Prediction of electric motor temperatures using machine learning models

Massine AKILAL
massine.akilal@gmail.com

Contents

1 Introduction

The presented project has been made to complete an online data science course. The course is delivered by “Machine Leaning Graz” [1]. The objective is to apply the leaned models, methods and libraries.

In order to perform this task, a dataset have been selected on Kaggle.

1.1 Dataset

The dataset has been found on Kaggle [2]. It has been previously used as a support data for in two studies where models are made for predicting the output temperatures that we try to predict [3] [4]

The dataset contains the recording of 12 physical variables, 998070 recordings for each variable. The samples are taken with 2Hz sampling frequency. The recordings are split in different batches representing testing sessions.

1.2 Background

A permanent magnet synchronous motor is composed by the different elements shown in the picture bellow [5]. Measuring the temperatures of the different parts may be difficult on a commercial car and expensive (sensor cost, ECU, specific technologies…). Finding a way of modeling these values without a direct measure is an industrial challenge. Therefore, in the project, we focus on predicting the 4 temperature values: magnet temperature (noted pm in the model), winding temperature (noted winding in the model), tooth temperature (noted tooth in the model) and a the Stator yoke temperature (noted yoke in the model)

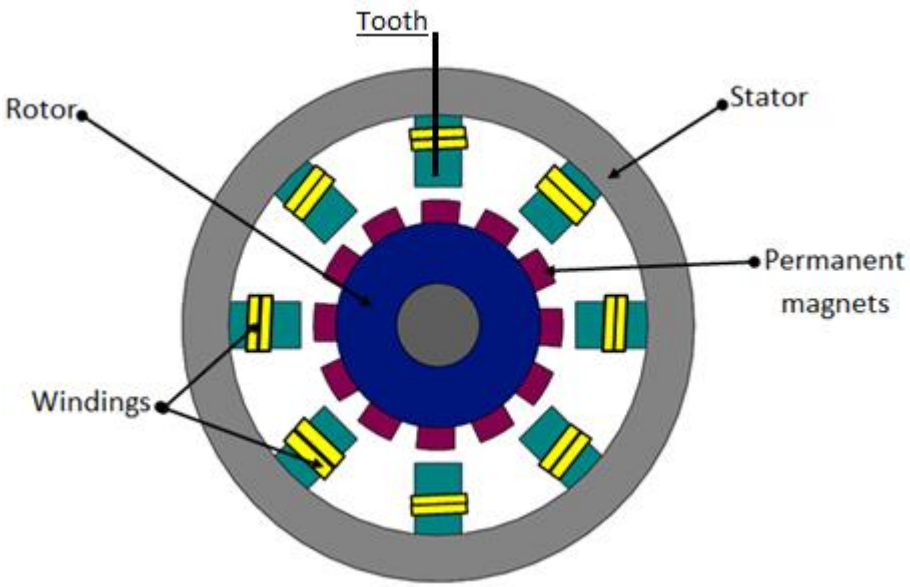

Figure 1 Permanent magnet synchronous motor [5]

1.3 Data selection

Among the 990000 lines available, we split the data into training data and testing data. Also, we split the inputs and the outputs (X_train, X_test, y_train, y_test). Among the available data, we ignore the torque information. The description explains that this value is not reliable due to measure process [2].

Table 1 Data split in this study

	Train (id!=20 and id!=65)	Test (id=20 and id=65)
Inputs	X_train	X_test
Temperatures	'ambient' and 'coolant'	
Voltage	'u_d' and 'u_q'	
Voltage	'i_d' and 'i_q'	
Engine RPM	'speed'	
Outputs	y_train	y_test
Temperatures	'pm', 'yoke', 'tooth', 'winding'	

We highlight the fact that we split the data info train and test by batches. It means we will tr to predict the values for batch 20 and 65 without showing the model any value coming from these batches. This is more difficult to predict but it is closer to the real situation. As a car manufacturer, you would like to predict these values on a real car based on test bench measurements.

1.4Scope of the study

In this work, we investigate the usage of different models in order to precisely predict the output temperatures. We will use a decision tree regressor, a random forest regressor and a neural network. The Mean Square Error (MSE) for each output variable will be used as a measure of the model performance. We will also use the function “score” to compute to performance over the 4 output variables. Basically, we will try to answer these following questions:

- How are these variables correlated?
- How do the three type of models perform on this regression problem?
- How do the hyperparameters influence the results?
- How can we reduce the noise in the predicted variable?
- How can we try to improve the model?

Also, the following topics are out of scope and will be note investigated even if they represent interesting features:

- The computation time and complexity of each model
- The comparison of the performance with other studies. In fact, this point would have been interesting as a benchmark, but the data have been modified by the owner to keep it confidential. Therefore, we don’t have a way of comparing the results to the previous studies. The study Motor Temperature Prediction with K-Nearest Neighbors and Convolutional Neural Network [6] would have been a good benchmark but the authors are predicting within test batch what we don’t do.

2 Data analysis

By looking at the data (figure 2) we see that the input variables are clearly discrete, while the output variables can be assumed continuous.

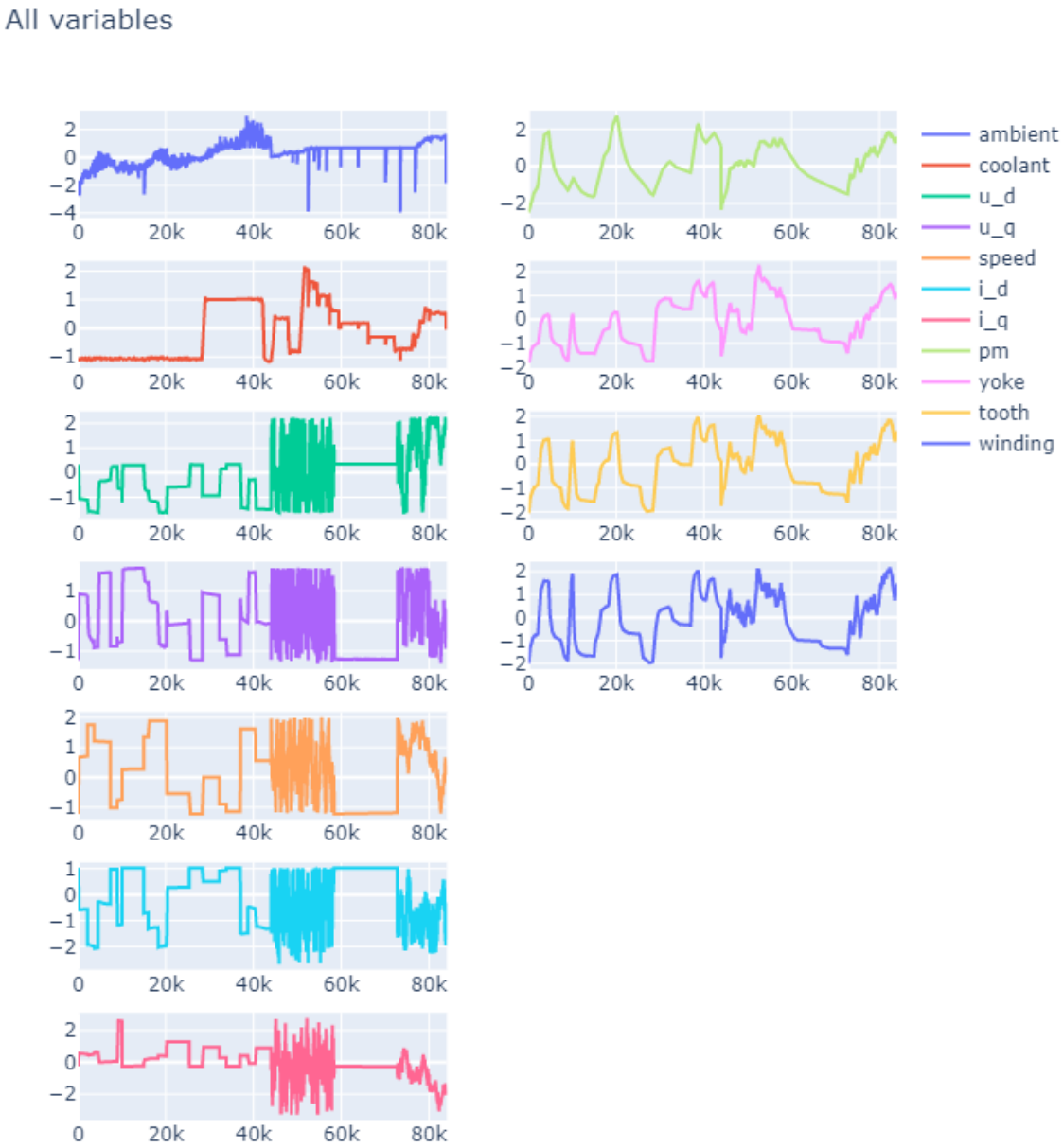

Figure 2 Representation of the variables in the test dataset. On the left, the input variables, on the right the output variables

We also notice that the correlations in the train and test datasets are similar. This comforts this choice of splitting the data. We notice also a high positive correlation between the output variables. But we don't see any high correlation (higher that 0.7 or lower than -0.7) between inputs and outputs. Only one exception is noticed. The correlation between Coolant temperature and Yoke temperature is 0.83. This result physically makes sense as the coolant is cooling the Yoke by convection.

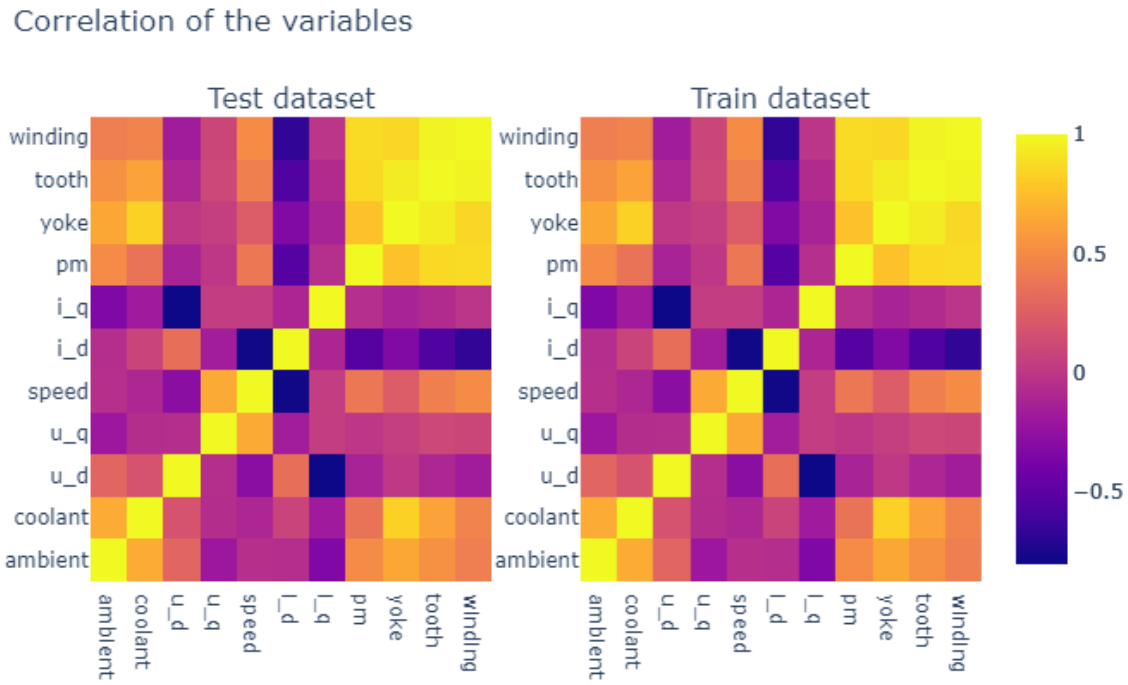

Figure 3 Representation of the correlation between variables in the test dataset and train dataset

3 Decision Tree

In this paragraph, we will implement a decision tree model to prediction the temperatures. This model may seem not adapted, but it will be used as a quick model to get a general idea of the performance.

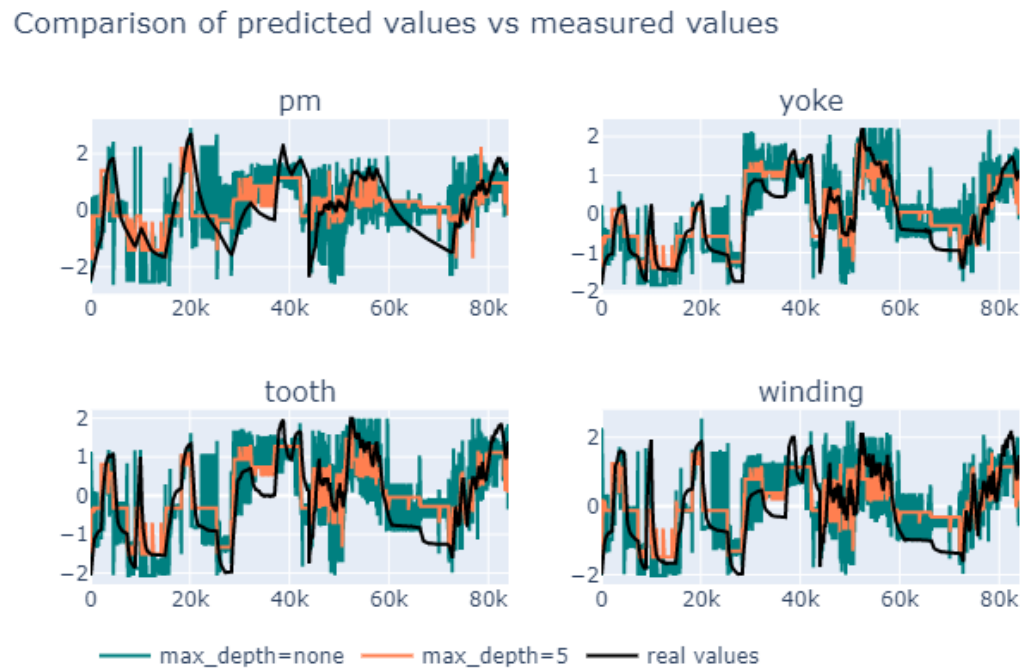

Figure 4 Predicted values with decision tree models: different parameters

We notice that the prediction is very noisy, with high variance. It cannot be a good prediction of the temperature in a real driving car. However, we might consider modifying the parameters in order to understand how this model changes. We will compute the model score against the tree depth for both a random split and a best split. We ran 14 different combinations of these parameters. We used the following maximum depths: 1, 2, 5, 10, 20, 30, 40. And we run each model with the best split solution and a random one. The results are represented in the graph bellow.

Table 2 Best results among the 14 tested versions of decision tree model

Depth	Splitter	Model score	MSE pm	MSE yoke	MSE tooth	MSE winding
5	Best	0.61	0.74	0.19	0.36	0.44
5	Random	0.62	0.71	0.23	0.34	0.41
10	Best	0.58	0.70	0.22	0.42	0.49
10	Random	0.64	0.72	0.18	0.31	0.38

For both random split and best split, the maximum reached score is between 0.60 and 0.64, and it is reached for a maximum depth between 5 and 10.

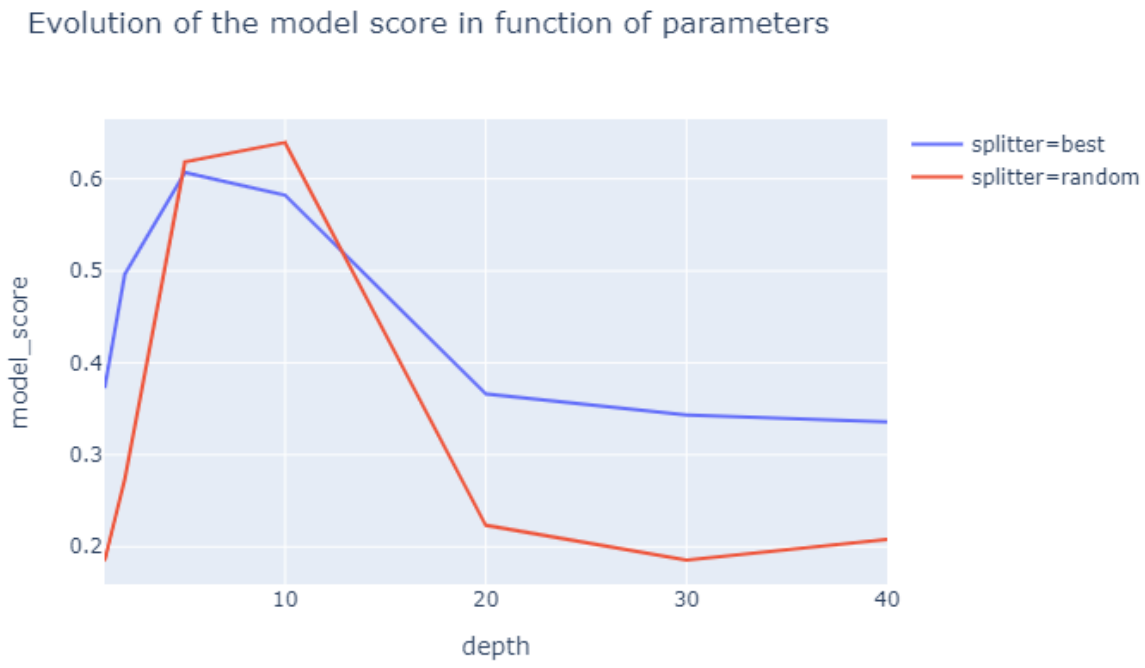

Figure 5 Measure of model score with different parameters

4 Random Forest

We try to improve the previous results by implementing a random forest model. By increasing the number of trees, data should be fit more accurately. Therefore, we investigate how the

number of trees influence the quality of the model. We realize that score is not a linear function of the trees. In our case, the best score reached is 0.56, achieved with a model containing 80 trees. This score is lower than the best one achieved by the decision tree regressions.

Table 3 Extraction of the results among the 12 tested versions of random forest model

Depth	Model score	MSE pm	MSE yoke	MSE tooth	MSE winding
20	0.54	0.86	0.24	0.43	0.52
80	0.56	0.79	0.23	0.43	0.51
100	0.53	0.85	0.25	0.44	0.52

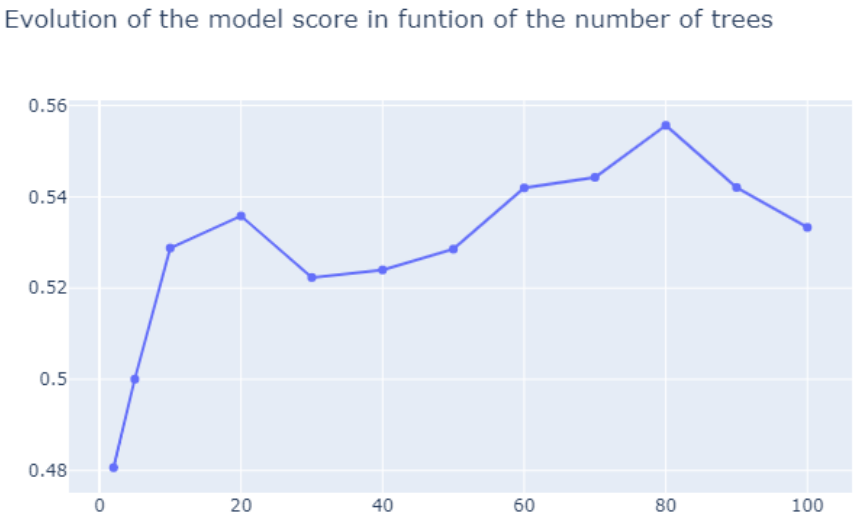

Figure 6 Measure of model score with different number of trees

5 Neural Networks

5.1 Network design and experiments

As the two precious models didn't reach a high score, we need to go for more complex models. We decide to use a Neural Network in order to model our variables. Like what we did previously, we will investigate the influence of the parameters on this model. However, Neural Networks are more complex than the previous models, they also use more parameters with unexpected effects on the model. We will consider the following hyperparameters:

- The Neural network architecture: we will use only feedforward networks [7], but we will vary the number of hidden layers and the number of neurons at each hidden layer.
- The activation functions: we will use the same activations for all the neurons. But we will test "tanh", "logistic" and "Relu" activation functions.
- Solver: we will use an "Adam" solver and a stochastic gradient descent
- The size of the training batches
- The learning rate
- A constant, an adaptative and an invscaling learning rate

Considering the number of hyperparameters, we cannot perform a full factorial experiment. There for, we fix a reference set of parameters (grey in the table below) and we vary only 1 parameter per experiment

Table 4 Organization of the experiment. The grey lines are the reference parameters. To investigate the influence of each parameters, we vary only 1 paramer at each step. In total, 28 experiments are made

<table>
<tr><th>Hidden layers</th><th>Activation</th><th>Solver</th><th>Batch</th><th>Learning</th><th>Learning type</th></tr>
<tr><td>(14, 14, 14, 12, 10, 8)</td><td rowspan="6">tanh</td><td rowspan="5">sgd</td><td rowspan="3">700</td><td rowspan="2">0.001</td><td rowspan="5">Constant</td></tr>
<tr><td>(20, 20, 20)</td></tr>
<tr><td>(20, 10, 20)</td></tr>
<tr><td>(10, 20, 20, 10)</td><td rowspan="2">0.0001 to 0.01</td></tr>
<tr><td>(11, 11, 11, 11, 11, 11)</td><td>100 to 5000</td></tr>
<tr><td>(5, 5, 5, 5, 5, 5, 5, 5)</td><td>Adam</td><td rowspan="3">700</td><td rowspan="3">0.001</td></tr>
<tr><td rowspan="3">(7, 7, 7, 6, 5, 4)</td><td>Relu</td><td rowspan="3">sgd</td><td>Invscaling</td></tr>
<tr><td>Logistic</td><td rowspan="2">Adaptive</td></tr>
<tr><td>tanh</td></tr>
</table>

The results of the 28 experiment allow us to understand the influence of each parameter. Therefore, we will use “adam” solver, the “tanh” activation function, 700 rows per batch, an adaptive learning rate with an initial value of 0.003 and 3 hidden layers with 20,10 and 20 neurons. This allow us to reach a score of 0.86 (34% higher than the best decision tree result)

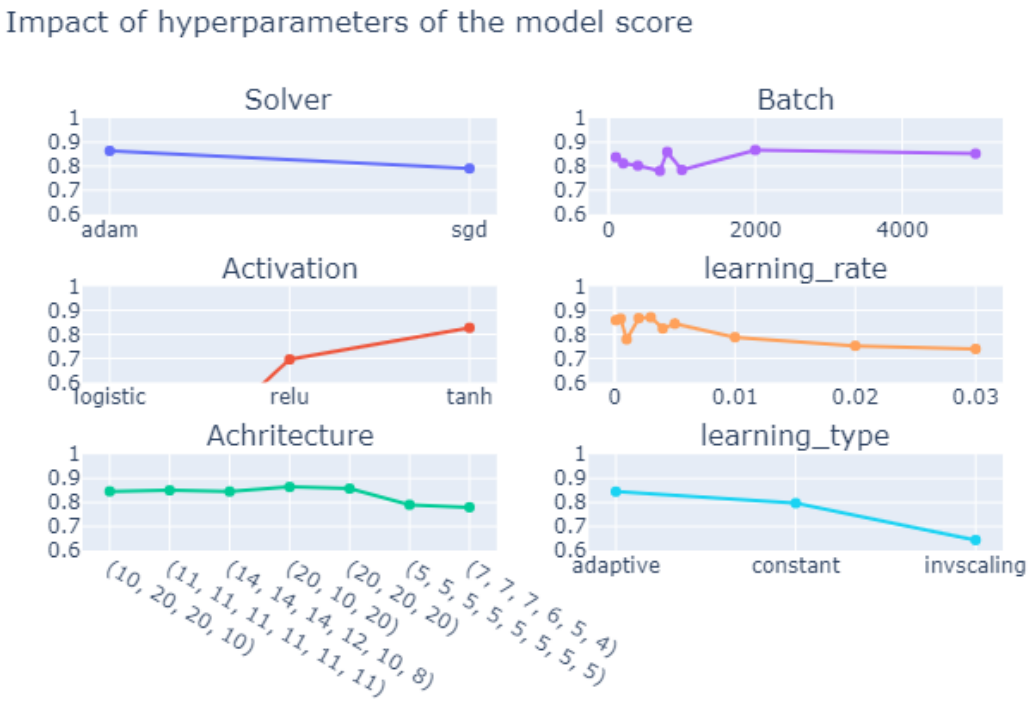

Figure 7 Impact of every hyperparameter of the model score

Table 5 Best result found

Best parameters found						Model score	MSE pm	MSE yoke	MSE tooth	MSE winding
20,10,20	tanh	Adam	700	0.003	Adaptive	0.86	0.24	0.08	0.13	0.16

We can see in the figure bellow that the model predicts less accurately during decreasing phases. Also, the model is sensitive to the noise.

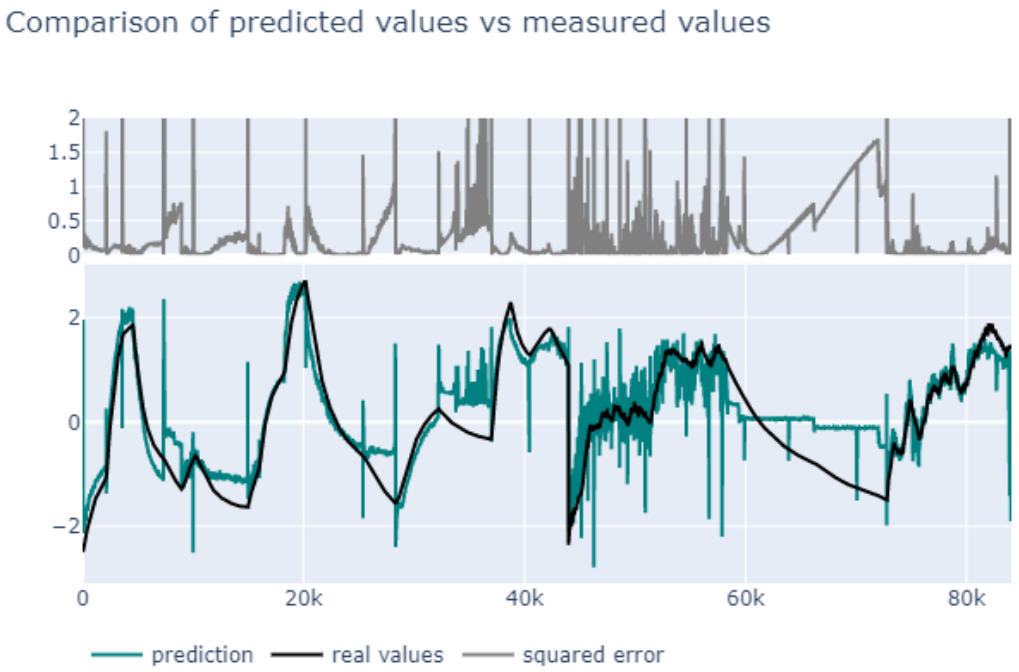

Figure 8 Display of the predicted values and measured values for the "pm" variable. The MSE for "pm" is 0.24 but locally, the errors explode

5.2Model validation

As we tuned the hyperparameters for some specific data, the network might be less performant if we change the training and testing datasets. In this part, we will randomly select the batch ids that we will give for testing and for training and we will look to the MSE and model score.

We can see in the figure 9, that the scores are very variables. In some tested cases, the scores are comparable to the ones we had (higher that 0.7) but in some cases, the scores drop under 0.6. Further investigations need to be done to understand the reason of this variability. We could start for example by looking at the data structure for the lowest results, id: 4, 30, 66, 43, 41, 48. This may give us a hint regarding the origin of the problem.

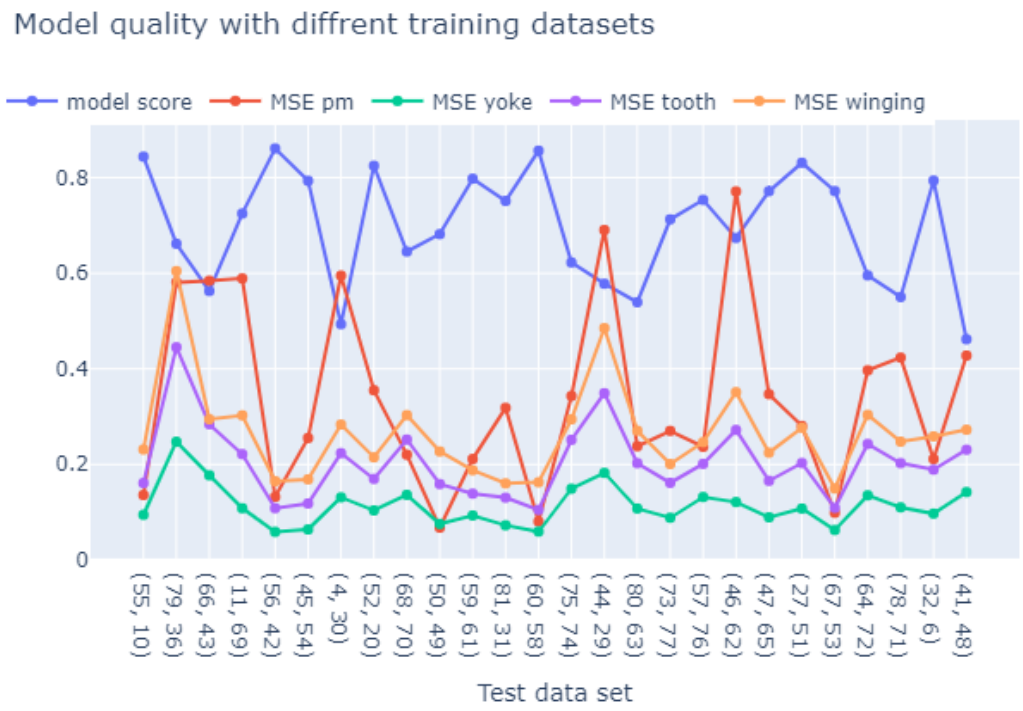

Figure 9 Model performance when we modify the batches we choose as input and output. x axis represents the test datasets. The train datasets are by default all other batches

5.3 Model improvement

In this paragraph, only a few ideas will be proposed to improve the model. These is no implementation done for this project for the moment.

We can notice in figure 8, that the increasing phases are well fitted, but the decreasing phases are not. Instead, these decreasing phases are predicted as constant functions. We can think about creating new input variables that will be an image of the current variables but with a certain continuity. For example, for a jump of +1 of the current input, we will build a function that transforms it to an exponential increasing variable. This new variable will be given to the network as a training variable. We might expect a better fit.

The second issue that we notice in figure 8 is the sudden high and low predictions, usually for only a few consecutive points. This might be solved by two ways. The prediction can be post processed to smooth the predictions by removing all the outliers and replacing them by the mean of previous and following predictions (but this cannot be implemented in a real time prediction solution). Or we modify the neural network and considering the prediction made at time t-1 for predicting the value at t. By setting a condition as the $|y(t)-y(t-1)|<e$. Where e is the maximum evolution between two consecutive points that we saw during the training.

6 Conclusion

In this study we have fist visualized the data and realized that the correlation between the variables is similar in training and testing datasets. We assumed then that splitting the data as we did was a correct use.

After applying different models, we could tune the hyperparameters to reach the best result for each parameter. The results could be summarized in the table below, and the neural network reached the best score.

Table 6 Summary of the best result of each tested model

Model	Model score	MSE pm	MSE yoke	MSE tooth	MSE winding
Decision Tree	0.64	0.72	0.18	0.31	0.38
Random Forest	0.56	0.79	0.23	0.43	0.51
Neural Network	0.86	0.24	0.08	0.13	0.16

Regarding the hyperparameters, there influent of the model quality have been proved by performing different tests. For all the three models, choosing the right parameters highly impacts the model quality. However, the model is not perfect as has several weaknesses that have been identified. We highlighted the noisy aspect of the prediction, the lake of fit for the decreasing phases and dependence of the score on the choice of training and testing datasets. As a next step of this project, these problems could be analyzed and solved. Indications and ideas have been discussed in paragraph 5.2 and 5.3

7 Bibliography

[1] A. Spataru and B. Andrusyak, "Machine Learning Graz," [Online]. Available: https://www.mlgraz.at/course.

[2] Kirgsn, "Electric Motor Temperature: 140 hrs recordings from a permanent magnet synchronous motor (PMSM)," University Paderborn, 2019. [Online]. Available: https://www.kaggle.com/wkirgsn/electric-motor-temperature. [Accessed 2020].

[3] W. Kirchgässner, O. Wallscheid and J. Böcke, Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors, 2019.

[4] W. Kirchgässner, O. Wallscheid and J. Böcke, Empirical Evaluation of Exponentially Weighted Moving Averages for Simple Linear Thermal Modeling of Permanent Magnet Synchronous Machines, 2019.

[5] 3 FEBRUARY 2017. [Online]. Available: https://alliedmarketresearch.wordpress.com/2017/02/03/permanent-magnet-synchronous-motor-pmsm-market-is-expected-to-garner-31-1-billion-globally-by-2022/.

[6] R. Le, K. Hu et A. Hu, «MotorTemperaturePredictionwithK-NearestNeighborsand ConvolutionalNeuralNetwor».

[7] A. Tch, «The mostly complete chart of Neural Networks, explained,» 4 August 2017. [En ligne]. Available: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464.