Predictive Analytics Final Project

Kartik Singh

166048199

ZAA

Table of Contents

Problem Statement

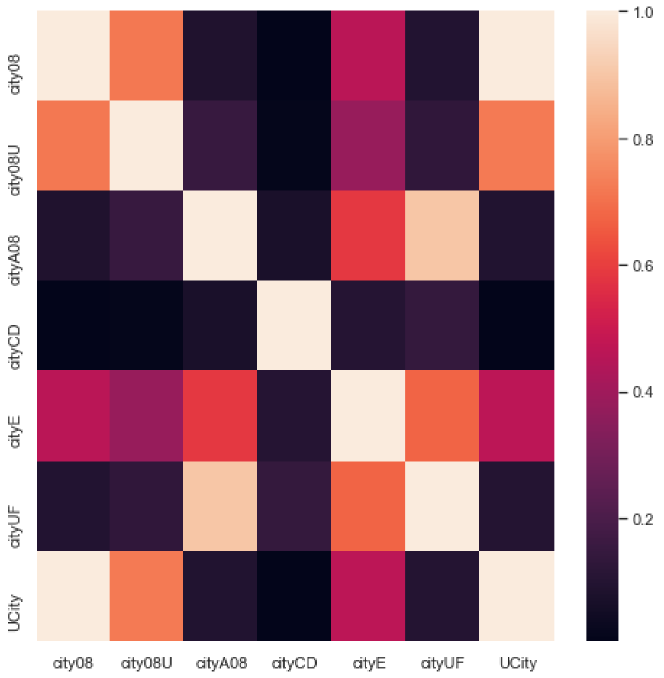
We're given fuel data of Vehicles from 1984 to 2022. Aim of the project is to predict UCity variable using other independent variables. Additionally, I have implemented correlation analysis, Exploratory Data Analysis, Univariate analysis and Bivariate analysis, Predictive Analysis Model Implementation, Generalization using k fold cross validation

Correlation Analysis

Analysing Correlation between Cityxxx and UCity

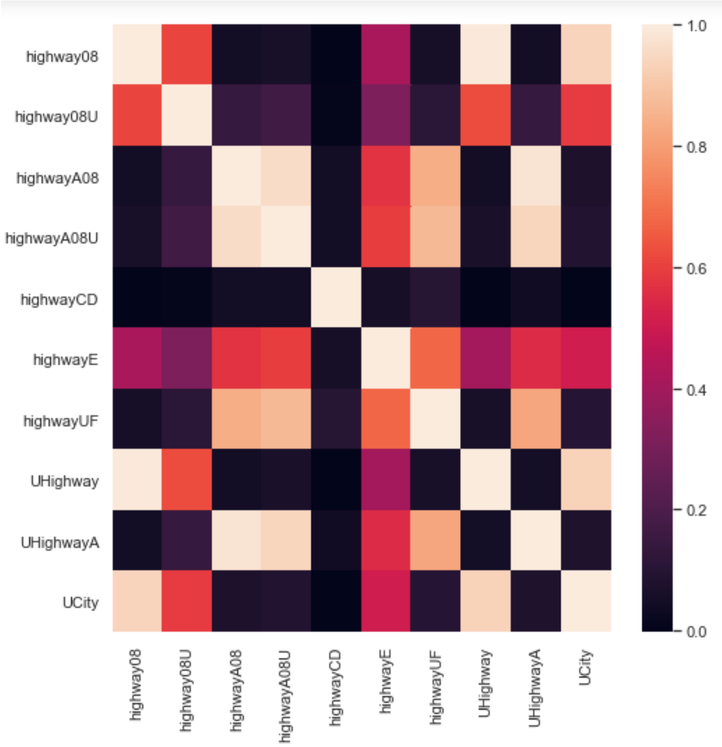
```
datanew=data1[['city08','city08U','cityA08', 'cityCD', 'cityE','cityUF', 'UCity']]
datanew.head()
```

	city08	city08U	cityA08	cityCD	cityE	cityUF	UCity
0	19	0.0	0	0.0	0.0	0.0	23.3333
1	9	0.0	0	0.0	0.0	0.0	11.0000
2	23	0.0	0	0.0	0.0	0.0	29.0000
3	10	0.0	0	0.0	0.0	0.0	12.2222
4	17	0.0	0	0.0	0.0	0.0	21.0000



There is a strong correlation between City08 and UCity (almost 1.0). Additionally, there is a high correlation between CityUF and City08.

Analysing Correlation between Highwayxxx

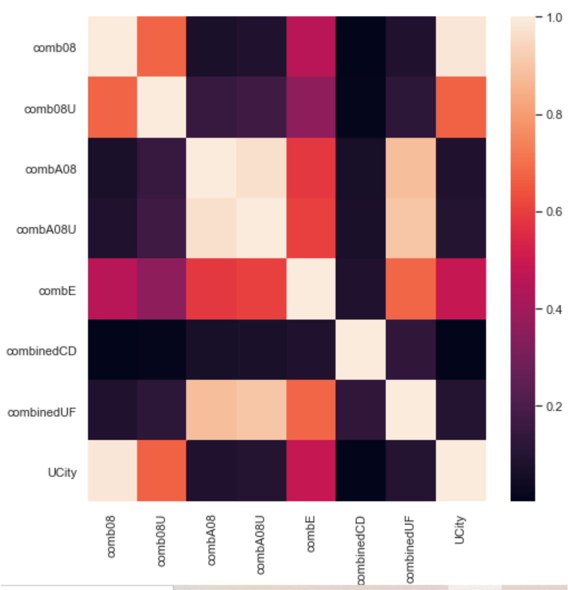


There is a high correlation between Highway08 and UCity

HighwayA08 and HighwayA08U appears to be similar and points towards multicollinearity

Additionally, there is a significant correlation between UHighway and UCity

Analysing Correlation between combxxx



Combo08 is a strong indicator of UCity. Combo08 and Combo08U are identical and has multicollinearity

Exploratory Data Analysis- Univariate & Bivariate Analysis

The data consist of 83 attributes and 43921 records

```
data.shape
```

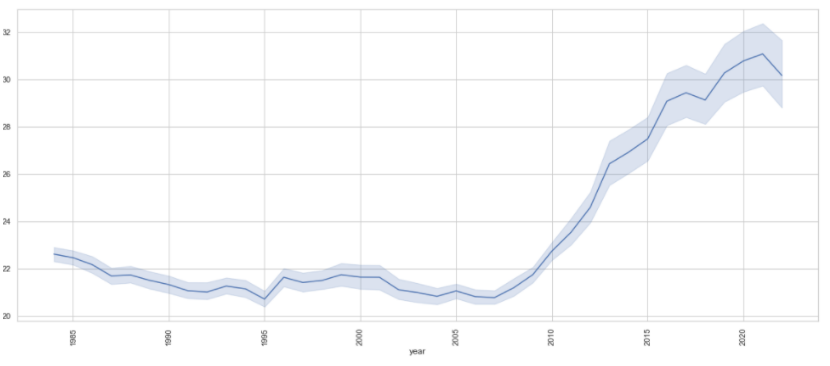
```
(43921, 83)
```

The attributes are mentioned in the image below

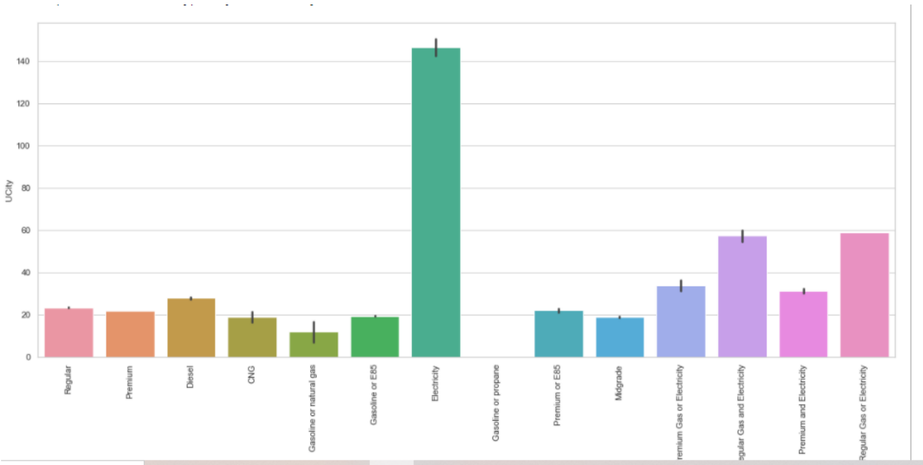
```
print(data.columns)
```

```
Index(['barrels08', 'barrelsA08', 'charge120', 'charge240', 'city08',
       'city08U', 'cityA08', 'cityA08U', 'cityCD', 'cityE', 'cityUF', 'co2',
       'co2A', 'co2TailpipeAGpm', 'co2TailpipeGpm', 'comb08', 'comb08U',
       'combA08', 'combA08U', 'combE', 'combinedCD', 'combinedUF', 'cylinders',
       'displ', 'drive', 'engId', 'eng_dscr', 'feScore', 'fuelCost08',
       'fuelCostA08', 'fuelType', 'fuelType1', 'ghgScore', 'ghgScoreA',
       'highway08', 'highway08U', 'highwayA08', 'highwayA08U', 'highwayCD',
       'highwayE', 'highwayUF', 'hlv', 'hpv', 'id', 'lv2', 'lv4', 'make',
       'model', 'mpgData', 'phevBlended', 'pv2', 'pv4', 'range', 'rangeCity',
       'rangeCityA', 'rangeHwy', 'rangeHwyA', 'trany', 'UCity', 'UCityA',
       'UHighway', 'UHighwayA', 'VClass', 'year', 'youSaveSpend', 'guzzler',
       'trans_dscr', 'tCharger', 'sCharger', 'atvType', 'fuelType2', 'rangeA',
       'evMotor', 'mfrCode', 'c240Dscr', 'charge240b', 'c240bDscr',
       'createdOn', 'modifiedOn', 'startStop', 'phevCity', 'phevHwy',
       'phevComb'],
      dtype='object')
```

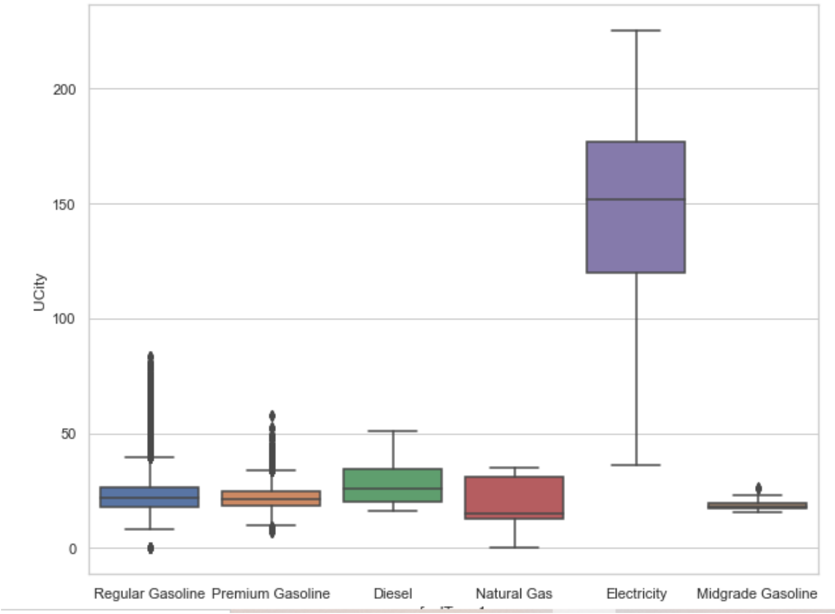
Analysing UCity average over the years points to the fact that after 2010 there has been a significant improvement in quality of vehicles as far as city average is concerned.



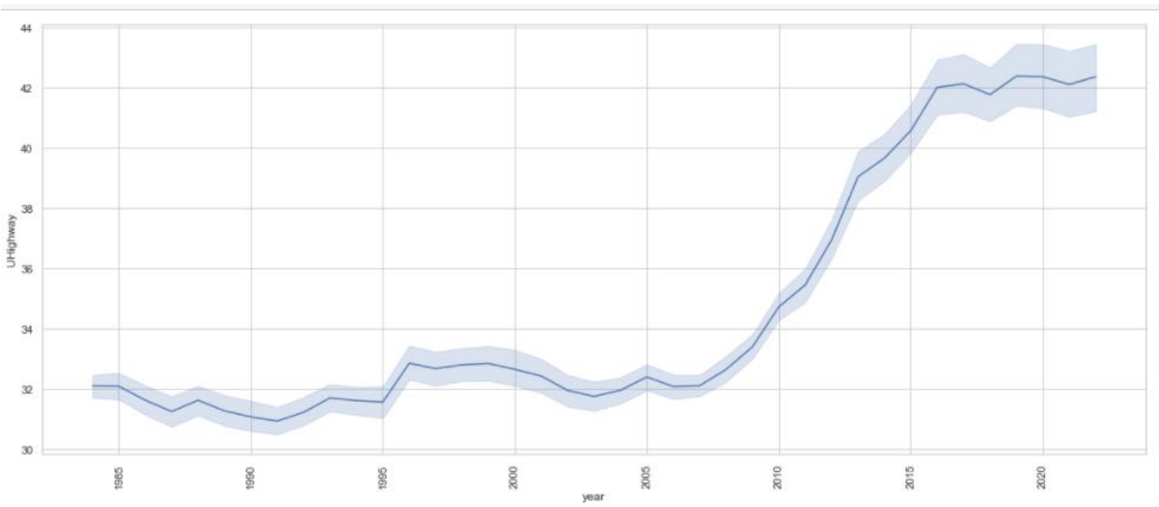
It could be the due to the fact of increased use of Electric vehicles which provide better average than most natural gas-based vehicles



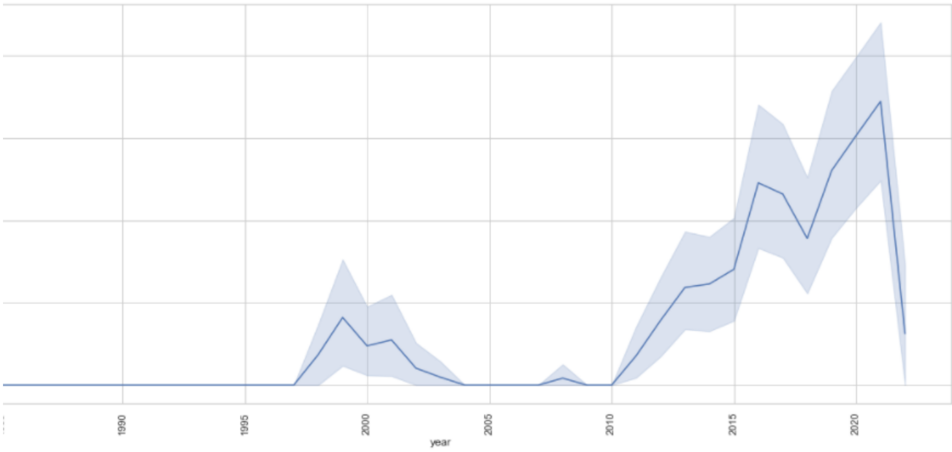

```
plt.figure(figsize=(10,8))
ax1 = sns.boxplot(x="fuelType1", y="UCity", data=data1)
```



A similar trend could be seen for Mileage on Highway:

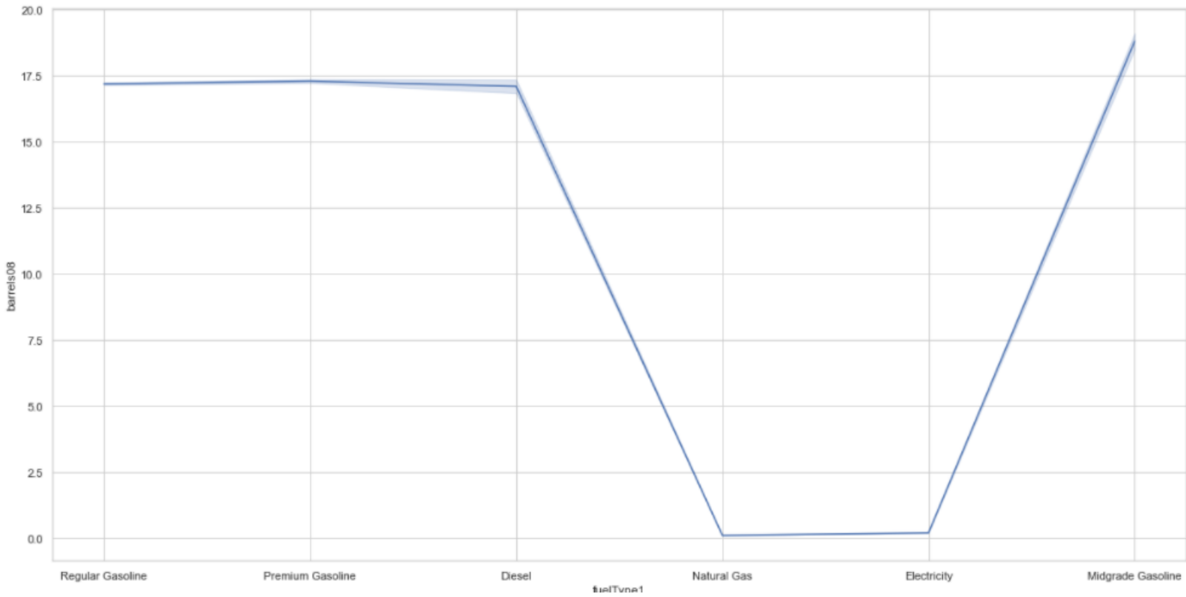


It could be the due to the fact of higher sales of Electric vehicles:

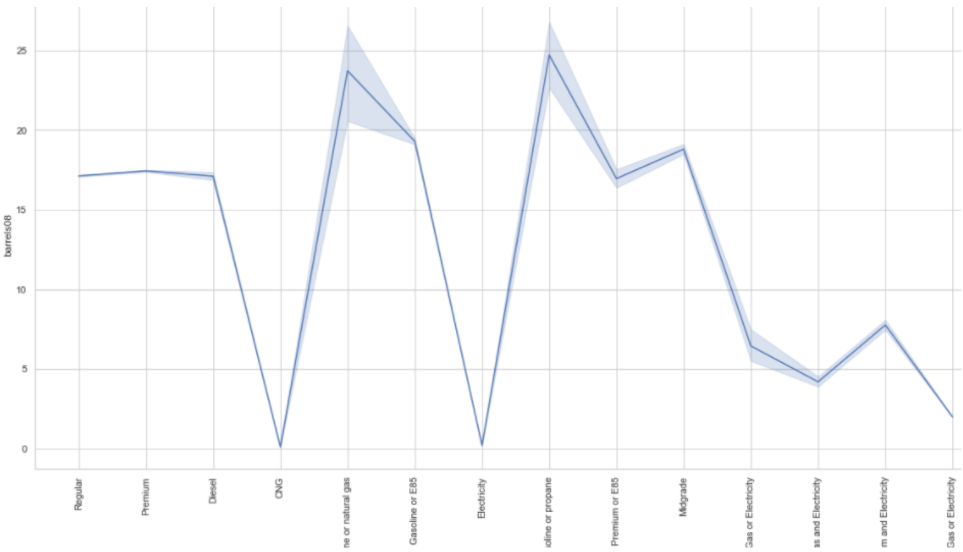


Mildgrade Gasoline vehicles uses highest barrels of fuels in comparison to other fuel operated vehicles:

```
plt.figure(figsize=(20,10))
ax1 = sns.lineplot(x="fuelType1", y="barrels08", data=data1)
```



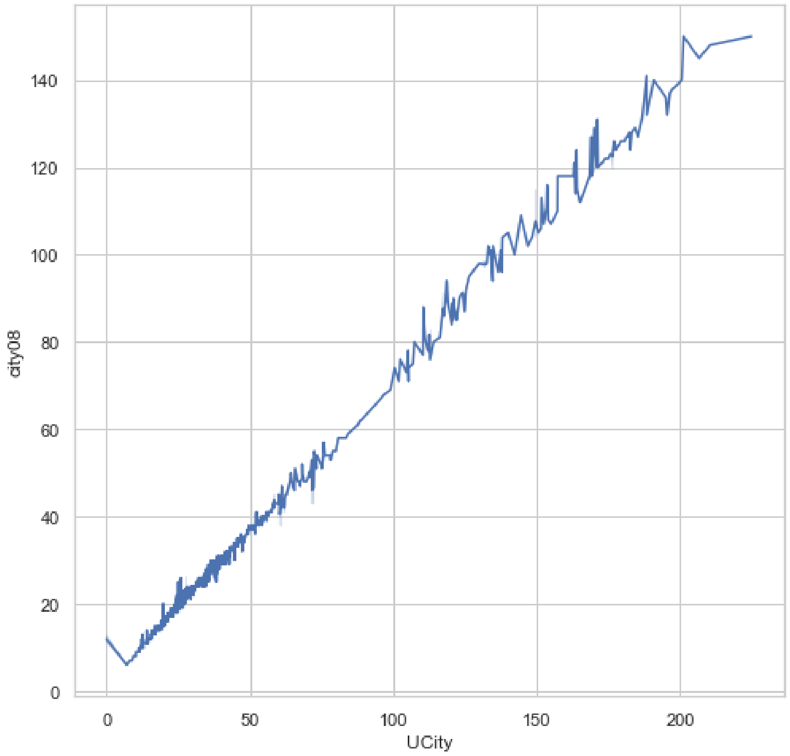
Upon further analysing it can be observed that Gasoline/ Natural Gas and Gasoline/Propane powered vehicles consumes highest barrels of fuel while all type of electric vehicles consume the least.



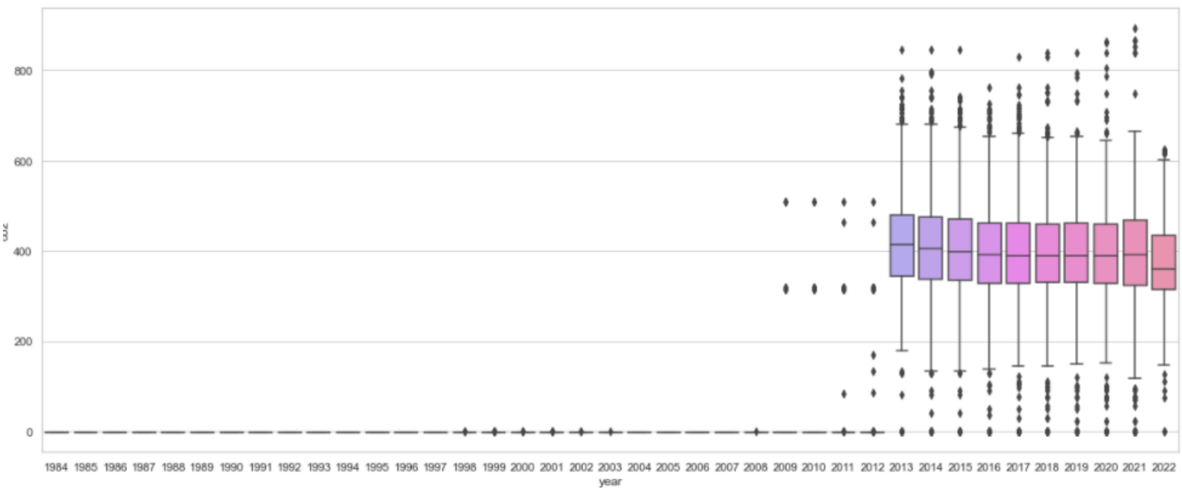
UCity and City08 has a strong positive correlation since City08 is unadjusted MPG for FuelType1 and hence including City08 is equivalent to including UCity in the model

```
sns.lineplot(x='UCity', y='city08', data=data1)
```

```
: <AxesSubplot:xlabel='UCity', ylabel='city08'>
```

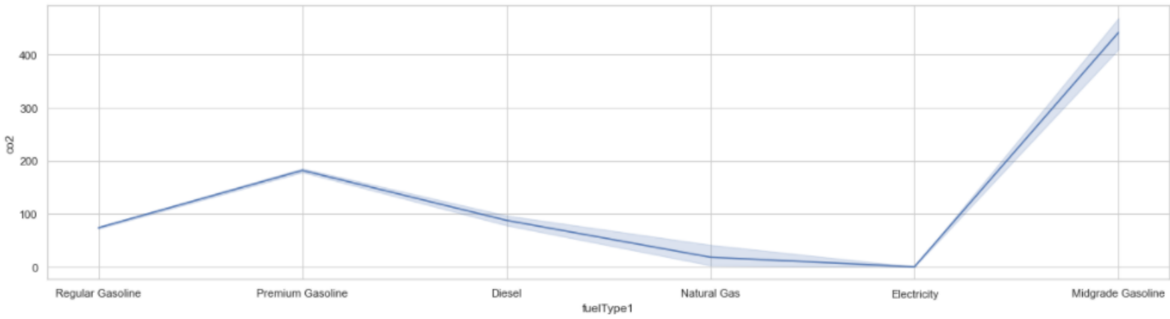


The dataset doesn't contain data for co2 before 2010 and hence is marked as -1. With the limited amount of data available it can be concluded **that co2 emission levels** for **vehicles have reduced over the years.**



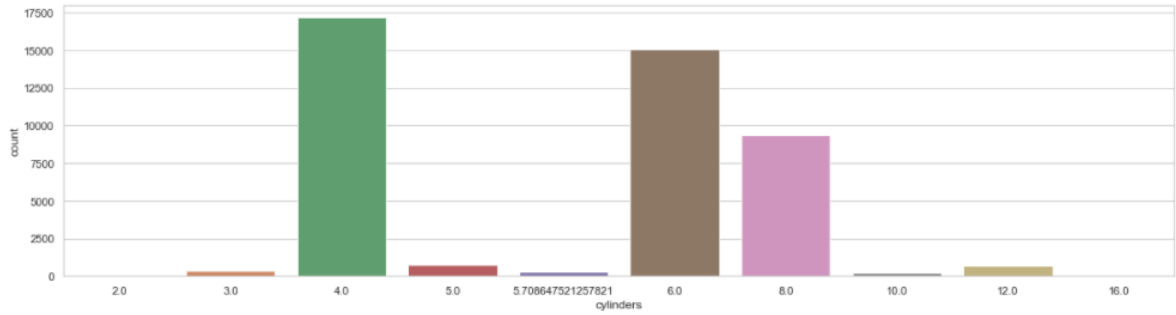
Mildgrade Gasoline Vehicles creates **highest co2** emissions as evident by the graph below

```
plt.figure(figsize=(20,5))
ax=sns.lineplot(x='fuelType1', y='co2', data=data1)
```



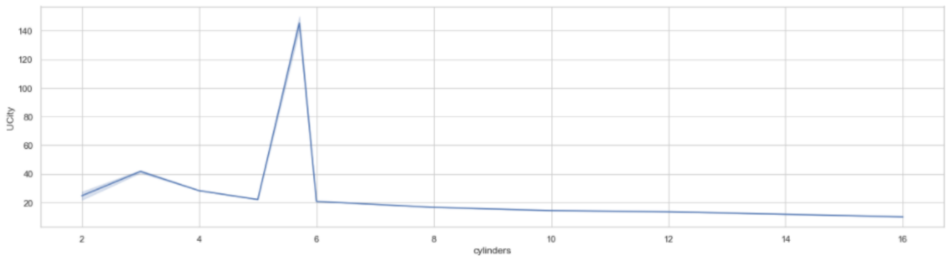
Most Vehicles uses **4 cylinders:**

```
plt.figure(figsize=(20,5))
ax=sns.countplot(x='cylinders', data=data1)
```



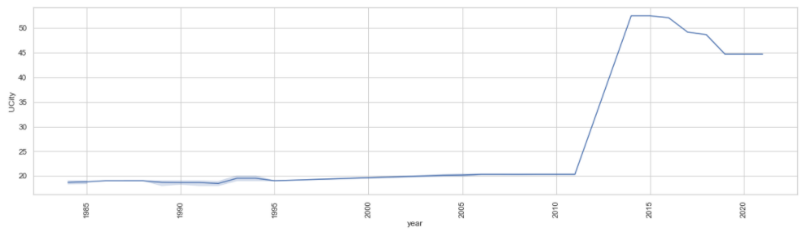
Vehicles using 4-6 Cylinders give highest average for UCity

```
plt.figure(figsize=(20,5))
ax=sns.lineplot(y="UCity", x="cylinders", data=data1)
```



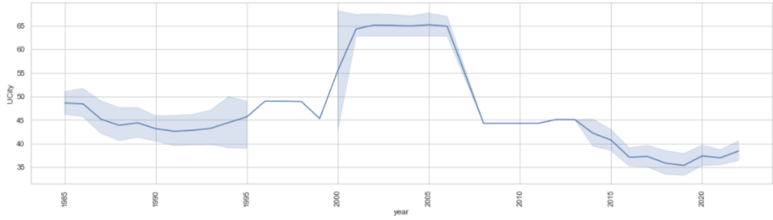
UCity average for vehicles using 2 cylinders have increased after 2010

```
plt.figure(figsize=(20,5))
plt.xticks(rotation=90)
ax=sns.lineplot(y="UCity", x="year", data=data1.loc[data1['cylinders']==2])
```



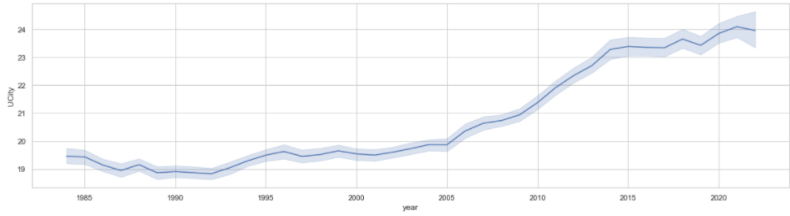
Vehicles using 3 cylinders had the highest average 2000-2007 and reduced significantly after that. Why? Maybe due to limited data available

```
plt.figure(figsize=(20,5))
plt.xticks(rotation=90)
ax=sns.lineplot(y="UCity", x="year", data=data1.loc[data1['cylinders']==3])
```



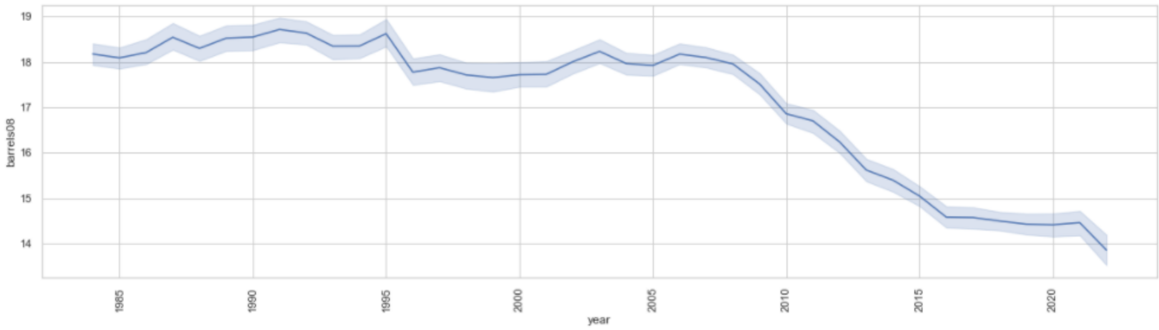
Vehicles using 6 cylinders has increased UCity average over the years since 2000

```
plt.figure(figsize=(20,5))
ax=sns.lineplot(y="UCity", x="year", data=data1.loc[data1['cylinders']==6])
```



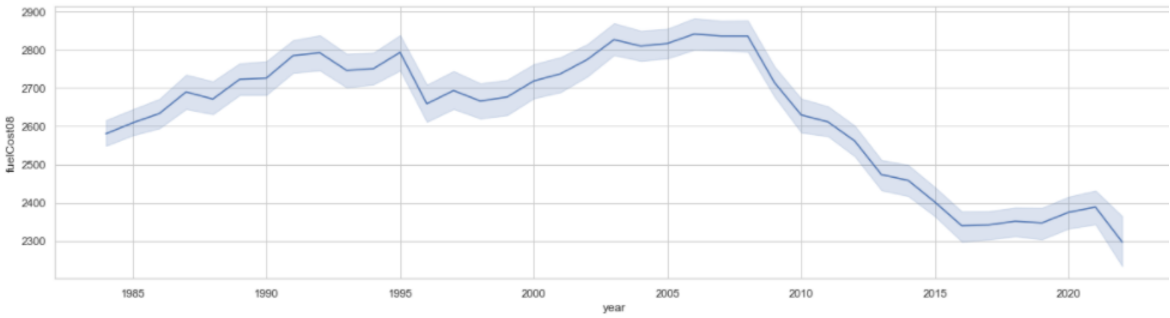
Usage of Barrels have reduced overtime

```
plt.figure(figsize=(20,5))
plt.xticks(rotation=90)
ax=sns.lineplot(x="year", y="barrels08", data=data1)
```



And thus with the reduce in barrels the fuelcost08 has reduced over time

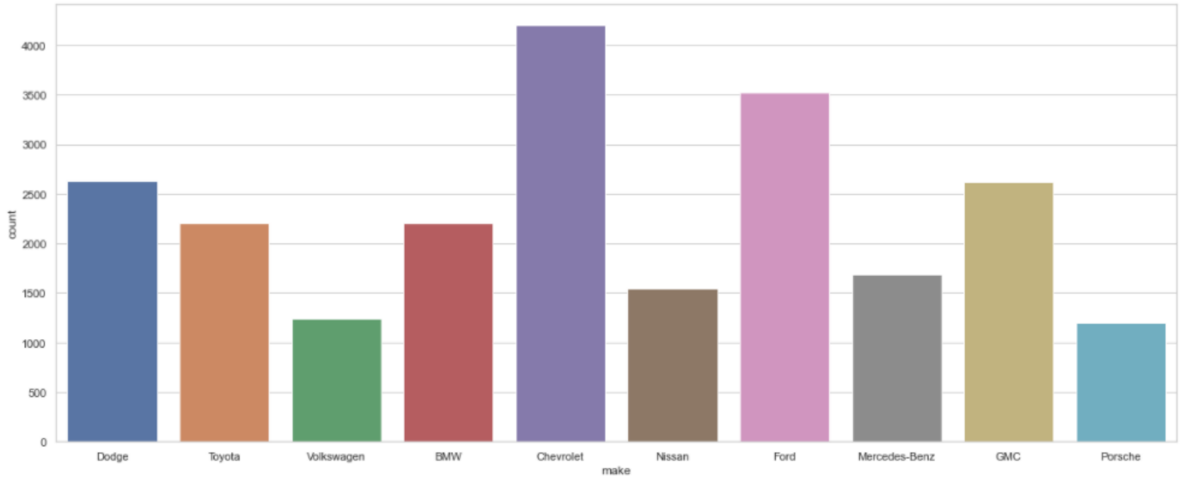
```
plt.figure(figsize=(20,5))
ax=sns.lineplot(y="fuelCost08", x="year", data=data1)
```



Chevrolet, Ford and GMC were highest sellers over time.

```
plt.figure(figsize=(20,8))
sns.countplot(x='make', data=data3)
```

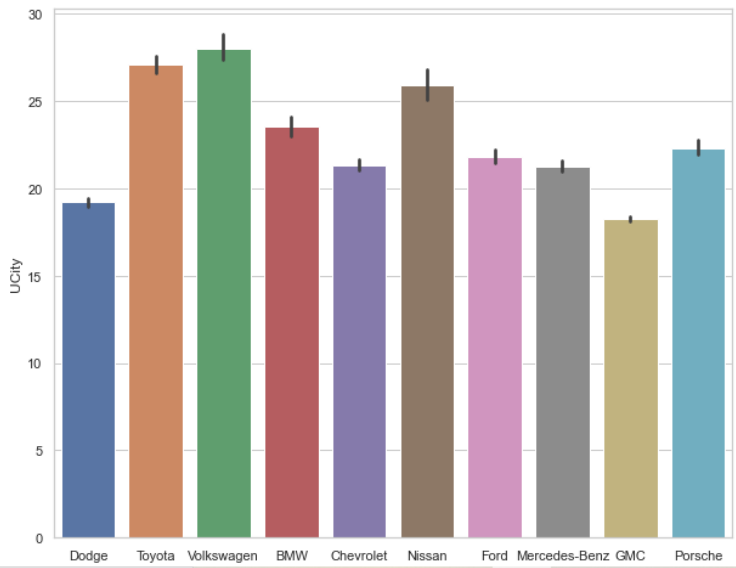
```
<AxesSubplot:xlabel='make', ylabel='count'>
```



Volkswagen, Toyota and Nissan has highest UCity average.

```
plt.figure(figsize=(10,8))
sns.barplot(x='make', y='UCity', data=data3)
```

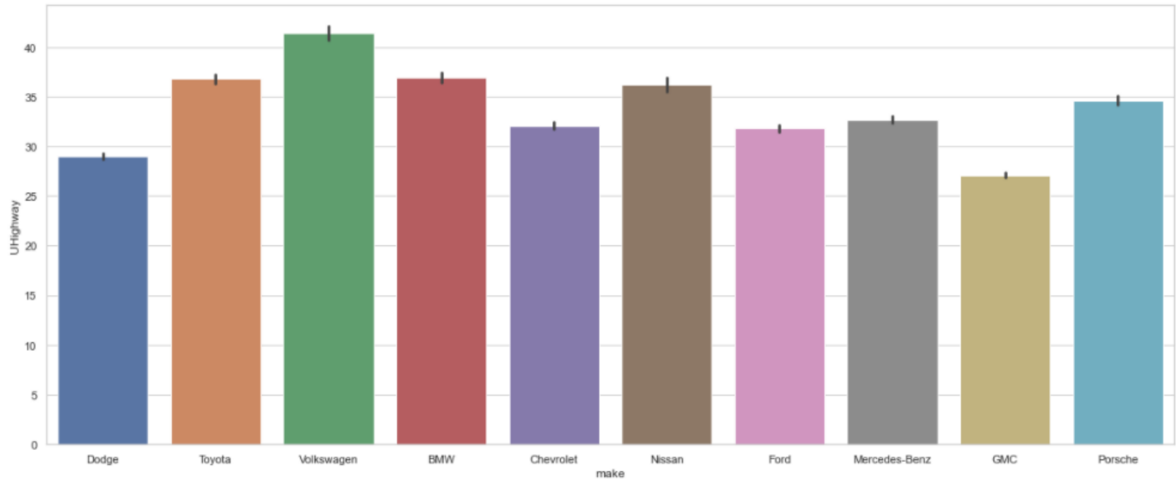
```
<AxesSubplot:xlabel='make', ylabel='UCity'>
```



This changes when we analyse the average on highway for these manufacturers. All manufactures have somewhat similar average on Highway (Volkswagen being an exception)

```
plt.figure(figsize=(20,8))
sns.barplot(x='make', y='UHighway', data=data3)
```

<AxesSubplot:xlabel='make', ylabel='UHighway'>

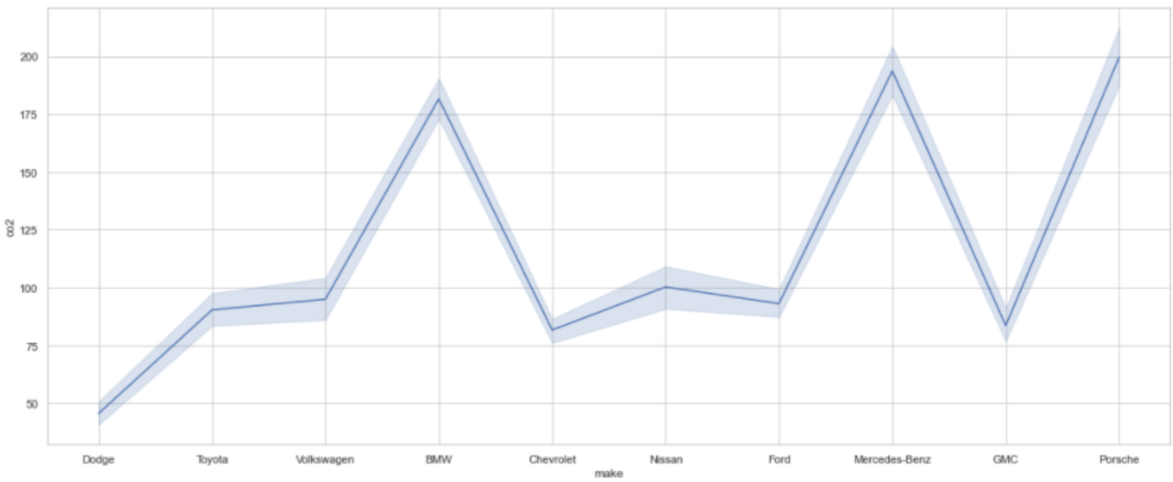


Volkswagen, Toyota and Nissan also produce lowest co2 emissions as compared to high end vehicles such as BMW, Mercedes and Porsche

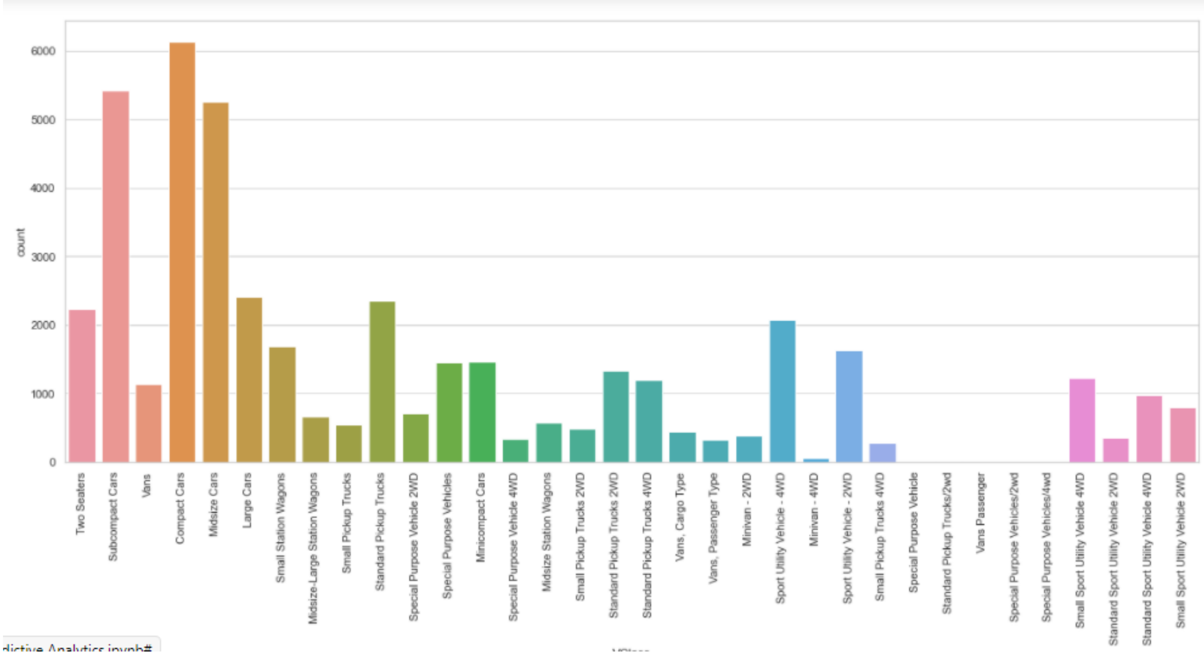
0

```
plt.figure(figsize=(20,8))
sns.lineplot(x='make', y='co2', data=data3)
```

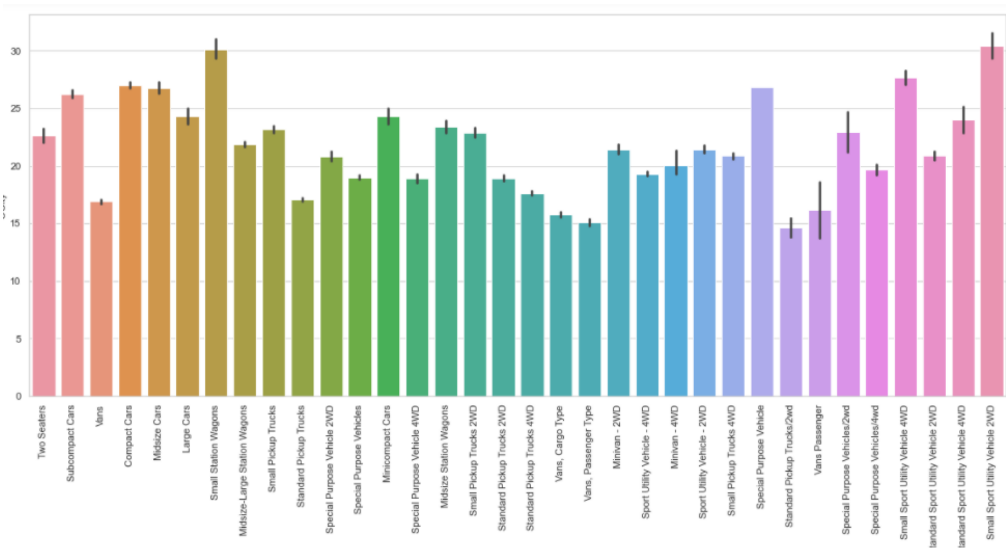
<AxesSubplot:xlabel='make', ylabel='co2'>



Compact cars are most sold type vehicles according to the dataset



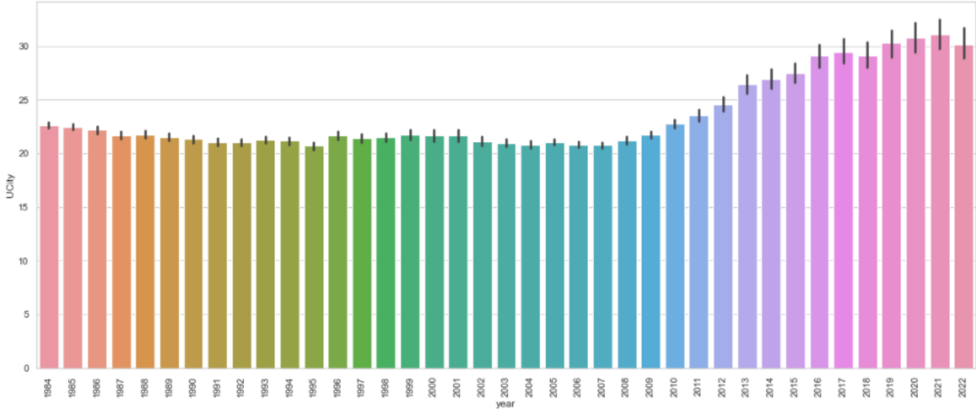
While small sports utility vehicles (2WD) produce the best city average closely followed by small station wagons



It can be further said that since 2010, UCity average has increased a lot.

```
plt.figure(figsize=(20,8))
plt.xticks(rotation=90)
sns.barplot(x='year', y='UCity', data=data1)
```

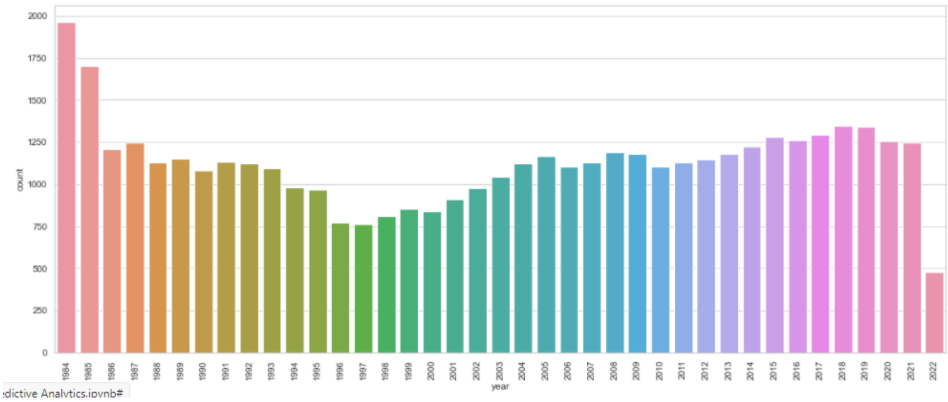
```
<AxesSubplot:xlabel='year', ylabel='UCity'>
```



While the most data is from 1984 and limited data from the 2000s

```
plt.figure(figsize=(20,8))
plt.xticks(rotation=90)
sns.countplot(x='year', data=data1)
```

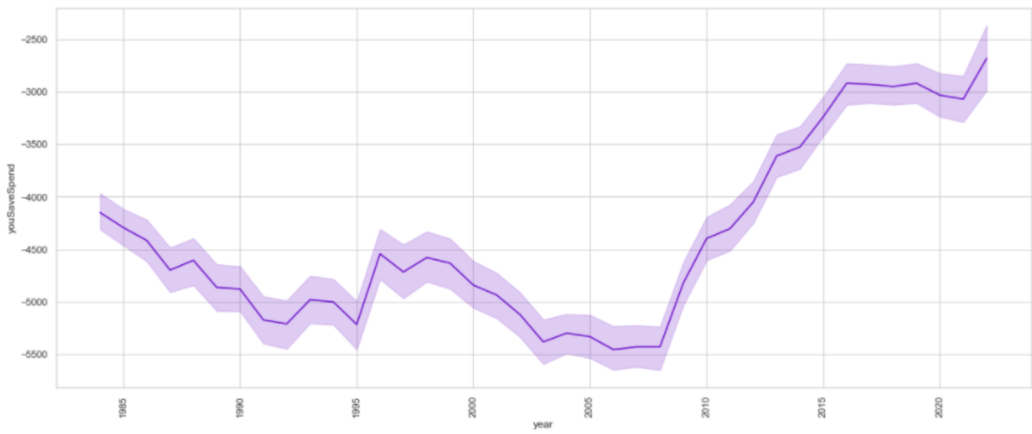
```
<AxesSubplot:xlabel='year', ylabel='count'>
```



It can be further concluded that cars are becoming more economical to drive over the years and save/spend ratio is improving every year

```
plt.figure(figsize=(20,8))
plt.xticks(rotation=90)
sns.lineplot(x='year', y='youSaveSpend', data=data1)
```

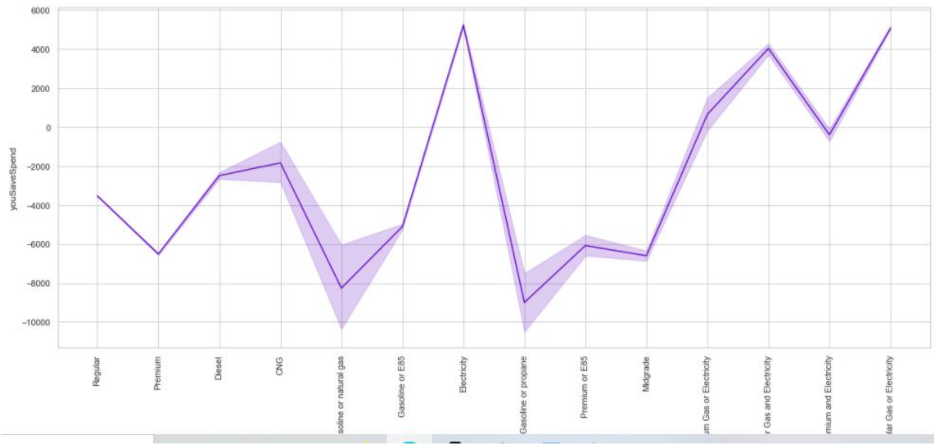
```
<AxesSubplot:xlabel='year', ylabel='youSaveSpend'>
```



We can further conclude that Electric vehicles are most economic to drive

```
plt.figure(figsize=(20,8))
plt.xticks(rotation=90)
sns.lineplot(x='fuelType', y='youSaveSpend', data=data1)
```

```
<AxesSubplot:xlabel='fuelType', ylabel='youSaveSpend'>
```



Outlier Analysis

Descriptive Statistics:

Descriptive statistics tells us that many columns has huge missing values (either 0 or -1) such as co2, co2a, charge240, barrelsA08, Co2TailpipeGpm, Co2TailpipeAGpm, feScore, ghgScore, ghgScoreA

	barrels08	barrelsA08	charge240	city08	city08U	cityUF	co2	co2A	co2TailpipeAGpm	co2TailpipeGpm
count	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000
mean	17.100731	0.222177	0.069337	18.711345	7.107325	0.002770	107.017987	5.846952	17.033195	461.315589
std	4.688838	1.142489	0.755676	8.873171	12.818082	0.038718	187.162456	57.051850	92.063818	125.486229
min	0.060000	0.000000	0.000000	6.000000	0.000000	0.000000	-1.000000	-1.000000	0.000000	0.000000
25%	14.330870	0.000000	0.000000	15.000000	0.000000	0.000000	-1.000000	-1.000000	0.000000	386.391304
50%	16.480500	0.000000	0.000000	17.000000	0.000000	0.000000	-1.000000	-1.000000	0.000000	444.350000
75%	19.388824	0.000000	0.000000	21.000000	16.039500	0.000000	270.000000	-1.000000	0.000000	522.764706
max	47.087143	18.311667	15.300000	150.000000	150.195800	0.927000	893.000000	713.000000	713.000000	1269.571429

comb08	cylinders	displ	feScore	fuelCost08	ghgScore	ghgScoreA	highway08	range	rangeHwyA	UCity
43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000
20.950297	5.708648	3.281768	0.641447	2624.832085	0.642563	-0.921974	24.823296	1.232053	0.142058	23.694636
8.506240	1.760981	1.352962	2.871225	737.596732	2.876897	0.652262	8.363189	17.114142	2.387078	12.619366
7.000000	2.000000	0.000000	-1.000000	450.000000	-1.000000	-1.000000	9.000000	0.000000	0.000000	0.000000
17.000000	4.000000	2.200000	-1.000000	2150.000000	-1.000000	-1.000000	20.000000	0.000000	0.000000	18.421200
20.000000	6.000000	3.000000	-1.000000	2600.000000	-1.000000	-1.000000	24.000000	0.000000	0.000000	21.700000
23.000000	6.000000	4.200000	3.000000	3050.000000	3.000000	-1.000000	28.000000	0.000000	0.000000	26.073700
142.000000	16.000000	8.400000	10.000000	8250.000000	10.000000	8.000000	133.000000	405.000000	114.760000	224.800000

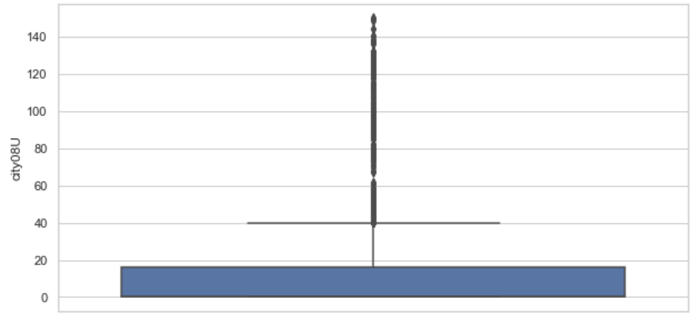
UCityA	UHighway	UHighwayA	year	youSaveSpend	phevCity
3921.000000	43921.000000	43921.000000	43921.000000	43921.000000	43921.000000
1.061427	34.841342	1.296095	2002.756927	-4360.527993	0.245236
8.744933	12.119023	8.853861	11.764501	3702.185122	3.618523
0.000000	0.000000	0.000000	1984.000000	-32500.000000	0.000000
0.000000	28.000000	0.000000	1992.000000	-6500.000000	0.000000
0.000000	33.400000	0.000000	2004.000000	-4250.000000	0.000000
0.000000	39.493800	0.000000	2013.000000	-2000.000000	0.000000
207.262200	187.100000	173.143600	2022.000000	6500.000000	97.000000

City08U:

City08u has values between 0-20 while 75% values are under 40 while there are a few outliers but EV Vehicles have higher average and thus these outliers are justified.

```
plt.figure(figsize=(10,5))
sns.boxplot(y='city08U', data=datafinal)
```

```
<AxesSubplot:ylabel='city08U'>
```

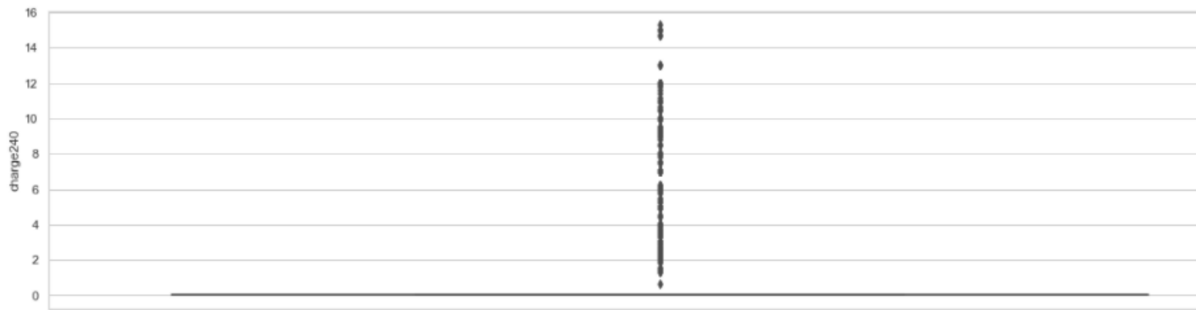


Charge240:

While Charge240 is defined for electric vehicles, some electric vehicles have high charge240 values

```
plt.figure(figsize=(20,5))
sns.boxplot(y='charge240', data=datafinal)
#no box shows very less values are above 0 as less vehicles are EV
```

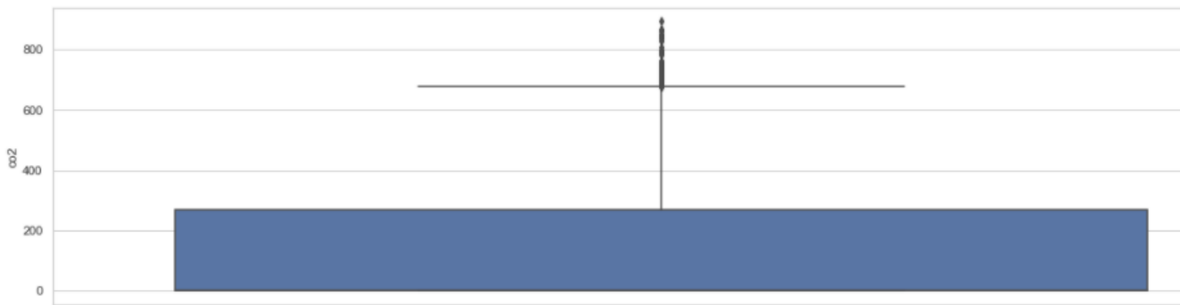
```
<AxesSubplot:ylabel='charge240'>
```



Co2 Emissions:

```
plt.figure(figsize=(20,5))
sns.boxplot(y='co2', data=datafinal)
#presence of outliers but can be milgrade petroleum vehicles
```

```
<AxesSubplot:ylabel='co2'>
```



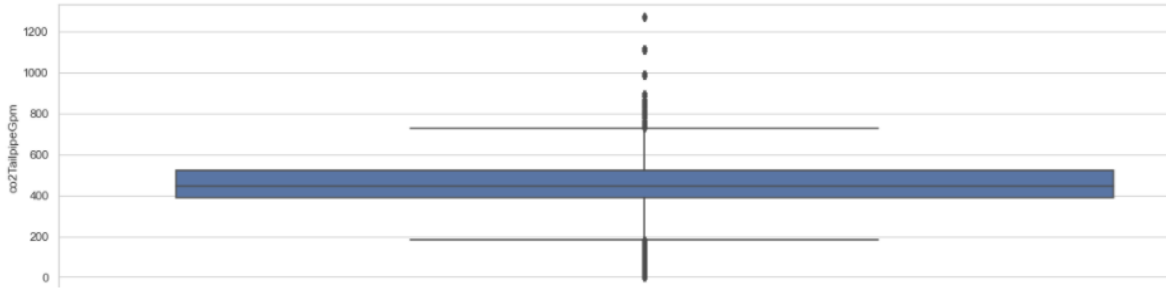
Presence of few outliers but can Mildgrade Gasoline Vehicles

Co2TailpipeGpm: Presence of outliers. They will be treated with scaling or Z Score methods

Presence of outliers over 1000 and will be treated with scaling

```
]: plt.figure(figsize=(20,5))
   sns.boxplot(y='co2TailpipeGpm', data=datafinal)
   #presence of significant outliers
```

```
]: <AxesSubplot:ylabel='co2TailpipeGpm'>
```

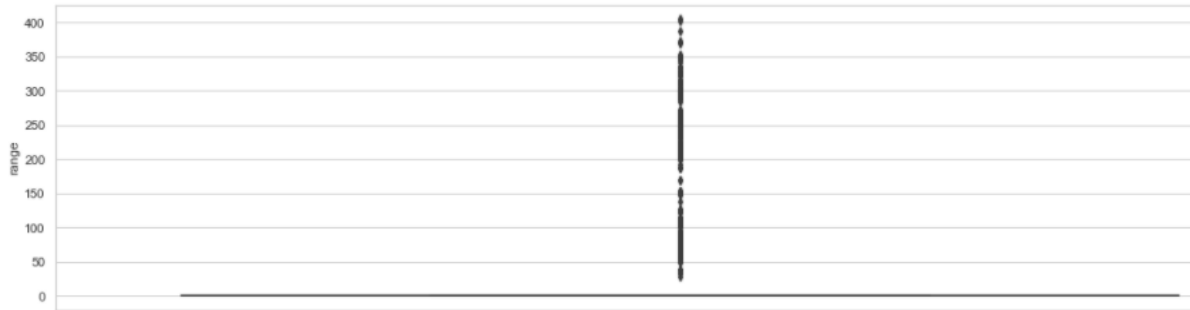


Range:

Range has too much missing data to detect outliers

```
plt.figure(figsize=(20,5))
sns.boxplot(y='range', data=datafinal)
#presence of significant outliers
```

```
<AxesSubplot:ylabel='range'>
```

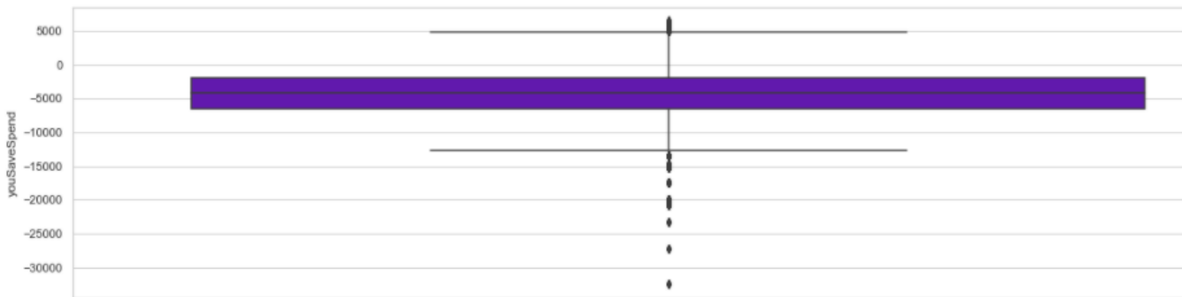


Yousavespend:

Presence of a few outliers but can be explained with use of expensive cars.

```
: plt.figure(figsize=(20,5))
  sns.boxplot(y='youSaveSpend', data=datafinal)
```

```
: <AxesSubplot:ylabel='youSaveSpend'>
```



Data Cleaning and Preparation

1) Analysing missing data:

```
data.isnull().sum()
```

```
barrels08          0
barrelsA08         0
```

```
drive           1186
engId              0
eng_dscr       16579
```

```
guzzler        41331
trans_dscr     28877
tCharger       35469
sCharger       42975
atvType        39831
fuelType2      42167
rangeA         42172
evMotor        42599
mfrCode        30808
c240Dscr       43803
charge240b         0
c240bDscr      43809
createdOn          0
modifiedOn         0
startStop      31689
```

There are few attributes where missing data is greater than 30,000 and total records are 43,000. Hence there is no point trying to fill the data and hence the attributes were dropped.

```
data1=data.drop(["guzzler", "trans_dscr","tCharger", "sCharger","atvType", "fuelType2", "rangeA", "evMotor", "mfrCode", "c240Dscr
```

The missing numerical data (except those marked as -1) was replaced with mean values

```
data=data.fillna(data.mean())
```

Missing character and categorical variables was replaced with “Not Available “and those containing -1 was replaced with 0.

```
]: data1.replace(np.NaN, "Not Available", inplace=True)
   data1.replace(-1, 0, inplace=True)
```

2) Preparing Data

Categorical variables were IntegerEncoded & OneHotEncoded:

```
datafinal1['make']=datafinal1['make'].astype(str)
datafinal1 = pd.get_dummies(datafinal1,prefix=['make'], columns = ['make'])
datafinal1 = pd.get_dummies(datafinal1,prefix=['fuelType'], columns = ['fuelType'])
datafinal1 = pd.get_dummies(datafinal1,prefix=['fuelType1'], columns = ['fuelType1'])
datafinal1 = pd.get_dummies(datafinal1,prefix=['VClass'], columns = ['VClass'])
datafinal1 = pd.get_dummies(datafinal1,prefix=['drive'], columns = ['drive'])
datafinal1 = pd.get_dummies(datafinal1,prefix=['trany'], columns = ['trany'])
#data2 = pd.get_dummies(data1,prefix=['Gender'], columns = ['Gender'])
```

Output:

```
make_AM General                              int64
make_ASC Incorporated                        int64
make_Acura                                   int64
make_Alfa Romeo                              int64
make_American Motors Corporation             int64
make_Aston Martin                            int64
make_Audi                                    int64
make_Aurora Cars Ltd                         int64
make_Autokraft Limited                       int64
make_Avanti Motor Corporation                int64
make_Azure Dynamics                          int64
make_BMW                                     int64
make_BMW Alpina                              int64
make_BYD                                     int64
make_Bentley                                 int64
make_Bertone                                 int64
make_Bill Dovell Motor Car Company           int64
```

MinMaxScaling the continuous variables to handle outliers and scaling variables:

```
continuous_vars = ['barrels08', 'barrelsA08', 'city08U', 'cityUF', 'co2', 'co2TailpipeGpm', 'co2TailpipeAGpm',
                   'comb08', 'fuelCost08', 'highway08' , 'rangeHwyA', 'UCityA', 'UHighway','youSaveSpend'
                   ]
minVec = datafinal[continuous_vars].min().copy()
maxVec = datafinal[continuous_vars].max().copy()
datafinal[continuous_vars] = (datafinal[continuous_vars]-minVec)/(maxVec-minVec)
datafinal.head()
```

barrels08	barrelsA08	city08U	cityUF	co2	co2A	co2TailpipeAGpm	co2TailpipeGpm	comb08	cylinders
0.332483	0.0	0.0	0.0	0.0	0	0.0	0.333333	0.103704	4.0
0.635900	0.0	0.0	0.0	0.0	0	0.0	0.636364	0.029630	12.0
0.258314	0.0	0.0	0.0	0.0	0	0.0	0.259259	0.148148	4.0
0.635900	0.0	0.0	0.0	0.0	0	0.0	0.636364	0.029630	8.0
0.367615	0.0	0.0	0.0	0.0	0	0.0	0.368421	0.088889	4.0

3) Test Train Split

Creating 80/20 train/test split.

UCity is the dependent variable while other variables are independent

```
: x = datafinal1.loc[:, datafinal1.columns != 'UCity']
  y = datafinal1.loc[:, datafinal1.columns == 'UCity']
```

```
: x_train, x_test, y_train, y_test = train_test_split(x, y , test_size = 0.2, random_state=0)
```

```
: x.shape
```

Training features consist of 238 attributes (including the model which has been hot encoded)

```
barrels08            int6
barrelsA08           int6
charge120            int6
charge240            int6
city08U              int6
cityA08              int6
cityA08U             int6
cityCD               int6
cityE                int6
cityUF               int6
co2                  int6
co2A                 int6
co2TailpipeAGpm      int6
co2TailpipeGpm       int6
comb08               int6
cylinders            int6
displ                int6
feScore              int6
fuelCost08           int6
```

```
make_Aston Martin                  int64
make_Audi                          int64
make_Aurora Cars Ltd               int64
make_Autokraft Limited             int64
make_Avanti Motor Corporation      int64
make_Azure Dynamics                int64
make_BMW                           int64
make_BMW Alpina                    int64
make_BYD                           int64
make_Bentley                       int64
make_Bertone                       int64
make_Bill Dovell Motor Car Company int64
make_Bitter Gmbh and Co. Kg        int64
make_Bugatti                       int64
make_Buick                         int64
make_CCC Engineering               int64
make_CODA Automotive               int64
make_CX Automotive                 int64
```

4) Dropping unnecessary data:

Several columns such as charge140, guzzler, tcharger, rangeA etc were dropped since they had more than 90% data missing. This would contribute significantly to the noise in the dataset and thus a decision of removing them was made.

Additionally, columns such as City08 was removed due to being identical to UCity and the model would have become greatly biased towards City08. Several other such as co2tailpipegpm, range columns with many missing, but masked as -1 instead of NaN values were dropped to remove bias towards -1/0.

```
data1=data.drop(["guzzler", "trans_dscr","tCharger", "sCharger","atvType", "fuelType2", "rangeA", "evMotor", "mfrCode", "c240Dscr
```

Model Creation

1) **Decision Tree Regressor:** Achieved a loss of 0.997. Successfully minimised the loss.

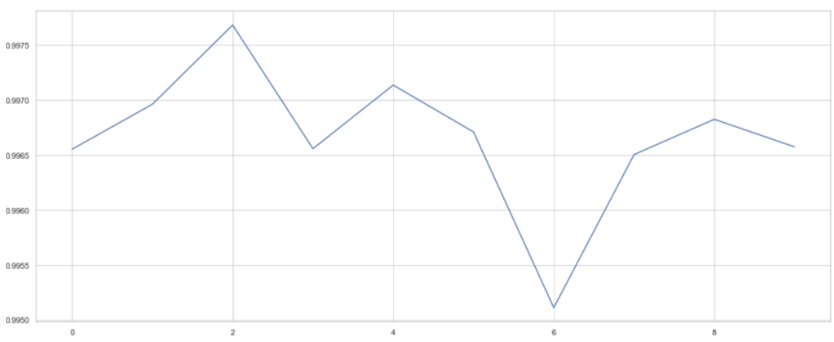
```
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor(criterion='mse',  max_depth=10)
tree.fit(x_train, y_train)
tree.score(x_train, y_train)
```

```
0.9978008193277822
```

K Fold Cross Validation for Loss:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(
    estimator=tree,
    X = x_train,
    y = y_train,
    cv=10
    )
print(accuracies)
plt.figure(figsize=(20,8))
sns.lineplot(y=accuracies,x= np.arange(0,10,1))
```

```
[0.99655233 0.99696126 0.9976813  0.99655705 0.99713547 0.99670927
 0.99511167 0.99650355 0.99682435 0.99657342]
```



2) **Artificial Neural Network Model:** Achieved a loss of 1.2 on training set and validation loss of 1.00 on test set which points to the fact the model is well trained

```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(512, input_dim=263, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=150, batch_size=10, validation_data=(x_test,y_test))
```

```
Epoch 148/150
3514/3514 [==============================] - 21s 6ms/step - loss: 1.3519 - val_loss: 0.8042
Epoch 149/150
3514/3514 [==============================] - 21s 6ms/step - loss: 1.0895 - val_loss: 0.8317
Epoch 150/150
3514/3514 [==============================] - 21s 6ms/step - loss: 1.2873 - val_loss: 1.0052
```

Model Explanation

Neural Network

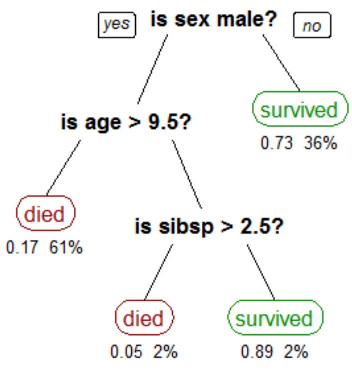
Neural Network models are also known as Multi-Layer Perceptron, Deep Learning models. These models were developed to imitate the human brain. A Neural Network Model has an input layer, hidden layers, and an output layer. The amount of perceptron in an input layer is equal to number of attributes. Each perceptron of the hidden layer uses a mathematical function to assign weights to attributes. Each perceptron in the layer uses a formula

$$\mathbf{y = b + W_i\, Xi}$$

to calculate the answer and then forward propagates it to the next layer. Each layer assigns weight to the inputs. These weights help determine the importance of any given variable. The larger the weight the more the significance of the attribute. All the inputs are multiplied by their weights, summed. These are then passed through an activation function (ReLu in our case) and forward propagated to next layer where this becomes an input. In the output layer using the sigmoid function, Neural Network calculates the probability of the record belonging to a certain class. With the help of the cost function, the error is calculated, and the goal is to reduce the error[2]. The error is then reduced by backpropagating the error and applying penalty to neurons, the weights are then recalculated and the whole process is repeated (an epoch).

Decision Tree Regressor

Decision trees are non-parametric supervised learning algorithms used for regression (and also classification). A decision tree predicts the value by creating decision rules by learning data features. A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/**internal node**, based on which the tree splits into branches/ **edges**[1]. The end node which has no further node is called leaf node. Leaf node provides an output after performing complex calculations



Decision trees utilises multiple algorithms to split into further nodes. The decision tree then decides where to place the N features to place. Generally, a feature with highest influence over data is placed on the root node and those with lower influence are placed further down the tree.

Conclusion

Both Decision Trees and Neural Network performs similarly in this case and either of the one can be chosen for this dataset. It can be further concluded that Electric Vehicles provide lots of benefits over natural resources powered vehicles.

Further it can be concluded that:

- Electric vehicles produces no carbon dioxide.
- Save/Spend ratio is highest for EV Vehicles
- Electric vehicles provide highest City and Highway averages of any vehicle
- Chevrolet, Ford and GMC are the most used vehicles
- Volkswagen, Toyota and Nissan has highest UCity average.
- Cars of expensive brands such as Mercedes, BMW, Porsche tends to pollute the environment more in comparison to non-luxurious manufacturers
- Mildgrade Gasoline vehicles produce highest co2 emissions of any other type of fuel and gives the least average
- 2010 was a turning point for vehicles as UCity average increased a lot after 2010

References

1) Gupta, Prashant. “Decision Trees in Machine Learning.” *Medium*, Towards Data Science, 12 Nov. 2017, towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.
2) Singh, Kartik. “Business Analytics Consulting Project”, Seneca College, 8 Aug. 2021

Declaration

I, Kartik Singh, declare that the attached assignment is my own work in accordance with the Seneca Academic Policy. I have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. I have not distributed my work to other students.