



— ISTANBUL — OKAN UNIVERSITY

Name and Surname: Ayşe Nur Akdeniz

Student Number: 170212001

Lecture: CENG498 Computer Engineering Graduation Project

Lecturer: Prof. Dr. Semih Bilgen

Report Date: 06.06.2021

PREDICTION OF FUEL CONSUMPTION OF VEHICLES

Ayşe Nur Akdeniz

Istanbul Okan University

ABSTRACT

A machine learning model is proposed for estimating fuel consumption for a vehicle, based on existing data on its MPG(Miles Per Gallon), cylinders, displacement, horsepower, weight, acceleration, model year and origin attributes. Exploratory data analysis was done first because if the model is fed with data which does not have any problem such as missing values, more consistent and realistic results will be obtained. Then, outliers from our dataset to get accurate predictions were excluded. After clearing data and analyzing them, features that were expected to be useful in explaining the target variable were determined. This step is called feature extraction. Regularization methods such as Lasso, Ridge, Elastic Net and XGB method are used to find the smallest Mean-Square Error. Then averages of two regularization methods which have minimum Mean-Square Error value were taken. Finally, an optimization exercise (Predicting the MPG) was carried out to minimize fuel consumption via modification of controllable vehicle characteristics such as acceleration.

ACKNOWLEDGEMENT

The completion of this study could not have been possible without the expertise of Prof. Dr. Semih Bilgen. I would like to sincere thanks to him for his constant guidance as well as for providing necessary information and direction.

Last but not least, I would like to thank my friends who are checking my project every step.

June, 2021

Ayşe Nur Akdeniz

Table of Contents

I. List of Abbreviations and Acronyms.....	5
II. List of Figures.....	5
III. Introduction.....	6
IV. Published Articles About Project.....	7
A) Artificial Neural Network for Forecasting Car Mileage per Gallon in the City.....	7
B) Predicting MPG for Automobile Using Artificial Neural Network Analysis.....	7
C) An Accurate Prediction of MPG (Miles per Gallon) using Linear Regression Model of Machine Learning.....	8
D) Prediction of Torque and Specific Fuel Consumption of a Gasoline Engine by Using Artificial Neural Networks.....	8
V. Project Plan.....	9
A) Work Contents.....	9
B) Programming Languages, Libraries, Data Sets.....	10
C) Test and Evaluation Part.....	10
VI. Project Schedule.....	11
VII. Project.....	12
VIII. Test and Evaluation.....	26
IX. Conclusion.....	27
X. References.....	28

I. LIST OF ABBREVIATIONS AND ACRONYMS

- **ANN:** Artificial Neural Network
- **EDA :** Explotary Data Analysis
- **IDE:** Integrated Development Environment
- **MPG:** Miles Per Gallon
- **UCI:** University of California, Irvine
- **XGB:** XGBoost

II. LIST OF FIGURES

Figure 1: Gannt Chart for the project schedule.....	11
Figure 2: Code Blocks of importing libraries and dataset.....	12
Figure 3: Code Blocks for finding missing values.....	13
Figure 4: Code Blocks for filling the null values and starting the EDA.....	14
Figure 5: Correlation Matrix of all Features.....	15
Figure 6: Correlation Matrix between features which has direct proportion.....	15
Figure 7: Pairplot schema for all features.....	16
Figure 8: Showing the categorical Features.....	17
Figure 9: Boxplot Schema.....	17
Figure 10: Detecting the outliers from boxplot schema.....	18
Figure 11: Drop the outlier from horsepower column.....	18
Figure 12: Drop the outlier from acceleration column	19
Figure 13: Skewness.....	19
Figure 14: Shows the skewness with displot method	19
Figure 15: Fixing the skewness.....	20
Figure 16: Applying the One Hot Encoding, Split and Standardization.....	21
Figure 17: Linear and Ridge Reression.....	22
Figure 18: Lasso and ElasticNet Regression.....	23
Figure 19: Applying the XGB model.....	24
Figure 20: Taking the Average of best two models	24
Figure 21: Take the Exponencial transform	25
Figure 22: Taking the Logarithm Transform.....	26
Figure 23: Take the Exponencial transform	26

III. INTRODUCTION

The rapid development of the automobile industry has created some requirements. In this development process, consumers are complaining about the increasing fuel prices. Also they are becoming more particular about the features. As a result of these, automobile manufacturers are constantly optimizing their processes to increase fuel efficiency.^[1] But what if we could have a chance to predict a car's MPG given some known features about the vehicle? There are many research and development projects on this subject.^[2] If we can successfully implement such a facility, we can beat our opponent in the sector and we can improve the fuel economy. Additionally, we can produce more desirable vehicles which are more efficient. So, let us make an introduction to how to create this model.

Vehicles achieve different fuel consumption levels according to a number of features. To give an example; fuel consumption differs according to weight, horsepower and number of cylinders a vehicle has. In this project, I will predict the fuel consumption (MPG) of the vehicles according to the main features such as cylinders, displacement, horsepower, weight, acceleration, model year and origin. We will access the necessary data to make this prediction from machine learning repository of UCI (University of California, Irvine)^[3]

I'm going to train linear regression models to predict the MPG of a given car using the scikit learn module in python. Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.^[4] That is why I choose the linear regression. The scikit learn module in python is basically used to perform machine learning tasks. It contains a number of machine learning models and algorithms that are very good and helpful to use such as XGB. Data regularization is a technique which makes minor changes to the learning algorithm such that the model generalizes better. This improves the performance of model on the invisible data as well and provides avoid the risk of overfitting.^[5] So, I will try three regularization methods Ridge, Lasso and Elastic Net. Ridge regression works well if there are many predictors of about the same magnitude. Lasso works well if there are few significant key features and the value of the others are close to zero. Elastic Net is a combination of both Lasso and Ridge.^[6] After data regularization, our prediction model will be ready to use.

IV. PUBLISHED ARTICLES ABOUT PROJECT

There are many projects about prediction of fuel consumption. Some use machine learning, whereas others use neural networks. While a Machine Learning model makes decisions based on what it learns from the data, a Neural Network organizes algorithms in such a way that it can make correct decisions on its own.^[7] In my project, I will use machine learning methods because I want my model parses the data, learn from it, and use these learned data for discovering meaningful patterns. So it would be better that I will review and compare the articles which use machine learning algorithms.

A) Artificial Neural Network for Forecasting Car Mileage per Gallon in the City^[8]

In this article, the authors use an ANN (Artificial Neural Network) for predicting MPG. An ANN is the piece of a computing system designed to simulate the way the human brain analyzes and processes information.^[9] The main aim of their study is to determine cars which would be suitable for the customer. In their model the features are used as an input layer in neural network model and they have two hidden layers and their output layer is MPG. Neural networks are black boxes, in which the user feeds in data and receives answers; they can fine-tune the answers, but they do not have access to the exact decision making process. When a new code is run on network results a different new weight matrix. This is a result of small random values set while the program is running.^[10] This is why ANN's are black boxes. In spite of this, the accuracy of predication of their model is 99.5% and this is a really good conclusion.

B) Predicting MPG for Automobile Using Artificial Neural Network Analysis^[11]

In this article, the authors also use the Artificial Neural Network for prediction of MPG and their aim is helping through later design and manufacturing of later automobile. In this model they use the system and both the Gradient Descent Algorithm which is an iterative optimization algorithm for finding the local minimum of a function^[12] and the Normalized Square Error Technique which facilitates the comparison between datasets or models with different scales.^[13] Also they explain the features such as weight, displacement, horsepower, acceleration, MPG and origin clearly. Data analysis was made and they created the histogram of each feature. Correlation and regression analysis was made by them also. As a result, they draw attention to the data pre-processing and they emphasized that it is necessary to proceed step by step while doing the project.

C) An Accurate Prediction of MPG (Miles per Gallon) using Linear Regression Model of Machine Learning^[14]

In this article, the authors used machine learning for the model but they used Root Mean Square Error. They figure the data that representing the scatter plot format of all the factors affecting MPG for cars of different companies and origin. Also they figure other features by using matplotlib and seaborn libraries for exploratory data analysis. This article consists of very close steps to my project. I am sure that the data preprocessing and analyzing the data are really important points in every project.

D) Prediction of Torque and Specific Fuel Consumption of a Gasoline Engine by Using Artificial Neural Networks^[15]

In this article, predicting fuel consumption depends on torque and brake. They are using ANN for this project which is really useful and effective for that situation. They firstly use a real engine for create database, and then with this database they use ANN for the prediction of torque and brake specific fuel consumption of an engine. They did not use a data set unlike mine; they used their own results of test engine. Also they used different features rather than mine such as spark advance, engine speed, and throttle positions. Although it is so different from mine, it also helped my study in terms of analysis.

A, **B** and **D** articles all use ANN to predict MPG, but **C** uses Machine Learning model. Article **A** goes deeper into ANN, while article **B** examines and analyzes the data. Step-by-step data analysis is performed in Article **C** and a machine learning model is developed. **B** and **C** have used different square error techniques, but I will use mean square error. The main advantage of the mean-square error technique is the analytical tractability that comes with it. Mean square error tells us how close a regression curve is to a set of points. If we much closer to the curve which means we have least mean square error value, we get the best accuracy.^[16] So, in my model there will be a machine learning model like the one in Article **C**, and it will include features and data preprocessing as in Article **B**.

V. PROJECT PLAN

A) Work Contents

Based on the articles in the previous section, I will use the UCI's data sets and it includes features such as mpg, cylinders, displacement, horsepower, weight, acceleration, model year, origin, car name. But car name is not important as it does not affect the MPG, so I will not use it. Also I will analyze the data and visualize some important points. But the first thing I will do is come up with a plan to go step by step. Here are the steps of my work on this project:

1. Import Libraries
2. Import data that I will use
3. Find missing values if there is and then for the column which has missing value, fill the mean value instead of null values
4. Do the Exploratory Data Analysis which helps determine how best to manipulate data sources to get the answers you need, making it easier for us to discover patterns, spot anomalies.
5. Determine the outliers and remove it from the columns that have outliers in our dataset.
6. In order to distribute the predicted values normally and have a symmetrical shape, the correction should be made by applying the Skewness method.
7. Apply StandardScaler method to fit and transform the data

Training the Model:

8. Apply Linear Regression (linear regression is used to fit a predictive model to an observed data set of values of the response and explanatory variables.)

(Regularization is a technique used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.)^[17]

9. Regularization 1: Ridge Regularization^[18] (also known as L2) adds regularization terms in the model which are function of square of coefficients of parameters. This technique minimizes the effect of irrelevant features as the strength of regularization growth.

10. Regularization 2: Lasso Regularization^[19] (also known as L1) adds regularization terms in the model which are function of absolute value of the coefficients of parameters. This technique can be used for feature selection and creating a model with fewer features.

11. Regularization 3: Elastic Net ^[20] adds regularization terms in the model which are combination of both L1 and L2 regularization.

12. Apply XGB ^[21] (XGB is an algorithm that designed for big, complex data sets.) because it has higher predictive power.

13. Averages Models ^[22]: Define a class that takes averages of two regularization methods which has minimum mean square error value. Then, compare the average model result with the result of the others. Whichever has the smallest mean squared error and accuracy is used in the Project.

B) Programming Languages, Libraries, Data Sets

In this project, I will code with Python Programming Language. Therefore, I will use Spyder and Jupyter Notebook which are the most useful ones of Anaconda Navigator. Anaconda also have different environment for different working field. In Machine Learning, Spyder and Jupyter are the most common use environments. Jupyter Notebook is an IPython notebook ("interactive python"). You can run each block of code separately. It is good at analyze data whether it be in pandas data frames or plots. Spyder is just an Integrated Development Environment (IDE) for python like visual studio, etc.

The Libraries that I will use are Numpy, Pandas, Seaborn, Matplotlib, SciPy, Scikit-Learn and XGB, using Tensor Flow backend.

I will use the UCI's data set that is "Auto MPG Data Set" from their machine learning repository ^[23]

C) Test and Evaluation Part

I will follow the steps that I mentioned in IV. A section. Then, I will go to the test phase after the coding is over. In test phase, I will give my model different data and evaluate the results. I will show the each code block running separately in Jupyter Notebook or Kaggle Notebook. Also, I will show the whole codes running in Spyder IDE.

VI. PROJECT SCHEDULE

I am planning to carry out this project according to this schedule.

(a) Gannt Chart

Progress of the Project.mvdx* x				
	Task Mode	Task Name	Start	End
1	☆	Progress of the Project	15.03.2021	15.05.2021
2	☆	Collecting Data Sets	15.03.2021	15.03.2021
3	☆	Exploratory Data Science	16.03.2021	16.03.2021
4	☆	Outlier Analysis and Feature Engineering	17.03.2021	19.03.2021
5	☆	Linear Regression and Regularization	20.03.2021	28.03.2021
6	☆	XGBoost Model	28.03.2021	1.04.2021
7	☆	Averaging Models and Conclusion	2.04.2021	16.04.2021
8	☆	Testing Period and Evaluation	17.04.2021	10.05.2021
9	☆	Documentation	26.02.2021	15.05.2021

(b) Gannt Chart

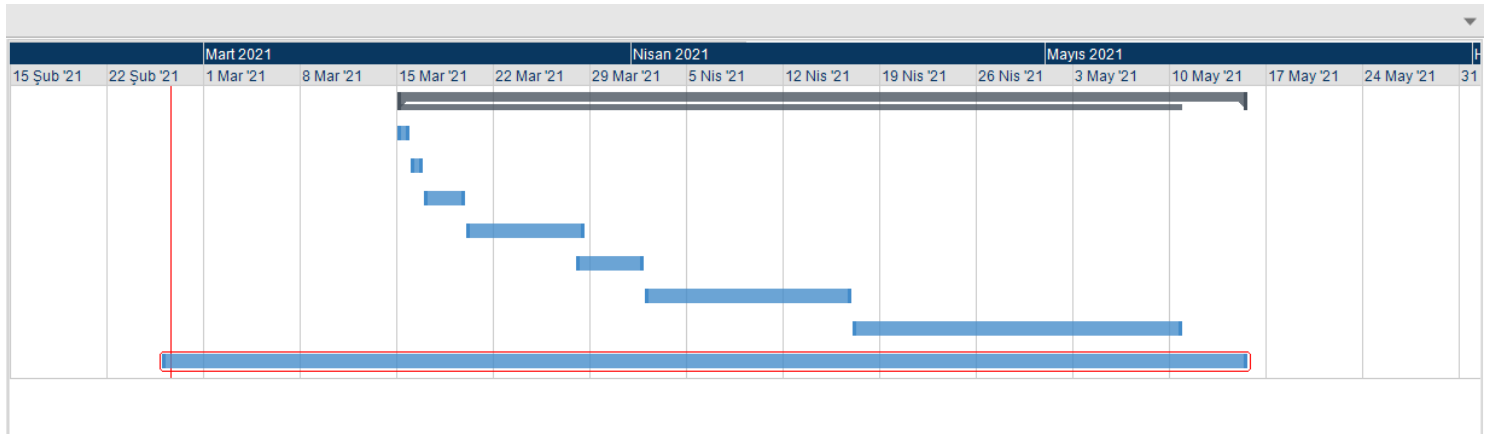
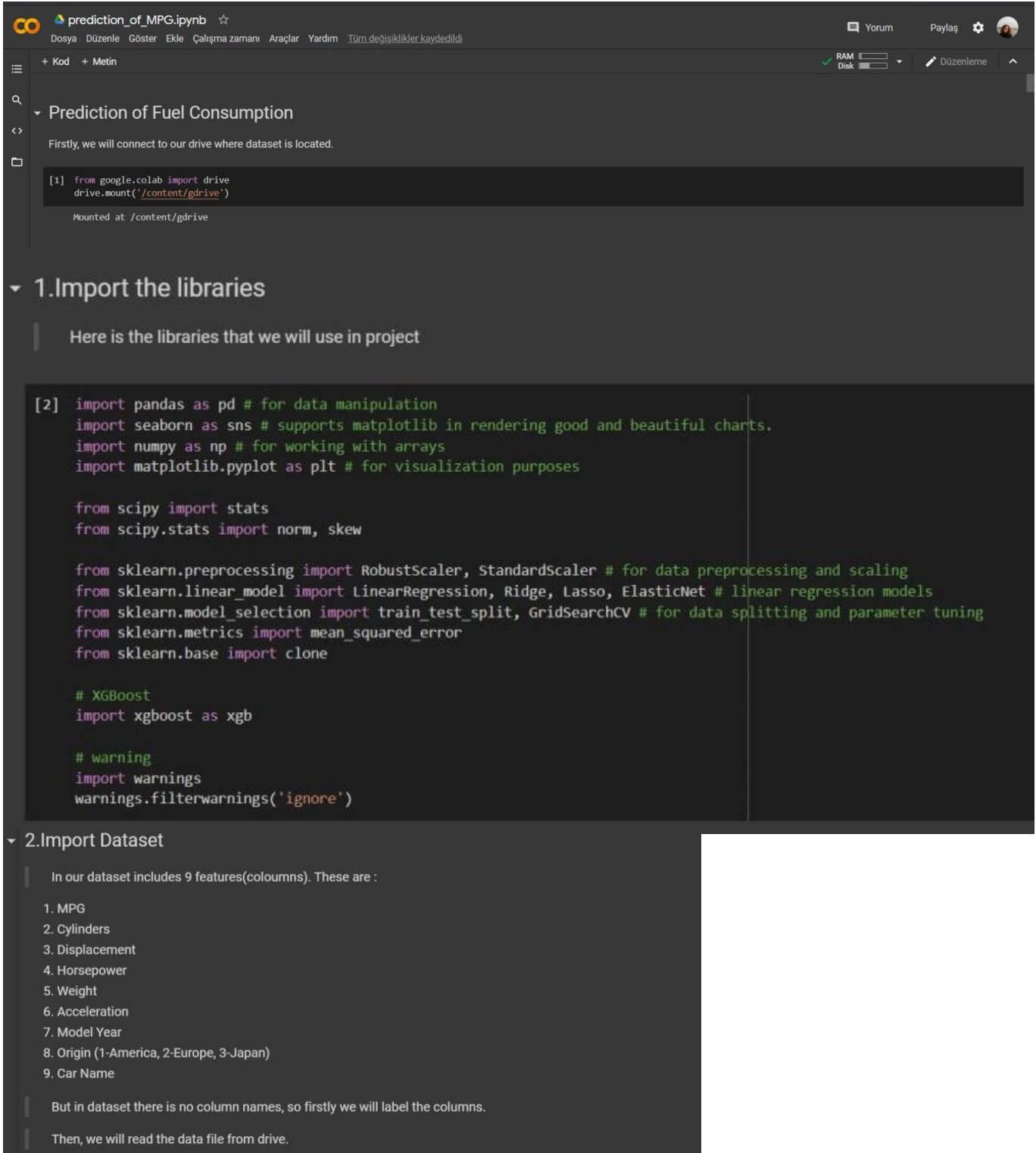


Figure 1. Gannt Charts

VII. PROJECT

I coded this Project on Colaboratory Platform which is a Jupyter notebook environment that runs in the browser using Google Cloud. Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. I also coded it on Spyder but I want to share with you this interactive python notebook (.ipynb) file because of Colab is more useful and easy to explain. ^[24]



The screenshot shows a Google Colab notebook interface. The title bar indicates the notebook is named 'prediction_of_MPG.ipynb'. The left sidebar shows the notebook's structure with sections for 'Prediction of Fuel Consumption' and '1.Import the libraries'. The main content area displays the following code:

```
[1] from google.colab import drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive
```

Below this, a section titled '1.Import the libraries' contains a code cell with the following imports:

```
[2] import pandas as pd # for data manipulation
import seaborn as sns # supports matplotlib in rendering good and beautiful charts.
import numpy as np # for working with arrays
import matplotlib.pyplot as plt # for visualization purposes

from scipy import stats
from scipy.stats import norm, skew

from sklearn.preprocessing import RobustScaler, StandardScaler # for data preprocessing and scaling
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet # linear regression models
from sklearn.model_selection import train_test_split, GridSearchCV # for data splitting and parameter tuning
from sklearn.metrics import mean_squared_error
from sklearn.base import clone

# XGBoost
import xgboost as xgb

# warning
import warnings
warnings.filterwarnings('ignore')
```

Below the code, a section titled '2.Import Dataset' contains a text block stating:

In our dataset includes 9 features(coloumns). These are :

1. MPG
2. Cylinders
3. Displacement
4. Horsepower
5. Weight
6. Acceleration
7. Model Year
8. Origin (1-America, 2-Europe, 3-Japan)
9. Car Name

Below the list, a text block states:

But in dataset there is no column names, so firstly we will label the columns.

Then, we will read the data file from drive.

```
[3] # We will not use car name attribute because it is not necessary for predicting MPG
column_name = ["MPG", "Cylinders", "Displacement", "Horsepower", "Weight", "Acceleration", "Model Year", "Origin"]

data = pd.read_csv("/content/gdrive/MyDrive/Prediction of Fuel Consumption/auto-mpg.data", names = column_name, na_values = "?", comment = "#", sep = " ", skipinitialspace = True)

# to be easy to understand we rename MPG to target
data = data.rename(columns = {"MPG": "TARGET"})
```

We will look at first 5 values and shape of the data

```
[4] print(data.head())
print("Data shape: ", data.shape)
```

	TARGET	Cylinders	Displacement	...	Acceleration	Model Year	Origin
0	18.0	8	307.0	...	12.0	70	1
1	15.0	8	350.0	...	11.5	70	1
2	18.0	8	318.0	...	11.0	70	1
3	16.0	8	304.0	...	12.0	70	1
4	17.0	8	302.0	...	10.5	70	1

```
[5 rows x 8 columns]
Data shape: (398, 8)
```

3. Missing Values

We can detect the data has null values or not

```
[5] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   TARGET          398 non-null    float64
1   Cylinders        398 non-null    int64
2   Displacement     398 non-null    float64
3   Horsepower       392 non-null    float64
4   Weight           398 non-null    float64
5   Acceleration     398 non-null    float64
6   Model Year       398 non-null    int64
7   Origin           398 non-null    int64
dtypes: float64(5), int64(3)
memory usage: 25.0 KB
```

```
[6] describe = data.describe()
print(describe)
```

	TARGET	Cylinders	...	Model Year	Origin
count	398.000000	398.000000	...	398.000000	398.000000
mean	23.514573	5.454774	...	76.010050	1.572864
std	7.815984	1.701004	...	3.697627	0.802055
min	9.000000	3.000000	...	70.000000	1.000000
25%	17.500000	4.000000	...	73.000000	1.000000
50%	23.000000	4.000000	...	76.000000	1.000000
75%	29.000000	8.000000	...	79.000000	2.000000
max	46.600000	8.000000	...	82.000000	3.000000

```
[8 rows x 8 columns]
```

```
[7] # How much missing value has each column?
```

```
print(data.isna().sum())
```

```
TARGET      0
Cylinders    0
Displacement 0
Horsepower   6
Weight       0
Acceleration 0
Model Year   0
Origin       0
dtype: int64
```

We can see the 'Horsepower' column has 6 missing values

```
[8] # Take the mean of horsepower and fill it into the null values

data["Horsepower"] = data["Horsepower"].fillna(data["Horsepower"].mean())

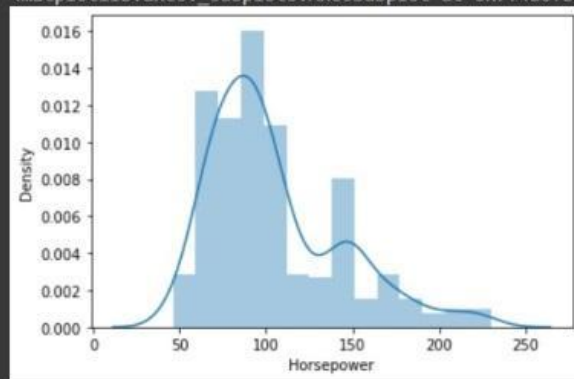
# Check missing values again

print(data.isna().sum())
```

```
TARGET      0
Cylinders    0
Displacement 0
Horsepower   0
Weight       0
Acceleration 0
Model Year   0
Origin       0
dtype: int64
```

```
[9] sns.distplot(data.Horsepower)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4d6fb47310>
```



4.Exploratory Data Analysis

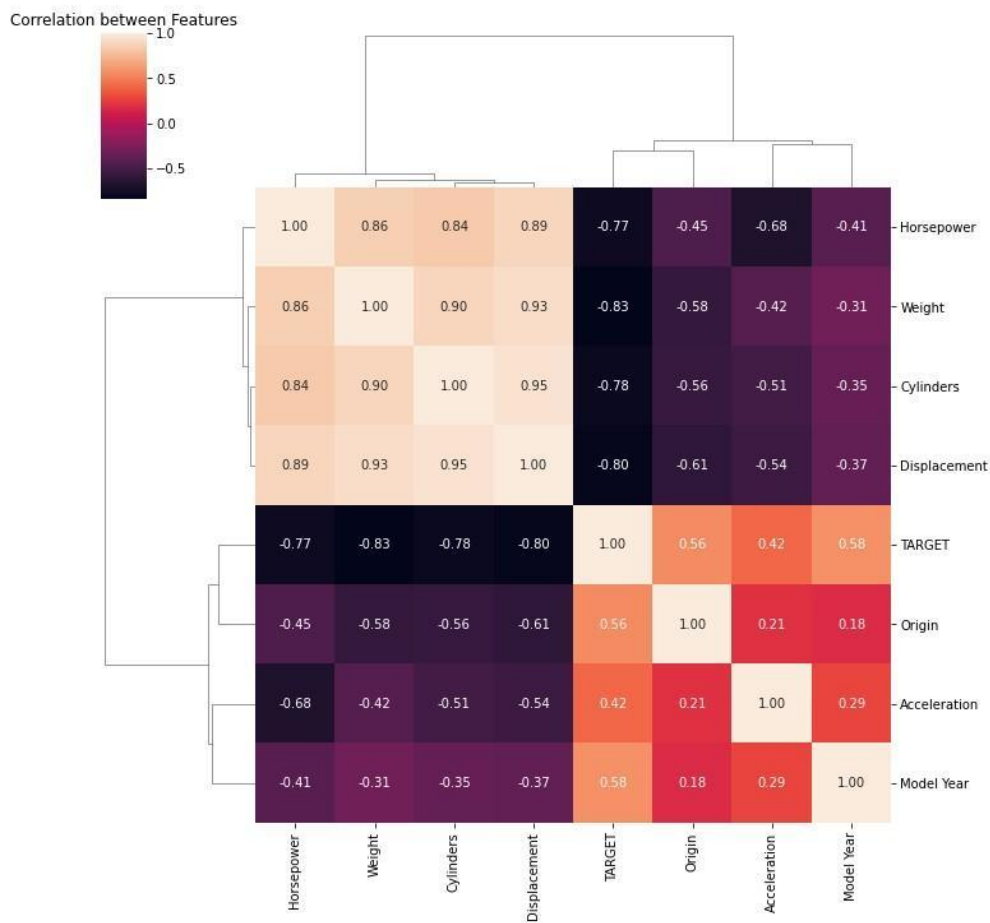
Firstly, we should look at correlation between features.

We can see the negative and positive correlation between features. Negative correlation means that they have inverse proportion, positive correlation means direct proportion

```
[10] # EDA (Exploratory Data Analysis)

corr_matrix = data.corr()
sns.clustermap(corr_matrix, annot = True, fmt = ".2f" )
plt.title("Correlation between Features")
plt.show()
```

Figure 4.Filling the null values and starting the EDA



We will put one filter in order to see features above 0.75.

The reason we do this is to make it easier to see in the pairplot and make the correlation matrix smaller.

```
[11]
threshold = 0.75
filter = np.abs(corr_matrix["TARGET"]) > threshold # take the thresholds related to dependence variable(Target)
corr_features = corr_matrix.columns[filter].tolist()
sns.clustermap(data[corr_features].corr(), annot = True, fmt = ".2f" )
plt.title("Correlation between Features")
plt.show()
```



With pairplot we can see the whole picture that relationship between them

```
sns.pairplot(data, diag_kind = "kde", markers = "+")  
plt.show()
```

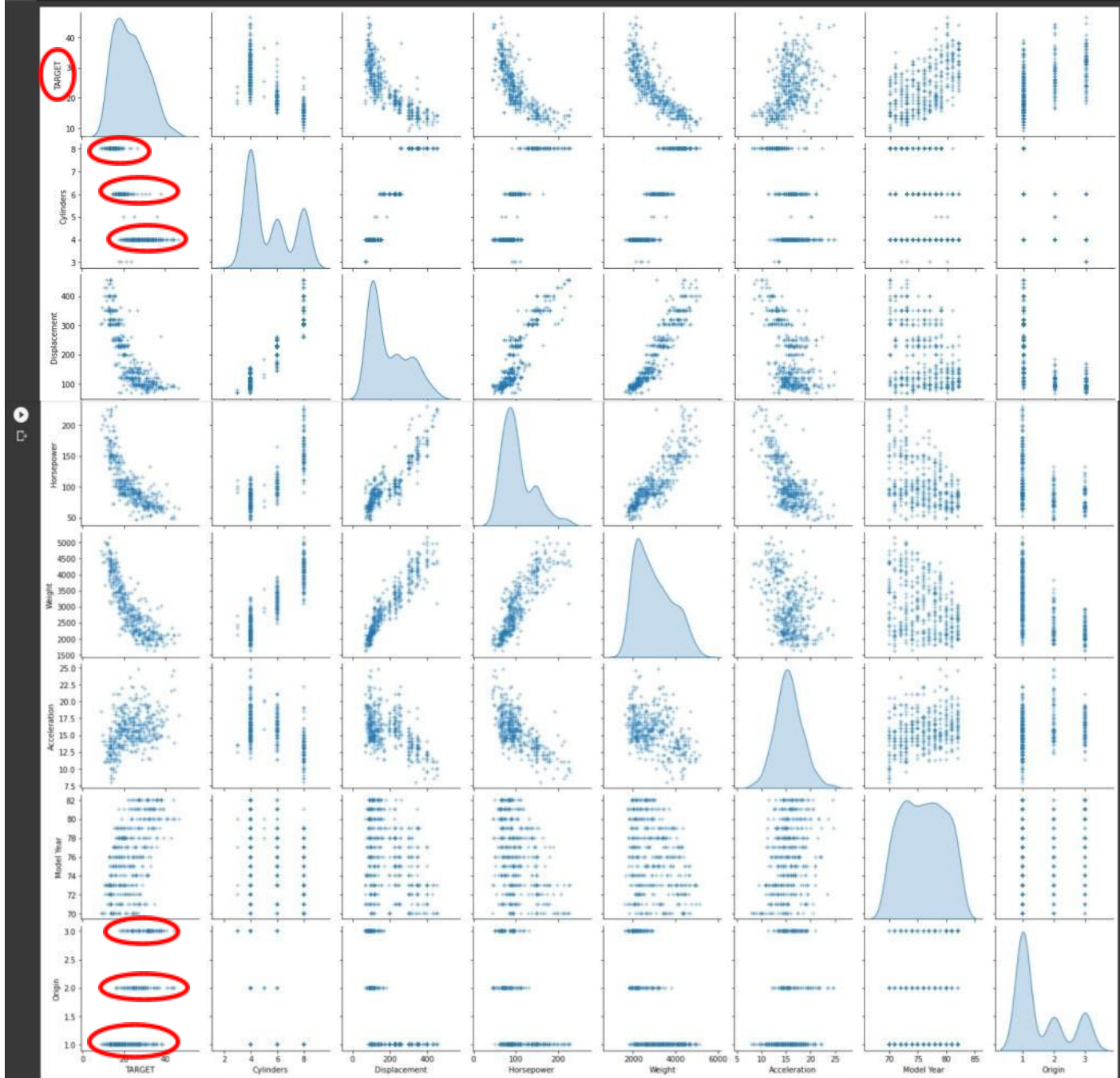
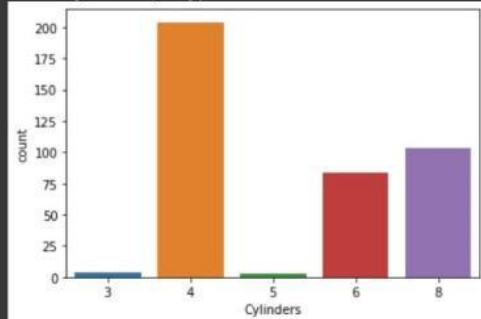


Figure7. Pairplot schema for all features

Looking at their relationship with target in the graphs above, we can see that 'Cylinders' and 'Origin' columns are categorical data.

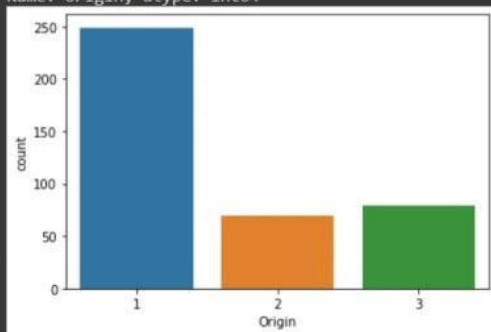
```
[13] plt.figure()
sns.countplot(data["Cylinders"])
print(data["Cylinders"].value_counts())
```

```
4    204
8    103
6     84
3      4
5       3
Name: Cylinders, dtype: int64
```



```
[14] plt.figure()
sns.countplot(data["Origin"])
print(data["Origin"].value_counts())
```

```
1    249
3     79
2     70
Name: Origin, dtype: int64
```



5.Outliers

We can see there is outliers or not with using boxplot

```
for c in data.columns:
    plt.figure()
    sns.boxplot(x = c, data = data, orient = "v")
```

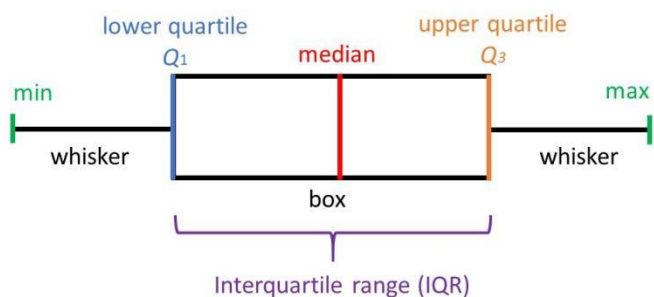


Figure 8. Showing the categorical features

You can see the boxplot schema here. We should remove the outliers from our data. Using this parameters, we will write the code that drops the outliers.

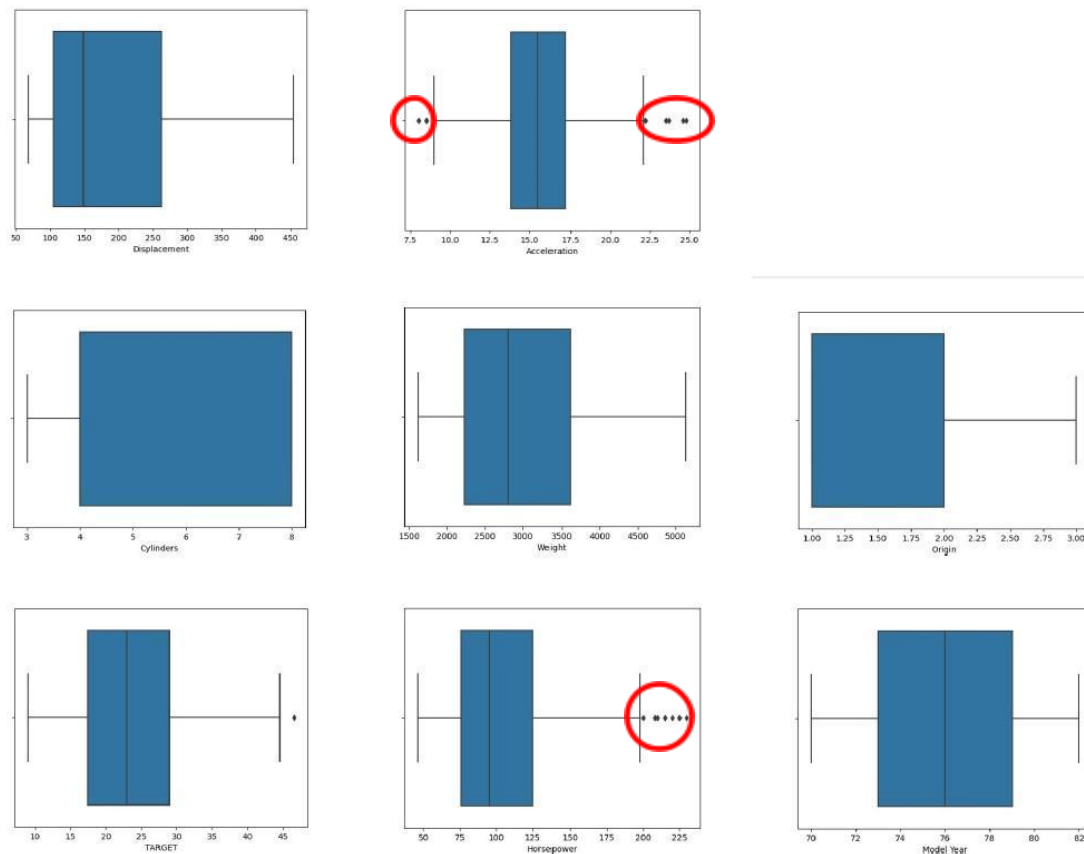


Figure 10. Detect the outliers from boxplot schema

As we can see, 'Horsepower' and 'Acceleration' columns has outliers.

So we should drop the outliers for well-working predicting model

```
[16]
thr = 2

horsepower_desc = describe["Horsepower"]

q3_hp = horsepower_desc[6]
q1_hp = horsepower_desc[4]
IQR_hp = q3_hp - q1_hp

top_limit_hp = q3_hp + thr*IQR_hp
bottom_limit_hp = q1_hp - thr*IQR_hp

filter_hp_bottom = bottom_limit_hp < data["Horsepower"]
filter_hp_top = data["Horsepower"] < top_limit_hp

filter_hp = filter_hp_bottom & filter_hp_top

data = data[filter_hp] # remove Horsepower outliers
```

Figure 11. Drop the outlier from horsepower column

```
[17] acceleration_desc = describe["Acceleration"]

q3_acc = acceleration_desc[6]
q1_acc = acceleration_desc[4]
IQR_acc = q3_acc - q1_acc # q3 - q1

top_limit_acc = q3_acc + thr*IQR_acc
bottom_limit_acc = q1_acc - thr*IQR_acc

filter_acc_bottom = bottom_limit_acc < data["Acceleration"]
filter_acc_top = data["Acceleration"] < top_limit_acc

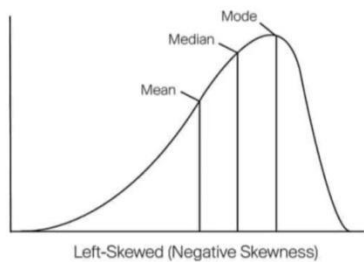
filter_acc = filter_acc_bottom & filter_acc_top

data = data[filter_acc] # remove Acceleration outliers
```

Figure 12. Drop the outlier from acceleration column

Skewness

Left Skewed or Negative Skewed



Right Skewed or Postive Skewed

So, the distribution which is right skewed have a long tail that extends to the right or positive side of the x axis, same as the below plot.

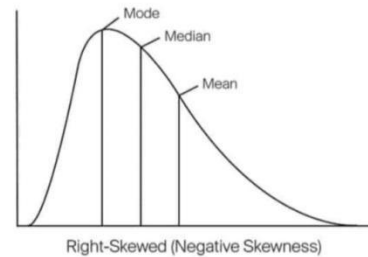


Figure
13. Skewness

6. Feature Engineering

Skewness

```
[18] # Target is dependent variable

sns.distplot(data.TARGET, fit = norm)

(mu, sigma) = norm.fit(data["TARGET"])

print("mu : {}, sigma : {}".format(mu, sigma))

mu : 23.472405063291134, sigma : 7.756119546409932
```

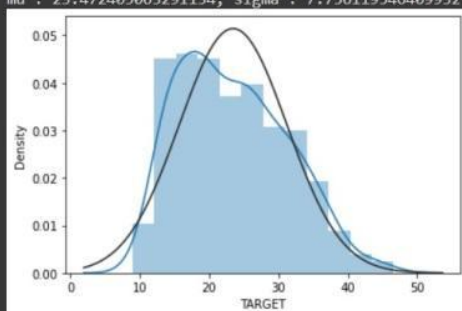
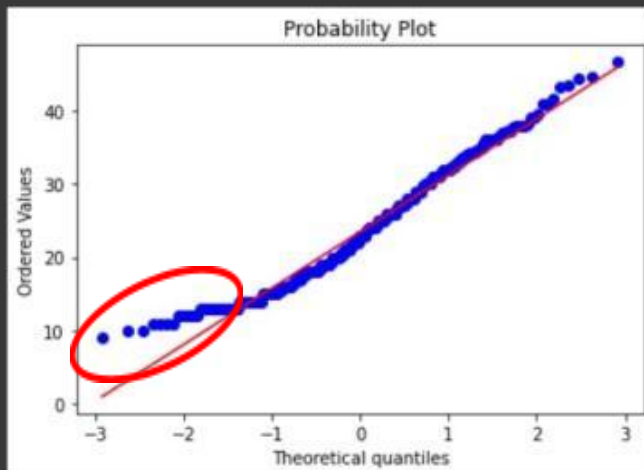


Figure 14.
Shows the
skewness
with displot
method

As we can see, we have Right-Skewed

```
[19] # qq plot (quantile quantile plot)

plt.figure()
stats.probplot(data["TARGET"], plot = plt)
plt.show()
```



Our data should be on the red line

```
[20]
data["TARGET"] = np.log1p(data["TARGET"])

plt.figure()
sns.distplot(data.TARGET, fit = norm)

(mu, sigma) = norm.fit(data["TARGET"])

print("mu : {}, sigma : {}".format(mu, sigma))

mu : 3.146474056830183, sigma : 0.3227569103044822
```

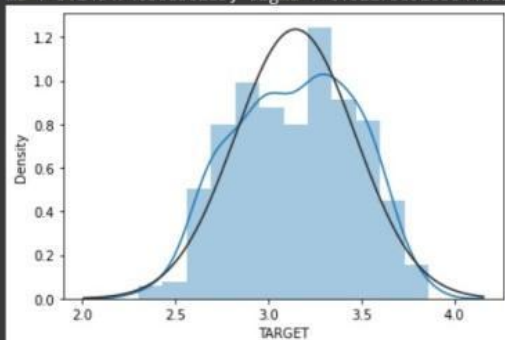
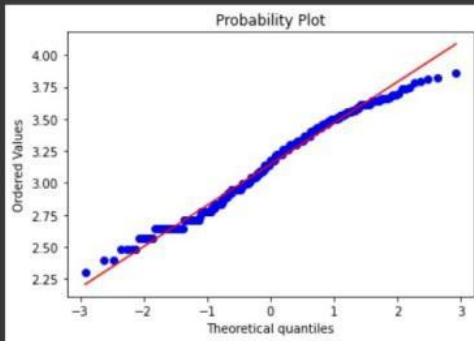


Figure 15. Fixing the skewness

```
[21] plt.figure()
stats.probplot(data["TARGET"], plot = plt)
plt.show()
```



```
[22] # feature - independent variable

skewed_feats = data.apply(lambda x : skew(x.dropna())).sort_values(ascending = False)
skewness = pd.DataFrame(skewed_feats, columns = ["Skewed"])
```

One Hot Encoding

```
[23] data["Cylinders"] = data["Cylinders"].astype(str)
data["Origin"] = data["Origin"].astype(str)

data = pd.get_dummies(data)
```

7.Split and Standardization

```
[24] # Split

x = data.drop(["TARGET"], axis = 1)
y = data.TARGET

test_size = 0.2
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = test_size, random_state = 42)
```

```
[25] # Standardization

scaler = StandardScaler() # RobustScaler or StandardScaler
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Figure 16. Applying the One Hot Encoding, Split and Standardization

Regression Models



```
# linear regression

lr = LinearRegression()
lr.fit(X_train, Y_train)

print("LR Coef : ", lr.coef_)

y_predicted_dummy = lr.predict(X_test)
mse = mean_squared_error(Y_test, y_predicted_dummy)

print("Linear Regression MSE : ", mse)
print("Test Accuracy: {}".format(lr.score(X_test, Y_test)))
```

```
LR Coef : [ 4.33539196e-02 -7.00351554e-02 -1.74807489e-01 -1.45415203e-02
 9.99434620e-02 -2.27927155e-02  3.74789759e-02  1.12061085e-02
-2.24108324e-02 -1.84760400e-02 -1.72042847e-02  1.38324896e-04
 2.05067107e-02]
Linear Regression MSE :  0.01339976714675724
Test Accuracy: 0.8448968040442395
```

```
[27] # Ridge Regression (L2) - minimizing (least-squared error +  $\lambda \cdot (\text{slope})^2$ ) --> advantage - prevents from overfitting
```

```
ridge = Ridge(random_state = 42, max_iter = 10000)

alphas = np.logspace(-4, -0.5, 30)

tuned_parameters = [{"alpha": alphas}]

n_folds = 5
```

```
[28] #classifier
clf = GridSearchCV(ridge, tuned_parameters, cv = n_folds, scoring = "neg_mean_squared_error", refit = True)
clf.fit(X_train, Y_train)
scores = clf.cv_results_["mean_test_score"]
scores_std = clf.cv_results_["std_test_score"]
```

```
[29] print("Ridge Coef : ", clf.best_estimator_.coef_)
ridge = clf.best_estimator_
print("Ridge Best Estimator : ", ridge)

y_predicted_dummy_rd = clf.predict(X_test)

mse = mean_squared_error(Y_test, y_predicted_dummy_rd)
print("Ridge MSE : ", mse)
print("Test Accuracy: {}".format(ridge.score(X_test, Y_test)))
```

```
Ridge Coef : [ 4.06552738e-02 -7.00085623e-02 -1.72917369e-01 -1.48054915e-02
 9.97118288e-02 -2.30116500e-02  3.72295686e-02  1.11324719e-02
-2.23907013e-02 -1.81369765e-02 -1.70931886e-02 -1.56489559e-05
 2.05155214e-02]
Ridge Best Estimator : Ridge(alpha=0.31622776601683794, copy_X=True, fit_intercept=True,
 max_iter=10000, normalize=False, random_state=42, solver='auto',
 tol=0.001)
Ridge MSE :  0.013421245209101306
Test Accuracy: 0.8446481940440793
```

Figure 17. Linear and Ridge Regression


```
[31] # Lasso Regression (L1) - minimizing (least-squared error *  $\lambda$  * slope) --> advantage - redundant coef values are assigned 0

lasso = Lasso(random_state = 42, max_iter = 10000)
alphas = np.logspace(-4, -0.5, 30)

tuned_parameters = [{'alpha': alphas}]
n_folds = 5

[32] clf = GridSearchCV(lasso, tuned_parameters, cv = n_folds, scoring = "neg_mean_squared_error", refit = True)
clf.fit(X_train, Y_train)
scores = clf.cv_results_["mean_test_score"]
scores_std = clf.cv_results_["std_test_score"]

[33] print("Lasso Coef : ", clf.best_estimator_.coef_)
lasso = clf.best_estimator_

y_predicted_dummy_lass = clf.predict(X_test)
mse = mean_squared_error(Y_test, y_predicted_dummy_lass)
print("Lasso MSE: ", mse)
print("Test Accuracy: {}".format(lasso.score(X_test, Y_test)))

Lasso Coef : [ 0.03139269 -0.06659594 -0.17176      -0.01408541  0.09942275 -0.01962127
 0.0527184   0.01407385 -0.0072404   -0.         -0.01585649  0.
 0.02037116]
Lasso MSE:  0.013403675670455173
Test Accuracy: 0.844851562622476

[35] # ElasticNet - minimizing (least-squared error +  $\lambda_1 * (\text{slope})^2 + \lambda_2 * \text{slope}$ ) Like mixed version of lasso and ridge

parametersGrid = {"alpha" : alphas,
                  "l1_ratio" : np.arange(0.0, 1.0, 0.05)}

eNet = ElasticNet(random_state = 42, max_iter = 10000)
clf = GridSearchCV(eNet, parametersGrid, cv = n_folds, scoring = "neg_mean_squared_error", refit = True)
clf.fit(X_train, Y_train)

eNet = clf.best_estimator_

print("ElasticNet Coef : ", clf.best_estimator_.coef_)
print("ElasticNet Best Estimator : ", clf.best_estimator_)

y_predicted_dummy_eNet = clf.predict(X_test)
mse = mean_squared_error(Y_test, y_predicted_dummy_eNet)
print("ElasticNet MSE : ", mse)
print("Test Accuracy: {}".format(eNet.score(X_test, Y_test)))

ElasticNet Coef : [ 0.02440704 -0.06980685 -0.16082902 -0.01634179  0.09817019 -0.02423724
 0.0360529   0.01068281 -0.02229063 -0.01647006 -0.01646491 -0.00095338
 0.02062713]
ElasticNet Best Estimator : ElasticNet(alpha=0.008531678524172814, copy_X=True, fit_intercept=True,
l1_ratio=0.0, max_iter=10000, normalize=False, positive=False,
precompute=False, random_state=42, selection='cyclic', tol=0.0001,
warm_start=False)
ElasticNet MSE :  0.013576740895132198
Test Accuracy: 0.8428483211360965
```

Figure 18. Lasso and Elastic Net Regression

```
[36] # XGBoost is an algorithm that designed for big, complex data sets.

parametersGrid = {"nthread" : [4], #when use hyperthread, xgboost may become slower
                  "objective" : ["reg:linear"],
                  "learning_rate" : [.03, 0.05, .07],
                  "max_depth" : [5, 6, 7],
                  "min_child_weight" : [4],
                  "silent" : [1],
                  "subsample" : [0.7],
                  "colsample_bytree" : [0.7],
                  "n_estimators" : [500, 1000] }

model_xgb = xgb.XGBRegressor()

[37] clf = GridSearchCV(model_xgb, parametersGrid, cv = n_folds, scoring = "neg_mean_squared_error", refit = True, n_jobs = 5, verbose = True)

clf.fit(X_train, Y_train)
model_xgb = clf.best_estimator_

y_predicted_dummy_xgb = clf.predict(X_test)
mse = mean_squared_error(Y_test, y_predicted_dummy_xgb)
print("XGBRegressor MSE: ", mse)
print("Test Accuracy: {}".format(model_xgb.score(X_test, Y_test)))

Fitting 5 folds for each of 18 candidates, totalling 90 fits
[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done 40 tasks | elapsed: 36.4s
[Parallel(n_jobs=5)]: Done 90 out of 90 | elapsed: 1.3min finished
XGBRegressor MSE: 0.008169698221682037
Test Accuracy: 0.905435199709151
```

Figure 19. Applying the XGB model

We should compare our Linear, Ridge, Lasso, Elastic Net and XGB Regularization models. As we can see that XGB model and Lasso gives us the best results. So we create a class that takes the average of these two models.

▼ Averaging Models

```
[38] class AveragingModels():

    def __init__(self, models):
        self.models = models

    # we define clones of the original models to fit the data in
    def fit(self, X, y):
        self.models_ = [clone(x) for x in self.models]

        # Train cloned base models
        for model in self.models_:
            model.fit(X, y)

        return self

    # Now we do the predictions for cloned models and average them
    def predict(self, X):
        predictions = np.column_stack([model.predict(X) for model in self.models_])
        return np.mean(predictions, axis = 1)

[39] averaged_models = AveragingModels(models = (model_xgb, lasso))
averaged_models.fit(X_train, Y_train)

y_predicted_dummy_avg = averaged_models.predict(X_test)
mse = mean_squared_error(Y_test, y_predicted_dummy_avg)
print("Averaged Models MSE: ", mse)

Averaged Models MSE: 0.00972521095311186
```

Figure 20.
Taking the
Average of
best two
models

As you can see that mean squared error of XGB model is the smallest. So we should predict our MPG in test data using XGB model.

We have seen that only XGBoost model better than average of XGBoost and Lasso. So we should use the predicted value of XGB

We have to use `np.exp1` transformation because we did `np.log1p` transformation before.

```
[43] y_test_value = np.exp1(y_test)
      predicted_value = np.exp1(y_predicted_dummy_xgb)
```

Figure 21. Take the Exponential transform

VIII. TEST AND EVALUATION

In this section, I will explain step by step what I changed and what happened after I changed some codes.

Firstly, in my codes, I did logarithm transform to reduce the skewness value at feature extraction part. If I do not do this, our prediction model is not running properly so my test accuracy becomes smaller than before. So I have to add a code line which has “**np.log1p**” transformation of my “**TARGET**” data value. But if I do this transform, at the end of my codes I have to rearrange this values. So I have to add a code line which will be inverse of the logarithm transform. That is to say, I have to use “**np.expm1**” transformation for necessary parameter.

```
# %% Feature Extraction

# Skewness

# Target dependent variable

sns.distplot(data.TARGET, fit = norm)

(mu, sigma) = norm.fit(data["TARGET"])

print("mu : {}, sigma : {}".format(mu, sigma))

# qq plot (quantile quantile plot)

plt.figure()
stats.probplot(data["TARGET"], plot = plt)
plt.show()

data["TARGET"] = np.log1p(data["TARGET"]) # we do the logarithm transform to reduce the skewness

plt.figure()
sns.distplot(data.TARGET, fit = norm)
```

Figure 22. Taking the logarithm transform

```
averaged_models = AveragingModels(models = (lasso, model_xgb))
averaged_models.fit(X_train, Y_train)

y_predicted_dummy_avg = averaged_models.predict(X_test)
mse = mean_squared_error(Y_test, y_predicted_dummy_avg)
print("Averaged Models MSE: ", mse)

y_test_value = np.expm1(Y_test)
predicted_value = np.expm1(y_predicted_dummy_avg)
```

Figure 23. Take the Exponencial transform

Secondly, in split and standardization part I have to adjust the test size in an effective way. So I decided to make it 0.6 which means 60% of data selected for the test, 40% of data selected for the train. But I know that if I keep the percentage of the training data higher, I get better accuracy of the test data. So I reduce the test size value 0, 1 by 0, 1 and I get the best accuracy when **test size** is **0, 2**.

Furthermore, I used the **Standard Scaler** in standardization instead of Robust Scaler. Actually, Robust is more irresponsive about outliers than Standard. We could use RobustScaler if we have outliers and want to reduce their influence. But we had dropped outliers manually, so we can use Standard Scaler if we need a relatively normal distribution.

Finally, I tried all the regression models that I mentioned it in Project Plan before. These are Linear, Ridge, Lasso, Elastic Net and XGB models. Each of them has different mean squared error and test accuracy value. We have to take the model which gives us the best accuracy. So took average of the models that gives best accuracy.

But taking every result into account, we can see that XGB model performs better than even the average of XGB and Lasso. So, at the end of my Project I decided to use XGB model for predicting fuel consumption.

IX. CONCLUSION

In this project, based on UCI database that is originally collected for the intent of fuel consumption prediction, I demonstrate an approach for finding a model which has a good accuracy. The model in my study includes the parameters which are cylinders, displacement, horsepower, weight, acceleration and origin. Using this attributes the XGB model gives us 90% accuracy for predicting fuel consumption.

Although I consider my work to have been successful, there are also sections which need improvement. For instance, there are other features that affect fuel consumption such as driver's behavior and environmental factors. I believe that if these factors included in the project there will be more successful software than I had.

Taking every study into account, there are a lot of researchers who have taken a lot of approaches to reduce fuel consumption. In the light of these researches, I believe transforming vehicles into autonomous vehicles may provide a better solution to this problem. Because features that are difficult to determine, such as driver behavior and speed, will become more predictable. With the integration of Artificial Intelligence into our lives, we can achieve a more sustainable life.

X. REFERENCES

- [1]: <https://www.iea.org/reports/fuel-consumption-of-cars-and-vans>
- [2]: 1- SIAMI-IRDEMOOSA, Elnaz; DINDARLOO, Saeid R. Prediction of fuel consumption of mining dump trucks: A neural networks approach. *Applied Energy*, 2015, 151: 77-84.
- 2- HU, Zhihui, et al. Prediction of fuel consumption for enroute ship based on machine learning. *IEEE Access*, 2019, 7: 119497-119505.
- 3- KIM, S. C.; KIM, K. U.; KIM, D. C. Prediction of fuel consumption of agricultural tractors. *Applied Engineering in Agriculture*, 2011, 27.5: 705-709.
- 4- ZHAO, Qi; CHEN, Qi; WANG, Li. Real-Time Prediction of Fuel Consumption Based on Digital Map API. *Applied Sciences*, 2019, 9.7: 1369.
- [3]: [UCI \(University of California, Irvine Campus\)](https://www.uci.edu/). They currently maintain free 585 data sets as a service to the machine learning community.
- [4]: <http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm>
- [5]: <https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a>
- [6]: <https://www.geeksforgeeks.org/lasso-vs-ridge-vs-elastic-net-ml/>
<https://towardsdatascience.com/what-is-regularization-and-how-do-i-use-it-f7008b5a68c6>
- [7]: <https://levity.ai/blog/difference-machine-learning-deep-learning>
- [8]: Afana, M., Ahmed, J., Harb, B., Abu-Nasser, B. S., & Abu-Naser, S. S. (2018). Artificial Neural Network for Forecasting Car Mileage per Gallon in the City.
- [9]: https://en.wikipedia.org/wiki/Artificial_neural_network
- [10]:
<https://towardsdatascience.com/machine-learning-how-black-is-this-black-box-f11e4031fdf>
- [11]: Al Barsh, Y. I., Duhair, M. K., Ismail, H. J., Abu-Nasser, B. S., & Abu-Naser, S. S. (2020). MPG Prediction Using Artificial Neural Network.
- [12]: https://en.wikipedia.org/wiki/Gradient_descent
- [13]: <https://www.marinedatascience.co/blog/2019/01/07/normalizing-the-rmse/>
- [14]: Shirbhayye, V., Kurmi, D., Dyavanapalli, S., Prasad, A. S. H., & Lal, N. (2020, January). An Accurate Prediction of MPG (Miles Per Gallon) using Linear Regression Model of Machine Learning. In 2020 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-5). IEEE.

^[15]: TOGUN, Necla Kara; BAYSEC, Sedat. Prediction of torque and specific fuel consumption of a gasoline engine by using artificial neural networks. Applied Energy, 2010, 87.1: 349-355.

^[16]: <https://sites.insead.edu/facultyresearch/research/doc.cfm?did=46875#:~:text=The%20two%20biggest%20advantages%20of,of%20the%20uncertainty%20in%20forecasting.>

^[17]: <https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a>

^[18]: <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>

^[19]: <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>

^[20]: <https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net>

^[21]: <https://en.wikipedia.org/wiki/XGBoost>

^[22]: KOTSIANTIS, Sotiris B.; KANELLOPOULOS, Dimitris; ZAHARAKIS, Ioannis D. Bagged averaging of regression models. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, Boston, MA, 2006. p. 53-60.

^[23]: <https://archive.ics.uci.edu/ml/datasets/Auto+MPG>

^[24]: https://drive.google.com/drive/folders/1y_0-ljk0ArqbZD9I7BJ0mEOTthC6jUQj?usp=sharing

(If it is not working with clicking, you should copy the link and paste it your browser)