# GA Data Science Capstone Project

Aryton - 2023

# Problem Statement

We want to be able to identify and predict how much efficiency and carbon footprint of the cars produced every year.

The goal of this project:

- Analyze Fuel consumption
- Analyze Carbon footprint contributed by each car based on the given technical specification and Car manufacturer record
- Create a model to predict the Fuel Usage related properties and Carbon footprint based on the given car specifications, Year and Car maker

# EDA
# (Explanatory Data Analysis)

# EDA Activity

- Dataset Analysis
- Picking features
- Clean NA
- Clean Duplicates
- Clean Outliers

# Data Analysis

Data Dimension (R x C) = 38113 x 80

- In Fuel Type features, there are detailed down of Primary (FT1) and Secondary Fuel Type (FT2). This drilldown us make to accommodate the Hybrid cars. In this Project we won’t consider this feature and only use Combined Fuel Type feature (Fuel Type) since its Redundant
- For Miles Per Gallon Feature, there’re also segregation based on the cars usage either in the City Only, Highway Only or Combined usage. In this project we will focus on the Combined usage
- For Fuel consumption and car Carbon produced related feature, there’re segregation based on the Fuel Type (FT1 or FT2). The data with FT2 are mostly have zero or really small compared to FT1 . With that said we will only focusing on the FT1 data only.

# Picking Features

I will split the features into 2 category :

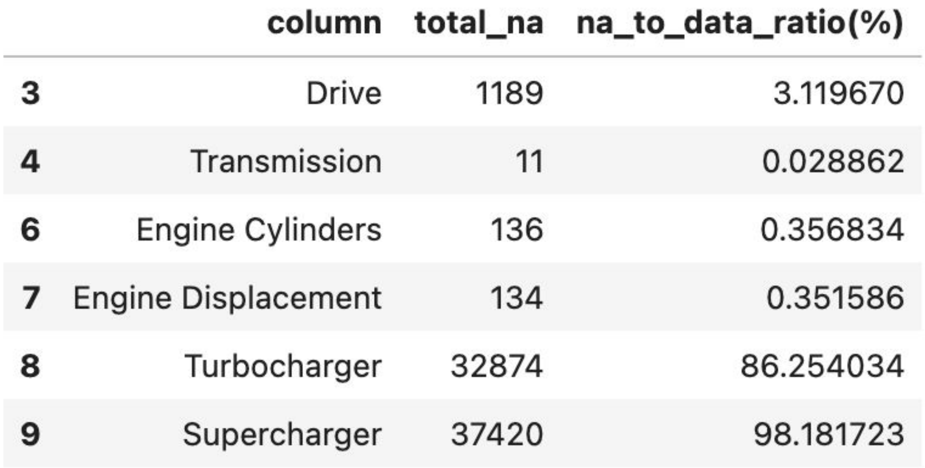
- **Independent** : This features will be our independent features that we will use to train our model. ( Year, Make, Class, Drive, Transmission, Engine Index, Engine Cylinders, Engine Displacement, Turbocharger, Supercharger, Fuel Type)

- **Dependent** : This contains all the dependent that we want to predict with our model.(Combined MPG (FT1), Combined Gasoline Consumption (CD), Annual Fuel Cost (FT1),  Annual Consumption in Barrels (FT1), Tailpipe CO2 (FT1), Fuel Economy Score, GHG Score)

**Data Dimension (R x C) after this activity = 38113 x 20**

# Clean NA - 1

Here are the statistics of the NA value in the features



| | column | total_na | na_to_data_ratio(%) |
|---|---|---|---|
| **3** | Drive | 1189 | 3.119670 |
| **4** | Transmission | 11 | 0.028862 |
| **6** | Engine Cylinders | 136 | 0.356834 |
| **7** | Engine Displacement | 134 | 0.351586 |
| **8** | Turbocharger | 32874 | 86.254034 |
| **9** | Supercharger | 37420 | 98.181723 |

# Clean NA - 2

| Features | Root Cause | Solution | Total Rows Deleted |
|---|---|---|---|
| Turbocharger | The feature has data either maintained as ‘T’ the car has Turbocharger or NaN if it don’t have | Replace the values with either 1(the car have turbocharger) or 0 (the car do not have turbocharger) | 0 |
| Supercharger | The feature has data either maintained as ‘S’ the car has Supercharger or NaN if it don’t have | Replace the values with either 1(the car have turbocharger) or 0 (the car do not have turbocharger) | 0 |
| Drive | There are around 1189 entry  with NA in this feature | Drop affected data since we are not able to identify the Drive each car | 1189 |

# Clean NA - 3

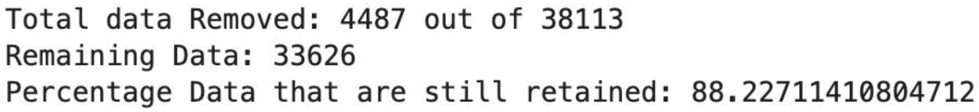
| Features | Root Cause | Solution | Total Rows Deleted |
|---|---|---|---|
| Transmission | All the electric cars have NaN value | We replace the value with ‘Not Applicable’ since electric cars do not have transmission | 0 |
| Engine Cylinders | There are 3 cars with Regular Fuel Type and 133 cars powered by Electric having NaN value | Delete the 3 entries with Regular Fuel Type and Replace the other 133 entries with value ‘Not Applicable’ since Electric car do not have engine | 3 |
| Engine Displacement | All NanN entries are electric cars | Replace the value with ‘Not Applicable’ | 0 |

# Clean Outliers

- We drop outliers entry in Combined MPG (FT1), Annual Fuel Cost (FT1) and Annual Consumption in Barrels.
- Combined Gasoline Consumption (CD) is removed because there are a lot of entry did not having values in this feature
- During the check we spot the values in GHG Score feature are identical with Fuel Economy Score so we drop this feature.
- For Tailpipe CO2(FT1) and Fuel Economy score, the cars that are made 2012 below did not have these features maintained hence we keep it as it is and we will select the data with those values maintained during the model creation later

# EDA Result

- We removed 4487 entries of data out of 38113. With that said, we retained around 88.23% of our data to be used for model training.
- On top of that we removed 2 features.

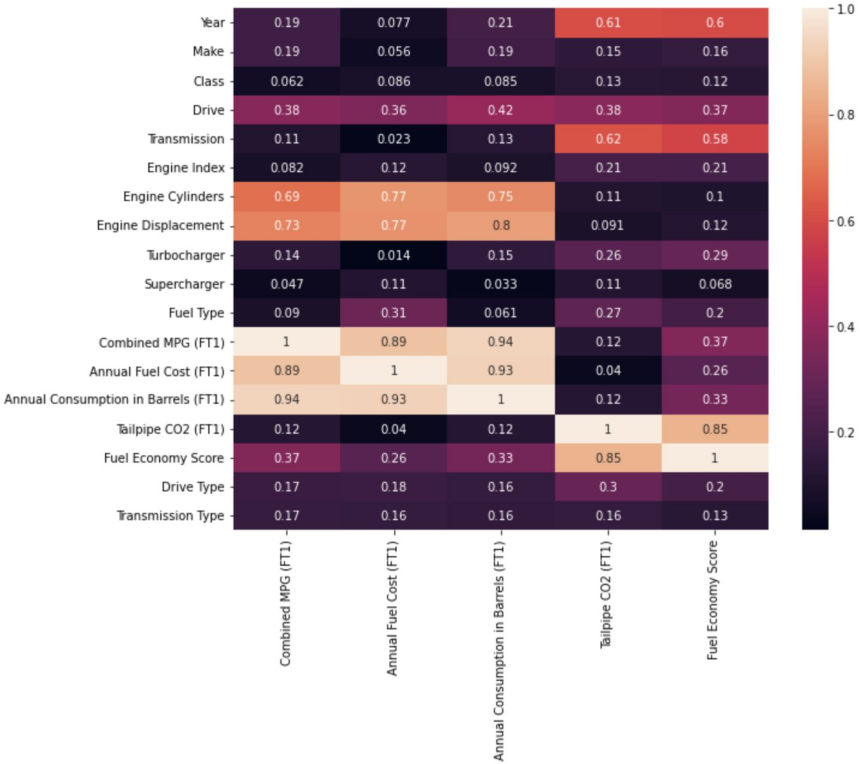


```
Total data Removed: 4487 out of 38113
Remaining Data: 33626
Percentage Data that are still retained: 88.22711410804712
```

**After EDA, our dataset dimension (R x C) become  33626 x 18**

# Basic Data Analytics

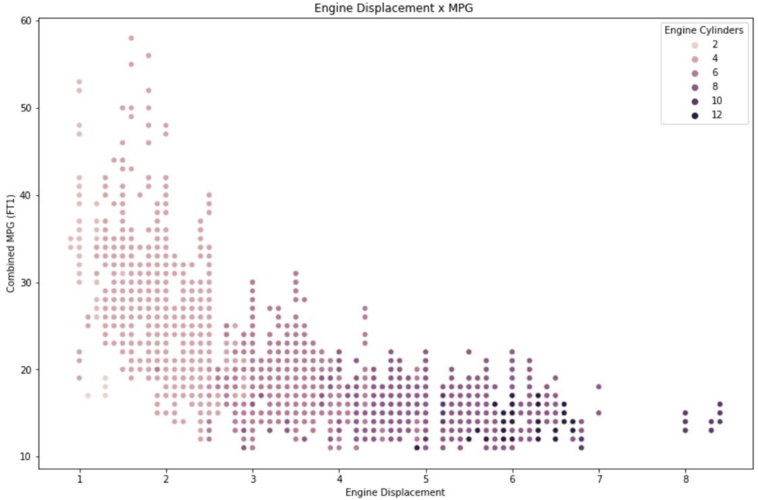
# Correlation Matrix

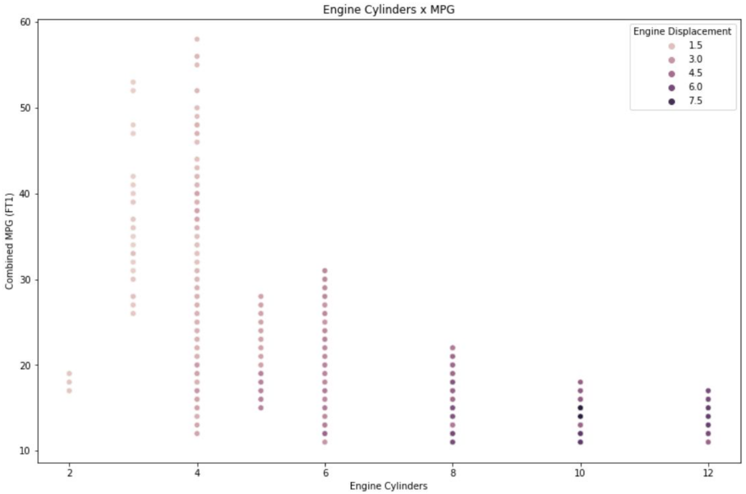


From the correlation we can assume that:

- How efficient the car fuel consumption is depending on the car engine features (Cylinder and Displacement).
- The transmission of the car affecting the carbon pollution produced by the car

# Engine X Miles Per Gallon



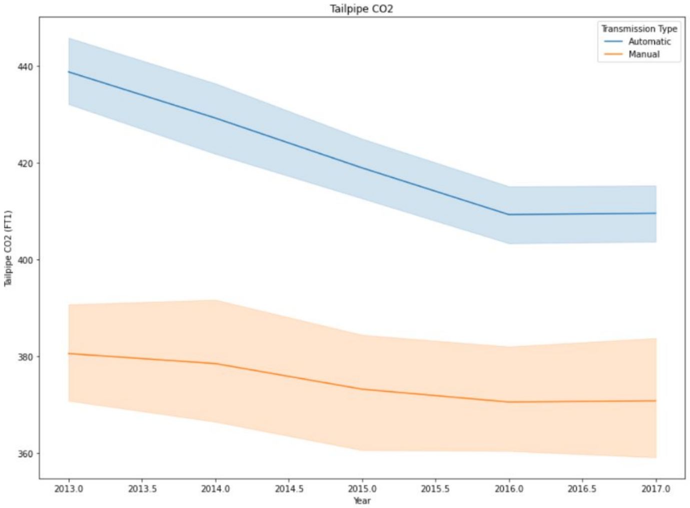


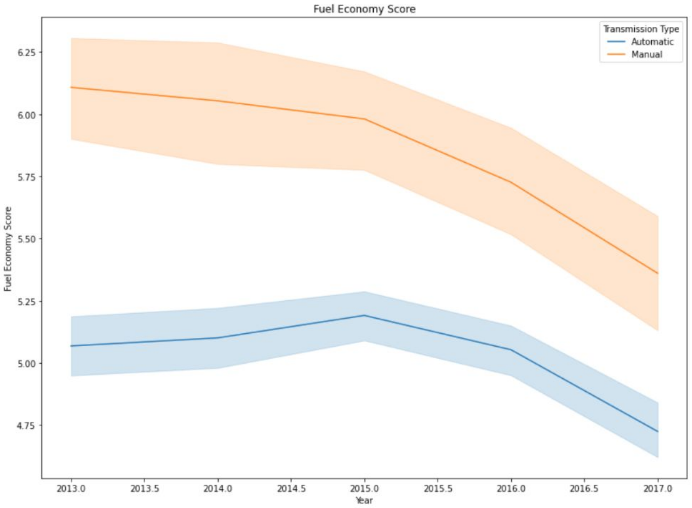
Based on the chart we can conclude :

A. The engine with 6 cylinder above will consume more fuel.

B. For 6 and below cylinder engine, the fuel consumption tends to be higher when the engine displacement is small.

# Transmission x Carbon Footprint

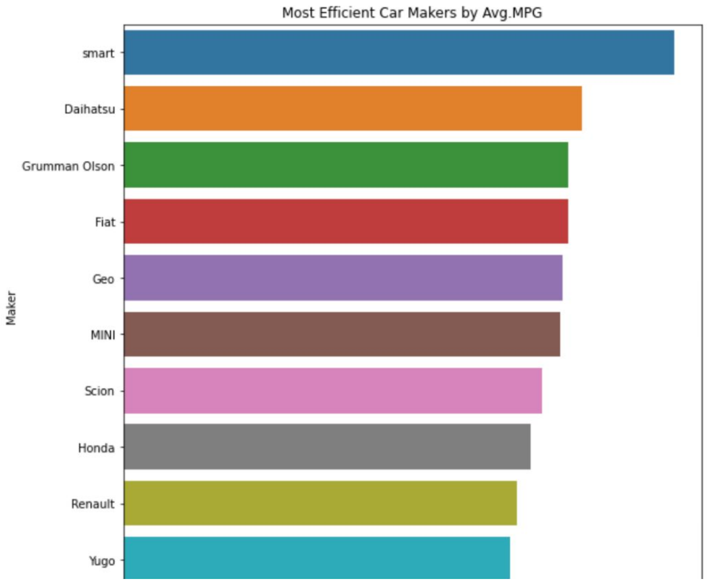


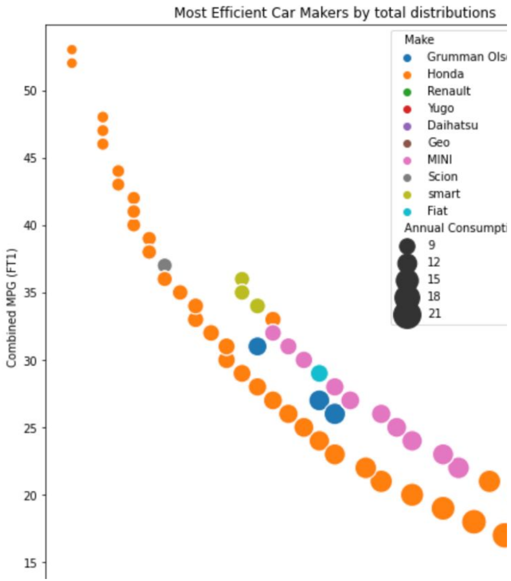


Based on above plot we can conclude:

1. Car with automatic transmission tend to produce more pollution than manual
2. There are huge deviation detected in Manual Car
3. The car polution reduced every year and the trend become stagnated from 2016 onwards

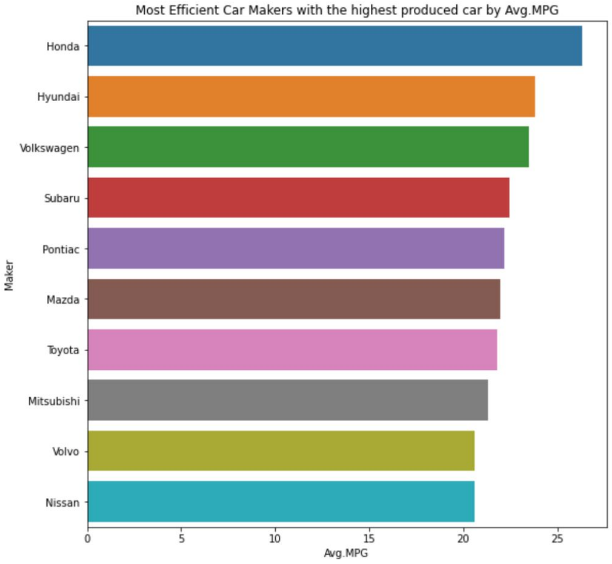
# Most Efficient Car Makers



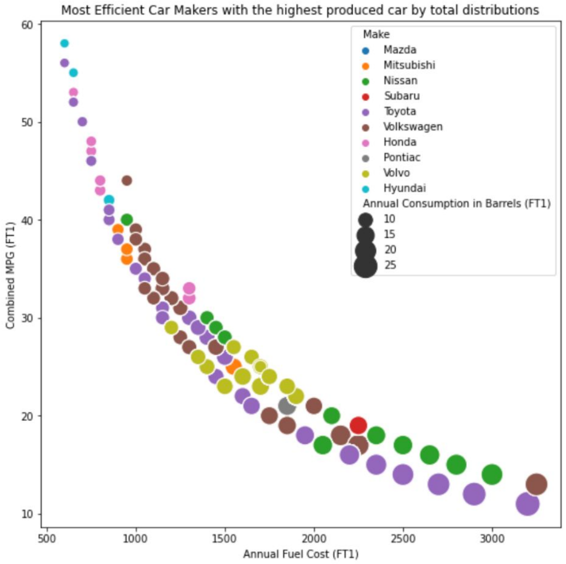


Based on above charts, SMART produced the most efficient cars based on the Average MPG, but they only have 3 cars in dataset. If we look at the 2nd chart, Honda dominated in terms of no of cars produced.

# Most Efficient Car Makers (With the highest Volume Production)



Most Efficient Car Makers with the highest produced car by Avg.MPG



Most Efficient Car Makers with the highest produced car by total distributions

Although Volkswagen is at the 3rd position in the left plot (based on avg MPG). it produced a lot of Efficient car in the data set comparing to Hyundai.

# Model

# Overview

| Type | Dependent Feature |
|---|---|
| Regression | Combine MPG (FP1) |
| | Annual Fuel Cost (FT1) |
| | Annual Consumption in Barrels |
| | Tailpipe CO2 |
| Classification | Fuel Economy Score |

# Data Encoding

Following features need encoding before we start use it in our model because they are in String format. Here are the details of the encoding that we will use for each feature

| Feature | Encoding Method | Total Unique Values |
|---|---|---|
| Make | Leave-one-out | 123 |
| Class | Base-N | 34 |
| Drive | Binary | 7 |
| Transmission | Base-N | 44 |
| Fuel Type | Binary | 11 |

# Regression Model

# Which Algorithm will be used ?

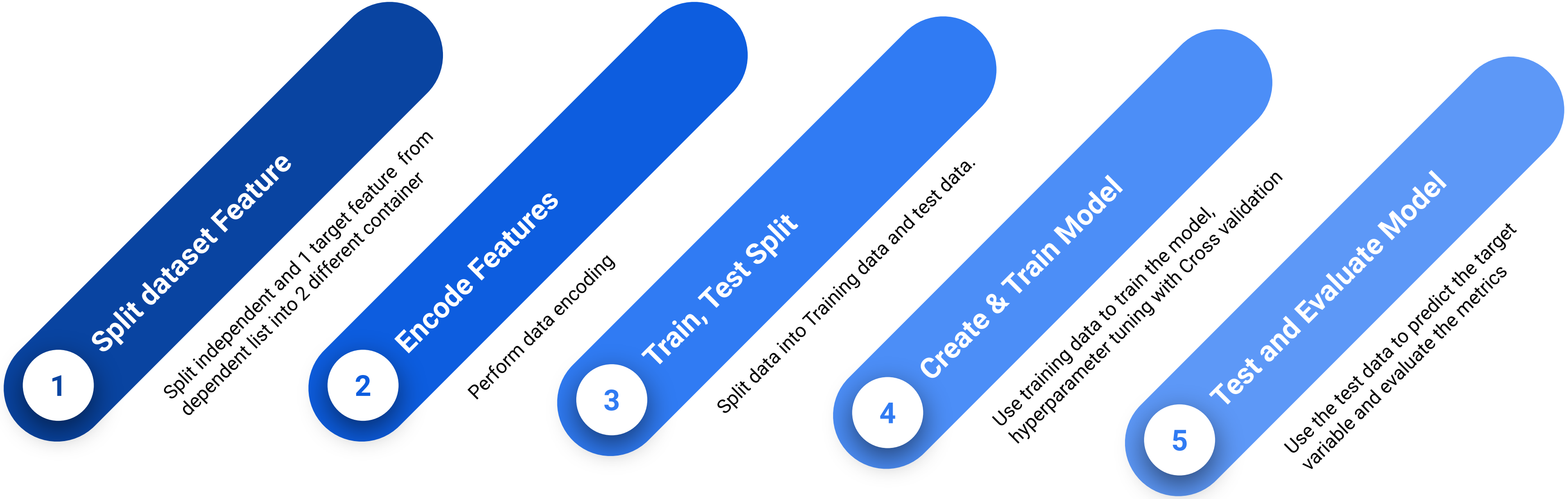
For each features we will create 3 regression model using below Algorithm:

- **Linear Regression**, We will use GridSearchCV to run the cross validation logic using this Algorithm. No hyperparameter tuning in this activities.

- **Ridge Regression**, we will use the RidgeRegressionCV which has built in Hyperparameter tuning and Cross validation.

- **Lasso Regression**, we will use the LassoRegressionCV which has the same feature as RidgeRegressionCV.

# How we will train each Algorithm?



1 **Split dataset Feature**

Split independent and 1 target feature  from dependent list into 2 different container

2 **Encode Features**

Perform data encoding

3 **Train, Test Split**

Split data into Training data and test data.

4 **Create & Train Model**

Use training data to train the model, hyperparameter tuning with Cross validation

5 **Test and Evaluate Model**

Use the test data to predict the target variable and evaluate the metrics

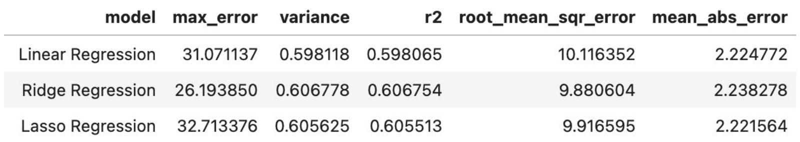
# Metrics to Evaluate the Model

We will use following metrics to evaluate how the model perform for each algorithm:

- Max Error
- Variance
- R2,
- Root Mean Square Error (RMSE)
- Mean Absolute Error (MAE)

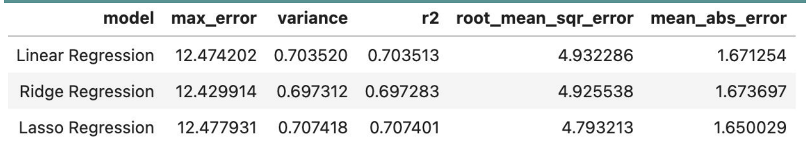
# Evaluation Result

## Combined MPG (FT1)



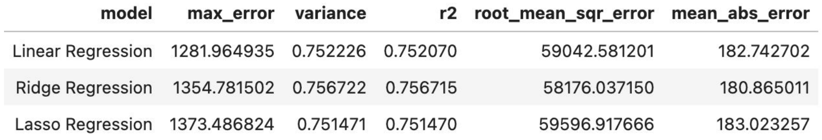
| model | max_error | variance | r2 | root_mean_sqr_error | mean_abs_error |
|---|---|---|---|---|---|
| Linear Regression | 31.071137 | 0.598118 | 0.598065 | 10.116352 | 2.224772 |
| Ridge Regression | 26.193850 | 0.606778 | 0.606754 | 9.880604 | 2.238278 |
| Lasso Regression | 32.713376 | 0.605625 | 0.605513 | 9.916595 | 2.221564 |

## Annual Fuel Consumption in Barrels



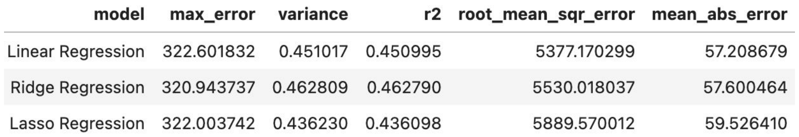
| model | max_error | variance | r2 | root_mean_sqr_error | mean_abs_error |
|---|---|---|---|---|---|
| Linear Regression | 12.474202 | 0.703520 | 0.703513 | 4.932286 | 1.671254 |
| Ridge Regression | 12.429914 | 0.697312 | 0.697283 | 4.925538 | 1.673697 |
| Lasso Regression | 12.477931 | 0.707418 | 0.707401 | 4.793213 | 1.650029 |

## Annual Fuel Cost



| model | max_error | variance | r2 | root_mean_sqr_error | mean_abs_error |
|---|---|---|---|---|---|
| Linear Regression | 1281.964935 | 0.752226 | 0.752070 | 59042.581201 | 182.742702 |
| Ridge Regression | 1354.781502 | 0.756722 | 0.756715 | 58176.037150 | 180.865011 |
| Lasso Regression | 1373.486824 | 0.751471 | 0.751470 | 59596.917666 | 183.023257 |

## Tailpipe CO2



| model | max_error | variance | r2 | root_mean_sqr_error | mean_abs_error |
|---|---|---|---|---|---|
| Linear Regression | 322.601832 | 0.451017 | 0.450995 | 5377.170299 | 57.208679 |
| Ridge Regression | 320.943737 | 0.462809 | 0.462790 | 5530.018037 | 57.600464 |
| Lasso Regression | 322.003742 | 0.436230 | 0.436098 | 5889.570012 | 59.526410 |

- Ridge regression show the best result in Combined MPG, Annual Fuel Cost and Tailpipe CO2 although they are not significant
- For Tailpipe CO2, we only use the data with value greater equal than 0 ( 5990 out of 33626 entries) and we don’t do clear any Outliers again for that slice of data which is the root cause why the RMSE and MAE so high.
- For Annual Fuel Cost, the only predictor we have is the car technical specs without any Avg. Car Usage per year which cause the model prediction having a high RMSE

# Classification Model

# Which Algorithm will be used ?

For each features we will create 3 Classification model using below Algorithm:

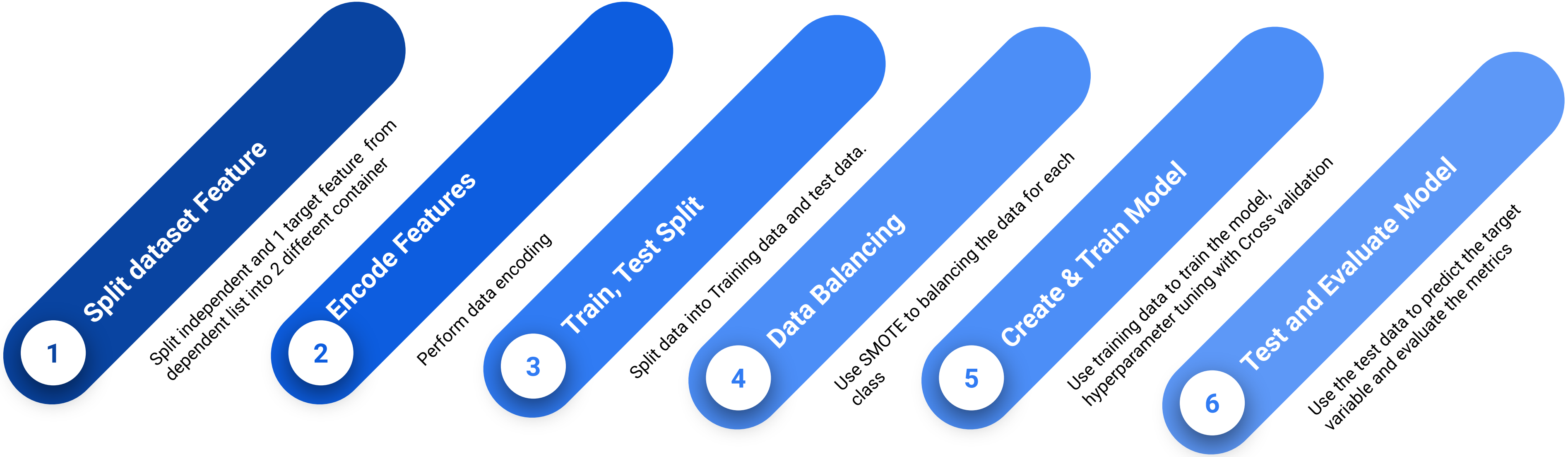
- **Logistic Regression**, We will use LogisticRegressionCV to run the cross validation logic using this Algorithm. This module has a built in hyperparameter tuning.

- **K-Nearest Neighbor**, we will use the GridSearchCV to perform cross validation and Hyperparameter tuning

- **Random Forest Classifier**, Same as K-Nearest Neighbor

All these models will be used to solve multiclass classification problem

# How we will train each Algorithm?



1. **Split dataset Feature** — Split independent and 1 target feature from dependent list into 2 different container
2. **Encode Features** — Perform data encoding
3. **Train, Test Split** — Split data into Training data and test data.
4. **Data Balancing** — Use SMOTE to balancing the data for each class
5. **Create & Train Model** — Use training data to train the model, hyperparameter tuning with Cross validation
6. **Test and Evaluate Model** — Use the test data to predict the target variable and evaluate the metrics

# Metrics to Evaluate the Model

We will use following metrics to evaluate how the model perform for each algorithm:

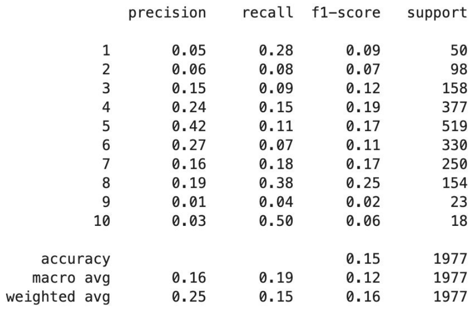
- Precision
- Recall
- F1 Score

# Precision, Recall ?

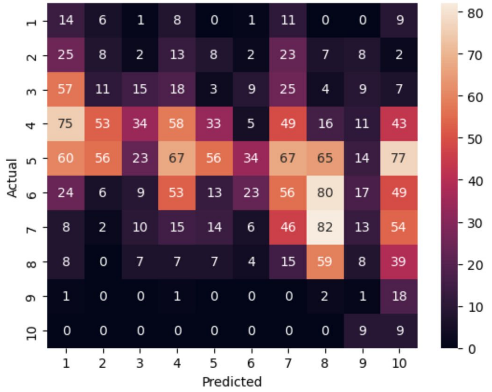
The fuel economy score rating has 10 types of classification from 1 to 10 which based on certain criteria that already standardized. With that said we need to aim for the highest **precision** in our model

# Evaluation Result - 1

## Logistic Regression

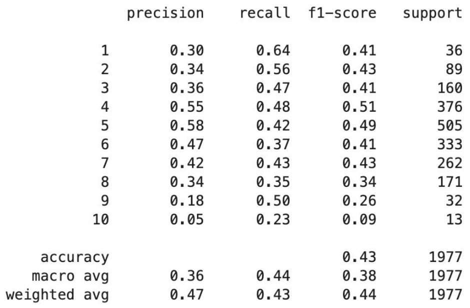


```
              precision    recall  f1-score   support

           1       0.05      0.28      0.09        50
           2       0.06      0.08      0.07        98
           3       0.15      0.09      0.12       158
           4       0.24      0.15      0.19       377
           5       0.42      0.11      0.17       519
           6       0.27      0.07      0.11       330
           7       0.16      0.18      0.17       250
           8       0.19      0.38      0.25       154
           9       0.01      0.04      0.02        23
          10       0.03      0.50      0.06        18

    accuracy                           0.15      1977
   macro avg       0.16      0.19      0.12      1977
weighted avg       0.25      0.15      0.16      1977
```

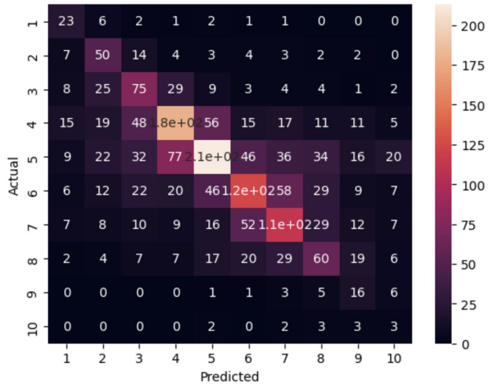


| Actual \ Predicted | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14 | 6 | 1 | 8 | 0 | 1 | 11 | 0 | 0 | 9 |
| 2 | 25 | 8 | 2 | 13 | 8 | 2 | 23 | 7 | 8 | 2 |
| 3 | 57 | 11 | 15 | 18 | 3 | 9 | 25 | 4 | 9 | 7 |
| 4 | 75 | 53 | 34 | 58 | 33 | 5 | 49 | 16 | 11 | 43 |
| 5 | 60 | 56 | 23 | 67 | 56 | 34 | 67 | 65 | 14 | 77 |
| 6 | 24 | 6 | 9 | 53 | 13 | 23 | 56 | 80 | 17 | 49 |
| 7 | 8 | 2 | 10 | 15 | 14 | 6 | 46 | 82 | 13 | 54 |
| 8 | 8 | 0 | 7 | 7 | 7 | 4 | 15 | 59 | 8 | 39 |
| 9 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 1 | 18 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 9 |

## K - Nearest Neighbour

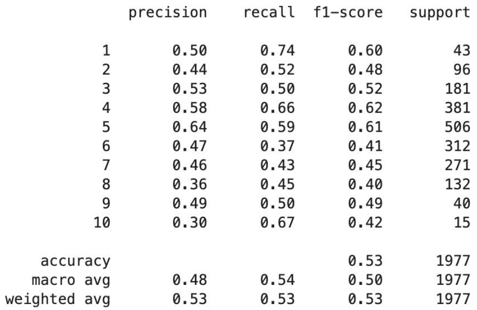


```
              precision    recall  f1-score   support

           1       0.30      0.64      0.41        36
           2       0.34      0.56      0.43        89
           3       0.36      0.47      0.41       160
           4       0.55      0.48      0.51       376
           5       0.58      0.42      0.49       505
           6       0.47      0.37      0.41       333
           7       0.42      0.43      0.43       262
           8       0.34      0.35      0.34       171
           9       0.18      0.50      0.26        32
          10       0.05      0.23      0.09        13

    accuracy                           0.43      1977
   macro avg       0.36      0.44      0.38      1977
weighted avg       0.47      0.43      0.44      1977
```

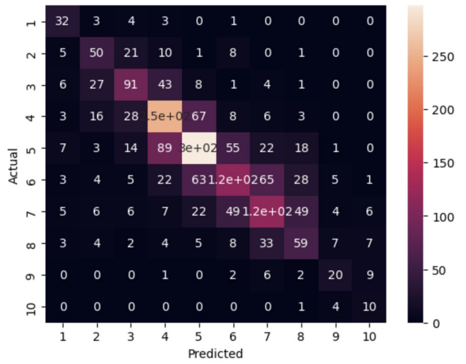


| Actual \ Predicted | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 23 | 6 | 2 | 1 | 2 | 1 | 1 | 0 | 0 | 0 |
| 2 | 7 | 50 | 14 | 4 | 3 | 4 | 3 | 2 | 2 | 0 |
| 3 | 8 | 25 | 75 | 29 | 9 | 3 | 4 | 4 | 1 | 2 |
| 4 | 15 | 19 | 48 | 1.8e+02 | 56 | 15 | 17 | 11 | 11 | 5 |
| 5 | 9 | 22 | 32 | 77 | 2.1e+02 | 46 | 36 | 34 | 16 | 20 |
| 6 | 6 | 12 | 22 | 20 | 46 | 1.2e+02 | 58 | 29 | 9 | 7 |
| 7 | 7 | 8 | 10 | 9 | 16 | 52 | 1.1e+02 | 29 | 12 | 7 |
| 8 | 2 | 4 | 7 | 7 | 17 | 20 | 29 | 60 | 19 | 6 |
| 9 | 0 | 0 | 0 | 0 | 1 | 1 | 3 | 5 | 16 | 6 |
| 10 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 3 | 3 | 3 |

## Random Forest Classifier



```
              precision    recall  f1-score   support

           1       0.50      0.74      0.60        43
           2       0.44      0.52      0.48        96
           3       0.53      0.50      0.52       181
           4       0.58      0.66      0.62       381
           5       0.64      0.59      0.61       506
           6       0.47      0.37      0.41       312
           7       0.46      0.43      0.45       271
           8       0.36      0.45      0.40       132
           9       0.49      0.50      0.49        40
          10       0.30      0.67      0.42        15

    accuracy                           0.53      1977
   macro avg       0.48      0.54      0.50      1977
weighted avg       0.53      0.53      0.53      1977
```



| Actual \ Predicted | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 32 | 3 | 4 | 3 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 5 | 50 | 21 | 10 | 1 | 8 | 0 | 1 | 0 | 0 |
| 3 | 6 | 27 | 91 | 43 | 8 | 1 | 4 | 1 | 0 | 0 |
| 4 | 3 | 16 | 28 | 2.5e+02 | 67 | 8 | 6 | 3 | 0 | 0 |
| 5 | 7 | 3 | 14 | 89 | 3e+02 | 55 | 22 | 18 | 1 | 0 |
| 6 | 3 | 4 | 5 | 22 | 63 | 1.2e+02 | 65 | 28 | 5 | 1 |
| 7 | 5 | 6 | 6 | 7 | 22 | 49 | 1.2e+02 | 49 | 4 | 6 |
| 8 | 3 | 4 | 2 | 4 | 5 | 8 | 33 | 59 | 7 | 7 |
| 9 | 0 | 0 | 0 | 1 | 0 | 2 | 6 | 2 | 20 | 9 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 4 | 10 |

# Evaluation Result - 2

Based on 3 algorithm that we used, we can get the accuracy of 53% for our model to predict this feature using the Random Forest classifier. The only caveat is we need a lot of computing resources to train the model.