Capstone Project

Niranjan Salimath

Machine Learning Engineer Nanodegree

July 27th, 2019

Definition

Project Overview

In most automotive applications, a high power and torque density as well as efficiency is required for any given motor, which makes the permanent magnet synchronous machine (PMSM) the preferred choice. In order to exploit the machine's full capabilities, high thermal stress on the machine's potentially failing components must be taken into account. A sensor-based temperature measurement would yield rather precise knowledge regarding the machine's thermal state, yet for the rotor part, it is technically and economically infeasible due to an electric motor's sophisticated internal structure and the difficult accessibility of the rotor. Hence, direct rotor monitoring techniques such as infrared thermography [1], [2] or classic thermocouples with shaft-mounted slip-rings [3] fall short of entering industrial series production. In contrast, stator winding temperature monitoring is measured on a sensor basis nowadays, yet in case of faults, these sensors cannot be replaced due to being firmly embedded in the stator. In addition, sensor functionality may deteriorate during the motor's life cycle. Taking into account the ever increasing importance of functional safety especially in the automotive industry, redundant temperature information becomes obligatory.

In this project, it will be shown that a supervised machine learning models, namely linear regression and support vector regression circumvents any kind of electric motor modeling by being fitted on measured test bench data with minor preprocessing directly and still achieves high estimation performance with time invariant properties. This approach ignores domain knowledge and, hence, does not require any conventional engineering expertise in electric machine thermal management.

Problem Statement

The goal is to create a algorithm to predict the rotor temperature in permanent magnet synchronous machines.

Permanent magnet synchronous machines (PMSMs) are a popular choice in many traction drive applications due to their high energy and power density and moderate assembly costs. However, electric motor thermal robustness in general is harmed by the lack of accurate temperature monitoring capabilities such that safe operation is ensured through oversized materials at the cost of its effective utilization. Classic thermal modeling is conducted through lumped parameter thermal networks (LPTNs), which help to estimate internal component temperatures rather precisely but also require expertise in choosing model parameters and lack physical interpretability as soon as their degrees of freedom are curtailed in order to meet the real-time requirement. It will be shown that, as an alternative to LPTNS, supervised learning algorithms like linear regression, logistic regression and support vector regressor achieve similar predictive performance with low computational complexity as long as input representations are preprocessed with exponentially weighted moving averages. Thus, domain knowledge becomes neglectable, and estimation performance depends entirely on collected data and considered input representations.

The tasks involved are the following:

1. Download and preprocess the Kaggle’s electric-motor-temperature dataset (pmsm_temperature_data.csv)
2. As input data for this project is a time series, whole of the data needs to be visualized, trained and tested by grouping.
3. Identify important features which are influencing the rotor temperature by visualizing the input data using “matplotlib” or “seaborn”.
4. Train regressor algorithms to predict rotor temperature.
5. Calculate evaluation metrics. And Compare with benchmark model.
6. Use advanced methods like ensembling methods to improve accuracy.

Evaluation Metrics

The evaluation metric to be used is mean squared error to quantify the performance of both the benchmark model and the solution model.
If y_i is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over $n_{samples}$ is defined as

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

The mean absolute error [MAE] is an average of the absolute errors. And MAE is more robust to outliers since it does not make use of square. But in the current scenario large deviation or large errors may cause greater ill effect in decision making when compared to smaller errors.

The mean squared error [MSE] is an average of the square of errors. In MSE, we are penalizing large deviations more. Square a big number, and it becomes much larger, relative to the others. Hence the effect of large errors or deviations can be caught and will be helpful in evaluating model properly.

Hence for the current scenario Mean squared error is more suitable.

Analysis

Data Exploration

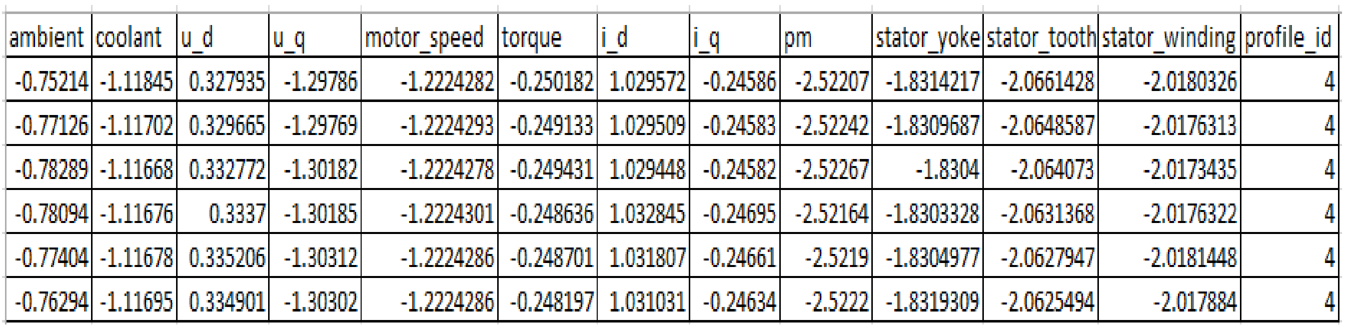
Kaggle Datasets link: pmsm_temperature_data.csv.

Dataset of Electric-Motor-Temperature Project (https://www.kaggle.com/wkirgsn/electric-motor-temperature)

Input fields of data:

1. Ambient – Ambient temperature as measured by a thermal sensor located closely to the stator.
2. Coolant - Coolant temperature. Motor is water cooled. Measurement is taken at outflow.
3. u_d - Voltage d-component
4. u_q - Voltage q-component
5. motor_speed - Motor speed
6. torque - Torque induced by current.
7. i_d - Current d-component
8. i_q - Current q-component
9. pm - Permanent Magnet surface temperature representing the rotor temperature. This was measured with an infrared thermography unit.
10. stator_yoke - Stator yoke temperature measured with a thermal sensor.
11. stator_tooth - Stator tooth temperature measured with a thermal sensor.
12. stator_winding - Stator winding temperature measured with a thermal sensor.
13. profile_id - Each measurement session has a unique ID. Make sure not to try to estimate from one session onto the other as they are strongly independent.

Data organization:

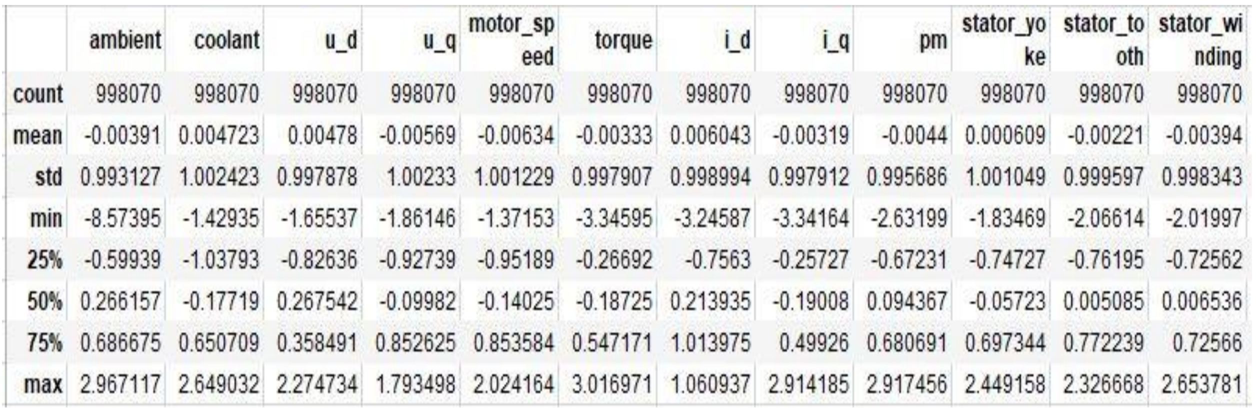


ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding	profile_id
-0.75214	-1.11845	0.327935	-1.29786	-1.2224282	-0.250182	1.029572	-0.24586	-2.52207	-1.8314217	-2.0661428	-2.0180326	4
-0.77126	-1.11702	0.329665	-1.29769	-1.2224293	-0.249133	1.029509	-0.24583	-2.52242	-1.8309687	-2.0648587	-2.0176313	4
-0.78289	-1.11668	0.332772	-1.30182	-1.2224278	-0.249431	1.029448	-0.24582	-2.52267	-1.8304	-2.064073	-2.0173435	4
-0.78094	-1.11676	0.3337	-1.30185	-1.2224301	-0.248636	1.032845	-0.24695	-2.52164	-1.8303328	-2.0631368	-2.0176322	4
-0.77404	-1.11678	0.335206	-1.30312	-1.2224286	-0.248701	1.031807	-0.24661	-2.5219	-1.8304977	-2.0627947	-2.0181448	4
-0.76294	-1.11695	0.334901	-1.30302	-1.2224286	-0.248197	1.031031	-0.24634	-2.5222	-1.8319309	-2.0625494	-2.017884	4

Target feature: Rotor temperature [“pm”]

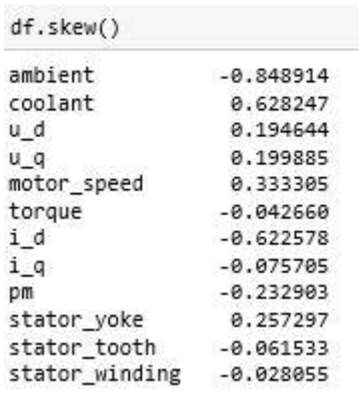
Shape of data (998070, 13)

Mean, standard deviation, minimum, maximum, percentiles of lower, middle and upper of the input dataset are specified in the below picture. Middle percentile itself is median.



	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding
count	998070	998070	998070	998070	998070	998070	998070	998070	998070	998070	998070	998070
mean	-0.00391	0.004723	0.00478	-0.00569	-0.00634	-0.00333	0.006043	-0.00319	-0.0044	0.000609	-0.00221	-0.00394
std	0.993127	1.002423	0.997878	1.00233	1.001229	0.997907	0.998994	0.997912	0.995686	1.001049	0.999597	0.998343
min	-8.57395	-1.42935	-1.65537	-1.86146	-1.37153	-3.34595	-3.24587	-3.34164	-2.63199	-1.83469	-2.06614	-2.01997
25%	-0.59939	-1.03793	-0.82636	-0.92739	-0.95189	-0.26692	-0.7563	-0.25727	-0.67231	-0.74727	-0.76195	-0.72562
50%	0.266157	-0.17719	0.267542	-0.09982	-0.14025	-0.18725	0.213935	-0.19008	0.094367	-0.05723	0.005085	0.006536
75%	0.686675	0.650709	0.358491	0.852625	0.853584	0.547171	1.013975	0.49926	0.680691	0.697344	0.772239	0.72566
max	2.967117	2.649032	2.274734	1.793498	2.024164	3.016971	1.060937	2.914185	2.917456	2.449158	2.326668	2.653781

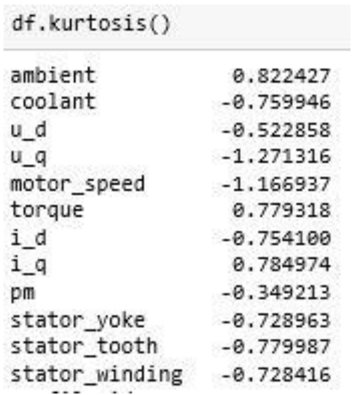
The skewness of input data is :



```
df.skew()

ambient          -0.848914
coolant           0.628247
u_d               0.194644
u_q               0.199885
motor_speed       0.333305
torque           -0.042660
i_d              -0.622578
i_q              -0.075705
pm               -0.232903
stator_yoke       0.257297
stator_tooth     -0.061533
stator_winding   -0.028055
```

The kutosis of input data is :



```
df.kurtosis()

ambient           0.822427
coolant          -0.759946
u_d              -0.522858
u_q              -1.271316
motor_speed      -1.166937
torque            0.779318
i_d              -0.754100
i_q               0.784974
pm               -0.349213
stator_yoke      -0.728963
stator_tooth     -0.779987
stator_winding   -0.728416
```

Exploratory Visualization

As the input data is a time series data, it needs to be grouped before splitting it into training set and testing set. Inside every group data needs to be split into training set and testing set. In this problem, as “profile_id” column indicates the different durations during which data is captured, hence we can group the data based on “profie_id”.

Before jumping to splitting data, if we see at individual group lengths[number of time stamps taken in particular profile_id] we can rule out some of the groups as they don’t carry much of the information regarding temperature change.

Measurement session lengths

Fig 1 shows that all measurement sessions range from 20 minutes to around 6 hours. The two short session ids "46" and "47" might be not very representative as temperatures inside electric motors need time to vary.

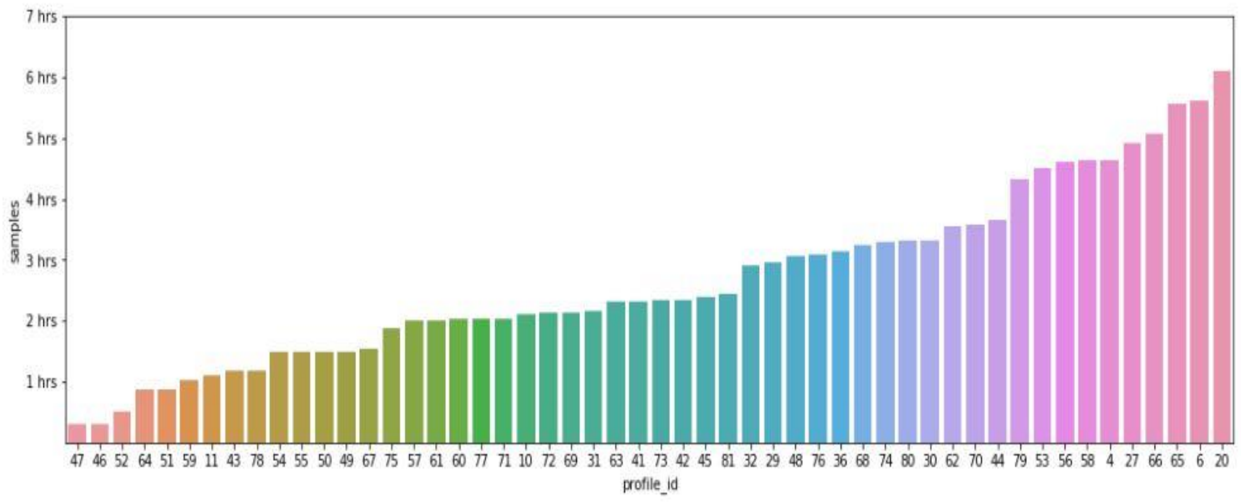


Fig 1

Effect of some uniformly picked recordings to select target features.

In real motor rotor temperature will mostly follow stator temperatures. Hence our target features will be stator_yoke, stator_winding, stator_tooth and pm(rotor temperature). Here temperatures of stator at different part of it are considered as target features as they will be resembling the rotor temperature with less deviation.

Other features like torque, motor_speed and coolant temperature will also be influencing the rotor temperature.

Visualization of the above mentioned features are captured in the Fig 2

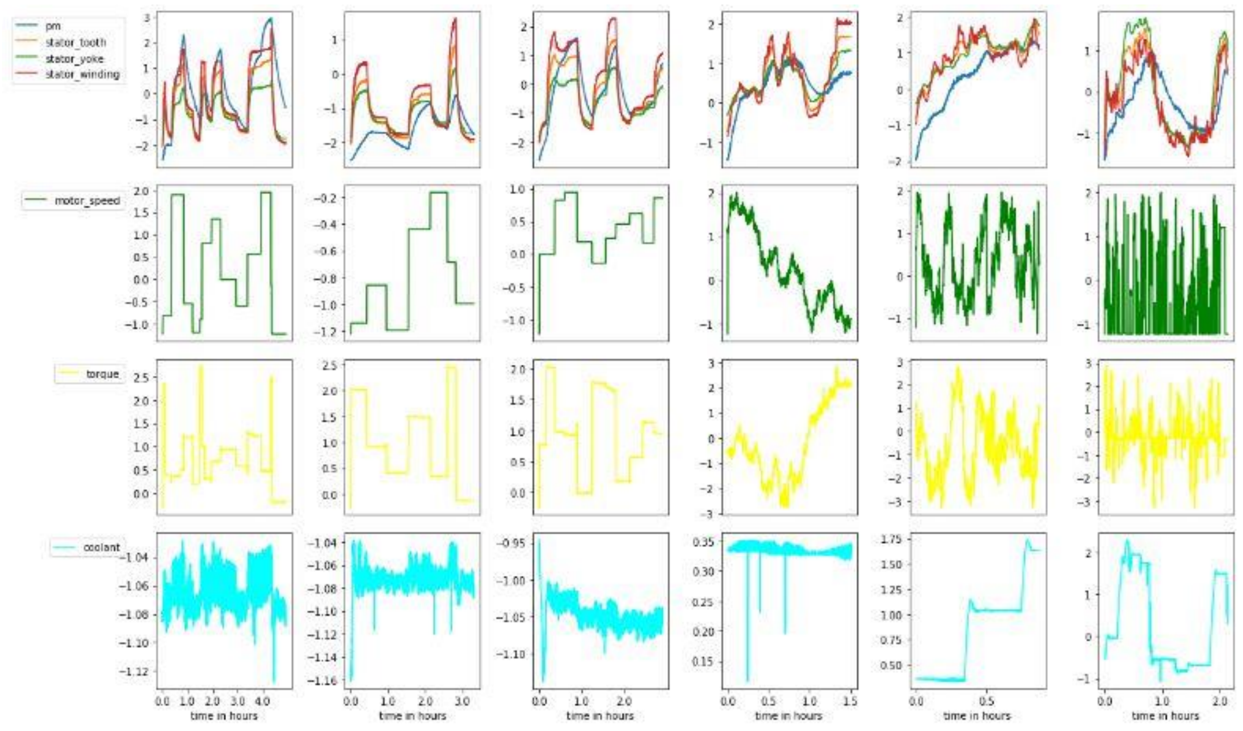


Fig 2

Findings from the above visualization

1. While motor excitations (motor_speed, torque, coolant) are sometimes of high dynamic, sometimes of stepwise nature, target temperatures always exhibit low-pass behavior with exponential rise and falls.
2. Coolant temperature suffers from measurement artefacts expressed by sharp drops in temperature, which recover as fast.
3. PM (Permanent Magnet -> Rotor) temperature expresses the slowest time constant and follows stator temperatures

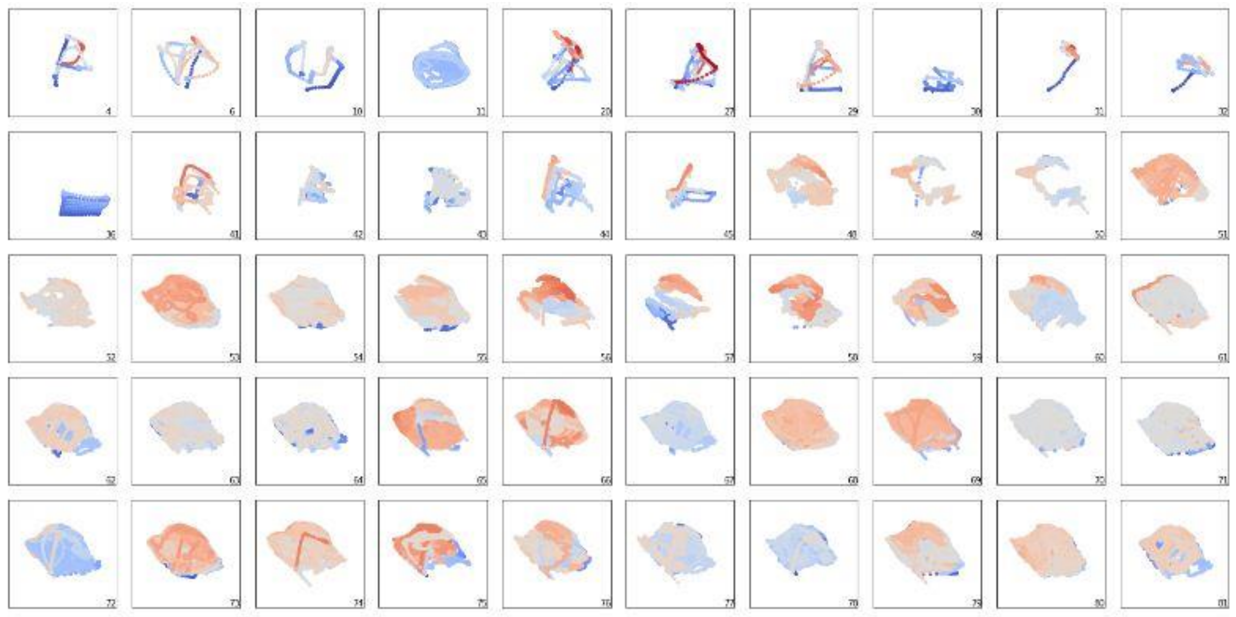


Fig 3

Finding from Fig 3:

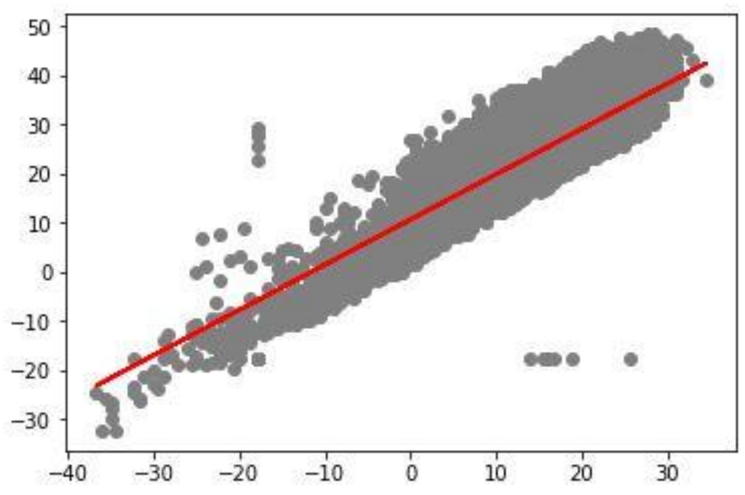
It is obvious that lower profile IDs are of simpler driving cycles, not moving much in feature space. Higher profile IDs are driving cycles of high dynamics - excitation happened through random walks in the motor_speed-torque-plane.

Algorithms and Techniques

Linear regression:

Linear regression is a very simple regression model which tries to find some linearity in the features and the target and predicts based on linear relationship between features and target variable.

Below figure shows linear regression model prediction and actual value



prediction vs test data

Here red line indicates the predicted value using linear regression model trained on some random data. Where gray colored points are the actual values for respective features.

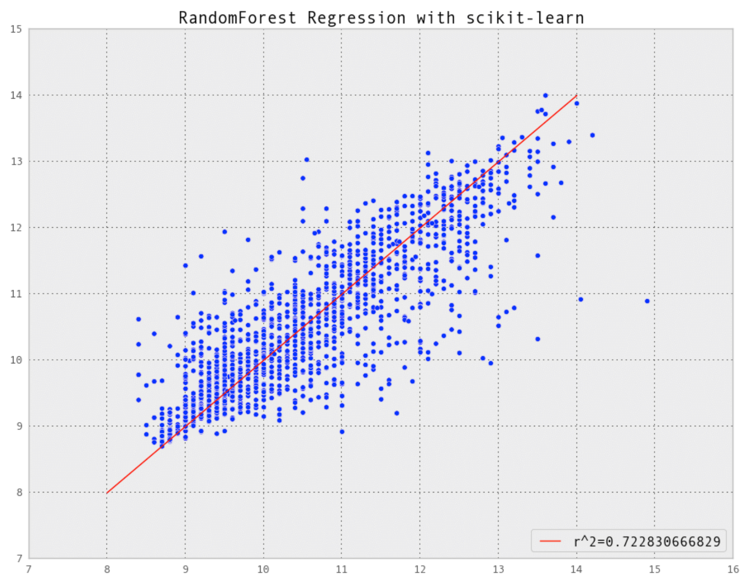
In the current scenario features like stator_yoke, stator_winding and stator tooth are almost linearly related to rotor temperature hence we can get some what good prediction.

So this model is chosen as benchmark model.

Random Forest Regression:

Random forest regressor is a meta estimator that fits number of classifying decision trees on various sub-samples of dataset and averaging to improve the predictive accuracy and control overfitting.

In current scenario input data is grouped based on profile IDs. Hence a multiple decision trees used in Random forest regression fit the data in different way and we can find variation in prediction by multiple trees. Hence this variation helps us in controlling overfitting of model. So random forest regressor is helpful in this problem.



GridSearchCV:

Gridsearch cross validation is usefull in finding proper parameters for a particular model by giving us the opportunity to choose the parameters for which the model needs to be fitted and gives the best parameter value suitable for the particular problem and model.

In current scenario, I am using randomforest regressor as estimator and I wanted to decide the proper value for parameter "max_depth" hence I gave max_depth as parameter.

So grid search algorithm is helpful in choosing the proper value for parameters in estimators.

XGBRegressor:

XGBoost stands for eXtreme Gradient Boosting.

Generally, XGBoost is fast. Really fast when compared to other implementations of gradient boosting. XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems.

Benchmark

To create an initial benchmark I used Linear regression model which is a very simple model but catches linearity in data. It gave me the mean squared error value of 0.3558. which is very poor in this case with keeping random test ids like 58 and 78.

Methodology

Data Preprocessing

Grouping of input data based on their profile Ids:

As this dataset is a time series and the data in different time series cannot be used for predicting the rotor temperature in different time series, I grouped all data points based on their profile IDs.

Splitting of data into training set and testing set is also done in all groups making sure that every group has both training and testing set.

Principal component analysis [PCA]:

PCA is used for extracting principal features which are affecting change in rotor temperature. Which is used for implementation of benchmark model linear regression model. Features other than target features like stator_winding, stator_tooth, stator_yoke and "pm" might be having some information which might be crucial in estimating rotor temperature. Hence we use PCA transform ed data to visualize and see the effect of other features.

Implementation

Implementation of the project starts from benchmark model linear regression model. For linear regression model I used principle component analysis as a data preprocessor. Linear regression model yielded me mean squared error of 0.3558 which is very high in this case.

Steps involved in implementation:

1. Load the input data into pandas data frame and remove NAs.

2. Identify important features which are influencing the rotor temperature by visualizing the input data as mentioned in data visualization section.
3. Transform the input data using Principal component analysis technique for benchmark model.
4. Split whole data into both the training and validation set in every group of data, preprocessing them as described in the data preprocessing section.
5. Initialize the RandomForest model with 10 estimators.
6. Fit the model.
7. Use grid search technique to finalize the parameters like max_depth used in Random_forest.
8. Finalize and fit the model with proper parameters.

Here random forest regression model is used for implementation which is very good in overcoming over fitting of decision trees.

Random forest regressor has multiple parameters to set but the main parameters generally used are "n_estimators" and "max_depth". By default n_estimators will be 10 in V*0.20* and max_depth will be "None" in that scenario depth will be considered until all the leaves are reached.

In current scenario input data is grouped based on profile IDs. Hence a multiple decision trees used in Random forest regression fit the data in different way and we can find variation in prediction by multiple trees. Hence this variation helps us in controlling overfitting of model.

Refinement

In using random forest regression model for implementation there are some parameters which needs to be set properly to improve the prediction capacity of the model. Hence "GridSearchCV" the technique used to get the best values for particular parameter of models.

Grid search cross validation technique uses the one base estimator on which grid search needs to run. Here in this case I am using random forest regression model.

Random forest regression model intern uses decision trees for them choosing the best max_depth value is difficult. Hence grid search is very helpful in this case.

Here I have given max_depth values as 4, 6, 10 for choosing the best max depth value.

Here I came to a decision that max_depth of 10 is suitable for this case.

Results

Model Evaluation and Validation

Final model built is Randomforest regressor.

Randomforest regressor model has least Mean squared error when we compare with benchmark model [Linear regression].

Time consumed in fitting benchmark model is very less but mean squared error is high. In random forest regressor model time consumed in fitting is approximately 2 minutes but mean squared error has been reduced drastically.

For best model for a particular dataset, mean squared error should be very low with acceptable amount of time consumption.

Results with profile IDs 58 and 78 are choosen randomly as test IDs.

	Linear Regression (benchmark model)	Random forest regression
Time consumed	< 1 minute	Apprx. 5 minutes
Mean squared error	0.3558	0.119

If we consider test IDs as 52 and 78 the results are:

	Linear Regression (benchmark model)	Random forest regression
Time consumed	< 1 minute	Apprx. 5 minutes
Mean squared error	0.342	0.121

In this problem the effect of choosing different test IDs is little bit more as it is a time series and data collected might not be started at the same point. It also depends on the state of vehicle and environment during the running.

Justification

The Benchmark model(linear regression model has a mean squared error of 0.3558 , while the final model has mean squared error of 0.119 which is very good for this dataset.

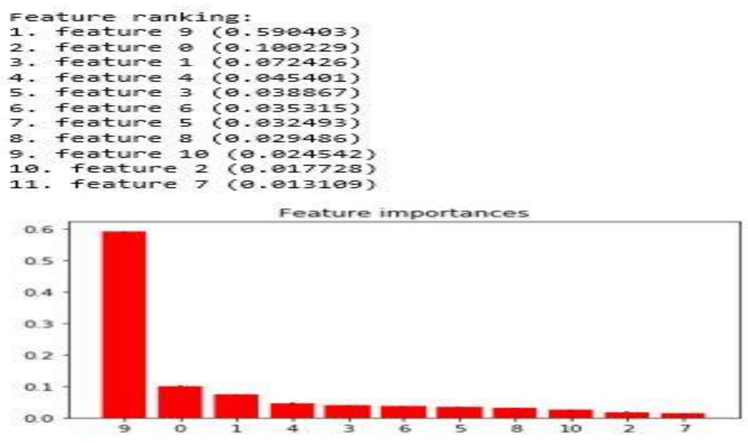
So, clearly final model is stronger model.

I think the model is significant enough for this problem's solution.

Random forest regression model is a robust bagging ensemble method with very low mean squared error for time series dataset.

Conclusion

Free-Form Visualization



```
Feature ranking:
1. feature 9 (0.590403)
2. feature 0 (0.100229)
3. feature 1 (0.072426)
4. feature 4 (0.045401)
5. feature 3 (0.038867)
6. feature 6 (0.035315)
7. feature 5 (0.032493)
8. feature 8 (0.029486)
9. feature 10 (0.024542)
10. feature 2 (0.017728)
11. feature 7 (0.013109)
```

The above figure clearly indicates that only 1 feature i.e. stator_tooth is influencing the rotor temperature with highest percentage of 59%. By this we can conclude that stator tooth data is very crucial in this dataset for prediction of rotor temperature.

But other features are also important when it comes to outliers and some deviation in stator_tooth data. These will improve the prediction by lesser margin but are important in deciding rotor temperature at final stage.

Reflection

The process used for this project can be summarized using the following steps:

1. Get data from Kaggle. The dataset is in csv format so, convert it to a pandas dataframe.
2. As input data for this project is a time series, whole of the data needs to be visualized, trained and tested by grouping the dataset based on profile IDs.
3. Identify important features which are influencing the rotor temperature by visualizing the input data using "matplotlib" or "seaborn".
4. Selected one benchmark model as linear regression model.
5. Train benchmark model to predict rotor temperature.
6. Calculated evaluation metrics. And saw it's criticality on problem statement.
7. Tried with powerful model random regression model and repeated steps 5 and 6.
8. For better performance I used grid search technique to select the best value for parameters, which robust with very good performance on time series dataset.
9. Finally I came to conclusion that random forest regression model is efficient enough for this problem.

Very interesting part of this implementation is splitting of data into groups. As normally all the data set what we get will be splitted directly into training set and testing set but in this case we can't split the data straight a way. Here we need to split the data by grouping them based on their profile IDs as the dataset is time series.

Very difficult task in this was to choose the parameters like number of estimators and max_depth for random_forest regression. Finally grid search technique helped me in finding solution for this problem.

Improvement

As an improvement part of the present implementation I have used XGBRegressor model of XGBoost library which is really very fast and accurate.

It is robust and accurate also. Mean squared error for XGBRegressor is 0.1157 with time less than 1 minute is very good for this dataset.