Capstone Project 2 – Mercedes Benz Greener Manufacturing

Background and Problem Statement:

Since the first automobile, the Benz Patent Motor Car in 1886, Mercedes-Benz has stood for important automotive innovations. These include, for example, the passenger safety cell with crumple zone, the airbag, and intelligent assistance systems. Mercedes-Benz applies for nearly 2000 patents per year, making the brand the European leader among premium carmakers. Daimler's Mercedes-Benz cars are leaders in the premium car industry. With a huge selection of features and options, customers can choose the customized Mercedes-Benz of their dreams.

To ensure the safety and reliability of every unique car configuration before they hit the road, Daimler's engineers have developed a robust testing system. But, optimizing the speed of their testing system for so many possible feature combinations is complex and time-consuming without a powerful algorithmic approach. As one of the world's biggest manufacturers of premium cars, safety and efficiency are paramount on Daimler's production lines.

Keeping the above problems in context our aim in this project is to reduce vehicle testing time on the testing bench by taking into consideration testing parameters and features.

Data Source:

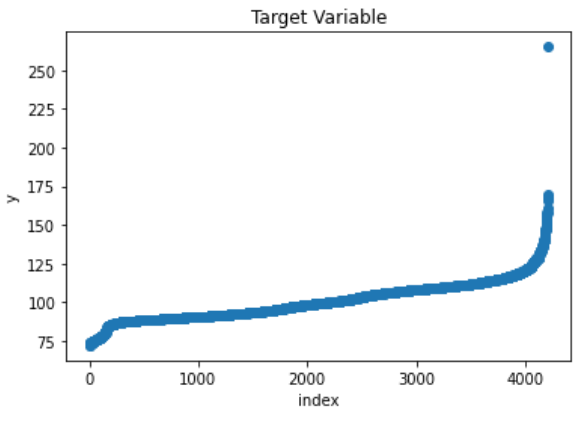
The data for this capstone project was taken from Kaggle Data science Competition

https://www.kaggle.com/c/mercedes-benz-greener-manufacturing/data

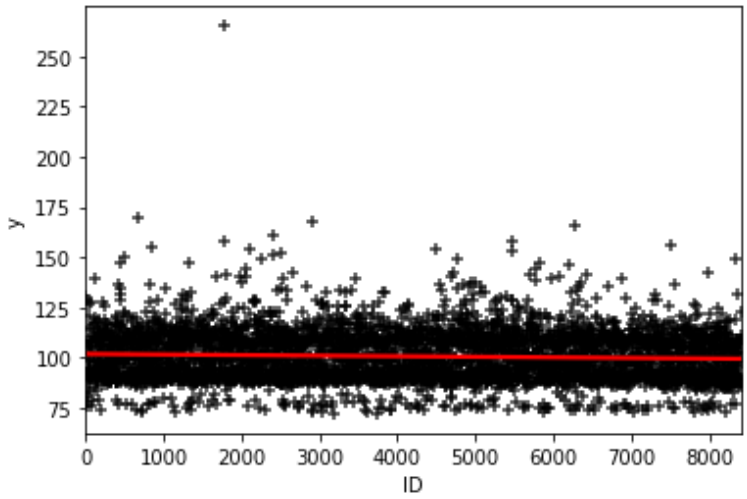
Data Wrangling & EDA:

In the given dataset we have 4209 observations and a total of 378 features, these features are in an encrypted format and their actual meaning cannot be ascertained, since this dataset is focused on finding the testing time of vehicle, we can relate these features to be names of different tests performed.

Our target variable is 'y' which denotes the total time taken to pass the test for the given set of variables and is distributed as shown



As seen above for the given observations our target variable lies in between the range of 75-170 seconds and an outlier value can be seen to be more than 250 seconds in our analysis we have taken the values less than 160 seconds. Also, there seems to be some relationship between the time taken with the 'ID' variable as our range increases a slight reduction in testing time was observed.



We have a total of 378 features in the dataset out of which 8 are categorical variables and the rest 370 are numerical variables leaving behind target variable y and ID there are 368 features an interesting fact is that these variables are having values of either 0s or 1.

Feature Engineering:

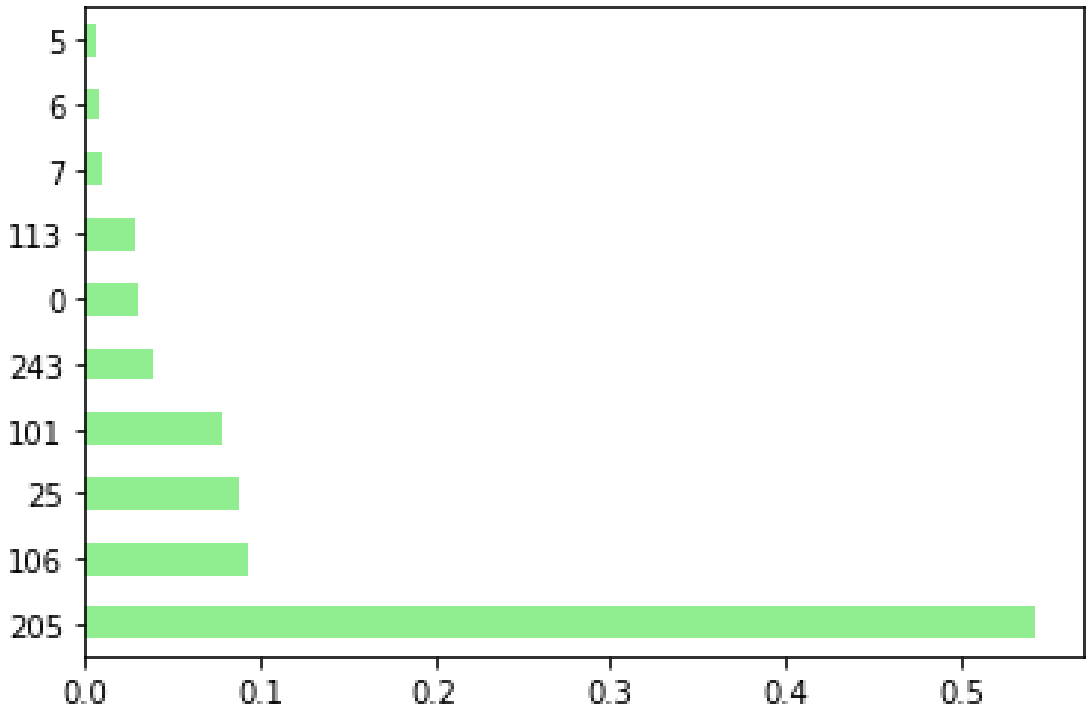
- There were no missing values in the given dataset.
- There were no duplicate rows but there were 56 duplicate columns, these columns were dropped for the analysis.
- Categorical features ‘X4’ was having a min variance with majority values as ‘d’ hence this column was dropped.
- For numerical features having variance as 0 were dropped.
- Correlation matrix for numerical features was computed features correlating greater than 0.95 exhibited multicollinearity hence these features too were dropped.
- After the above steps, there were still 292 features present.
- Categorical variables were ordinal and hence label encoded, one hot encoding was initially applied but this led to an increase in dimensions and also lead to a decrease in model predictive power.
- Similar feature engineering steps were applied for the test data having 4209 observations.
- All the features were scaled using a standard scalar.
- Validation data was formed using training data by keeping split as 30:70.
- PCA with 2 components was used to further reduce the dimensionality on training test and validation datasets.
- R2 score was selected as evaluation metric.
- We need to predict testing time for the given test data and submit it in Kaggle kernel for getting public score for the competition.

Model Preparation and Comparison:

- Linear Regression was used as the base model.
- LassoCV, RidgeCV, EnetCV were used as regularization models to improve r2 scores over the base model.
- Random Forest regressor was used as a tree-based model and 5-fold grid search hyperparameter optimization was done to get the best parameters for the model. This helped in increasing the r2 score furthermore feature importance plot was calculated to determine the important features.

{'max_depth': 5, 'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 50}

- Feature importance's using Random Forest Regressor:

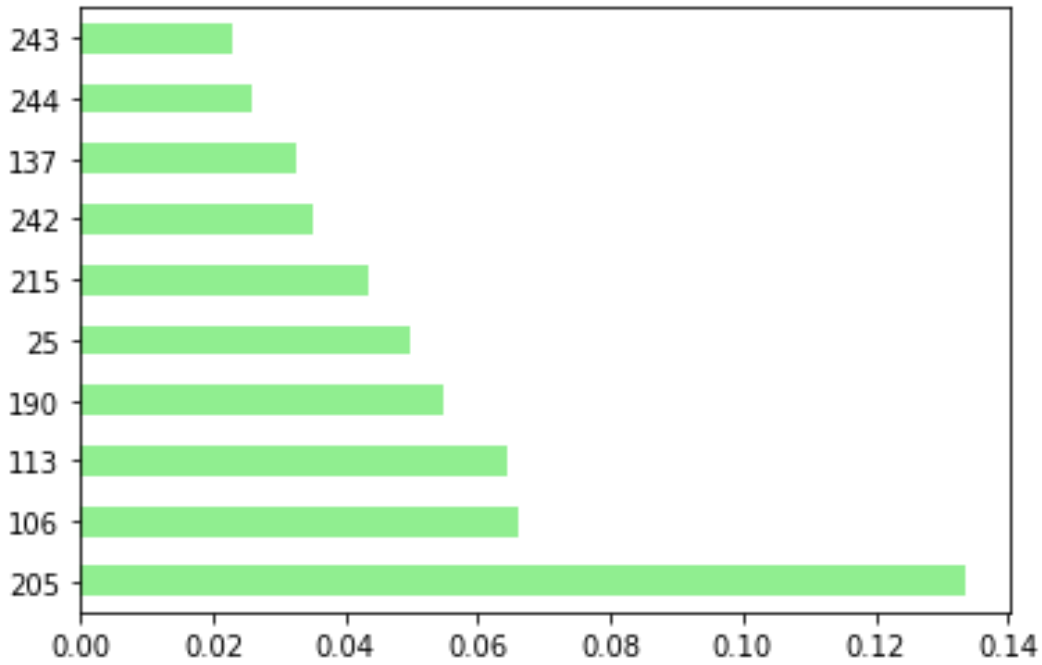


- As seen from the above plot features X205, X106, X25, X101, X243 were having the maximum impact in our random forest model.

State of the art XGboost Regressor model was used to further improve our r2 score
Also, a 5-fold Grid search Hyper optimization was done to get the best parameters for our model. This helped in increasing the r2 score furthermore feature importance plot was calculated to determine the important features.

{'colsample_bytree': 0.7,'learning_rate': 0.01,'max_depth': 3,'min_child_weight': 1,
'n_estimators': 500,'objective': 'reg: squared error',' subsample': 0.7}

- Feature importance's using XGboost Regressor:



- As seen from the above plot features X205, X106, X113, X190, X25, X215, X242, were having the maximum impact in our XGboost model.

Model Summaries

Sr No	Models	Training_Score	Validation_Score
1	Linear_Regression	0.58	-2.92
2	LassoCV	0.56	0.6
3	RidgeCV	0.58	0.57
4	EnetCV	0.56	0.59
5	Random Forest	0.61	0.6
6	XgBoost	0.55	0.61

As seen above since our validation score is high while using XGboost the model we will use on test data to predict the testing time of vehicles.

After Submission of the predicted Target value in Kaggle we have got test Public score of 0.54751.

Suggestions/ Future Research:

While predicting testing of vehicles features such as X205, X106, X113, X190, X25, X215, X242, X101, X243 are having maximum impact and can be focused more in order to reduce testing time of vehicles.

For categorical variables to get better sense Mean encoding can be used or extra features can be created to get better inference.

Instead of using standard scalar model vector normalization can be used.

Further to increase the predictive power of our models we can use ensemble techniques such as stacking and blending.