Sujeeth Shetty 09/01/2020

Open Data Science Conference(ODSC) Hackathon

Improving Efficiency & Production Process of Electric Vehicles using Data Science Techniques

Introduction:

This challenge has been designed to provide you with hands-on understanding of data science problems in commercial EVs' production and optimization in most advanced motor technologies used by companies like Tesla, BMW and Ford.

It is often a challenging and complex task to measure rotor and stator temperatures in commercial electric vehicles. Even if these specific tasks can be completed successfully, these testing processes cannot be classified as economical for manufacturers. Keeping in mind that the temperature data have significant importance on dynamical responses of vehicles and motors' performances, there is an emerging need for new proposals and scientific contributions in this domain.

Consider, one manufacturer of electric cars hired you to propose an estimator for the stator and rotor temperatures and design a predictive machine learning or deep learning model. Such a model could significantly help your new company to utilize new control strategies of the motors and maximize their operational performances. If you build an accurate ML/DL model, the needs of the company for implementing additional temperature sensors in vehicles will be reduced. The potential contribution will directly result in lowering car construction and maintenance costs and will convince the company to invest further in hiring DS experts like you.

The objective of this is project is to predict the rotor/stator temperatures and to use Root-Mean-Square Error(RMSE) to access the fit of regression model.

Initial Considerations:

- The motors are excited by reference torques and reference velocities. These reference signals are achieved by adjusting motor currents ("i_d" and "i_q") and voltages ("u_d" and "u_q") within appropriate control strategy.
- Temperature estimations should be real-time, and not based on future values for current predictions. Real-time predictions shall protect the motor from overheating.
- The motor torque increases in inverse proportion to the decreased temperature.
- A steady state of a motor can be achieved faster at lower temperatures.
- Phase currents increase with increased magnet temperature.

About the Data:

Each row in the csv files represents complete measurement information from sensors in one-time step and one row is recorded every 0.5 seconds. Individual measurement sessions last between 1 and 6 hours and can be identified with the "profile_id" column. The training dataset has 9 predictors, 4 target variables(pm, stator_yoke, stator_tooth & stator_winding) and 846,368 records. The following table provides variables of interest and their short descriptions.

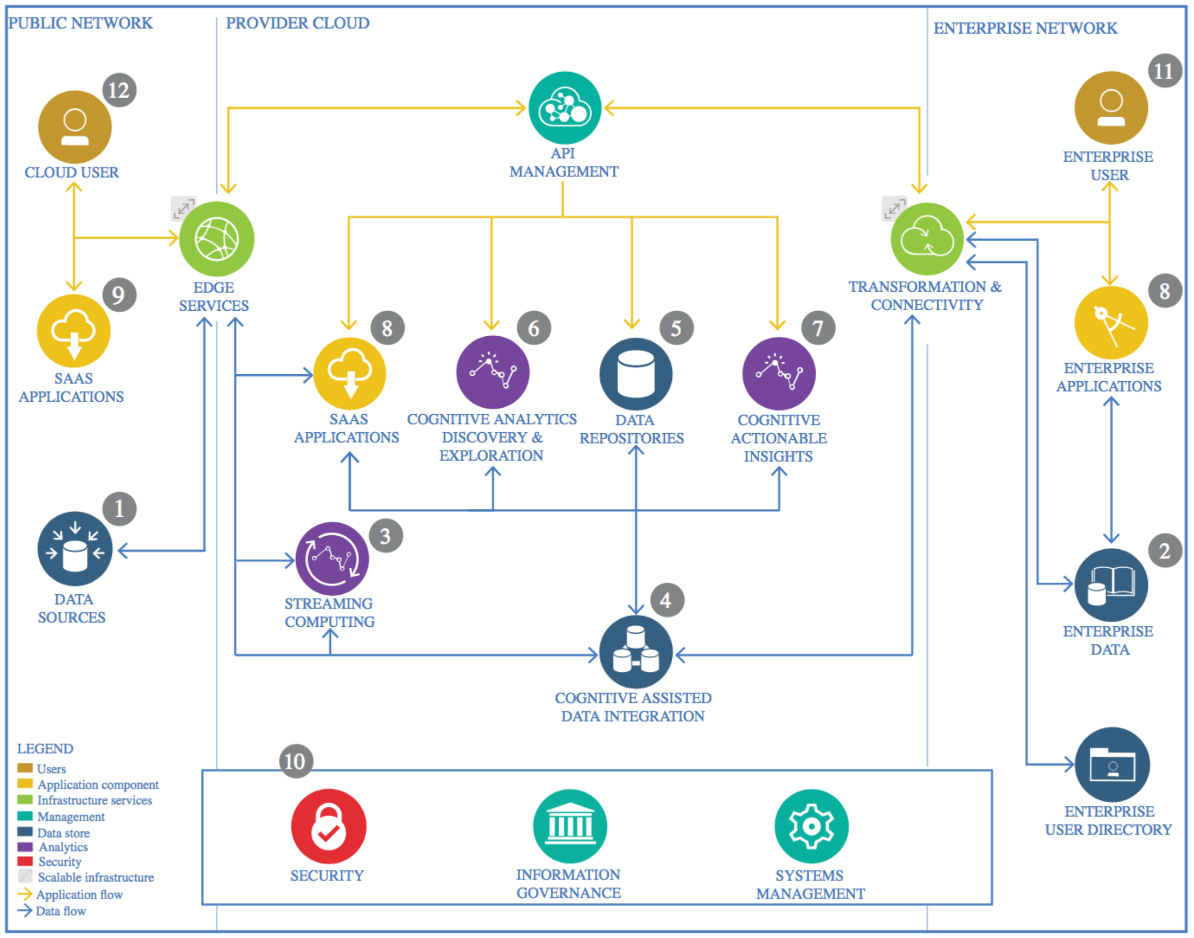
Variable	Description
Ambient	Ambient temperature – measured by a thermal sensor
coolant	Coolant temperature measured at outflow.
u_d	Voltage d-component
u_q	Voltage q-component
motor_speed	Motor speed
torque	Torque induced by current.
i_d	Current d-component
i_q	Current q-component
pm	Permanent Magnet surface temperature (the rotor temperature) – measured with an infrared thermography unit
stator_yoke	Stator yoke temperature – measured by a thermal sensor.
stator_tooth	Stator tooth temperature – measured by a thermal sensor.
stator_winding	Stator winding temperature – measured by a thermal sensor.
profile_id	Each measurement session with a unique ID.

The dataset is sourced from the following publications:

Kirchgässner, Wilhelm & Wallscheid, Oliver & Böcker, Joachim. (2019). Empirical Evaluation of Exponentially Weighted Moving Averages for Simple Linear Thermal Modeling of Permanent Magnet Synchronous Machines.

Kirchgässner, Wilhelm & Wallscheid, Oliver & Böcker, Joachim. (2019). Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors.

Architecture Components Overview:



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

Project Outline:

Tools

The project was run on **Google Cloud AI Platform Notebook**. It offers an integrated and secure JupyterLab environment for data scientists and machine learning developers to experiment, develop, and deploy models into production. This instance come pre-installed with the latest data science and machine learning frameworks.

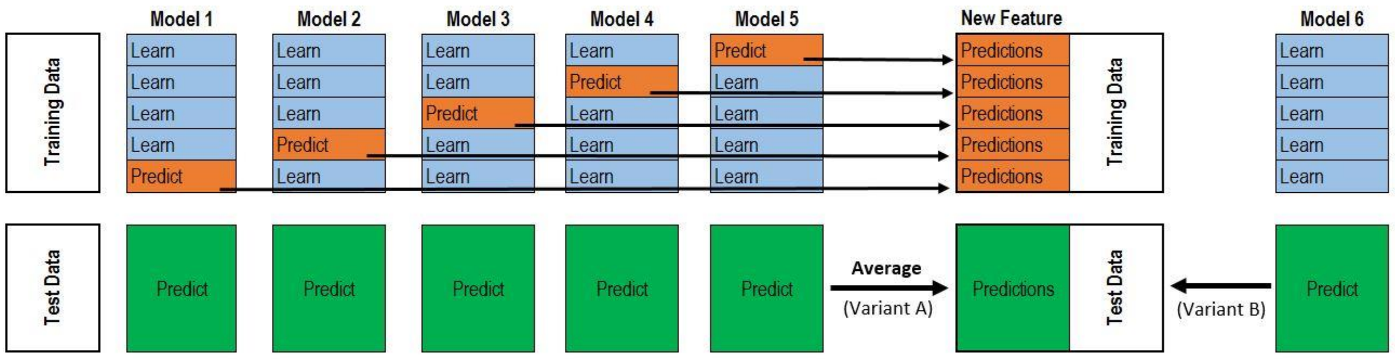
Algorithm Implementation

Following algorithms is implemented using scikit-learn package is evaluated using repeated k-fold cross-validation.

- LASSO Regression
- Elastic-Net Regression

- Random Forest Regressor
- XGBoost Regressor
- Light Gradient Boosting Machine

The model performance of each algorithm is reported using the Root Mean Square Error. Next, we combined all these models into a single ensemble model using stacking. We began with this simple approach of averaging base models. We built a new class to extend scikit-learn with our model and also to leverage encapsulation and code reuse. We just averaged four models here **Lasso, ENet & Random Forest**. This encouraged us to go further and explore less simple stacking approach called **Meta Ensembling**.



(Image taken from Faron)

As shown in the pic, in this approach, we added a meta-model on averaged base models and used the out-of-folds predictions of these base models to train our meta-model.

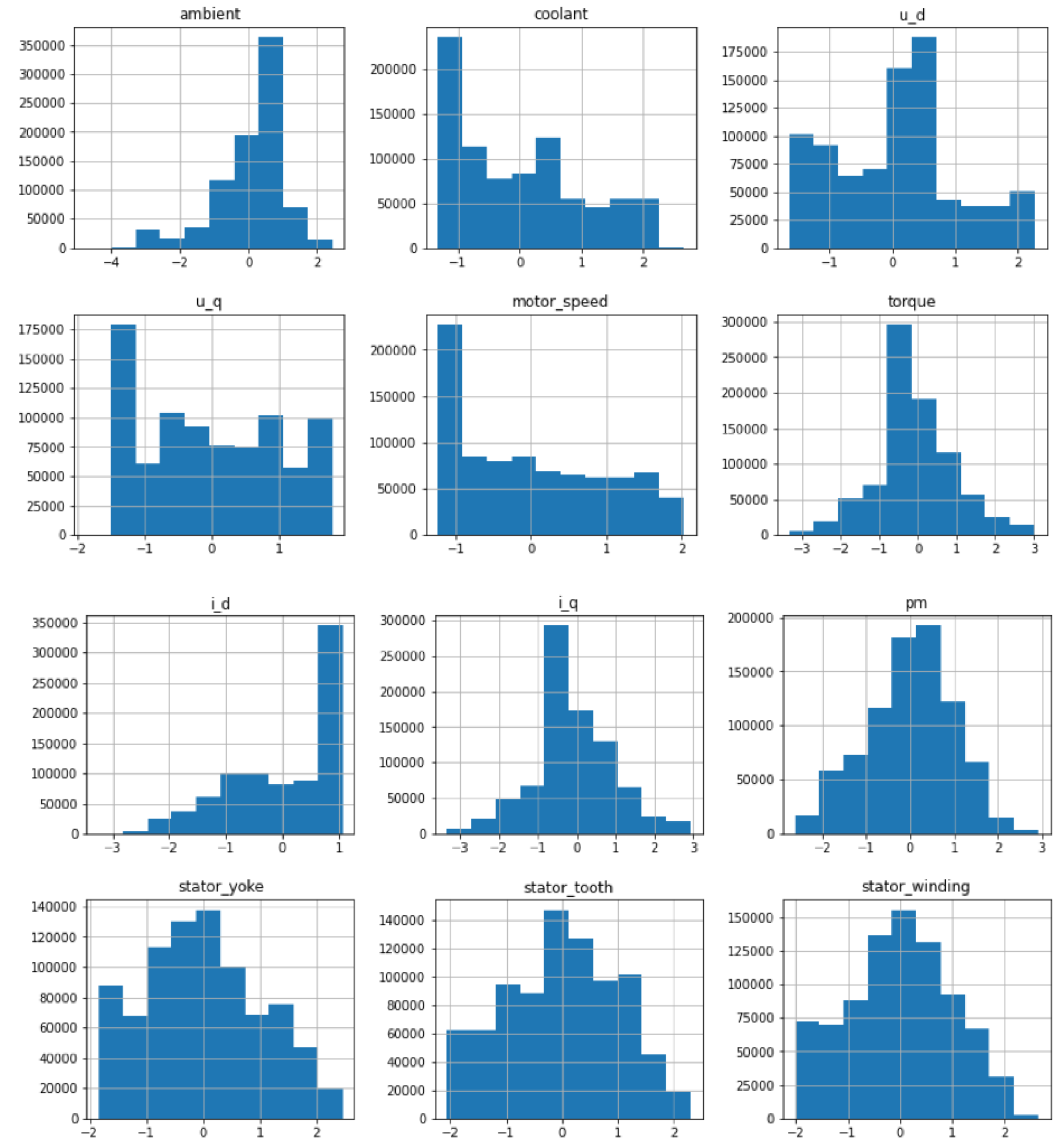
The procedure, for the training part, is described as follows:

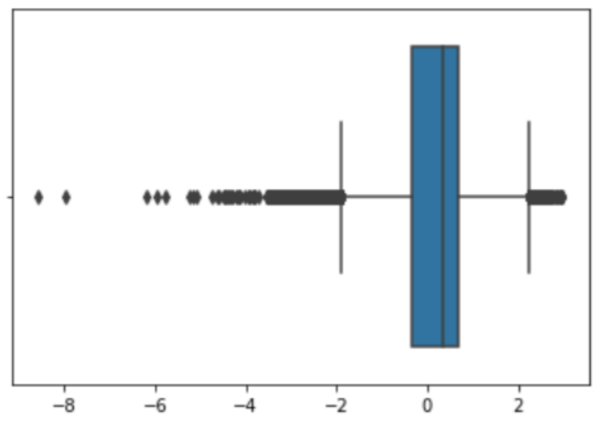
1) Split the total training set into two disjoint sets (here **train** and .**holdout**)
2) Train several base models on the first part (**train**)
3) Test these base models on the second part (**holdout**)
4) Use the predictions from Step 3 (called out-of-folds predictions) as the inputs, and the correct responses (target variable) as the outputs to train a higher-level learner called **meta-model**.

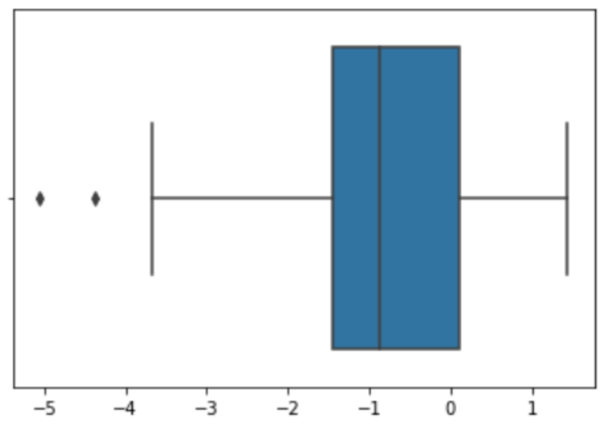
For the prediction part , we averaged the predictions of all base models on the test data and used them as meta-features on which, the final prediction was done with the meta-model.

Data Preparation and Exploratory Data Analysis

The objective of this project is to accurately predict Rotor and Stator Temperatures. From the figure you can see that, the distribution of the pm, stator_yoke, stator_tooth & stator_winding is normally distributed, so scaling was not performed on this dataset. Instead we did outlier analysis using Boxplot and removed outliers based on 'ambient' column data.

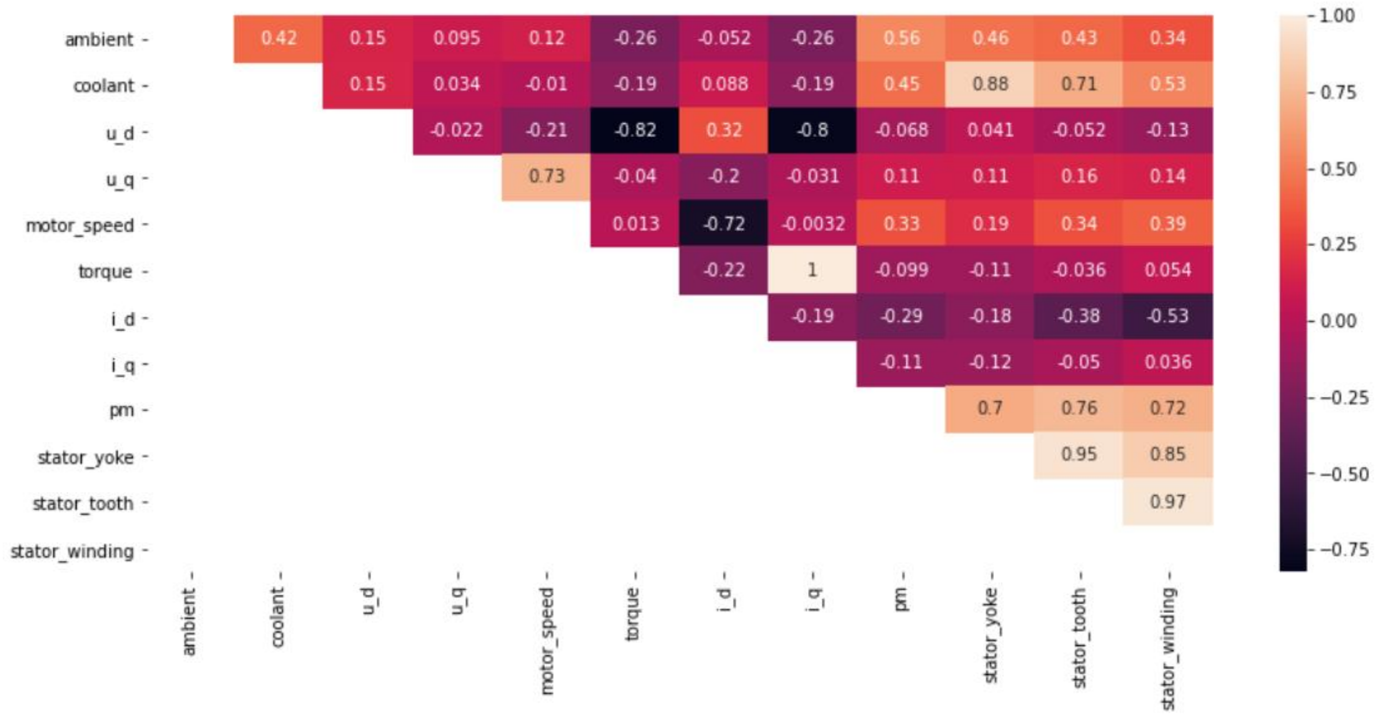

ambient
coolant
u_d
u_q
motor_speed
torque
i_d
i_q
pm
stator_yoke
stator_tooth
stator_winding





Outliers analysis(Ambient) on Train using Box-Plot *Outliers analysis(Ambient) on Test using Box-Plot*

The correlation plot indicated that torque and i_q is highly correlated. To avoid multicollinearity, we removed torque from the dataset.



As the training and test dataset are from different distribution, 25% of test data were added to the training set and 25% were taken for validation(this is to maintain test/validation from same distribution). The final prediction was performed on the entire test dataset and all four target variables were individually trained/predicted.

Implementation and Results

Mainly 3 experimentation was undertaken using the dataset in which we explored the effect of individual model(Lasso, Elastic-Net, XGBoost, Light-GBM), Model averaging in an ensemble approach and Meta Ensembling. Finally, the results of the experiments were discussed in the results section and further improvement opportunities were discussed.

Implementation

1. Base Model

XGBoost with dept of 4 was trained to create a baseline and the model was fitted using MultiOutputRegressor to predict all 4 target variables at once. We achieved good result on validation dataset with RMSE of 0.25505. However, we had high variance problem as the test error was 0.3737 and the model was able to capture only 80.36% of variance.

Modelling for Stacked Regression we have trained following models and calculated Cross-Variation score.

Model		Lasso	Elastic-Net	Random Forest	XGboost(Tuned)	LGBM
CV Score(Train)	pm	0.7327	0.7327	0.7953	0.7497	0.7483
	Stator_yoke	0.4086	0.4086	0.4299	0.4199	0.3986
	Stator_tooth	0.5742	0.5742	0.5647	0.5603	0.5384
	Stator_winding	0.6494	0.6493	0.6355	0.6031	0.5803

2. Stacking

We began with this simple approach of averaging base models. We averaged 3 model here **Lasso, ENet & Random Forest**. In the Meta Ensembling, we added meta-model on the averaged base model. To make the two approaches comparable (by using the same number of models) , we just average **ENet, RandomForest and XGBoost**, then we add **Lasso as meta-model**.

Model		Average Base Model	Meta Ensemble
CV Score(Train)	pm	0.7268	0.7593
	Stator_yoke	0.3983	0.4572
	Stator_tooth	0.5481	0.6163
	Stator_winding	0.6221	0.6635

3. Ensembling Stacked Regressor, XGBoost & LightGBM

We picked the Meta Ensemble model, tuned XGBoost and LightGBM for the final prediction. We achieved an **overall RMSE of 1.5918** on test dataset and the RMSE score of each target variable on train/test is captured below.

Target Variables	pm	stator_yoke	stator_tooth	stator_winding
RMSE(Train)	0.3036	0.2108	0.2838	0.3279
RMSE(Test)	0.3513	0.5084	0.4118	0.32

Results

Through experimentation, we found that stacking multiple models and ensembled model drastically improves the performance. Since the test/train dataset are from different distribution we faced high variance problem, but Ensembling reduced the variance to a great extent. There is still potential to improve the predictions of stator_yoke and stator_tooth. This could be achieved by stacking Neural Network.