DTS303TC Big Data Security and Analytics Assessment 1

Yuqiang Li 1920183
Jiayuan Zhu 1931097
Zhiyi Zhao 1930878

October 19, 2022

1 Introduction

Global greenhouse effect and air pollution are intensifying, and worldwide countries have been committed to stringent fuel consumption and emission regulation. The automobile industry is also putting a massive effort into developing new energy vehicles as a response to these regulations (Al-Gabalawy, Elmetwaly, Younis, & I. Omar, 2022). As the most representative products of new energy vehicles, electric vehicles have been widely used in transportation (Denic, Cirkovic, Petković, Nesic, & Mehmedi, 2022). It is crucial to investigate the thermal performance of these vehicle products for its significant impacts on their efficiency. Therefore, this work extracts the heart of the most vital part of these vehicles, which is the electronic motor. To date, permanent magnet synchronous motor (PMSM) is commonly installed on electric vehicles (EVs) due to their high energy density and low self-discharge rate. (Kirchgässner, Wallscheid, & Böcker, 2021).

However, with the accumulation of highly intensive operations and the aging of a car's critical parts, Electric vehicles failures happen more frequently, introducing a challenging task to the vehicle's safety control systems (He, Wu, Nie, & Shi, 2022). According to the (Lee, Brown, & Ryan, 2017), the issues of motors account for more than half of the EV failures. Considered as the heart of EVs, PMSM is of vital importance to maintaining the car's performance. Its fuel emission and dynamic performance are greatly affected by permanent magnets (PMs), and the performance of demagnetization will occur with the overheating phenomenon of PM temperature (Nan et al., 2022). The problem of preventing motor from overheating meanwhile reaching its maximum viable power has been difficult for entrepreneurs to solve.

It is essential to detect the temperature of PMs in the rotor to prevent the performance degradation of PMSM. PMSM's operation above its thermal limits has terrible effects. In the rotor, namely the excessive temperature can lead to irreversible demonetization of the magnets, resulting in partial discharge, lifespan reduction, or even irreversible degradation. Traditionally, methods on the early detection of the PM temperature are highly dependent on human expert knowledge. It includes signal-processing-based. The signal-processing-based method is a direct detection method that requires the installation of several temperature sensors (Thosar, Patil, Singh, Thamke, & Gonge, 2020). However, the direct detection method requires temperature sensors and an extra data transformation device, which largely increases the cost of the system (Yang et al., 2021). Sometimes it is difficult to install sufficient sensors to measure the motor's temperature accurately. In addition, with the gradual expansion of the market, the PMSM motor market has transitioned from the original trucks and buses to private cars, which means that all instruments, including motors and sensors, need to be downsized and scaled up for mass production. Measuring these devices' temperatures is practically impossible with direct methods. Sensor-based temperature measurement is only possible for the outer-motor chassis as it is difficult to access the rotor because of its complicated structure. Companies have to invest more in R&D and maintenance (Gaona, Wallscheid, & Böcker, 2017).

Motivated by the contents aforementioned, this project's main aims and highlights are taking indirect measurements to calculate the PMSM temperature using Spark, a data-driven software framework (Nan et al., 2022). It requires expertized domain knowledge, and a module will be devoted to this topic below: 1) Analyze data using Spark, the large-scale data cluster. 2) Estimate the PMSM temperature based on Gradient-Boosting Trees (GBT). 3) Perform pre-processing on the original data to ensure appropriate data and feature representation. 4) Test a few machine learning models, namely Linear Regression (LR), Gradient-Boosting Tree (GBT), Decision Tree (DT), and Random Forest (RF).

1.1 Background

In the automotive industry, reducing the weight and volume of the drive is essential (Gaona et al., 2017). Thus, permanent magnet synchronous machines (PMSMs) are widely used the reliable operation, lightweight, and high efficiency (He et al., 2022). However, it is essential to monitor the machine temperature in real-time to maximize the utilization of the drive.

1.2 State-of-the-Art

Detecting methods can be classified into two general types: directed detection and indirect temperature estimation. Directed detection is a traditional method highly dependent on human expert knowledge. It is demonstrated by the fact that researchers assess the effects of thermal stress by measuring the components of a machine with sensors. To prevent the winding insulator from melting, sensors are needed to detect the temperature of the winding in many applications (Nan et al., 2022). The researchers determined the extent of demagnetization of PMSM by studying the increase in temperature. It is still used today in the national defense industry and everyday life in agricultural production (He et al., 2022). In recent years, temperature estimation has been widely studied as indirect temperature measurement. There are two main types: classical statistical learning-based and data-driven-based methods (Lee et al., 2017; Kirchgässner et al., 2021). Classical statistical methods begin to place specific requirements on modeling and are mainly applied to the task of descriptive analysis. Usually, the researcher uses all the statistical methods readily available in the front-end R statistical environment in an extensible way, divides the data into a similar structure, and performs the expected analysis on each subset. This approach is still relevant today for tasks with tiny datasets. With the rapid growth of digital signal processing (DSP) and the broad application of big data statistical regulation, information in PMSMs can be fully monitored and recorded in real time, making it possible for extensive multi-dimensional data analysis. Thosar et al. (2020) developed a data-driven method using linear regression, an early warning function of the motor based on identification, statistics, and analysis of GAN data, achieved RMSE 0.2368, MAE 0.1515, and R^2 0.9439. The data-driven approach using past data and machine learning can provide a timely temperature prediction in PMSM.

Researchers generally like to use machine learning tools or deep learning algorithms. Savant and other researchers (2020) used Support Vector Regressor and Random forest to estimate parameters in a permanent magnet synchronous motor(PMSM), achieving the performance of R^2 in 0.7405 and 0.7562, respectively. At the same time, a deep neural network(DNN) model for PMSM temperature prediction was constructed. The results showed the prediction error of the model(MAE) was 0.1515, the RMSE was 0.2368, R-squared was 0.9439, and the goodness of fit in a train set data and a test set data((Savant, Kumar, & Ghatak, 2020)).

The rest of the paper is: In section 2, Methodology, including problem definition. Section 3 presents the investigation on the dataset as well as pre-processing steps. Section 4 will discuss the results of the applied algorithms and a comparison is made from different angles. Finally, the conclusions are presented in section 6.

2 Methodology

2.1 Problem Definition

In summary, directed detection provides the most direct indication of temperature trends but undoubtedly presents challenges in sensor development, installation, and maintenance. Traditional statistics-based method allows users to leverage all of the statistical methods readily available in R while abstracting the technical programming details. People hardly ever analyze data on a single machine because the data are too large to store or the statistical method used is too computationally intensive. Deep learning tools have advantages in handling multiple inputs, output, and nonlinear data. Despite taking uncertainties into account in multi-objective objective optimization, the objective function is difficult to establish, requires high computer performance, and takes too much time to compute (He et al., 2022).

3 Dataset

3.1 Overview

The dataset is sourced from the Kaggle Dataset: Electric Motor Temperature, which includes data collected from several sensors on the permanent magnet synchronous motor (PMSM) deployed on the test bench (Kaggle, n.d.). Permanent magnet synchronous machine (PMSM) drive is used in a wide range of motion control applications, which are known for their low torque ripple, excellent dynamic performance, high efficiency and high power density. The dataset contains a total of 133,086 entries, each with 13 columns. All entries are sampled at a frequency of 2Hz. The dataset consists of multiple measurement periods, which is can be identified in column **profile_id**. Currents in d/q-coordinates (columns **i_d** and **i_q**) and voltages in d/q-coordinates (columns **u_d** and **u_q**) result from a standard control strategy trying to follow the reference speed and torque. The columns and their specific description are shown in Table 1.

Table 1: Columns of the Dataset

Columns	Description	Unit
coolant	Coolant temperature	℃
stator_winding	Stator winding temperature measured with thermocouples	℃
u_d	Voltage d-component measurement in dq-coordinates	
stator_tooth	Stator tooth temperature measured with thermocouples	℃
motor_speed	Motor speed	rpm
i_d	Current d-component measurement in dq-coordinates	
i_q	Current q-component measurement in dq-coordinates	
pm	Permanent magnet temperature measured with thermocouples and transmitted wirelessly via a thermography unit.	℃
stator_yoke	Stator yoke temperature measured with thermocouples	℃
ambient	Ambient temperature	℃
torque	Motor torque	Nm
profile_id	Measurement session id. Each distinct measurement session can be identified through this integer id.	

3.2 Data Distribution

The box-plots of all column are plotted in Figure 1 to check the data distribution. It shows that column **i_q**, **u_d**, **ambient**, and **torque** all have obvious outliers while the others do not. Thus, we considered the treatment of outliers from these columns during pre-processing.

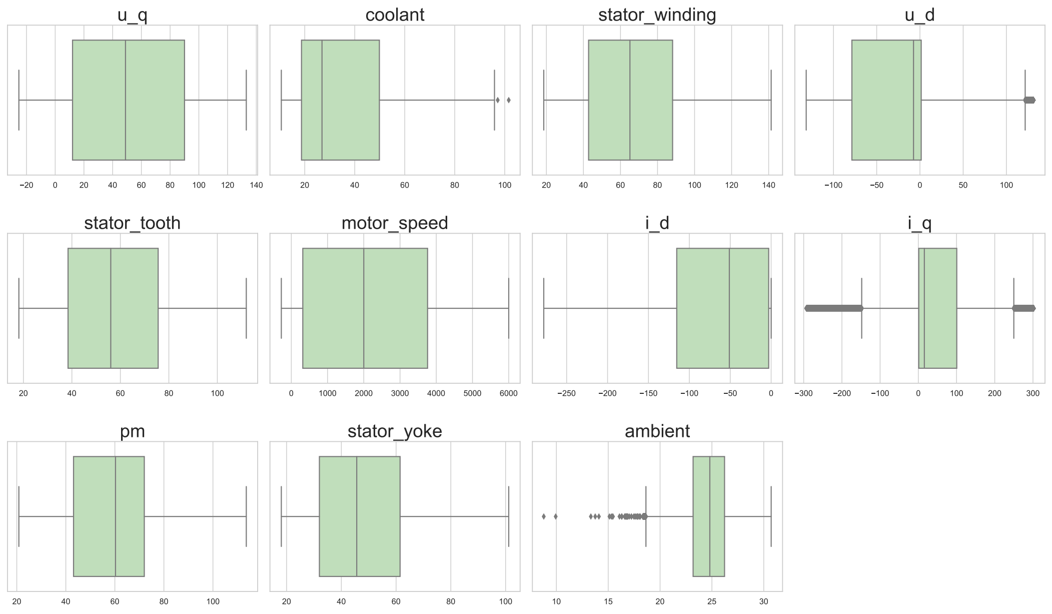
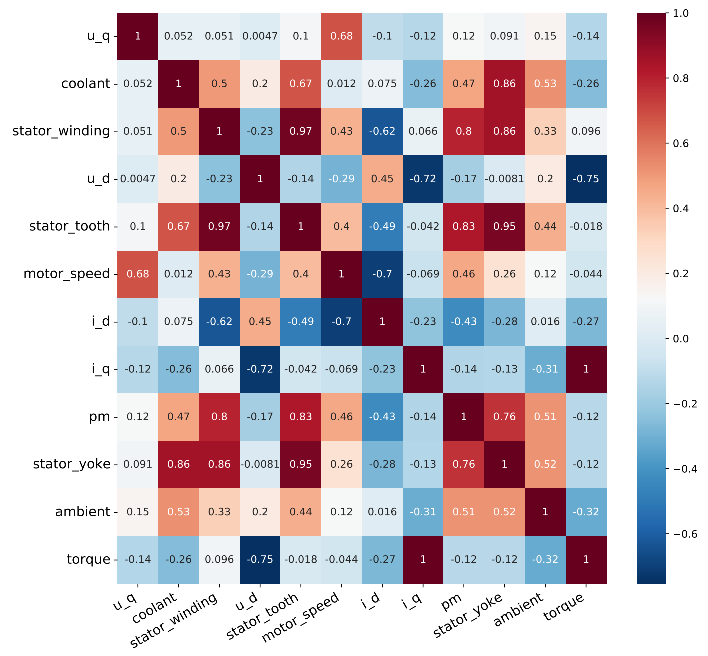


Figure 1: Data distribution

To explore the association between the different columns of data, we calculated the correlation coefficients between each numeric attribute, namely remove the column **profile_id**. The correlation matrix is shown in Figure 2. Since column **pm** is the to-be-predicted attribute of the task, columns **torque** and **u_d**, which have the lowest absolute value of correlation coefficients with **pm**, $|-0.12|$ and $|0.12|$, respectively, are considered for discarding in the following pre-processing stage for better performance.

To visualize the relationship between each column, we randomly sampled 1000 rows of the dataset and plotted a scatter matrix to display the distribution between each two columns, as shown in Figure 3. Notice that there is a clear linear relationship between **pm** and multiple columns, such as **stator_winding**, **stator_tooth**, and **stator_yoke**. This suggests that linear regression model could be a potential baseline candidate for its average performance.



	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
u_q	1	0.052	0.051	0.0047	0.1	0.68	-0.1	-0.12	0.12	0.091	0.15	-0.14
coolant	0.052	1	0.5	0.2	0.67	0.012	0.075	-0.26	0.47	0.86	0.53	-0.26
stator_winding	0.051	0.5	1	-0.23	0.97	0.43	-0.62	0.066	0.8	0.86	0.33	0.096
u_d	0.0047	0.2	-0.23	1	-0.14	-0.29	0.45	-0.72	-0.17	-0.0081	0.2	-0.75
stator_tooth	0.1	0.67	0.97	-0.14	1	0.4	-0.49	-0.042	0.83	0.95	0.44	-0.018
motor_speed	0.68	0.012	0.43	-0.29	0.4	1	-0.7	-0.069	0.46	0.26	0.12	-0.044
i_d	-0.1	0.075	-0.62	0.45	-0.49	-0.7	1	-0.23	-0.43	-0.28	0.016	-0.27
i_q	-0.12	-0.26	0.066	-0.72	-0.042	-0.069	-0.23	1	-0.14	-0.13	-0.31	1
pm	0.12	0.47	0.8	-0.17	0.83	0.46	-0.43	-0.14	1	0.76	0.51	-0.12
stator_yoke	0.091	0.86	0.86	-0.0081	0.95	0.26	-0.28	-0.13	0.76	1	0.52	-0.12
ambient	0.15	0.53	0.33	0.2	0.44	0.12	0.016	-0.31	0.51	0.52	1	-0.32
torque	-0.14	-0.26	0.096	-0.75	-0.018	-0.044	-0.27	1	-0.12	-0.12	-0.32	1

Figure 2: Correlation Matrix

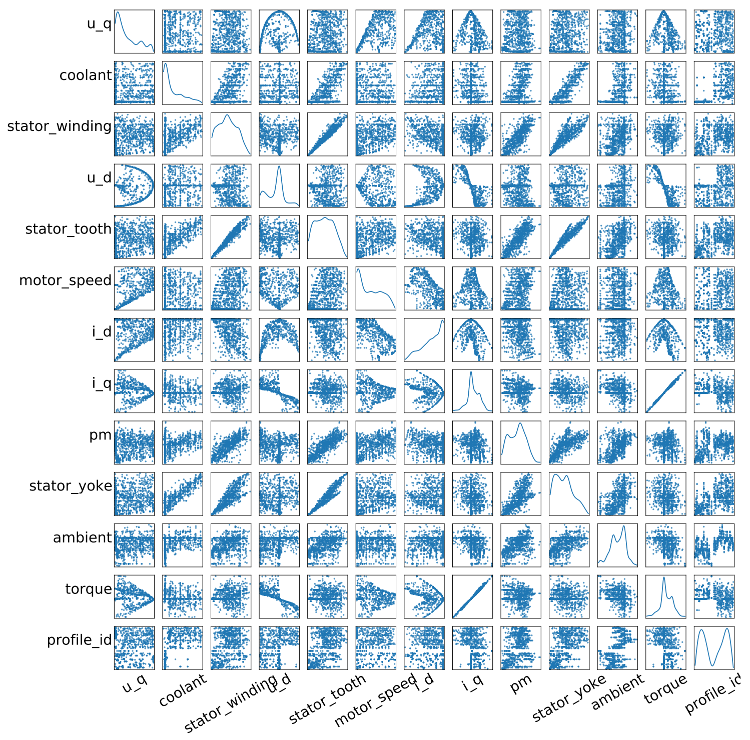


Figure 3: Matrix Scatter Plot

3.3 Pre-processing

Firstly, based on the correlation coefficients of column **pm** above, we remove columns **torque** and **u_d**, which have the two lowest in numeric columns, which can improve the efficiency of the model while minimising the impact on its accuracy. Then, for the only categorical column **profile_id**, we encode it in two steps. First the column is mapped to the corresponding label indices via the **StringIndexer** method provided by PySpark, the result landing in $[0, \text{numLabels})$. This process allows the corresponding features to be indexed, making it possible for use certain algorithms to process categorical variables. Since each category in column **profile_id** does not have a comparative relationship, one-hot encoding is applied to this column using **OneHotEncoder** by PySpark. After processing the dataset, the processed attributes are assembled to a new column for more convenient model fitting and prediction. The assembled column was standardized to eliminate the effect of samples with different attributes of different magnitudes, which would lead to the dominance of attributes of larger magnitudes and slow convergence of iterations. Standardizing features by removing the mean and scaling to the unit variance for each attribute property so that for each attribute/column all data are clustered around 0 with the standard deviation of 1, that is, the processed dataset now has a variance of 1 and a mean of 0.

4 Experiment Setup

4.1 Gradient-Boosting Tree

Gradient-Boosting Tree (GBT) is a Boosting Tree-based method, regarded as one of the best-performing methods among statistical approaches. The Boosting Tree Model uses decision trees as base classifiers and the forward distribution algorithm for boosting. For the regression task, suppose the training dataset is $T = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, $x_i \in X \subseteq \mathbb{R}^n$, X is the input space, $y_i \in Y \subseteq \mathbb{R}$, Y is output space. To be able to fit the training data, X is divided into J disjoint regions $R_1, R_2, ..., R_J$, with the constant c_j. The tree can be represented as

$$T(x;\Theta) = \sum_{j=1}^{J} c_j \mathrm{I}[x \in R_j] \tag{1}$$

where parameter $\Theta = (R_1, c_1), (R_2, c_2), ..., (R_J, c_j)$ indicates the division of the tree into regions and the constants on each region. J is the complexity of the regression tree, namely the number of leaf nodes. $\mathrm{I}[\cdot]$ is the indicator function that returns 1 if the condition inside is true otherwise 0. The initial boosting tree is set as $f_0(x) = 0$ and the model for the m-th step is

$$f_m(x) = f_{m-1} + T(x;\Theta_m) \tag{2}$$

where f_{m-1} is the model at step m. The parameters for the next decision tree are determined by Empirical Risk Minimization (ERM).

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i-1}^{N} L(y_i, f_{m-1}(x_i) + T(x;\Theta_m)) \tag{3}$$

When Root Mean Square Error (RMSE) is adopted as the loss function

$$L(y, f(x)) = (y - f(x))^2 \tag{4}$$

The loss 4 can be expanded as

$$\begin{aligned} L\left(y, f_{m-1}(x) + T(x;\Theta_m)\right) &= \left[y - f_{m-1}(x) - T(x;\Theta_m)\right]^2 \\ &= \left[r - T\left(x;\Theta_m\right)\right]^2 \end{aligned} \tag{5}$$

Here,

$$r = y - f_{m-1}(x) \tag{6}$$

is the residual of the current model fitted data. For regression tasks, the boosting tree model is trained by simply fitting a reduction in the residuals of the current model.

However, in Gradient-Boosting Tree Model (GBT), gradient boosting proposed by Freidman (Friedman, 2001) is used instead of residuals to simplify the optimisation. It utilized the negative gradient of the loss function of the current model,

$$-\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)} \tag{7}$$

as an approximation to the residuals in the booster tree algorithm for regression problems to fit the regression tree.

4.2 Other Models for Comparison

In the experiments, Linear Regression Model, Decision Tree Model, and Random Forest Model are also implemented for comparison.

4.2.1 Linear Regression

Linear regression is used to predict the relationship between two variables by looking at the data using a linear equation. It assumes that each variable is independent of the other and the effects are superimposable.

$$f(x_1, x_2, ..., x_n) = w_0 + \sum_{i=1}^{n} w_i x_i = w^T x + b \tag{8}$$

The loss function can be represented as

$$L(w) = \frac{1}{2} \sum_{j=1}^{m} \left[y^{(j)} - \sum_{i=1}^{n} w_{(i)} x_i^{(j)} - b \right]^2 \tag{9}$$

Then the gradient descent is used to train to reduce the loss to optimize the model.

4.2.2 Decision Tree

The decision tree for regression tasks recursively divides each region into two word regions on the output space where the training set is located and determines the output values on each sub-region, to construct a binary decision tree. The optimal cut variable j and cut point s are obtained from

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \tag{10}$$

by iterating over the variable j to get the smallest value of the formula. Then continue to divide the two sub-regions as above until the input space is divided into $R_1, R_2, ..., R_M$, generating the decision tree

$$f(x) = \sum_{m=1}^{M} \hat{c}_m \mathrm{I}\left[x \in R_m\right] \tag{11}$$

4.2.3 Random Forest

Random Forest adopts the bagging method to ensemble decision trees. Suppose there are N samples in the training set. N samples are randomly selected with a put-back, and are used to train a decision tree as the samples at the root node. When a node of the decision tree needs to be split, m attributes are randomly selected from a total of M attributes, satisfying the condition $m \ll M$. From these m attributes. And by using certain criteria such as information gain to select an attribute as the splitting attribute for that node. Then repeat the previous steps until no further splitting is possible and the random forest model is finally generated.

5 Results & Evaluation

This research uses a train-test split of 0.8 and 0.2. Cross validation was not employed because of the extra training time. All the evaluation metrics, namely R^2, RMSE, and MAE, were obtained form the inference result on the test data set.

All the models were trained with the same training set and testing set aforementioned, using the same framework PySpark (1.5.0) which runs on Cloudera environment (5.5.0).

5.1 R^2, RMSE, and MAE

Figure 4 plots the model performance in terms of R^2, RMSE, and MAE. The target variable (PM) has not been normalized and thus have the unit of Celsius degree.

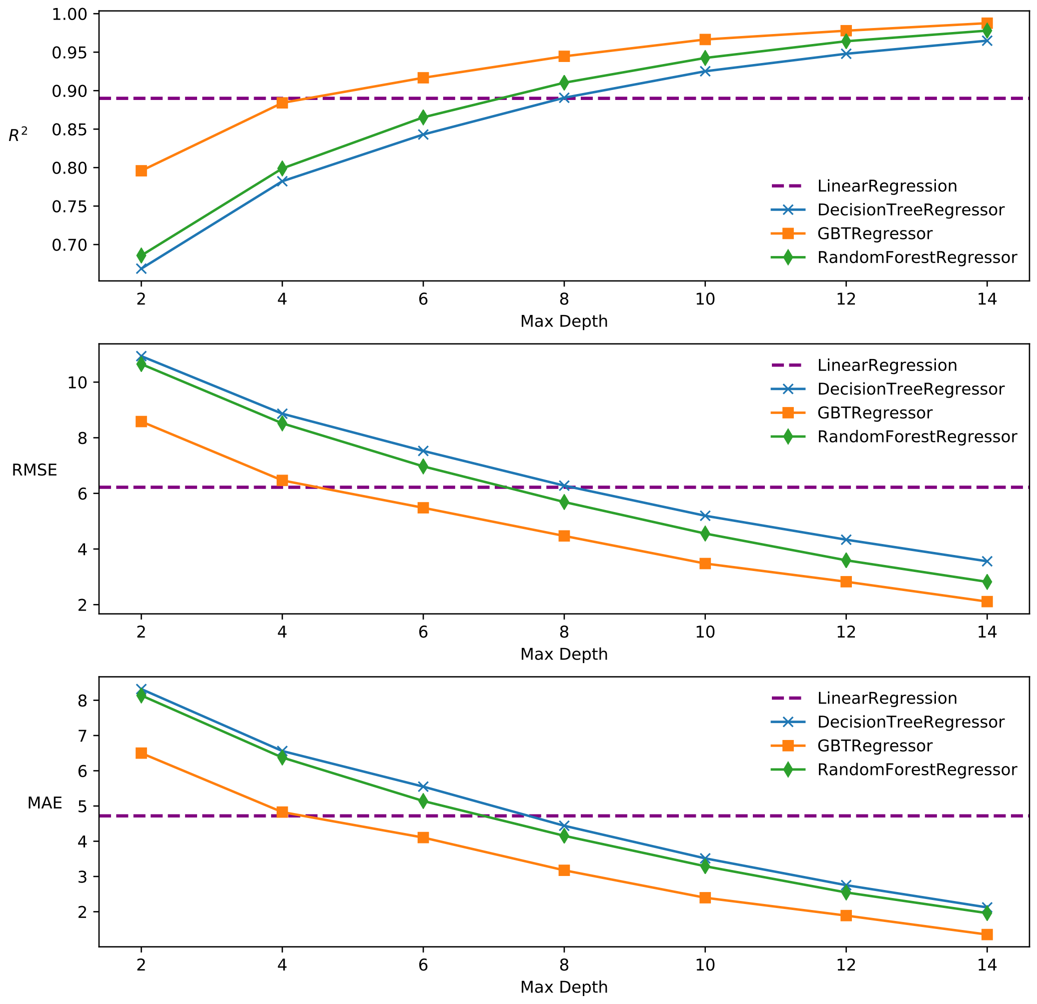


Figure 4: Model Accuracy Metrics

The simplest linear regression model, denoted by the dashed purple line, reached the accuracy of R^2 = 0.89. The proposed GBT models' R^2 increases with the hyper-parameter **maxDepth**, which limits the decision tree's maximum depth. The highest R2 GBT model reaches is 0.987, at depth = 14. For comparison purpose, to other two tree-based regression methods (Decision Tree and Random Forest) also show an increase of R2 but not as prominent as GBT. This suggests the advantage when the gradient-boosting process brings to the tree-based models. The same comparison can be made from the RMSE and MAE plots.

Notice that it took only slightly more than 4 layers in depth for a GBT model to surpass a linear regression model while it takes 7 and 8 for Random Forest Regressor and Decision Tree Regressor, respectively. Again, this indicates the effectiveness of the gradient-boosting process. The number of ensemble trees here are set to around 20, meaning that a 5-layer GBT model has around 500 parameters, which apparently has a stronger representation capability compared to a linear model.

5.2 Model Efficiency

Figure 5 plots all the training and inference time of the selected models. Although the GBT models display a clear advantage in terms of the accuracy, the required training time noticeably increase with the depth as the hyper-parameter. it took around 900 seconds for a 14-layer GBT model to be fully trained to reach test $R^2 = 0.987$ while this time cost is about 10 times the decision tree regressor (test $R^2 = 0.947$. This is also the reason why researchers tend to prefer max depth fewer than 15 layers. The main explanation for this sharp increase is that GBT build one tree at a time dedicated to correct the error from the previously generated tree according to its error gradient while RF builds trees that are independent, which does not involve the gradient calculation. The calculation involves as many layers as a GBT tree has, which at least linearly increases as the number of nodes in the tree; while other models does not involve gradient calculation and thus not slow down quite much.

It is also worth noticing that in the inference stage the GBT's time expense is much less than random forest and close to linear and decision tree regressor.

In production environment, the estimation of PM temperature has to be low-latency, fast-responsive, and frequent enough. Regardless of other time cost, from the plot we can estimate that a deployed GBT model can produce predicted temperature records about 2 to 3 times as a decision tree regressor does, of course also 2 to 3 times slower than a linear model.

Usually the model training is considered as an offline process while the inference time accounts for the real time cost. Based on this, the latter time is prioritized. The results suggest that a GBT model is a decent model that balances the trade-off of the efficiency and the accuracy: linear regression and decision tree regressor are faster but their accuracy might not cater to the demand of the car's control system. On the other hand, the model accuracy actually reduces the time expense on error correction, such as some filtering algorithm, which may also be the overhead.

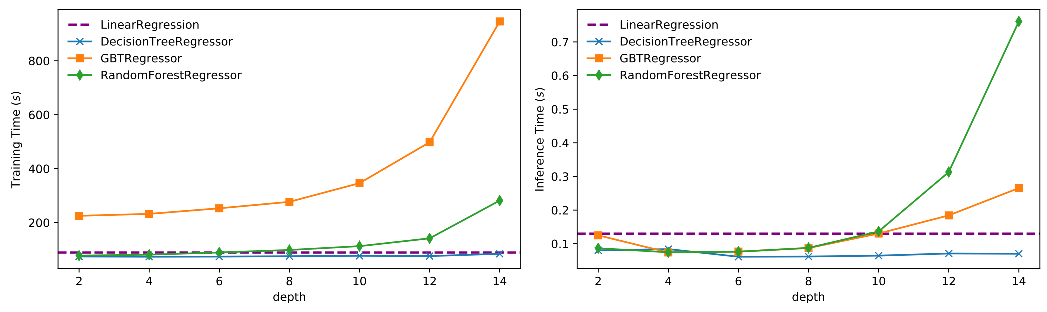


Figure 5: Model Training & Inference Time

5.3 Model Portability

In general, the number of parameters of all the selected four models can be ranked in descending order as: GBT $\approx$ RF $>$ DT $>$ LR. For low-end devices, such as micro chips, hard-encoded model usually store quite limited number of parameters. Linear regression does not suffer from this problem in this context. However, GBT and RF as ensemble learners face the problem of model compression. Possible methods include pruning and voting, which shrink the model size while preserving the accuracy. This will not be expanded in detail due to the content length limitation.

5.4 Hyper-Parameter Optimization

Among the selected models, GBT is the most sensitive one regarding hyper-parameter tuning, mainly because the gradient-related learning rate is involved. Learning rate cannot be either too small, which slows down the progress and may result in model loss objective stuck in local minima, or too big, which leads to numerical instability. Normally learning rate for a GBT model is 0.1, which shrinks the error by the previous model. 1 means no change will be made on the calculated gradient.

There are three other hyper-parameters that have been optimized in this study, namely:

- **loss type**: either 'MSE' or 'MAE'.
- l_2: the regularization term for GBT model. Range from $[0, 1]$
- **max depth**: the maximum depth of a GBT model. Integer, $5 \leq d < 15$

All these hyper-parameters are tuned using the `optuna` library with the objective to maximize set to R^2 on the testing set. Because of the small number of hyper-parameters, Optuna uses Bayesian optimization to automatically selects new candidates. The results are shown as below:

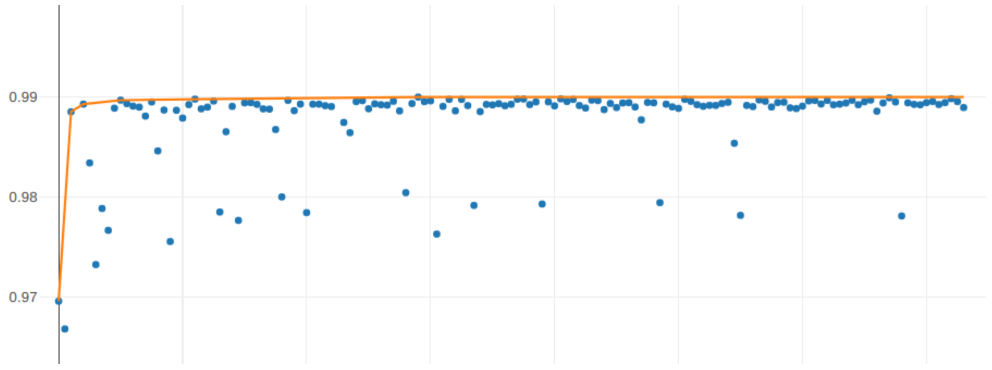


Figure 6: Hyper-Parameter Optimization Progress

Figure 6 shows the best R^2 as the optimization runs. The global best result is from trial 58, with $R^2 = 0.990$ and hyper-parameters as `{'loss_type': 'squared_error', 'lr': 0.7353, 'max_depth': 11, 'l2': 0.4136}`.

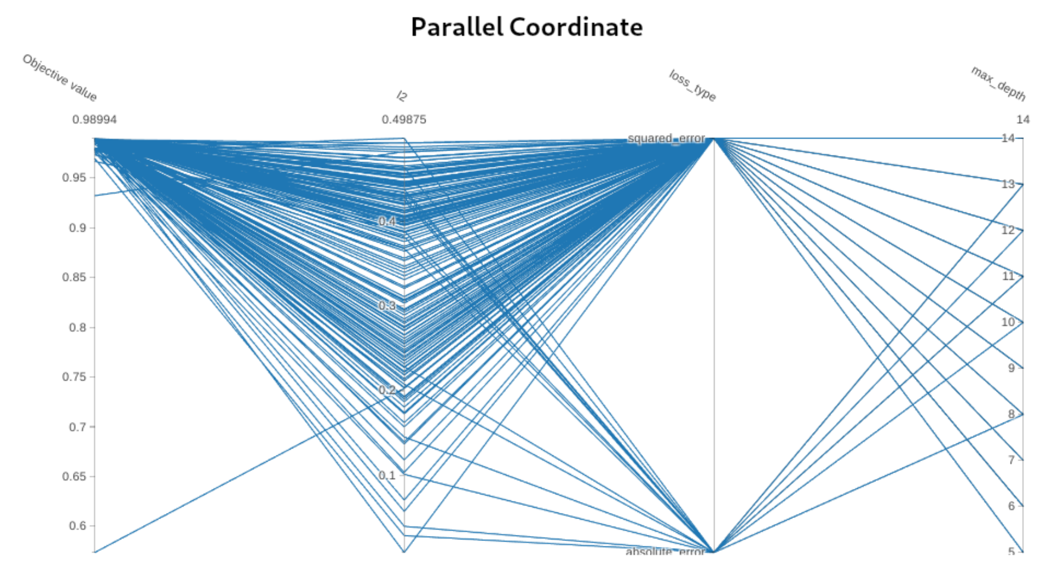


Figure 7: Hyper-Parameter Parallel Coordinate Plot

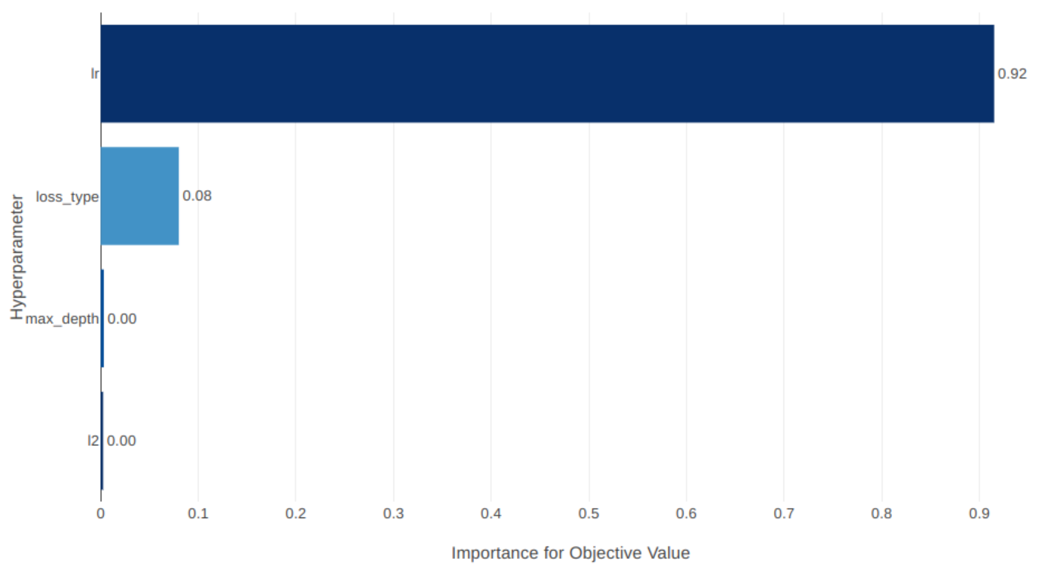


Figure 8: Hyper-Parameter Importance

Figure 7 and 8 illustrates the importance of different hyper-parameters in 2 ways. The Parallel Coordinates plot depicts all the model candidates. In their hyper-parameter distribution, we can see denser area indicate better hyper-parameter values resulted from better trials. From the bar plot, it is straightforward that by calculation, the learning has the greatest impact on the objective R^2.

In summary, the learning-rate is crucial to gradient-based models like GBT.

5.5 Limitations

The GBT model employed in this study has 3 main drawbacks:

First, it is slow to train if its model accuracy is to be maximally exploited. GBT does not converge as fast as linear models and require long hyper-parameter optimization process. This can only be resolved by reducing the model complexity and scarifying accuracy to some extent.

GBT is prone to over-fit. Unlike random forest which does not use the gradient feedback, GBT has the clear objective to minimize and all the gradients come from the training samples. This can be improved by tweaking the regularization term l_2, also with some accuracy loss.

Finally, GBT lacks strong interpretability compared to linear models. This is a common problem for ensemble learning models and the corresponding solution is, also, to shrink the model size, by, for instance, pruning or using fewer nodes.

6 Conclusion

Great importance has been shown regarding the accurate estimation of PMSM. As the dataset size increase, many widely-applied conventional mathematical models, such as linear regression and logistic regression, start to suffer from the accuracy and scalability problems despite their neat closed-form algebraic formula. Gradient-Boosting Trees (GBT) as a relatively newer machine learning technique applied on decision tree, out-stands in terms of its high accuracy and portability. This article utilizes GBT on the task of PMSM temperature estimation and critically analyzes the performance and compares it with a few typical state-of-the-art methodologies. Results suggest that fitted models could be feasibly deployed to low-end devices while still preserving its decent performance.

References

Al-Gabalawy, M., Elmetwaly, A., Younis, R., & I. Omar, A. (2022, May). Temperature prediction for electric vehicles of permanent magnet synchronous motor using robust machine learning tools. *Journal of Ambient Intelligence and Humanized Computing*. doi: 10.1007/s12652-022-03888-9

Denic, N., Cirkovic, B., Petković, D., Nesic, Z., & Mehmedi, S. (2022, September). Neuro Fuzzy Estimation of the Optimal Parameters for Prediction of Permanent Magnet Synchronous Motor (PMSM) Temperature. *Journal of Vibration Engineering & Technologies*. doi: 10.1007/s42417-022-00710-w

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, *29*(5), 1189–1232. Retrieved 2022-10-18, from `https://www.jstor.org/stable/2699986` (Publisher: Institute of Mathematical Statistics)

Gaona, D., Wallscheid, O., & Böcker, J. (2017, December). Fusion of a lumped-parameter thermal network and speed-dependent flux observer for PM temperature estimation in synchronous machines. In *2017 IEEE Southern Power Electronics Conference (SPEC)* (pp. 1–6). doi: 10.1109/SPEC.2017.8333640

He, L., Wu, X., Nie, Y., & Shi, W. (2022, July). Loss Prediction of Vehicle Permanent Magnet Synchronous Motor Based on Deep Learning. *Journal of Electrical Engineering & Technology*. doi: 10.1007/s42835-022-01153-9

Kaggle. (n.d.). *Electric motor temperature*. `https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature`.

Kirchgässner, W., Wallscheid, O., & Böcker, J. (2021, September). Data-Driven Permanent Magnet Temperature Estimation in Synchronous Motors With Supervised Machine Learning: A Benchmark. *IEEE Transactions on Energy Conversion*, *36*(3), 2059–2067. doi: 10.1109/TEC.2021.3052546

Lee, J. Y. L., Brown, J. J., & Ryan, L. M. (2017, July). Sufficiency Revisited: Rethinking Statistical Algorithms in the Big Data Era. *The American Statistician*, *71*(3), 202–208. doi: 10.1080/00031305.2016.1255659

Nan, J., Deng, B., Cao, W., Hu, J., Chang, Y., Cai, Y., & Zhong, Z. (2022, July). Big Data-Based Early Fault Warning of Batteries Combining Short-Text Mining and Grey Correlation. *Energies*, *15*, 5333. doi: 10.3390/en15155333

Savant, R., Kumar, A. A., & Ghatak, A. (2020, December). Prediction and Analysis of Permanent Magnet Synchronous Motor parameters using Machine Learning Algorithms. In *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)* (pp. 1–5). Bengaluru, India: IEEE. Retrieved 2022-10-19, from `https://ieeexplore.ieee.org/document/9339479/` doi: 10.1109/ICAECC50550.2020.9339479

Thosar, P., Patil, J., Singh, M., Thamke, S., & Gonge, S. (2020, October). Prediction of Motor Temperature using Linear Regression. In *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)* (pp. 7–12). doi: 10.1109/ICSTCEE49637.2020.9277184

Yang, C., Zha, M., Wang, W., Yang, L., You, S., & Xiang, C. (2021, December). Motor-Temperature-Aware Predictive Energy Management Strategy for Plug-In Hybrid Electric Vehicles Using Rolling Game Optimization. *IEEE Transactions on Transportation Electrification*, *7*(4), 2209–2223. doi: 10.1109/TTE.2021.3083751