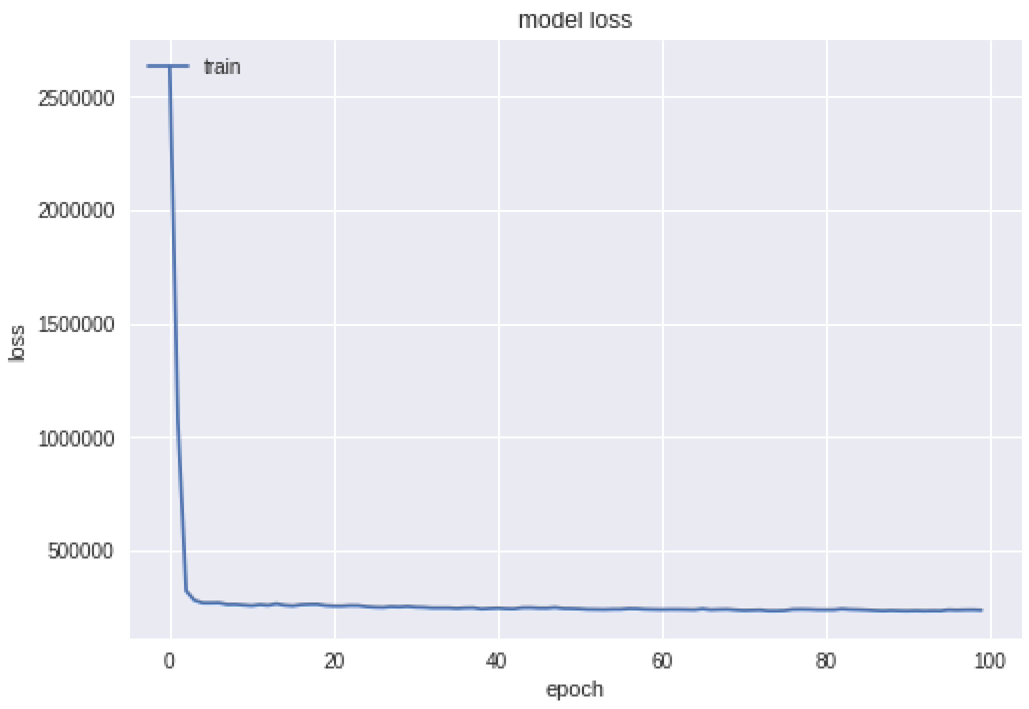
Our basic model is:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_5 (LSTM)                (None, 200)               162400    
_________________________________________________________________
dense_9 (Dense)              (None, 100)               20100     
_________________________________________________________________
dense_10 (Dense)             (None, 7)                 707       
=================================================================
Total params: 183,207
Trainable params: 183,207
Non-trainable params: 0
```

.

And after training for 100 epochs the loss is as follows



Next, we use walk forward validation to test our model and get a result of 402 RMSE while using only active power feature.

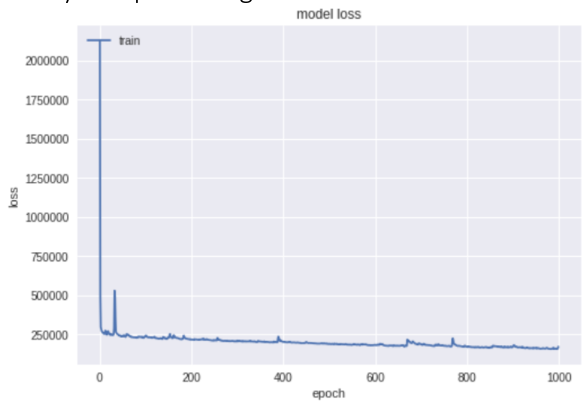
We can see the model trained only on the active power feature didn't do better than the ml approaches using the same features.

Some suggestion to reduce prediction loss:

1. Predict for the next week based on two weeks before.
2. Try combination of features.
3. More complex network and hyperparameter tunings.

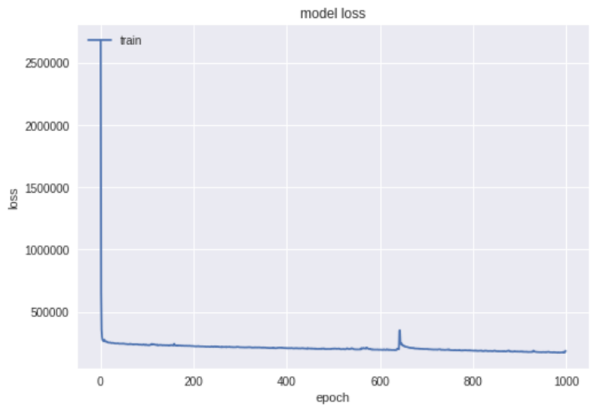
We chose to focus the 2 first suggestions and the results are:

1. 14 days for predicting next week:



The loss is higher on the test – 418, we think that increasing the epochs could have helped, looking at the training phase we could note that the loss is reducing over epochs, it may be hard to notice it through the graph loss due to the large loss at the first epochs.
Moreover, it seems that the loss tends to increase at some points for few epochs, rough estimation is that the gradient oscillates, we think that momentum optimizer with learning rate scheduler could have helped to guide the gradient better.

2. The combination of features we chose based on the correlation matrix above. As we can see 'active power' correlate best with feature 0 (correlate with himself – make sense) and feature 3, before we try any further features we would like to check if adding feature 3 help the model to reduce the loss:



The loss is far worse than before – 435.24, we would have anticipated that with more features – the model would approximate better the correct target value, or at least, ignore the feature which leads him to false predication. Though it does make sense that the model gets the best result using the target feature for prediction – especially at such kind of tasks.

Final remarks

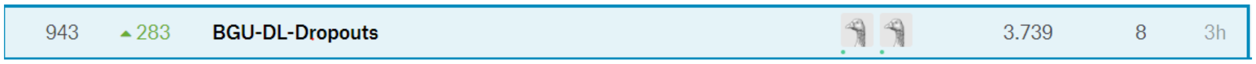
Working with time series is challenging. There is a lot of possibilities to define the problem and fit the data for it so there is no really ground truth to check. We could just as easily try to predict the next day given the previous one and get different result. You need to define your problem carefully and explore the data and fit the model accordingly.

ANN seems to work fine for time series analysis and a simple network gave close results to classic machine learning algorithm.

A lot of work was put into figuring out how to transform the problem into a supervised one. And how to implement the sliding window and the walk forward validation.

Part 3: Embedding:

We join the **the** elo **multi-target regression competition** and out team name is "BGU-DL-Dropouts. In the time of writing this report our place in the leader board is as follows (the notebook of this submission is included in the submission zip and would be explained briefly later).



943	▲283	BGU-DL-Dropouts	3.739	8	3h

b. At first we used a very naïve approach to get on the leader board and get a benchmark result for the data – we used random forest regressor with 100 estimators max depth 3 and random state 42, only on the data in the train and later we used naïve approach which we gave all the predications 0 (which put us in the 1000 place, right in the middle of the leader board somehow).

c. We used embedding to represent the connection between the original features in the train data to particular feature in the history transaction table. As we understood the point of the embedding in this exercise was to create new features that symbolized connection between categorical features.

The embedding we created is a combination of the categorical features in the train and the state_id categorical feature and another embedding which not appear in the notebook is relative to the amount feature, where the loss measure by categorical cross entropy and mse relatively.

d. Using only the embedding we created to predict the loyalty value did not gave good results. This might be because the embedding process we did wasn't good or because there is no real connection between the features which can be used in the embedding.

e. Using the originals features and the embeddings we crated to predict the loyalty value did not gave better results than before.

f. Doing this assignment and using embedding we understood better (we hope) the meaning of embedding and how to use it to understand connection between features in the data. We understood the need of choosing the right features and defining a proper task for embedding or else the results might be meaningless and cannot be used to present robust correlation between features. From the embedding we did we can see there is no clear connection to be understood from the 3 original features in the train data to the two features we examine in the history dataset.

g. We used feature extraction from the last model (using the embedding and the features) as input to random forest regressor which gave better result than the ANN model. This led us to believe for the elo dataset classic machine learning works better than DL models, we would assume the DL models should work fine as well, but in practice, we have seen that the model coverage to relatively high loss, and struggles to train further, regardless to optimization method we tried to use.

Submission:

To get our result in the leaderboard we used feature engineering and the lgb model. We created new features on the history and new transaction datasets using aggregation function and then merged the dataframes with the train and test datasets. We ran lgb first with its default values and later using grid search for several features on our local machine and looking at public kernels we tune the hyper parameter to get the result we submitted.

Final remarks:

It took a lot of time to understand how to use embeddings in this assignment and from talk with other in the class there are still different opinions of what we should have done.

The elo data set was very unclear and un organized which made the work much harder (which we guess would be the case in real world projects). One difficulty we had is to work with limited RAM memory relative to the high tables size (mainly history transaction), meanwhile the Collaboratory and Kaggle collapsed, we learned to use correctly GC, and Dask Dataframe – library which resemble to pandas saves the data on disk. And later we used implemented method in this one of the competitor's kernel which reduce memory for features that can be used with 'lighter' data types.

Up until now we thought that embedding is only used to represent a data in smaller more condensed way but after this work we understand that it can be use to represents new unseen connection in the data and that there is a lot of work that can be done to create a good and informative embedding that would improve model performances.

Unlike in previous works when it was very clear how to show the improvements in the model in terms of loss and accuracy it wasn't that clear in the case of the embedding and we should asses if it was good other than the final results on the test set.