Smart Energy In The Information Age Project Report
Electric Car Range Prediction and feature analysis using Machine Learning

Harshith Reddy Kallu
112674629

Viraj Kamat
112818603

Rohit Rawat
112963417

Objective
Given a dataset containing the telemetry of an Electric Vehicle, that includes parameters such as battery charge, speed, and other environmental factors we wish to use machine learning models to predict the range of the Electric Vehicle based on the current charge state and the exogenous factors. While also using conventional machine learning models to perform our predictions we also wish to propose and implement a novel approach that involves ensembling multiple machine learning models to further improve the accuracy of our predictions.

Approach & Challenges
We faced a few challenges with respect to our task, some of which are highlighted below:

1. Cleaning our dataset, removing outliers would be a concern, unprocessed data could adversely affect the performance of our model.
2. Resampling our dataset such that second timesteps would be downsampled to minute timesteps correctly without losing important information in our features.
3. The choice of our singular model was our primary concern, while Deep learning models are the trend, we did not have a concrete idea of which standard machine learning models would be appropriate.
4. The method to obtain a better performing model by ensembling singular models was not clear to us; do we weight each model by a parameter or do we simply take an average.
5. What metric would be used to evaluate the performance of our model and what features would matter in training our model, introducing arbitrary features could worsen the performance of our model during model training.

Dataset
- We received our dataset from Dr. Maximilian Holland whom we contacted via mail after a lot of search over Google gave us no results. The dataset in question is that of the **Nissan Leaf** Electric Car with a 30KWh capacity. The data was captured from the car using the Leaf spy Android App and saved in a CSV format. We found that the dataset contained information about the car for every second of the travel made.

  We faced the following issues with it:

1. Since a car is stopped and started on a particular time there would gap in the logged data, this would create an issue when resampling

2. To overcome this we could simply copy the past data over those timesteps that were absent, however, repeated timesteps affected time-series based prediction models such as L.S.T.M.

- We decided it was best to resample the data from second to minute intervals and to keep the time-related gaps in the dataset. We used the Python Pandas module to read our data and to resample our data from second timesteps to minute timesteps.

- After resampling the data our next step was to find the important columns and come up with a formula for the range of the car based on the current state of the car. We found the following columns to be useful:

1. Odo(km) - The distance traveled by car up until that point
2. SOC - State of the charge in percent
3. Ambient Temperature in Celsius
4. A/c power used in Watts
5. Aux power consumed in Watts
6. Motor power consumed in Watts
7. Tire pressure
8. Motor Temperature in Celsius
9. Torque

- The use of Odometer distance was to accurately gauge the distance traveled per timestep, this would be important for our range prediction. Tire pressure is a well-known factor known to affect the mileage of a well, this along with electricity consumed by other components such as Motor power, A/C power, etc would be critical for our machine learning process. Also important would be the exogenous factors such as ambient temperature and motor temperature that would affect the performance of the vehicle.

- We also included other parameters that would help our machine learning model perform better. Almost all the attributes were numerical with very few empty cells.

- In order to define the range left in our car which would be our true values that would be used in our predictions, we came up with the following formula:

  Odometer Difference = Odo reading at current timestep - Odo reading at the previous timestep

  S.O.C Difference = S.O.C reading at current timestep - S.O.C reading at the previous timestep

  Expected Range = (OdoMeter Difference/S.O.C Difference) * S.O.C at current timestep

  If the car was stationary, both the Odometer Difference and the State of Charge Difference would be zero in which case we would calculate the range by multiplying the average known mileage for Nissan Leaf i.e ~ 1.6km per percent charge to obtain the expected range of our car.

- **Parametrizing Driver behavior**

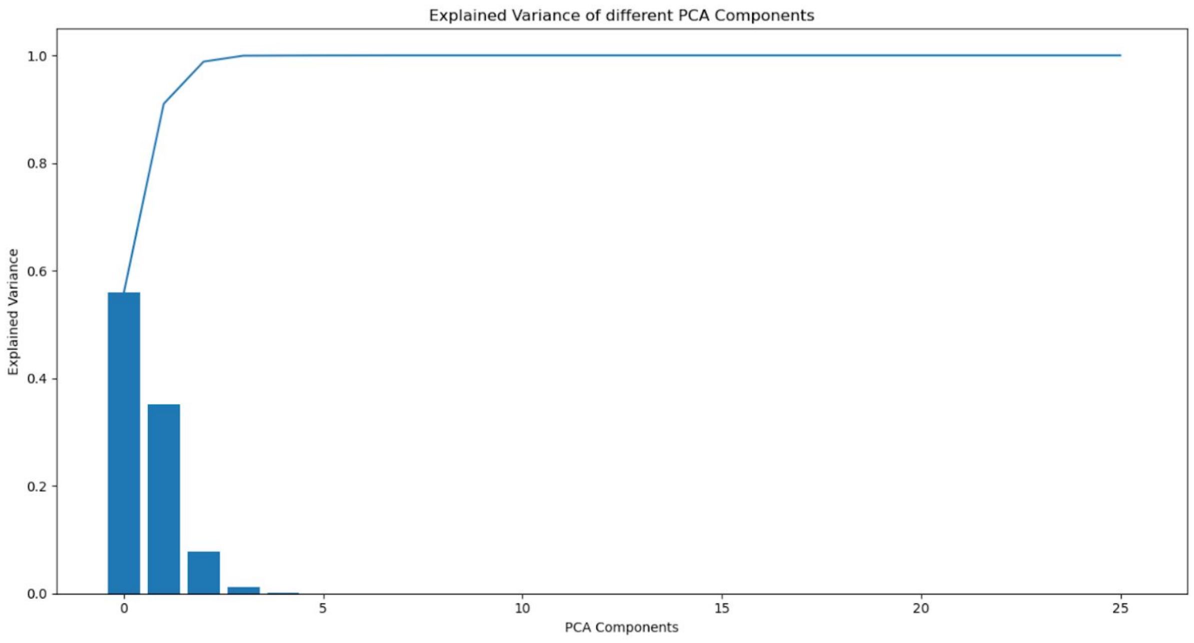
We had to quantify the driver's aggression i.e how fast the car was being driven. The following steps were used to do so :

1. We first calculated the percentile rank of each of the data-points based on speed based with the help of the Pandas pct function
2. For each of the data-points, we initialized a field called "driver_aggression"
3. Based on the percentile rank, if the car speed was more than 90 percentile of the other data-points it was given an aggression value of 10. Similarly, if the speed was more than 70 percentile of the values it was given a score of 8 and so on.
4. This would help the machine learning model better understand the higher speeds equal to higher driver aggression

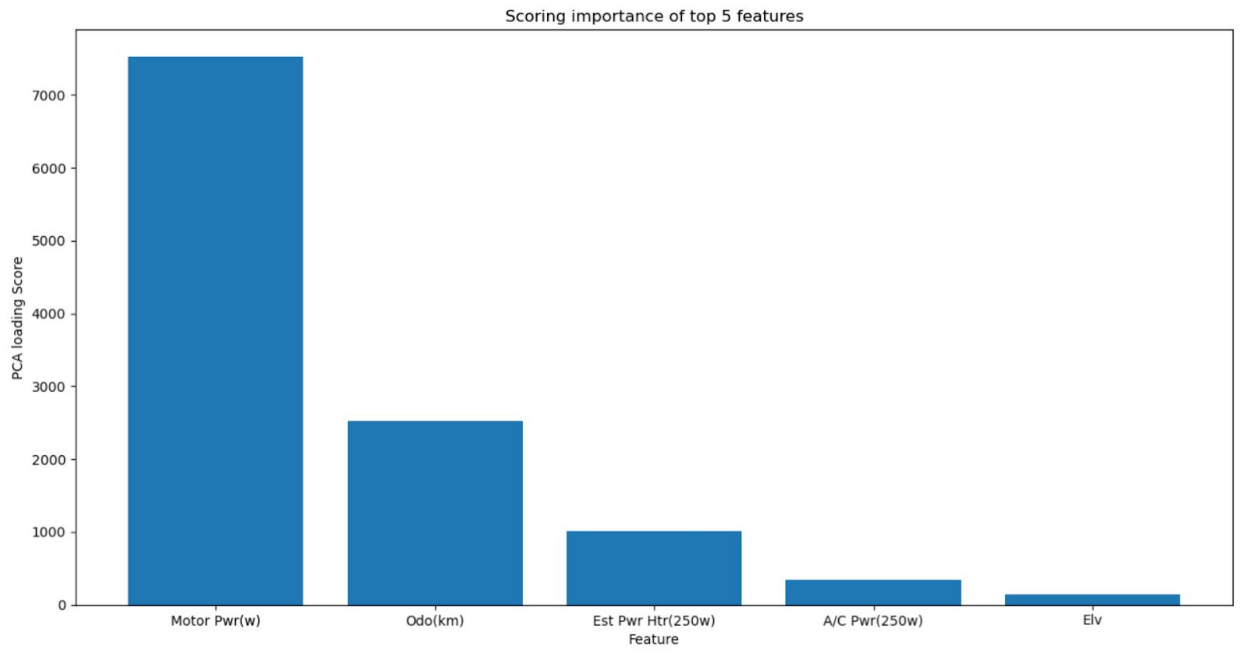
Data and Features Analysis

Principal Component Analysis of Features:

- PCA is used for dimension reduction i.e. it helps in understanding the most important features from the dataset that best explains the variance of the whole dataset. It projects the values of features on some fixed number of dimensions which is also known as PCA Components. Then on the basis of these projected values on the components, it compares the variability of the features in the dataset and computes it as PCA loadings. On the basis of the loadings score, we can estimate the most important features.
- We implemented the PCA on the dataset and drew the following scree plot to understand the behavior of the components:



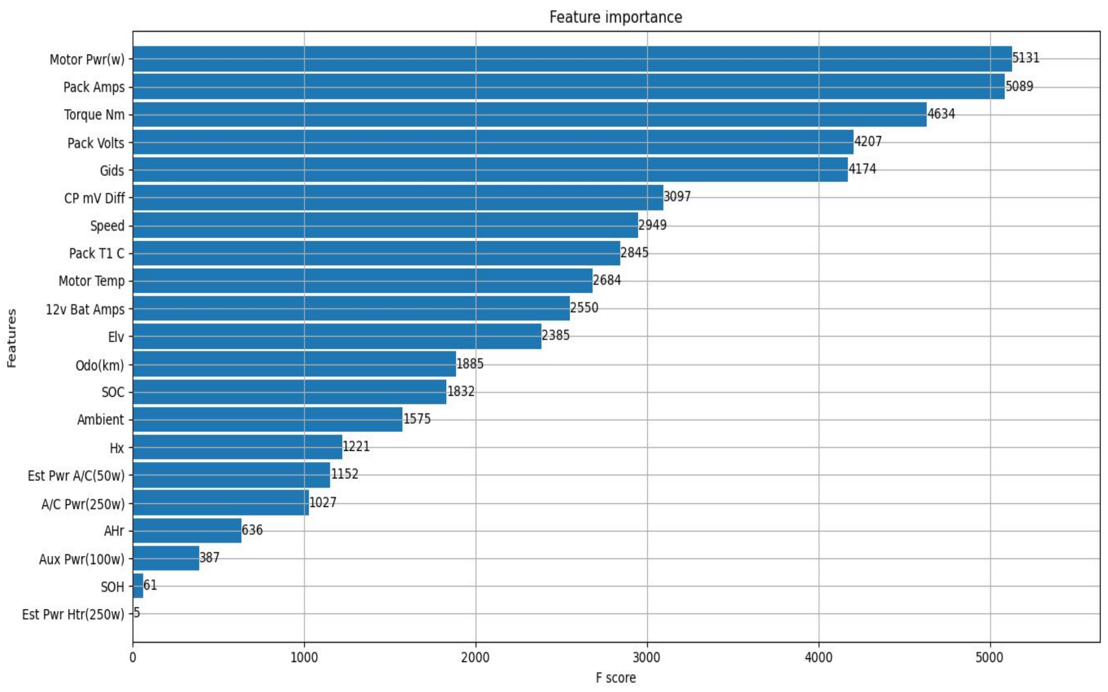
- We can observe from the line plot of cumulative explained variance that 75% of the variability of the dataset can be implemented using the first two to three PCA components. On the basis of these components, we calculated the PCA loadings score of all the features and selected out the top 5 features that explain the variability of the whole dataset.
- Those features or columns are: 'Motor Pwr(w)': Drive motor power, 'Odo(km)': Odometer reading, 'Est Pwr Htr(250w)': Estimated Cabin PTC Heater power in 250-watt units, 'A/C Pwr(250w)': Power used by the Air Conditioning System power in 250-watt units, and 'Elv': Elevation which is received from the GPS hardware in the Android device. The Scoring importance of these top 5 features is shown below:



- We can estimate that the first feature(Motor Pwr) is strongly correlated to the rest of the features while showing a PCA loading score of around 7000. After that Odometer reading of the distance traveled is showing good correlation and so on.
- By understanding the Principal Component Analysis of our data, we can presumably say that these features are mostly going to explain the range of the Electric Vehicle.

XGBoost Features Importance:

- We also used the XGBoost machine learning model to perform feature analysis, before doing so we tuned the hyperparameters of the model and when we had the best-trained model with the dataset we explored the importance of the features.

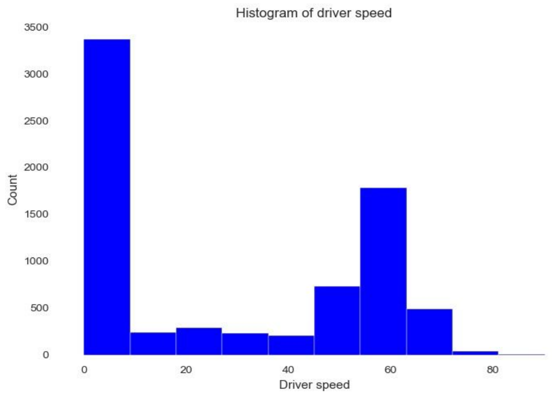


- The XGboost library has an inbuilt plot_importance method that takes our best-trained model as the parameter. As can be seen above for our best-trained model the score for the most important features is given. We can draw the following inferences:

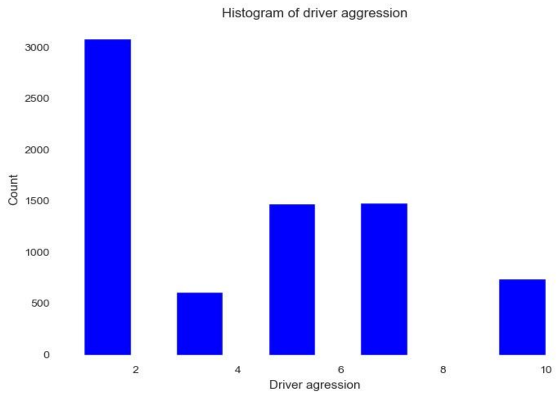
1. Power related metrics such as Amps, Motor Power, Voltage, and Voltage difference play a crucial role in our Machine Learning Model which would make sense since we are training our model with features from an electric car.
2. The torque of a car is an important feature when the torque of a car is high more power is consumed thereby affecting mileage, when the torque is low the mileage is higher.
3. Ambient temperature and motor temperature are also important features, the performance of our car is dependent on exogenous factors that would affect the mileage.

Driver's behavior analysis:

- Above, we have the histogram for the driver speed and driver aggression. The driver speed was to be found to be mostly in the 60 km/hour range which was optimal for the car which in our case is the Nissan Leaf. We did not notice drivers being very aggressive but there were instances where the driver was indeed faster than usual and scored a 10 on the driver aggressions scale. We expect higher driver speed to adversely affect the mileage of the car.



Histogram: Driver's Speed



Histogram: Driver's Aggression

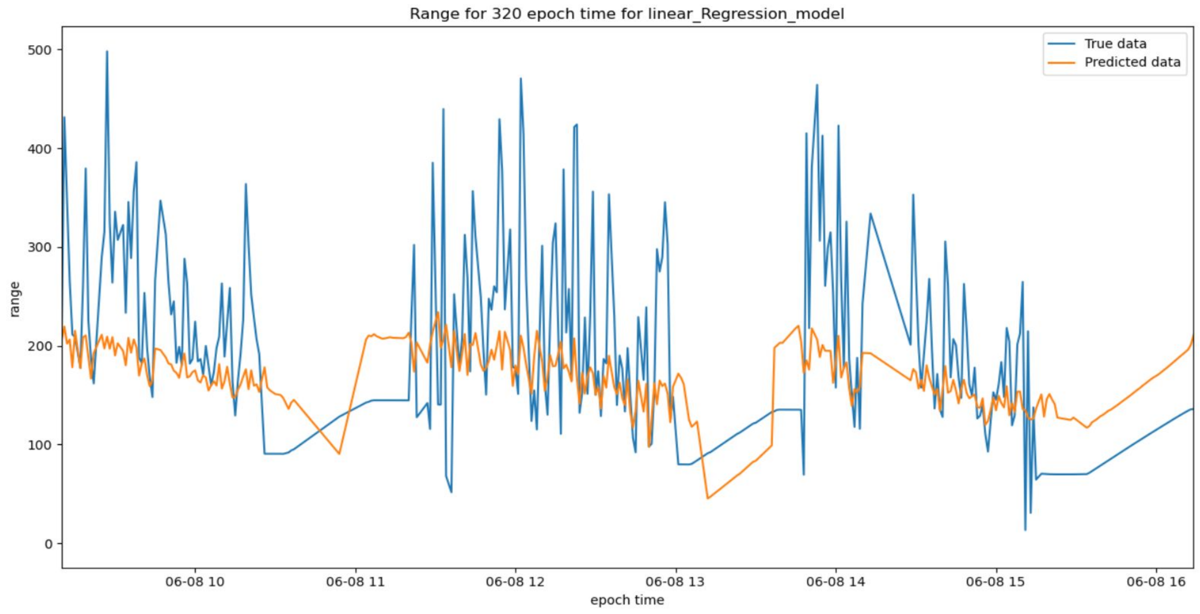
Individual Models or Non - Ensembled Model

- We have tried different types of regressors i.e. Linear Regression, AdaBoost Regressor, Bagging Regressor, Extra-Trees Regressor, and a Recurrent Neural Network i.e. Long short-term machine. We have used the Root Mean Squared Error(**RMSE**) value to compare these models, that is calculated as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Where y_hat: Predicted values of Range of EV,
y: True values of Range of EV,
n: Number of values

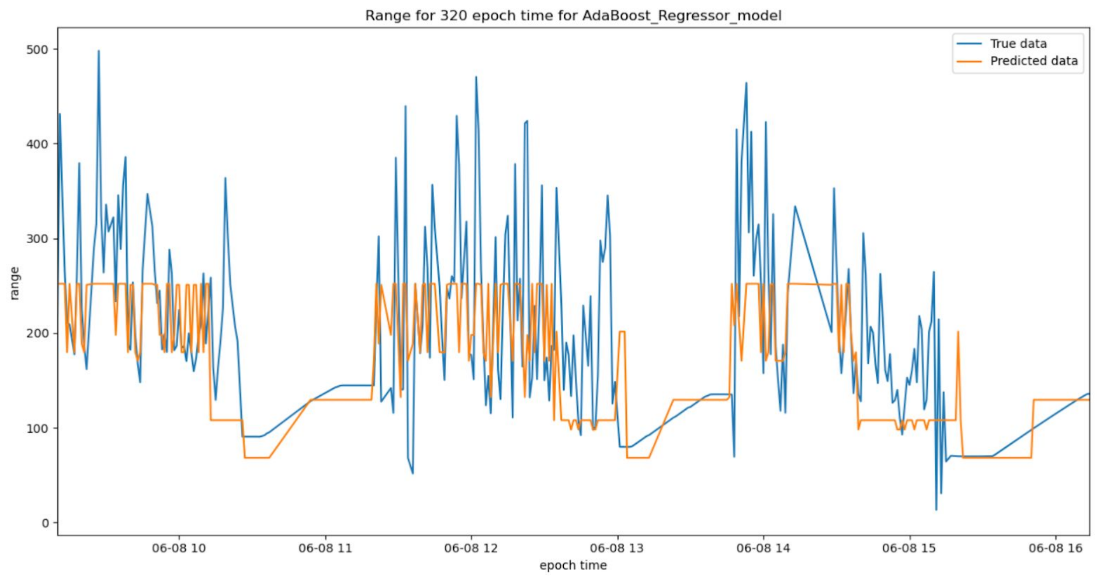
- **Linear Regression Model**



We started with fitting our dataset on a Linear Regression Model, which tries to fit the data on a line. For linear Regression, we got an **RMSE score of 81.78280563366944.**

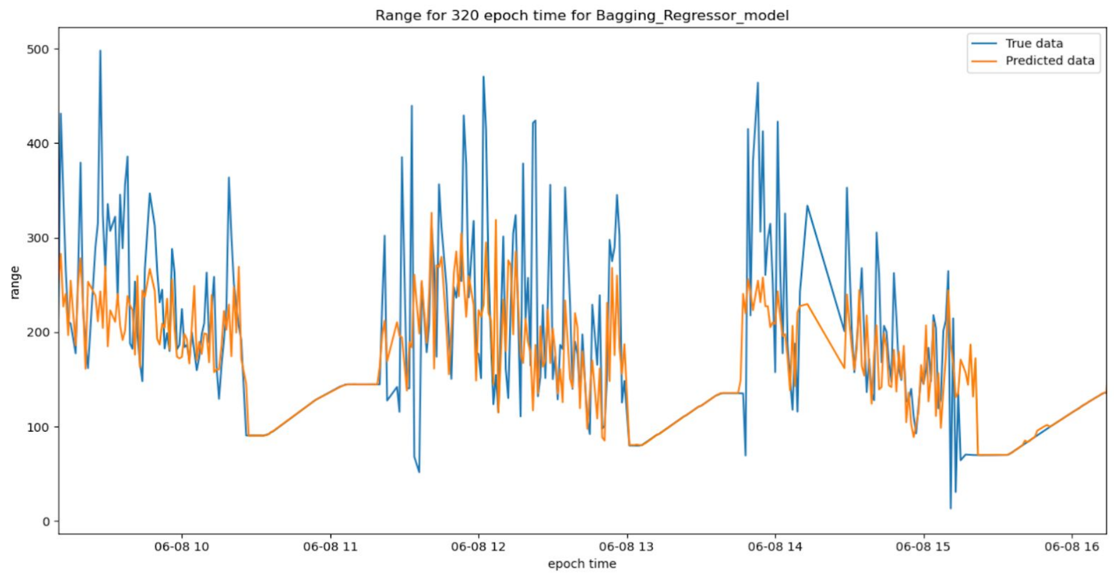
- **AdaBoost Regressor Model:**

Next, we have used the AdaBoost Regressor Model to fit our data, which is a meta-estimator. It fits a regressor on the dataset and then again fits the copy of that regressor on the dataset but with adjustment of weight of instances on the basis of errors. For AdaBoost Regression, we got an **RMSE score of 75.63687847244836** which is better than Linear Regression.



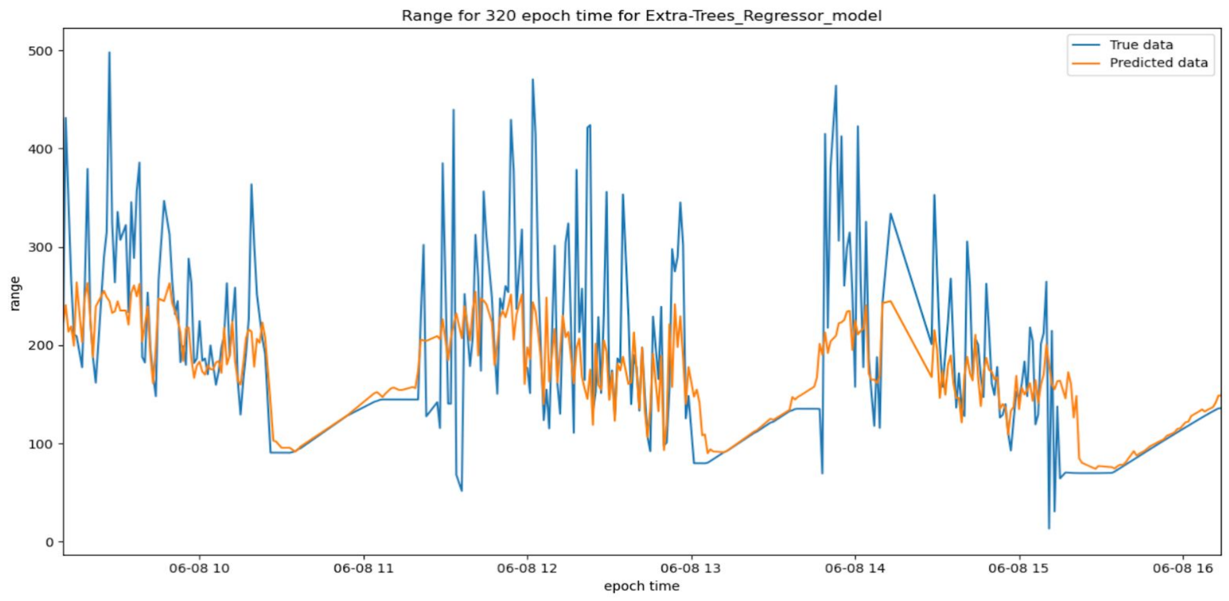
- **Bagging Regressor Model**

The Bagging Regressor Model is also a meta-estimator like AdaBoost Regressor but it fits the regressors on random subsets of the original data and chooses the regressor by voting. We got the **RMSE Score of 70.94967401057501** for this model.

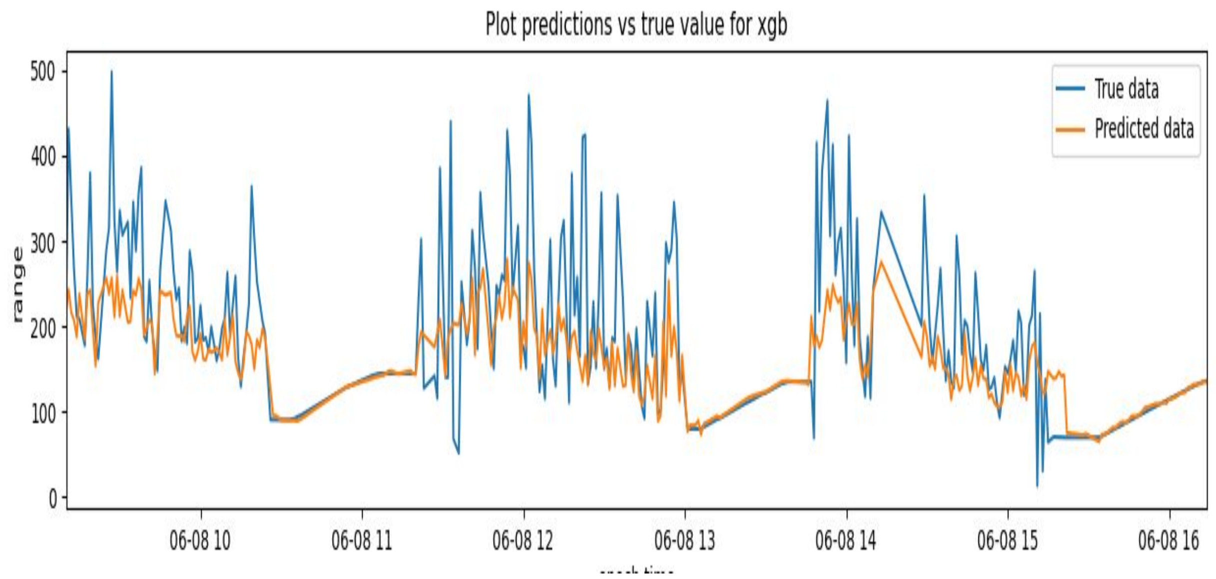


- **Extra-Trees Regressor Model**

  This model fits a number of randomized decision trees on various sub-samples of the dataset. It uses averaging to prevent over-fitting. That's the reason it performed well with respect to the above models. We got an **RMSE Score of 68.20844281020672** for ExtraTreesRegressor_model.



- **XGBoost Regressor Model**



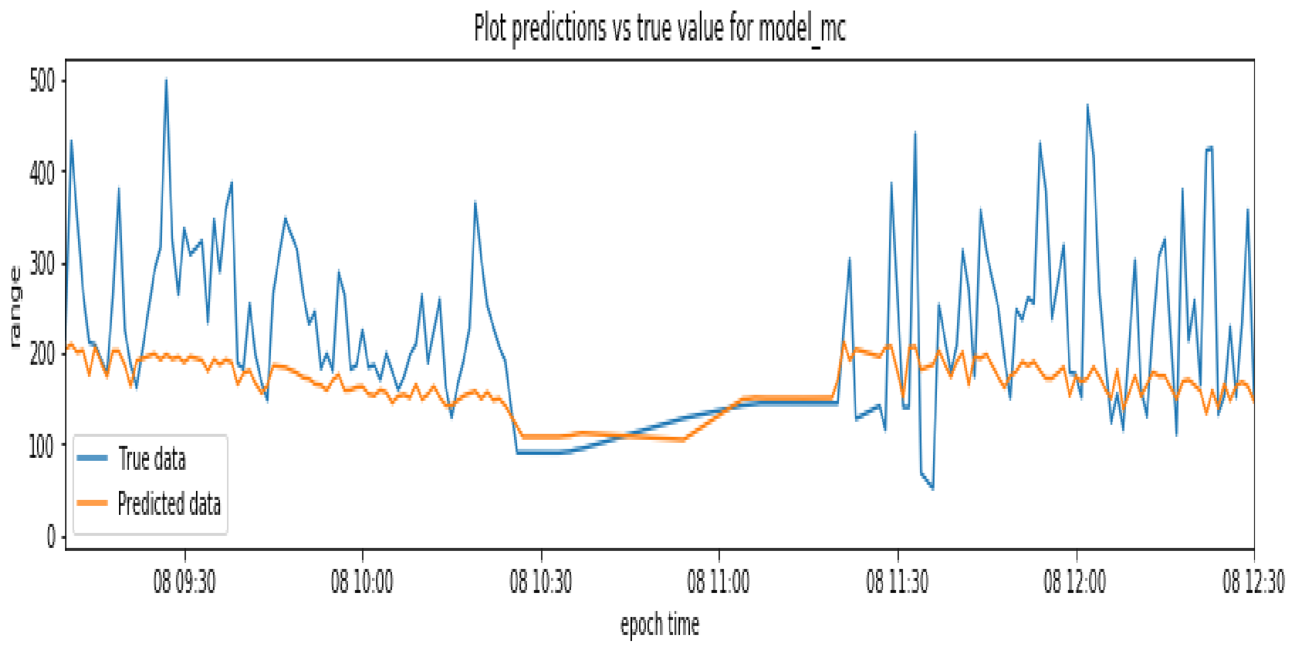
Of all the singular models we built that had no ensembling, Extreme gradient Boost provided the least RSME of them all.
XGBoost is a decision tree based boosting model that combines multiple decision trees in a stage-wise fashion. Each decision tree is a weak learner which would make mistakes when learning on a subset of training data, the next decision tree would then

make an attempt to improve upon this by correcting for errors and biases i.e it tries to minimize a loss function.
When training the model we had to focus on tuning the hyperparameters which included the following :

1. Number of iterations
2. Max depth of the decision tree
3. Max leaf nodes of the decision tree
4. The number of columns to be randomly sampled for a decision tree

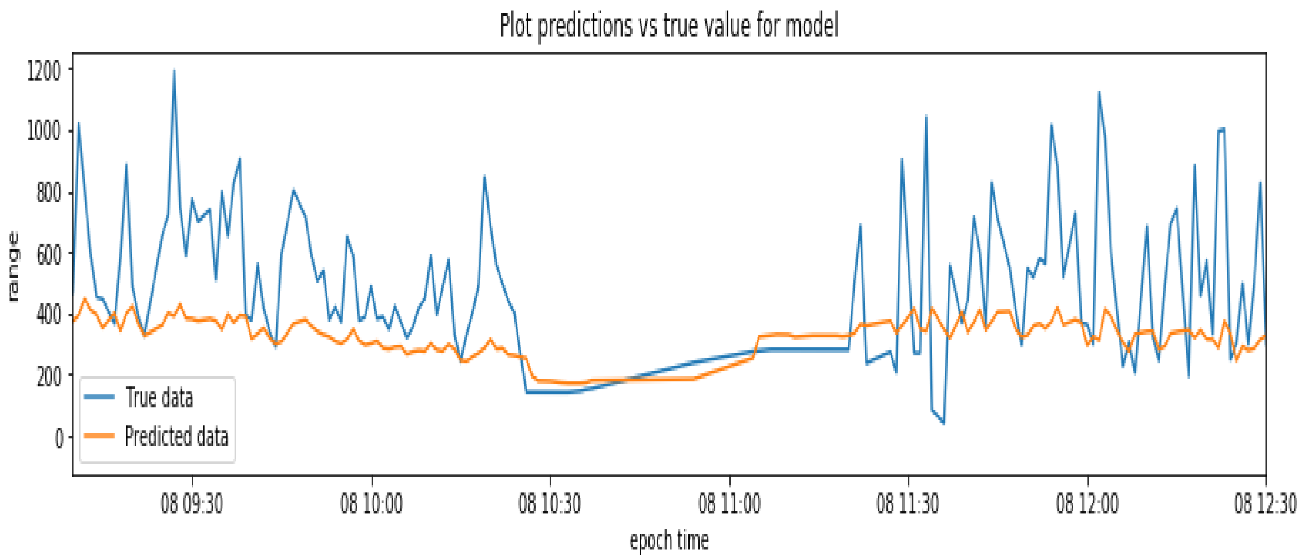
There are several more, but we found these to be the most important. After tuning these parameters on the training set by trying to lower the Root Mean Squared Error yet not trying to overfit the data the following was the **R.M.S.E on the test data: 66.45261942378283**.

- **Neural Network Model**
  We use a 5 layer artificial neural network to train the neural network on the features to predict the range. Our neural net consists of 4 dense layers and 1 dropout layer. The dense layer is a deeply connected neural network layer and the number of units in the output layer is 200.
  We use ReLu as our activation function. We also have a dropout layer to avoid the overfitting of the data. It is a form of regularization. We set the probability of dropping out nodes to 0.2, i.e. we drop 20% of the selected nodes.
  We got an **RMSE Score of 70.82734268643283**.



- **Long Short Term Memory Model(LSTM Model)**
  LSTM is designed to avoid long term dependency problems. In LSTM the repeating module has a different structure. Instead of having a single neural network, there are

four. The key parameter in LSTM is the cell state which consists of some linear interactions.
LSTM can add or remove information from the cell state. Gates enable us the option to let information through. The sigmoid function which outputs 0 or 1 helps us decide how much of each value would be let through the gate. And we got an **RMSE score of 198.45261437643283**.



- **Optimizations Techniques:**

We used the following optimizers to optimize the loss function of the Neural Networks that we have used i.e. LSTM and 5 layered-NN:

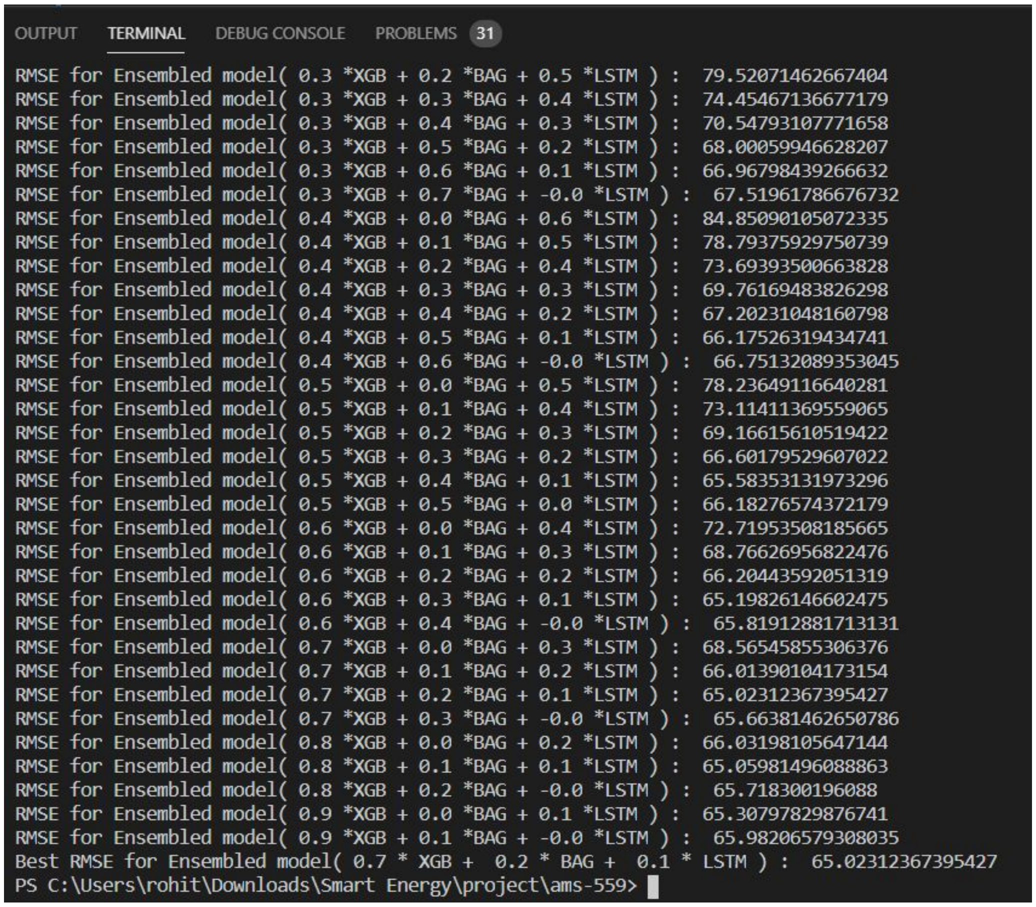
1. **Adam:** It is an adaptive learning rate algorithm that calculates the learning rate specifically for training deep neural networks for each parameter. It also has the advantage of Adagrad which works well with sparse gradients but performs badly in non-convex optimization and RMSProp which tackles some of the issues of Adagrad and works really well in the online setting.
2. **SGD:** Stochastic gradient descent optimizer computes the gradient of the cost function not in a single update operation which is its advantage over batch gradient descent. It can be used to learn online. With SGD fluctuation, we could potentially reach a better minimum but also face the problem of overshooting, which is why we keep reducing the learning rate with iterations so that we end up at the global minimum.
3. **RMSProp:** It is an adaptive learning rate method. It was developed from the need to resolve Adagrad's radically diminishing. RMSProp divides the learning rate by an exponentially decaying average of squared gradients. A good default value for learning rate η is 0.001

Ensembled Model

- Up to this point we trained and used individual models i.e. without ensembling of more than one model. Here we tried combining more than one model to calculate the predicted values for the Range of Electric Vehicle.

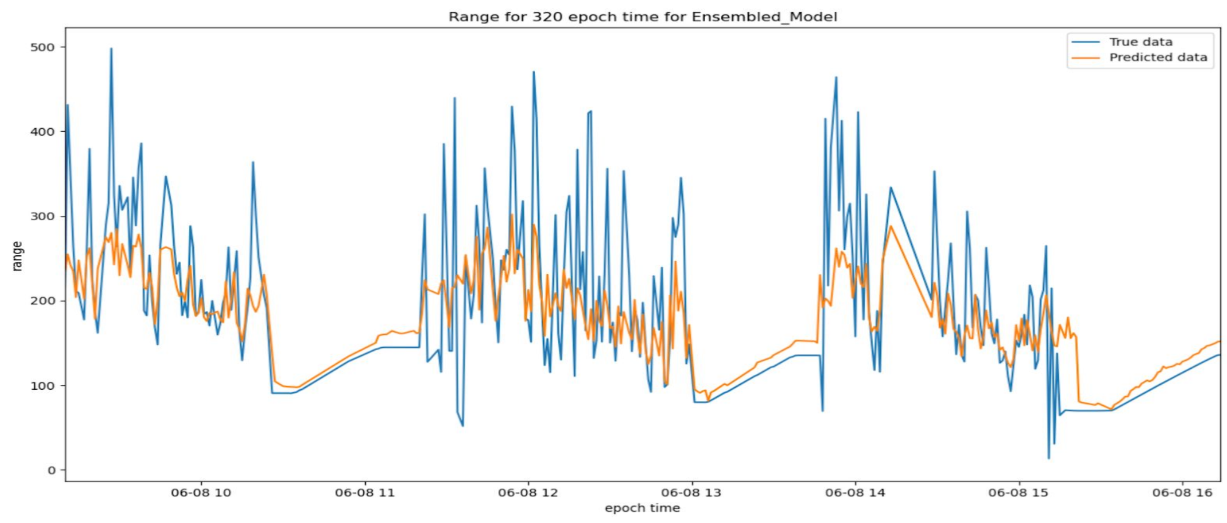
- We have tried the combination of XGB Regressor, Bagging Regressor, Extra-Trees Regressor, LSTM with AdaBoost, LSTM with RMSProp Optimizer and NN with Adam Optimizer because these models gave better RMSE scores individually.

- Out of many combinations of mentioned models, we got the best results for the combination of **XGB + BAG + LSTM with Adam**. Then we calculated the best weights for these models which estimate the least RMSE Score as shown in the figure below.



```
RMSE for Ensembled model( 0.3 *XGB + 0.2 *BAG + 0.5 *LSTM ) :  79.52071462667404
RMSE for Ensembled model( 0.3 *XGB + 0.3 *BAG + 0.4 *LSTM ) :  74.45467136677179
RMSE for Ensembled model( 0.3 *XGB + 0.4 *BAG + 0.3 *LSTM ) :  70.54793107771658
RMSE for Ensembled model( 0.3 *XGB + 0.5 *BAG + 0.2 *LSTM ) :  68.00059946628207
RMSE for Ensembled model( 0.3 *XGB + 0.6 *BAG + 0.1 *LSTM ) :  66.96798439266632
RMSE for Ensembled model( 0.3 *XGB + 0.7 *BAG + -0.0 *LSTM ) :  67.51961786676732
RMSE for Ensembled model( 0.4 *XGB + 0.0 *BAG + 0.6 *LSTM ) :  84.85090105072335
RMSE for Ensembled model( 0.4 *XGB + 0.1 *BAG + 0.5 *LSTM ) :  78.79375929750739
RMSE for Ensembled model( 0.4 *XGB + 0.2 *BAG + 0.4 *LSTM ) :  73.69393500663828
RMSE for Ensembled model( 0.4 *XGB + 0.3 *BAG + 0.3 *LSTM ) :  69.76169483826298
RMSE for Ensembled model( 0.4 *XGB + 0.4 *BAG + 0.2 *LSTM ) :  67.20231048160798
RMSE for Ensembled model( 0.4 *XGB + 0.5 *BAG + 0.1 *LSTM ) :  66.17526319434741
RMSE for Ensembled model( 0.4 *XGB + 0.6 *BAG + -0.0 *LSTM ) :  66.75132089353045
RMSE for Ensembled model( 0.5 *XGB + 0.0 *BAG + 0.5 *LSTM ) :  78.23649116640281
RMSE for Ensembled model( 0.5 *XGB + 0.1 *BAG + 0.4 *LSTM ) :  73.11411369559065
RMSE for Ensembled model( 0.5 *XGB + 0.2 *BAG + 0.3 *LSTM ) :  69.16615610519422
RMSE for Ensembled model( 0.5 *XGB + 0.3 *BAG + 0.2 *LSTM ) :  66.60179529607022
RMSE for Ensembled model( 0.5 *XGB + 0.4 *BAG + 0.1 *LSTM ) :  65.58353131973296
RMSE for Ensembled model( 0.5 *XGB + 0.5 *BAG + 0.0 *LSTM ) :  66.18276574372179
RMSE for Ensembled model( 0.6 *XGB + 0.0 *BAG + 0.4 *LSTM ) :  72.71953508185665
RMSE for Ensembled model( 0.6 *XGB + 0.1 *BAG + 0.3 *LSTM ) :  68.76626956822476
RMSE for Ensembled model( 0.6 *XGB + 0.2 *BAG + 0.2 *LSTM ) :  66.20443592051319
RMSE for Ensembled model( 0.6 *XGB + 0.3 *BAG + 0.1 *LSTM ) :  65.19826146602475
RMSE for Ensembled model( 0.6 *XGB + 0.4 *BAG + -0.0 *LSTM ) :  65.81912881713131
RMSE for Ensembled model( 0.7 *XGB + 0.0 *BAG + 0.3 *LSTM ) :  68.56545855306376
RMSE for Ensembled model( 0.7 *XGB + 0.1 *BAG + 0.2 *LSTM ) :  66.01390104173154
RMSE for Ensembled model( 0.7 *XGB + 0.2 *BAG + 0.1 *LSTM ) :  65.02312367395427
RMSE for Ensembled model( 0.7 *XGB + 0.3 *BAG + -0.0 *LSTM ) :  65.66381462650786
RMSE for Ensembled model( 0.8 *XGB + 0.0 *BAG + 0.2 *LSTM ) :  66.03198105647144
RMSE for Ensembled model( 0.8 *XGB + 0.1 *BAG + 0.1 *LSTM ) :  65.05981496088863
RMSE for Ensembled model( 0.8 *XGB + 0.2 *BAG + -0.0 *LSTM ) :  65.718300196088
RMSE for Ensembled model( 0.9 *XGB + 0.0 *BAG + 0.1 *LSTM ) :  65.30797829876741
RMSE for Ensembled model( 0.9 *XGB + 0.1 *BAG + -0.0 *LSTM ) :  65.98206579308035
Best RMSE for Ensembled model( 0.7 * XGB +  0.2 * BAG +  0.1 * LSTM ) :  65.02312367395427
PS C:\Users\rohit\Downloads\Smart Energy\project\ams-559>
```

- The best combination of weight we found for our ensembled model was 0.7 for XGB regressor, 0.2 for Bagging Regressor, and 0.1 for LSTM Regressor i.e. **[0.7 * XGB + 0.2 * BAG + 0.1 * LSTM]** with least and best **RMSE Score of 65.02312367395427**. We can observe that in the following plot:

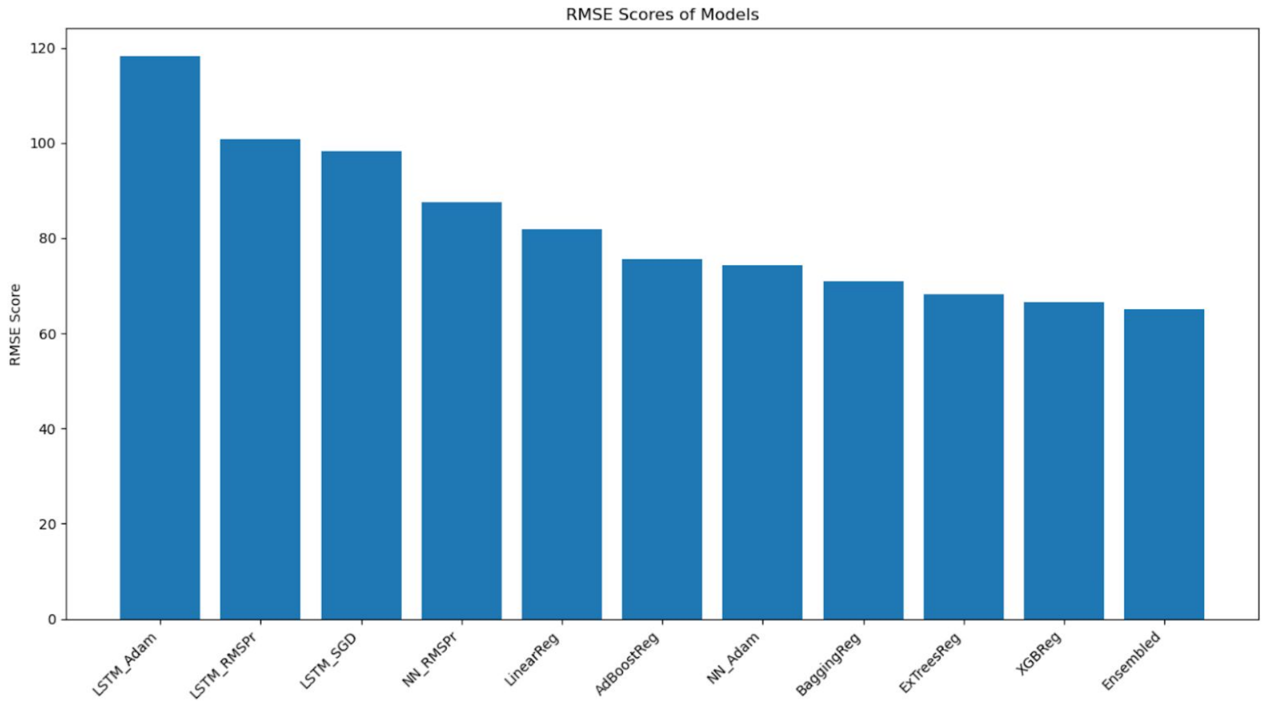


Overall Results

- From the beginning to this point, we have used various individual models and ensemble models(combinations of models). Resultantly, we got stats as shown in the table below.
- The best non-ensembled model or individual model was the XGBoost Regressor as highlighted with blue color in the table with the RMSE Score of **66.45261942378283**. And the overall best performance was given by our ensembled model(highlighted with red color) with a score of **65.02312367395427.**

Model	RMSE Score
Linear Regression	81.78280563366944
AdaBoost Regressor	75.63687847244836
Bagging Regressor	70.94967401057501
Extra-Trees Regressor	68.20844281020672
XGB Regressor	66.45261942378283
LSTM with Adam Optimizer	118.21029465014225
LSTM with RMSProp Optimizer	100.80652164432739
LSTM with SGD Optimizer	98.24036065279401
NN with Adam Optimizer	74.25204016912768
NN with RMSProp Optimizer	87.64033313157483
Ensembled Model **(0.7*XGB + 0.2*BAG+0.1*LSTM)**	**65.02312367395427**

- **Analysis:** We can explain these results by saying that the Range of Electric Vehicle shows linear relation with time(i.e. decreasing with time as battery charge goes down). Because of that linear relation, it was better explained by our models like XGB Regressor in individual models. Similarly, In ensembled model weight of XGB and BAG Regressor was higher as compared to LSTM, where LSTM is a Recurrent Neural Network that generally best explains back memory-based relations which lacks here due to more linearity.
- Below is the comparison of RMSE scores of the different models using a bar graph:



Conclusion

- Given our dataset containing electrical vehicle telemetry for Nissan Leaf, we preprocessed and filtered our data. The features like external environment, driver characteristics, vehicle design, and other hardware and sensors related, we covered in our dataset.
- Under Data and Features Analysis, we have performed PCA Analysis, Driver's aggression analysis, and XGBoot feature importance analysis to add important features to our dataset or to understand the behavior of those features.
- The goal was to predict electrical vehicle ranges based on the current statistics of the car. To do so, we implemented machine learning models that would perform accurate predictions of the electric vehicle range based on parameters of the car such as the state of Charge, the power consumed from various sources, and exogenous factors such as driver style and temperature.

- We tried fitting our dataset on pre-existing models like Regressors, RNN, or NN where Regressors performed well. Then, we built our own novel approach, an ensembled weighted model based on combinations of multiple singular models that gave us a better accuracy in predicting the range of an electric vehicle.

GitHub Repository Link

- Project GitHub Repository: https://github.com/viraj-kamat/ams-559
- All the instructions to run the different scripts of the repository are mentioned in README.md which is present in the repository.

Special Thanks

- Searching for an Electrical Vehicle telemetry data online was hard. We could not find any data on any website online; repeated searches on Google datasets, Kaggle yielded nothing tangible to work with.

- We asked several forums and subreddits (reddit.com/r/electricvehicles,reddit.com/r/cars) but users were reluctant to share data. We even tweeted Elon Musk requesting some datasets. Finally, we came across the Nissan Leaf Cars forum where a user had been aggregating Nissan Leaf Telemetry data.

- We are grateful to **Dr. Maximilian Holland**, a well-known Analyst, Writer and Social Theorist who aggregated Nissan Leaf telemetry data and was kind enough to share CSV files containing such information through the mail. Without his help this project would not have been possible.

References

- https://www.shirin-glander.de/2018/11/ml_basics_gbm/

- https://www.reddit.com/r/electricvehicles/

- https://blog.cambridgespark.com/hyperparameter-tuning-in-xgboost-4ff9100a3b2f

- https://www.sciencedirect.com/science/article/abs/pii/S0306261919314199

- https://www.hindawi.com/journals/jat/2019/4109148/

- https://ieeexplore.ieee.org/document/8719243