Pranav Veerubhotla
MEEN 423-500
5/3/2022

Pranav Veerubhotla
Spring 2022
MEEN 423-500
Final Project

Executive Summary

The purpose of this project is to determine whether key environmental data regarding vehicles can be predicted using readily available consumer information. Specifically, this report aims to use Machine Learning (ML) techniques to make predictions regarding a vehicles combined fuel consumption and CO_2 emissions using regression techniques. Both Multiple Linear Regression (MLR) and Artificial Neural Networks (ANNs) will be tested to determine which ML technique best suits the data used when predicting the aforementioned environmental vehicle statistics.

Through experimentation, it was observed that the ANN models consistently outperformed the MLR models implemented. Using mean-squared error (*mse*) as the evaluation metric, the best preforming model was found to be a simple one hundred node, one-layer neural network utilizing a 'logistic' activation function. However, general trends recorded during testing demonstrated the 'tanh' activation function to produce more consistent results; with single layer and multi-layer ANNs achieving similar *mse* values, whilst 'logistic' ANNs with multiple layers had significantly larger *mse* values.

Testing with the MLR models did reveal interesting information regarding the data set used. Consulting a scatter matrix of the data, features 'Vehicle Class' and 'Cylinders' were theorized to be medium to highly correlated with the resulting combined fuel consumption and CO_2 emissions. Thus, an additional MLR model was trained for each prediction case with access to only the two aforementioned features. Using the r^2 value calculated through `SkLearn_metrics score()` function, it was noted that the MLR models with access to only two features preformed nearly as well at the models provided access to all features. Therefore, we can conclude through our manual testing that the 'Vehicle Class' and 'Cylinders' features function as principal components within our dataset.

The final recommendation for choice of ML model would be the best preforming model as indicated by the preforming tests; *ANN-1* a simple one hundred node, one-layer neural network utilizing a 'logistic' activation function.

ML Problem

Today, the best-selling cars are not actually cars, or sedans rather, themselves but instead Sport Utility Vehicles or SUVs. Over the past three decades, North Americans have proven their interest in vehicles of this segment with their purchasing history. Today, it is more likely that a family has an SUV or truck instead of a sedan, especially if that is the only vehicle owned.

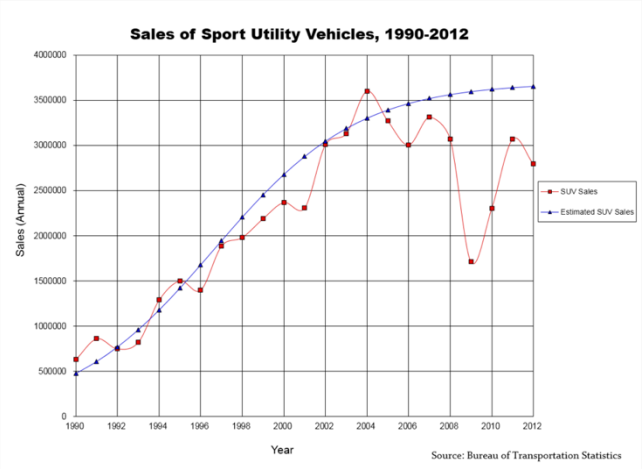

Figure 1: SUV Sales (Projected and Actual) in North America [1990-2012]

Despite SUVs offering greater protection in the event of a crash, and an elevated driving position for the driver, they are not without their costs. Aside from obviously being more expensive to purchase initially compared to their sedan counterparts, they are also more expensive to run and worse for the environment. Due to their heavy weight and poor aerodynamic profile, SUVs require larger engines with more power and torque to achieve the same speeds as a smaller, lighter sedan. Thus, due to the larger engines, sometimes in conjunction with all-wheel or four-wheel drive systems which further compound drivetrain losses, the SUVs often return poor fuel economy and release substantial amounts of carbon dioxide into the atmosphere.

The question set to be answered during this project is the following:

*Is it possible to **predict** fuel economy & CO_2 emissions using basic consumer vehicle data?*

To answer this question, the dataset detailed below was used in conjunction with various machine learning methods. The following are the main two ML methods that were applied to answer the above question:

I. Multiple Linear Regression
II. Artificial Neural Networks

In addition to answering the above question, it will also be determined whether the same conclusion can be drawn when the number of features are greatly decreased.

Dataset Analysis

The data set used to conduct the analyses was: *2022 Fuel Consumption Ratings* (source :: Rini) and was published on April 5th of this year. Thus, it will provide data reflective of the current state of the American car market. Data collected included the following measurements:

model year (all 2022), make, model, vehicle class, engine size (L), cylinders, transmission, fuel type, fuel consumption (City [L/100 km]), fuel consumption (Highway [L/100 km]), fuel consumption (Combined [L/100 km]), fuel consumption (Combined [mpg]), CO_2 emissions [g/km], CO_2 rating, & smog rating.

The above columns are then split into 'features and 'response' variables as such. As the data set also consists of categorical measurements, some features must be encoded using the following scheme. (*Appendix* – full encoding process dictionary files)

Table 1: Features and Response Variables along with Encoding Scheme

Variable	Type	Categorical	Encoded Range
'Make'	*Feature*	*Y*	$0 < N < 38$
'Model'	*Feature*	*Y*	$0 < N < 714$
'Vehicle Class'	*Feature*	*Y*	$0 < N < 13$
'Engine Size (L)'	*Feature*	*N*	*N/A*
'Cylinders'	*Feature*	*N*	*N/A*
'Transmission'	*Feature*	*Y*	$0 < N < 22$
'Fuel Type'	*Feature*	*Y*	$0 < N < 3$
'Fuel Consumption (City (L/100 km)'	*Response*	*N*	*N/A*
'Fuel Consumption (Hwy (L/100 km))'	*Response*	*N*	*N/A*
'Fuel Consumption (Comb (L/100 km))'	*Response*	*N*	*N/A*
'Fuel Consumption (Comb (mpg))'	*Response*	*N*	*N/A*
'CO2 Emissions (g/km)',	*Response*	*N*	*N/A*
'CO2 Rating'	*Response*	*N*	*N/A*
'Smog Rating'	*Response*	*N*	*N/A*

Having converted all data into numbers, we are then able to produce a scatter matrix potentially demonstrating any correlations between all columns in the dataset.

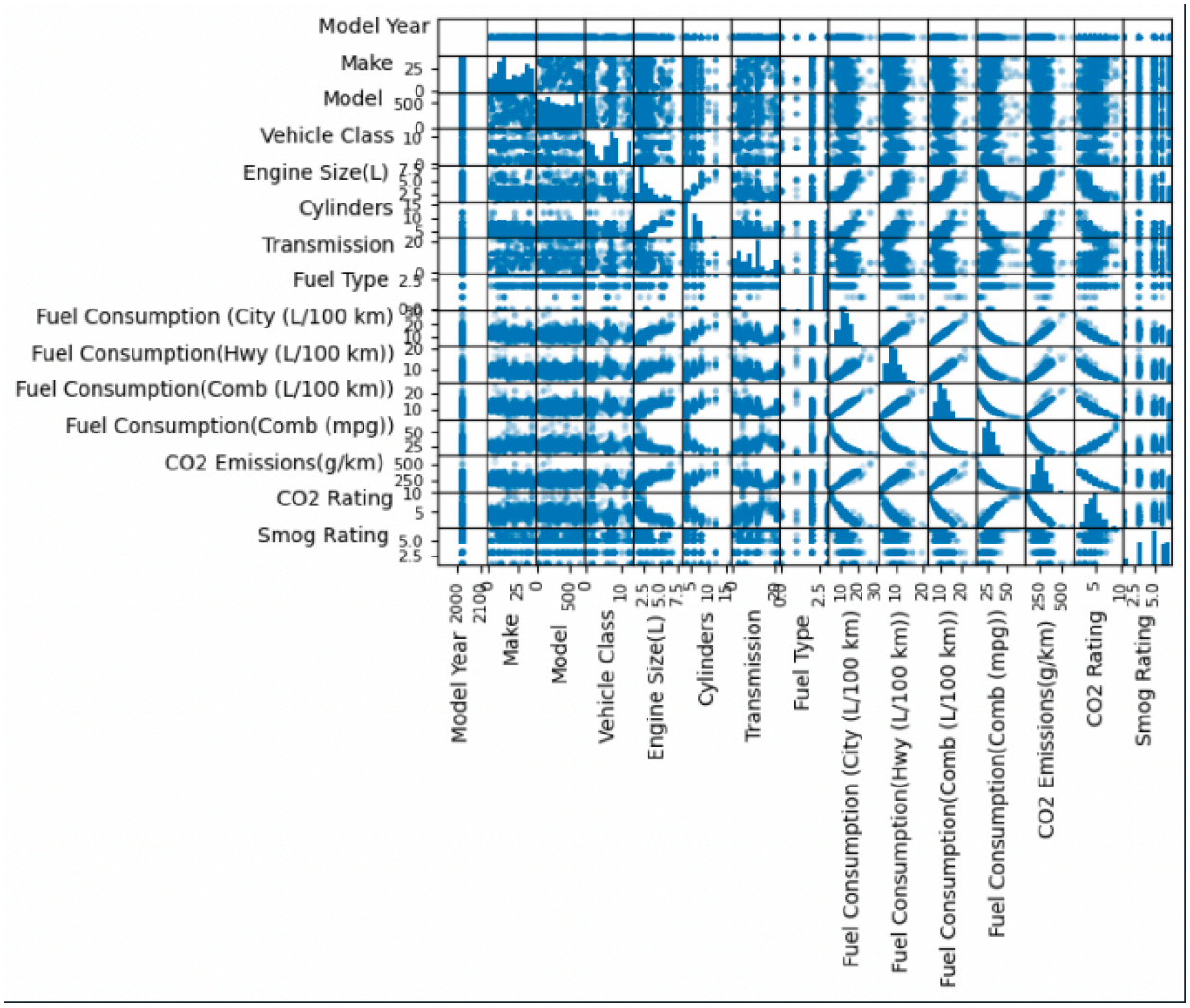

Figure 2: Scatter Matrix of <u>all</u> Features and Response Variables

From the scatter matrix above, correlations (although somewhat noisy), it can be determined that correlations exist between the engine size (L), cylinders, transmission, and faintly with vehicle class. These correlations will be noted later to attempt reproduction of the results using fewer feature columns, in the vein of *Principal Components Analysis (PCA)*. To

further visualize the data being worked with, the following boxplot was generated depicting depict the distribution and spread of the variables with respect to each other.

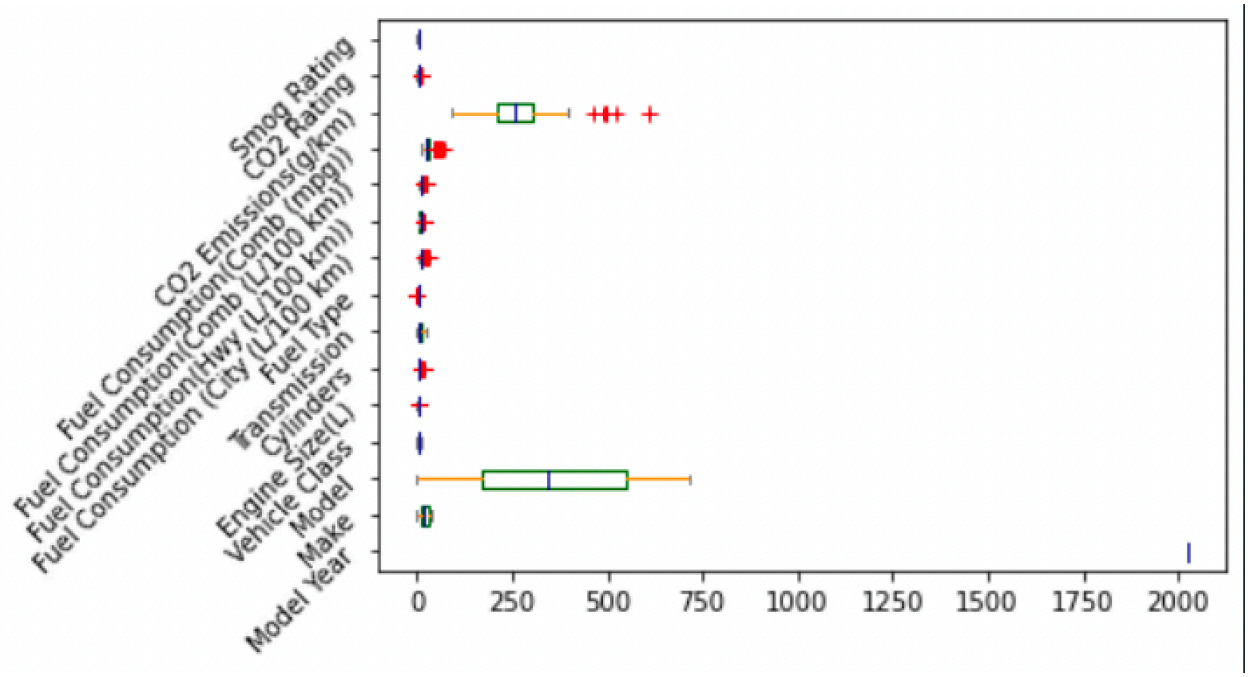

Figure 3: Boxplot of all Features and Variables

From the boxplot above, it can be determined that all variables have similar magnitude and spread apart from the vehicle model and CO_2 emissions. The vehicle model is not significant to the analyses as it will not be considered, however the CO_2 emissions will need to be re-scaled prior to model training and evaluation. Additionally, the presence of the outliers for the CO_2 emissions may weaken the strength of the model's prediction, possibly yielding high error during model evaluation.

Training and Validation Strategy

As described above, the combined fuel consumption [mpg] and carbon emissions [g/km] will be predicted using both Multiple Linear Regression (MLR) and a series of Artificial Neural Networks (ANNs). MLR will be used as it is significantly less complex than the ANN setups tested (less weights due to lack of hidden layers), and thus will provide a good metric to evaluate the performance of the ANNs to judge whether the additional complexity provides any tangible gain. Several ANN configurations will be tested in hopes that hopefully a suitable configuration can be determined such that it has a low propensity to get stuck in local minima during stochastic solving (stochastic gradient descent).

Multiple Linear Regression, MLR

Multiple linear regression will be the first technique utilized to predict both: combined fuel consumption [mpg] and CO_2 emissions [g/km]. MLR will be conducted twice for each predicted quantity, once utilizing all available features and the latter only having access to two features. The models will be generated and fitted using the `lienar.model` and `linearRegression` classes from ***`SciKit Learn`***. Prior to fitting, all MLR models will have the training and test data re-scaled using the Standard scaler as such:

```
scaler = StandardScaler().fit(features)
linear_modeld = LinearRegression()
linear_modeld.fit(scaler.transform(X5_train),y5_train)
```

Figure 4: Scaling and Fitting MLR Model using SciKit Learn

Subsequently, predictions are made for the MLR model using the following command. The model will then be evaluated using the mean-squared error (mse) and correlation coefficient (r^2) value to evaluate the prediction error.

```
y5_test_pred = linear_modeld.predict(scaler.transform(X5_test))
mse_all = mean_squared_error(y5_test, y5_test_pred)
r2_all = r2_score(y5_test, y5_test_pred)
```

Figure 5: Making Predictions with an MLR model using SciKit Learn

Finally, for the model only utilizing two features (the other MLR has access to all features), a 3-Dimensional plot will be made plotting the actual points against the 'prediction' plane generated by the MLR model

Artificial Neural Networks, ANNs

In addition to the simpler MLR models, ANNs will also be utilized to determine whether the added complexity (mostly through storing & accessing the numerous additional weights) provides any benefit, and whether the benefits outweigh the additional computational costs. ***`SciKit Learn`*** is also utilized to scale, build, fit and evaluate the ANNs; using the `neural_network` class and the `MLPRegressor()` function. The following five ANN setups will be tested in hopes of yielding an idea of which configurations produce higher preforming models.

Table 2: ANN Naming Scheme and Corresponding Configuration

ANN-#	Number of Layers	Nodes Per Layer	Activation Function
1	1	100	*Logistic*
2	1	20	*Logistic*
3	4	25	*Logistic*
4	1	100	*Tanh*
5	4	25	*Tanh*

Each of the ANN models will be built, fitted, and evaluated using the following procedure and methods from ***SciKit Learn.*** Just as the MLR models, the ANNs will be evaluated using the mean-squared error (mse) and correlation coefficient (r^2) value although it is not directly applicable to our ANN structure, it will provide a useful comparison metric between our methods. Below is an example of the implementation for the ANN-1 configuration model.

```
'''
ARTIFICIAL
NUERAL
NETWORKS
'''
print('\n ~ARTIFICIAL NUERAL NETWRORKS')
'''We are then going to use an ANN of 100 nodes & 1 hidden layer -- LOGISTIC'''
scaler = StandardScaler().fit(features)

ann_model1 = MLPRegressor((100),
                      activation='logistic',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model1.fit( scaler.transform(X4_train), y4_train)

mse_train_ann1 = mean_squared_error(y4_train, ann_model1.predict(scaler.transform(X4_train)))

Y_pred_ann1 = ann_model1.predict(scaler.transform(X4_test))
mse_test_ann1 = mean_squared_error(y4_test, Y_pred_ann1)

print('[ ANN1 : 1HL-100N {LOGISTIC}] Training error (MSE): ', mse_train_ann1)
print('[ ANN1 : 1HL-100N {LOGISTIC}] Testing error (MSE):  ', mse_test_ann1)
```

Figure 6: Building, Fitting, and Evaluating ANN-1

Results and Discussion

Having trained both MLR and ANN models to predict the combined fuel economy [mpg] and the CO_2 emissions [g/km], the following results were generated mostly consisting of mean-squared error and r^2 values to compare between models. The following are the results broken down by ML implementation.

Multiple Linear Regression, MLR :: Results

The python script containing all the models and test data was ran and generated the following results for display in the user console regarding the performance of the combined fuel consumption and CO_2 emissions predictions respectively. First the results for the fuel consumption will be analyzed.

```
~MULTIPLE LINEAR REGRESSION

[MLR-ALL Features], MSE on the test set considering ALL FEATURES is 28.266908773567383
[MLR-ALL Features], R2 on the test set considering ALL FEATURES is 0.5453701194472356

[MLR-VC & Cylinders], R2 value considering VC & Cylinders is 0.49965023899819916
```

Figure 7: Console Output Result for MLR Model Predictions for Combined Fuel Economy [mpg]

Analyzing the above results, its apparent that the model struggled to predict the combined fuel economy [mpg], achieving an r^2 value of 0.54537 which is quite low. However, using the knowledge gathered from the scatter matrix, the second MLR model was only provided the 'Vehicle Class' and 'Cylinders' as features and managed to achieve an r^2 value of 0.49965 which is quite close to the correlation coefficient using all the features. Thus, it can be concluded that although the model was not able to predict very the combined fuel economy very well, by utilizing the scatter matrix the main contributing components of the data were isolated and then subsequently used to train the second model. Additionally, the following 3-dimensional plot was made to visual the 'prediction' plane used by the MLR model, specifically how it passes through (or misses) the data points.

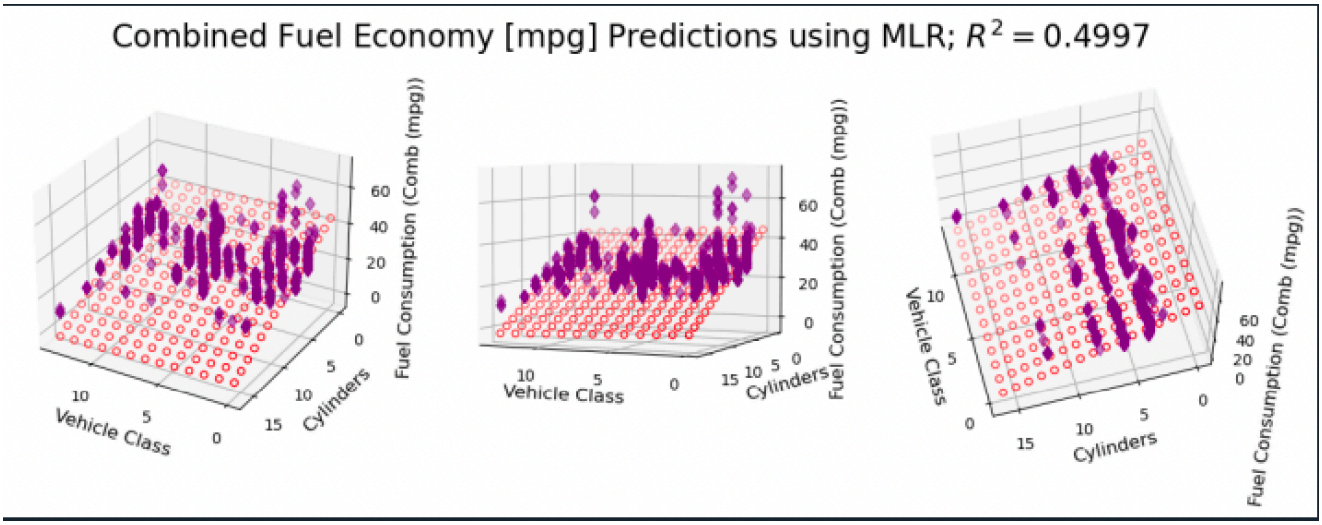

Figure 8: 3-dimensional Plot Showcasing Prediction 'Plane' (red-circles) and Actual Data Points (purple-diamonds) Using Limited Features

Observing the above 3-dimensional plot, the 'prediction' plane struggles to reach the outlier points, specifically those with low values for both 'Vehicle Class' and 'Cylinders'. Overall, this model can roughly predict the combined fuel consumption and excels with vehicles

encoded with larger values for their vehicle type with many cylinders; it struggles with the outlier points.

Similar processes were followed for predicting the CO_2 emissions [g/km], with the following results output to the console.

```
~MULTIPLE LINEAR REGRESSION

[MLR-ALL Features], MSE on the test set considering ALL FEATURES is 1124.716098999065
[MLR-ALL Features], R2 on the test set considering ALL FEATURES is 0.719048734502937

[MLR-VC & Cylinders], R2 value considering VC & Cylinders is 0.7059305135918496
```

Figure 9: Console Output Result for MLR Model Predictions for CO_2 Emissions [g/km]

Similar trends were observed to those generated during the fuel economy predictions. However, this time our mean-squared error was much larger whilst still maintaining a relatively high r^2 value of 0.71905. These metrics are possibly indicative of outliers affecting our model, however as it is desired in this case to preserve the integrity of the data set and not omit any points (to provide a more holistic view of the car market), they will still be included for these analyses. Once again it was observed that the r^2 value for the model using only two features was very close to the model trained with all features, this time generating an r^2 value of 0.70593. The following 3-dimensional plot was also made to better visualize the performance of the model with limited feature access.

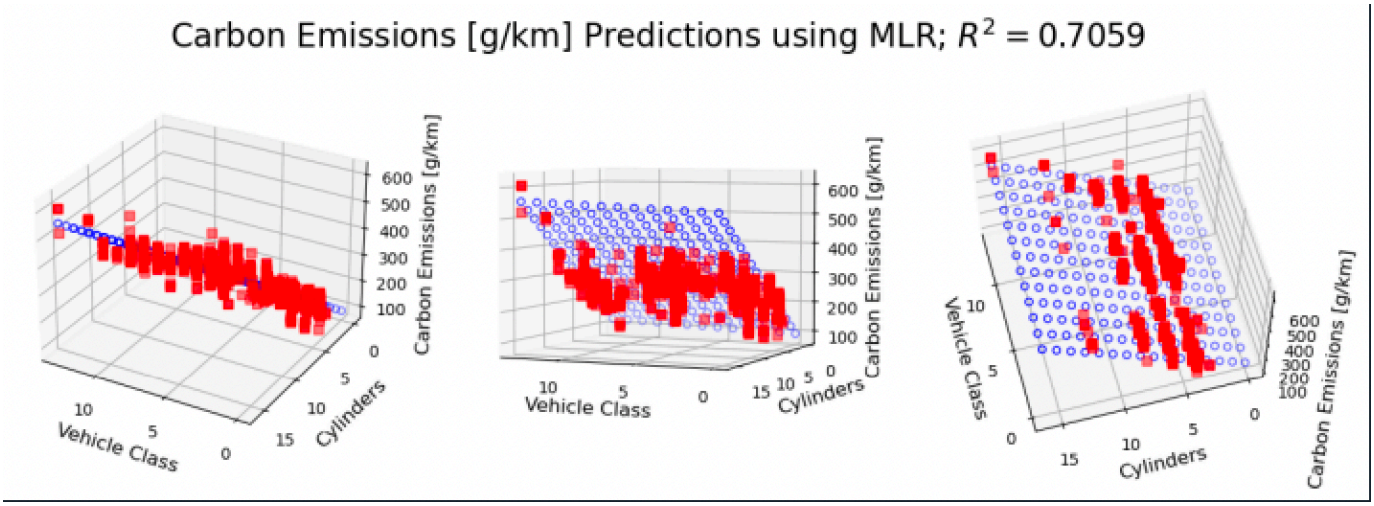

Figure 10: 3-dimensional Plot Showcasing Prediction 'Plane' (blue-circles) and Actual Data Points (red-squares) Using Limited Features

From the 3-dimensional pot, it can be determined that the MLR model trained was much more effective at predicting Carbon Emissions than Combined Fuel Consumption, as demonstrated by the 'prediction' plane (blue circles) passing through a greater proportion of the data points (red squares).

We generalize our results predicting both combined fuel consumption and carbon dioxide emissions in the subsequent table.

Table 3: Tabulated Results for Both MLRs Across Both Predictions

Model	Combined Fuel Consumption [mpg]		CO_2 emissions [g/km]	
	mse	r^2	*mse*	r^2
MLR-all features	28.267	0.545	1124.715	0.719
MLR-2 features	--	0.499	--	0.706

The above table is reflective of the trends observed visually in the 3-dimensional plots.

Artificial Neural Networks, ANNs:: Results

Using the procedure outlined above, the five ANN models were built, fitted, and evaluated by the python script and generated the following console outputs respectively.

```
Fuel Econonomy (Comb [mpg]) Predictions

 ~MULTIPLE LINEAR REGRESSION

[MLR-ALL Features], MSE on the test set considering ALL FEATURES is 28.266908773567383
[MLR-ALL Features], R2 on the test set considering ALL FEATURES is 0.5453701194472356

[MLR-VC & Cylinders], R2 value considering VC & Cylinders is 0.49965023899819916

 ~ARTIFICIAL NUERAL NETWRORKS
[ ANN1 : 1HL-100N {LOGISTIC}] Training error (MSE):  2.012793596132705
[ ANN1 : 1HL-100N {LOGISTIC}] Testing error (MSE):   10.479334333849309
[ ANN2 : 1HL-20N {LOGISTIC}] Training error (MSE):  8.77801695516009
[ ANN2 : 1HL-20N {LOGISTIC}]Testing error (MSE):   13.1477279477452
[ ANN3 : 4HL-25N/E {LOGISTIC}] Training error (MSE):  14.823571758010985
[ ANN3 : 4HL-25N/E {LOGISTIC}]Testing error (MSE):   19.016074994633172
[ ANN4 : 1HL-100N {TANH}] Training error (MSE):  3.906222493155867
[ ANN4 : 1HL-100N {TANH}] Testing error (MSE):   10.479334333849309
[ ANN5 : 4HL-25N/E {TANH}] Training error (MSE):  2.1665307266973746
[ ANN5 : 4HL-25N/E {TANH}]Testing error (MSE):   11.864091977645641
```

Figure 11: Console Output for Combined Fuel Economy [mpg] Predictions

```
CO2 Emissions [g/km]  Predictions

 ~MULTIPLE LINEAR REGRESSION

[MLR-ALL Features], MSE on the test set considering ALL FEATURES is 1124.716098999065
[MLR-ALL Features], R2 on the test set considering ALL FEATURES is 0.719048734502937

[MLR-VC & Cylinders], R2 value considering VC & Cylinders is 0.7059305135918496

 ~ARTIFICIAL NUERAL NETWRORKS
[ ANN1 : 1HL-100N {LOGISTIC}] Training error (MSE):  203.11039838119146
[ ANN1 : 1HL-100N {LOGISTIC}] Testing error (MSE):   442.6402543397792
[ ANN2 : 1HL-20N {LOGISTIC}] Training error (MSE):  443.9165823383097
[ ANN2 : 1HL-20N {LOGISTIC}]Testing error (MSE):   669.5205974750216
[ ANN3 : 4HL-25N/E {LOGISTIC}] Training error (MSE):  695.9373438345655
[ ANN3 : 4HL-25N/E {LOGISTIC}]Testing error (MSE):   795.6599408307221
[ ANN4 : 1HL-100N {TANH}] Training error (MSE):  226.1690239605717
[ ANN4 : 1HL-100N {TANH}] Testing error (MSE):   582.6112319516511
[ ANN5 : 4HL-25N/E {TANH}] Training error (MSE):  4185.047373708121
[ ANN5 : 4HL-25N/E {TANH}]Testing error (MSE):   4003.244437798747
```

Figure 12: Console Output for CO_2 Emissions [g/km] Predictions

For easier interpretation and subsequent analyses, the above results are also tabulated below.

Table 4: Training and Test MSE Results Across Five ANN Configurations Tested

Model	Combined Fuel Consumption [mpg]		CO_2 emissions [g/km]	
	Training mse	*Test mse*	*Training mse*	*Training mse*
ANN-1	2.012	10.479	203.110	442.640
ANN-2	8.778	13.148	443.916	669.521
ANN-3	14.824	19.016	695.937	795.650
ANN-4	3.906	10.479	226.169	582.611
ANN-5	2.167	11.864	4185.047	4003.244

Thus, it can be concluded that the strongest preforming models were ANN-1 (1HL-100N {Logistic}) and ANNs-4&5 (1HL-100N {tanh} & 4HL-25N {tanh} respectively). Therefore, the 'tanh' activation functions appear to be better at aiding the solving of the gradient descent than the 'logistic' activation function. Furthermore, it was observed that between the three ANNs using the 'logistic' activation function the optimal number of nodes appears to be between twenty and one hundred nodes as the one hundred node, one-layer model outperformed the one-layer model of only twenty nodes.

Across all five ANN models, it was observed that they were much better at predicting the combined fuel economy rather than the CO_2 emissions. Across the board, both the training and test *mse* values were found to be roughly a magnitude of one to two smaller for the fuel consumption predictions compared to the carbon dioxide emissions.

Final Model Selection

The decision to choose our final model choice is assisted by our mean-squared error (*mse*) metric. Our results are aligned, being the ANN models performed better than the MLR models across the board for *mse* when predicting combined fuel economy [mpg]. Additionally, the same is noted for the CO_2 emissions; the MLR models having *mse* values greater than all but one (*ANN-5*) of the ANN models.

Thus, the ANN models will be chosen as the superior ML method used to answer the initial problem. Within the five ANN models tested, the recommendation would be *ANN-1*, a simple ANN consisting of only one hidden layer and one hundred nodes. Across both predictions, it was found to have the lowest training and test *mse*.

The choice to recommend a simple ANN architecture is further affirmed by both the scatter matrix (*Figure:*) and the 3-dimensional plots (*Figure:*). Across all three plots, non-linear correlations were observed in multiple dimensions. Specifically in the 3-dimensional plots, the prediction 'plane' was demonstrated struggling to reach the outlier points above and below the plane. Thus, we can conclude that this problem has relationships that may be too complex to be captured by an MLR model as implemented, and thus the final recommendation is *ANN-1*.

Appendix

Python Code

```
"""
Created on Sat Apr 30 11:57:18 2022

@author: pranavveerubhotla
"""

import numpy as np
import pandas as pd
import csv
from kmeans import kmeans
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.neural_network import MLPRegressor

df2 = pd.read_csv("MY2022 Fuel Consumption Ratings.csv")
print('ORIGINAL; DATAFRAME TYPES...\n',df2.dtypes, '\n')

my_list = []
my_values = []
for col in ['Make', 'Model', 'Vehicle Class', 'Transmission', 'Fuel Type']:
    df2[col] = df2[col].astype('category')
    my_list.append(df2[col].cat.codes)
    my_values.append(df2[col].tolist())
print('After Encoding Categogorical Data; DATAFRAME TYPES...\n',df2.dtypes, '\n')
cat_columns = df2.select_dtypes(['category']).columns
df2[cat_columns] = df2[cat_columns].apply(lambda x: x.cat.codes)


#Organize the dataframe (seperate features & labels)
feature_names = ['Model Year', 'Make', 'Model',
                 'Vehicle Class', 'Engine Size(L)', 'Cylinders', 'Transmission',
                 'Fuel Type']
features = df2[feature_names]

results = []
for i in my_values:
    results.append(list({value:"" for value in i}))

fnames = ['make_enc.csv', 'model_enc.csv', 'vc_enc.csv','trans_enc.csv', 'fuel_enc.csv']

print('We ~ENCODE~ using the following scheme... \n')
for i in range(0, len(results)):
    print('\n~', cat_columns[i])
    print('++++++++++++++++++++')
    file = open(fnames[i], 'w')
    writer = csv.writer(file)
    k=0
    for j in results[i]:
       # print('\t', j, '\t\t\t\t\t', k)
       # PRINT & also write to appropriate csv
       print(f"{'Entity: ' + j:<70} Encoded #: {k}")
       writer.writerow([j, k])
       k+=1
    file.close()
```

```
X = features
y1 = df2['Fuel Consumption (City (L/100 km)']
y2 = df2['Fuel Consumption(Hwy (L/100 km))']
y3 = df2['Fuel Consumption(Comb (L/100 km))']
y4 = df2['Fuel Consumption(Comb (mpg))']
y5 = df2['CO2 Emissions(g/km)']
y6 = df2['CO2 Rating']
y7 = df2['Smog Rating']

all_handles = ['Model Year', 'Make', 'Model',
                'Vehicle Class', 'Engine Size(L)', 'Cylinders', 'Transmission',
                'Fuel Type', 'Fuel Consumption (City (L/100 km)', 'Fuel Consumption(Hwy (L/100 km))',
                'Fuel Consumption(Comb (L/100 km))', 'Fuel Consumption(Comb (mpg))', 'CO2 Emissions(g/km)',
                'CO2 Rating', 'Smog Rating']
#Split the data
X1_train, X1_test, y1_train, y1_test = train_test_split(X,
                                                        y1,
                                                        train_size=0.8,
                                                        random_state=42)
X2_train, X2_test, y2_train, y2_test = train_test_split(X,
                                                        y2,
                                                        train_size=0.8,
                                                        random_state=42)
X3_train, X3_test, y3_train, y3_test = train_test_split(X,
                                                        y3,
                                                        train_size=0.8,
                                                        random_state=42)
X4_train, X4_test, y4_train, y4_test = train_test_split(X,
                                                        y4,
                                                        train_size=0.8,
                                                        random_state=42)
X5_train, X5_test, y5_train, y5_test = train_test_split(X,
                                                        y5,
                                                        train_size=0.8,
                                                        random_state=42)
X6_train, X6_test, y6_train, y6_test = train_test_split(X,
                                                        y6,
                                                        train_size=0.8,
                                                        random_state=42)
X7_train, X7_test, y7_train, y7_test = train_test_split(X,
                                                        y7,
                                                        train_size=0.8,
                                                        random_state=42)


'''Data ANALYSIS'''
'''Transform Categorical Data'''
types_of_vehicles = []
for i in df2['Vehicle Class']:
    types_of_vehicles.append(i)
unique_types_of_vehicles = set(types_of_vehicles)

transmissions_of_vehicles = []
for i in df2['Transmission']:
    transmissions_of_vehicles.append(i)
unique_transmissions_of_vehicles = set(transmissions_of_vehicles)

cylinders_of_vehicles = []
for i in df2['Cylinders']:
    cylinders_of_vehicles.append(i)
unique_cylinders_of_vehicles = set(cylinders_of_vehicles)

fuel_types_of_vehicles = []
for i in df2['Fuel Type']:
    fuel_types_of_vehicles.append(i)
unique_fuel_types_of_vehicles = set(fuel_types_of_vehicles)

#Plot quantities
grid_cleansed = df2.hist(xlabelsize = 5, grid = False, xrot = 45)
plt.savefig('grid.jpeg')
#grid_cleansed_stacked = df2.plot.hist(stacked = True, bins = 1000)
```

```
color = {
    "boxes": "DarkGreen",
    "whiskers": "DarkOrange",
    "medians": "DarkBlue",
    "caps": "Gray",
}

bp = df2.plot.box(color = color, sym = 'r+', vert = False, rot=45)
plt.savefig('bp.jpeg')
correlation_df = df2.corr()
correlation_df.plot(title='Correlation Plot of ALL Features & Response Vars')
plt.savefig('corr.jpeg')

pscatter_matrix = scatter_matrix(df2, alpha = 0.5, figsize = (7,9))

axes_sm = pd.plotting.scatter_matrix(df2, alpha=0.2)
for ax in axes_sm.flatten():
    ax.xaxis.label.set_rotation(90)
    ax.yaxis.label.set_rotation(0)
    ax.yaxis.label.set_ha('right')

plt.tight_layout()
plt.gcf().subplots_adjust(wspace=0, hspace=0)
plt.show()
plt.savefig('sm.jpeg')

print('_____________________________________________________________________________________________')
print('_____________________________________________________________________________________________')
print('                          Fuel Econonomy (Comb [mpg]) Predictions                            ')
print('_____________________________________________________________________________________________')
print('\n ~MULTIPLE LINEAR REGRESSION')
'''
MULTIPLE
LINEAR
REGRESSION
'''

'''Preform Multiple Linear Regression for Fuel Economy in GENERAL'''
'''USING ALL FEATURES'''
linear_modeld = LinearRegression()
linear_modeld.fit(X4_train,y4_train)
y4_test_pred = linear_modeld.predict(X4_test)
mse_all = mean_squared_error(y4_test, y4_test_pred)
r2_all = r2_score(y4_test, y4_test_pred)
print('\n[MLR-ALL Features], MSE on the test set considering ALL FEATURES is', mse_all)
print('[MLR-ALL Features], R2 on the test set considering ALL FEATURES is', r2_all)



'''Multiple Linear Rregression -- more specific -- vehicle type and transmission against y1'''
'''USING VEHICLE CLASS & CYLINDERS'''
X = df2[['Vehicle Class', 'Cylinders']].values.reshape(-1,2)
Y = y4

x = X[:, 0]
y = X[:, 1]
z = Y

x_pred = np.linspace(0, 13, 13)   # range of Vehicle Class indexes [formerly CATEGORICAL]
y_pred = np.linspace(0, 16, 16)  # range of Cylinders
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T

mlr = linear_model.LinearRegression()
model = mlr.fit(X, Y)
predicted = model.predict(model_viz)

r2 = model.score(X, Y)

print('\n[MLR-VC & Cylinders], R2 value considering VC & Cylinders is', r2)

plt.style.use('default')
```

```
fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

for ax in axes:
    ax.plot(x, y, z, color='purple', zorder=15, linestyle='none', marker='d', alpha=0.5)
    ax.scatter(xx_pred.flatten(), yy_pred.flatten(), predicted, facecolor=(0,0,0,0), s=20, edgecolor='red')
    ax.set_xlabel('Vehicle Class', fontsize=12)
    ax.set_ylabel('Cylinders', fontsize=12)
    ax.set_zlabel('Fuel Consumption (Comb (mpg))', fontsize=12)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')



ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('Combined Fuel Economy [mpg] Predictions using MLR; '+'$R^2 = %.4f$' % r2, fontsize=20)

fig.tight_layout()
plt.savefig('Combined Fuel Economy.jpeg')

'''
ARTIFICIAL
NUERAL
NETWORKS
'''
print('\n ~ARTIFICIAL NUERAL NETWRORKS')
'''We are then going to use an ANN of 100 nodes & 1 hidden layer -- LOGISTIC'''
scaler = StandardScaler().fit(features)

ann_model1 = MLPRegressor((100),
                      activation='logistic',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model1.fit( scaler.transform(X4_train), y4_train)

mse_train_ann1 = mean_squared_error(y4_train, ann_model1.predict(scaler.transform(X4_train)))

Y_pred_ann1 = ann_model1.predict(scaler.transform(X4_test))
mse_test_ann1 = mean_squared_error(y4_test, Y_pred_ann1)

print('[ ANN1 : 1HL-100N {LOGISTIC}] Training error (MSE): ', mse_train_ann1)
print('[ ANN1 : 1HL-100N {LOGISTIC}] Testing error (MSE):  ', mse_test_ann1)

'''We are then going to use an ANN of 20 nodes & 1 hidden layer'''
scaler = StandardScaler().fit(features)

ann_model2 = MLPRegressor((20),
                      activation='logistic',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model2.fit( scaler.transform(X4_train), y4_train)
```

```
mse_train_ann2 = mean_squared_error(y4_train, ann_model2.predict(scaler.transform(X4_train)))

Y_pred_ann2 = ann_model2.predict(scaler.transform(X4_test))
mse_test_ann2 = mean_squared_error(y4_test, Y_pred_ann2)

print('[ ANN2 : 1HL-20N {LOGISTIC}] Training error (MSE): ', mse_train_ann2)
print('[ ANN2 : 1HL-20N {LOGISTIC}]Testing error (MSE):  ', mse_test_ann2)

'''We are then going to use an ANN of 100 nodes & 4 hidden layer'''
scaler = StandardScaler().fit(features)

ann_model3 = MLPRegressor((25, 25, 25, 25),
                      activation='logistic',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model3.fit( scaler.transform(X4_train), y4_train)

mse_train_ann3 = mean_squared_error(y4_train, ann_model3.predict(scaler.transform(X4_train)))

Y_pred_ann3 = ann_model3.predict(scaler.transform(X4_test))
mse_test_ann3 = mean_squared_error(y4_test, Y_pred_ann3)

print('[ ANN3 : 4HL-25N/E {LOGISTIC}] Training error (MSE): ', mse_train_ann3)
print('[ ANN3 : 4HL-25N/E {LOGISTIC}]Testing error (MSE):  ', mse_test_ann3)

'''We are then going to use an ANN of 100 nodes & 1 hidden layer -- TANH'''
scaler = StandardScaler().fit(features)

ann_model4 = MLPRegressor((100),
                      activation='tanh',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model4.fit( scaler.transform(X4_train), y4_train)

mse_train_ann4 = mean_squared_error(y4_train, ann_model4.predict(scaler.transform(X4_train)))

Y_pred_ann4 = ann_model4.predict(scaler.transform(X4_test))
mse_test_ann4 = mean_squared_error(y4_test, Y_pred_ann1)

print('[ ANN4 : 1HL-100N {TANH}] Training error (MSE): ', mse_train_ann4)
print('[ ANN4 : 1HL-100N {TANH}] Testing error (MSE):  ', mse_test_ann4)

'''We are then going to use an ANN of 100 nodes & 4 hidden layer -- TANH'''
scaler = StandardScaler().fit(features)

ann_model5 = MLPRegressor((25, 25, 25, 25),
                      activation='tanh',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model5.fit( scaler.transform(X4_train), y4_train)

mse_train_ann5 = mean_squared_error(y4_train, ann_model5.predict(scaler.transform(X4_train)))

Y_pred_ann5 = ann_model5.predict(scaler.transform(X4_test))
mse_test_ann5 = mean_squared_error(y4_test, Y_pred_ann5)
```

```
print('[ ANN5 : 4HL-25N/E {TANH}] Training error (MSE): ', mse_train_ann5)
print('[ ANN5 : 4HL-25N/E {TANH}]Testing error (MSE):  ', mse_test_ann5)


print('_____________________________________________________________________________________________')
print('_____________________________________________________________________________________________')
print('                              CO2 Emissions [g/km]  Predictions                              ')
print('_____________________________________________________________________________________________')
print('\n ~MULTIPLE LINEAR REGRESSION')
'''
MULTIPLE
LINEAR
REGRESSION
'''

'''Preform Multiple Linear Regression for Fuel Economy in GENERAL'''
'''USING ALL FEATURES'''
scaler = StandardScaler().fit(features)
linear_modeld = LinearRegression()
linear_modeld.fit(scaler.transform(X5_train),y5_train)
y5_test_pred = linear_modeld.predict(scaler.transform(X5_test))
mse_all = mean_squared_error(y5_test, y5_test_pred)
r2_all = r2_score(y5_test, y5_test_pred)
print('\n[MLR-ALL Features], MSE on the test set considering ALL FEATURES is', mse_all)
print('[MLR-ALL Features], R2 on the test set considering ALL FEATURES is', r2_all)



'''Multiple Linear Rregression -- more specific -- vehicle type and transmission against y1'''
'''USING VEHICLE CLASS & CYLINDERS'''
X = df2[['Vehicle Class', 'Cylinders']].values.reshape(-1,2)
Y = y5

x = X[:, 0]
y = X[:, 1]
z = Y

x_pred = np.linspace(0, 13, 13)   # range of Vehicle Class indexes [formerly CATEGORICAL]
y_pred = np.linspace(0, 16, 16)  # range of Cylinders
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T

mlr = linear_model.LinearRegression()
model = mlr.fit(X, Y)
predicted = model.predict(model_viz)

r2 = model.score(X, Y)

print('\n[MLR-VC & Cylinders], R2 value considering VC & Cylinders is', r2)

plt.style.use('default')

fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

for ax in axes:
    ax.plot(x, y, z, color='red', zorder=15, linestyle='none', marker='s', alpha=0.5)
    ax.scatter(xx_pred.flatten(), yy_pred.flatten(), predicted, facecolor=(0,0,0,0), s=20, edgecolor='blue')
    ax.set_xlabel('Vehicle Class', fontsize=12)
    ax.set_ylabel('Cylinders', fontsize=12)
    ax.set_zlabel('Carbon Emissions [g/km]', fontsize=12)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')
```

```
ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('Carbon Emissions [g/km] Predictions using MLR; '+'$R^2 = %.4f$' % r2, fontsize=20)

fig.tight_layout()
plt.savefig('Carbon Emissions.jpeg')

'''
ARTIFICIAL
NUERAL
NETWORKS
'''
print('\n ~ARTIFICIAL NUERAL NETWRORKS')
'''We are then going to use an ANN of 100 nodes & 1 hidden layer'''
scaler = StandardScaler().fit(features)

ann_model1 = MLPRegressor((100),
                      activation='logistic',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model1.fit( scaler.transform(X5_train), y5_train)

mse_train_ann1 = mean_squared_error(y5_train, ann_model1.predict(scaler.transform(X5_train)))

Y_pred_ann1 = ann_model1.predict(scaler.transform(X5_test))
mse_test_ann1 = mean_squared_error(y5_test, Y_pred_ann1)

print('[ ANN1 : 1HL-100N {LOGISTIC}] Training error (MSE): ', mse_train_ann1)
print('[ ANN1 : 1HL-100N {LOGISTIC}] Testing error (MSE):  ', mse_test_ann1)

'''We are then going to use an ANN of 20 nodes & 1 hidden layer'''
scaler = StandardScaler().fit(features)

ann_model2 = MLPRegressor((20),
                      activation='logistic',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model2.fit( scaler.transform(X5_train), y5_train)

mse_train_ann2 = mean_squared_error(y5_train, ann_model2.predict(scaler.transform(X5_train)))

Y_pred_ann2 = ann_model2.predict(scaler.transform(X5_test))
mse_test_ann2 = mean_squared_error(y5_test, Y_pred_ann2)

print('[ ANN2 : 1HL-20N {LOGISTIC}] Training error (MSE): ', mse_train_ann2)
print('[ ANN2 : 1HL-20N {LOGISTIC}]Testing error (MSE):  ', mse_test_ann2)

'''We are then going to use an ANN of 100 nodes & 4 hidden layer'''
scaler = StandardScaler().fit(features)

ann_model3 = MLPRegressor((25, 25, 25, 25),
                      activation='logistic',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)
```

```
ann_model3.fit( scaler.transform(X5_train), y5_train)

mse_train_ann3 = mean_squared_error(y5_train, ann_model3.predict(scaler.transform(X5_train)))

Y_pred_ann3 = ann_model3.predict(scaler.transform(X4_test))
mse_test_ann3 = mean_squared_error(y5_test, Y_pred_ann3)

print('[ ANN3 : 4HL-25N/E {LOGISTIC}] Training error (MSE): ', mse_train_ann3)
print('[ ANN3 : 4HL-25N/E {LOGISTIC}]Testing error (MSE):  ', mse_test_ann3)

'''We are then going to use an ANN of 100 nodes & 1 hidden layer -- TANH'''
scaler = StandardScaler().fit(features)

ann_model4 = MLPRegressor((100),
                      activation='tanh',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model4.fit( scaler.transform(X5_train), y5_train)

mse_train_ann4 = mean_squared_error(y5_train, ann_model4.predict(scaler.transform(X5_train)))

Y_pred_ann4 = ann_model4.predict(scaler.transform(X5_test))
mse_test_ann4 = mean_squared_error(y5_test, Y_pred_ann4)

print('[ ANN4 : 1HL-100N {TANH}] Training error (MSE): ', mse_train_ann4)
print('[ ANN4 : 1HL-100N {TANH}] Testing error (MSE):  ', mse_test_ann4)

'''We are then going to use an ANN of 100 nodes & 4 hidden layer -- TANH'''
scaler = StandardScaler().fit(features)

ann_model5 = MLPRegressor((25, 25, 25, 25),
                      activation='tanh',
                      solver='adam',
                      alpha=0.0001,
                      learning_rate='constant',
                      learning_rate_init=0.001,
                      max_iter=50000,
                      random_state=42,
                      verbose=False)

ann_model5.fit( scaler.transform(X5_train), y5_train)

mse_train_ann5 = mean_squared_error(y5_train, ann_model5.predict(scaler.transform(X5_train)))

Y_pred_ann5 = ann_model5.predict(scaler.transform(X5_test))
mse_test_ann5 = mean_squared_error(y5_test, Y_pred_ann5)

print('[ ANN5 : 4HL-25N/E {TANH}] Training error (MSE): ', mse_train_ann5)
print('[ ANN5 : 4HL-25N/E {TANH}]Testing error (MSE):  ', mse_test_ann5)
```

Encoding Scheme

For the results of the Encoding Scheme, please see the included '.csv' files within the .zip file.