The 2021 EPA Automotive Trends Report

Greenhouse Gas Emissions, Fuel Economy, and Technology since 1975

MBAN 6110
Delina Ivanova
August 3, 2022

Prepared by: Chris Kershaw, Farida Madkour, Ankur Basu, Naila Mahmood, Shijie Chen, Balu Nair

Executive Summary

Objective – Rising inflation rates have affected economies worldwide highlighting the need for automakers to fine-tune their forecasting methodologies in order to determine ideal levels of output. The report explores in detail how the *Cars and Fuel Economy* dataset was analyzed to identify which characteristics are important in influencing manufacturing quantities for specific types of vehicles.

Data preparation – Prior to running regression models, the data went through several stages of data cleaning and data transformations. Some of those are

- Removing additional variables that were already included in the data set.
- Modifying column names to meet Python syntax requirements.
- Imputing Null values.
- Data normalization to eliminate bias from the model.
- Dummy variable creation to account for the effect of categorical data.

Post-data preparation correlation analysis showed relationships between our chosen dependent variable (**Production in Thousands**) and the other variables in the data set. Correlation analysis showed strong relationships between dependent variables and the following variables which were ultimately utilized in forming the final regression model.

1. Acceleration (0-60 time in seconds)	2. Powertrain Gasoline
3. Manual Total Share	4. Sedan/Wagon
5. GM	6. Mazda
7. Drivetrain – Rear	8. Real World MPG
9. Car SUV	10. Stop Start

This robust regression equation was used to test 5 models 1) Linear Regression, 2) Ridge Regression, 3) Lasso Regression, 4) Elastic-Net and 5) Random Forest using the same dependent and independent variables. A comparison of their variation (R2) indicated Random Forest model showed the highest variation while the elastic net displayed the lowest variation (R2).

Conclusion & Recommendation – The Forest Regressor model yielded the best results compared to the other 4 models, and the R2 of the Forest Regressor model is 99%. To strengthen the model, it is recommended to integrate external variables that are not currently in the dataset that would be meaningful for the prediction of production levels. Overall, applying a Forest Regressor forecasting technique will support in reducing the ripple effects of production shortages, and alleviate the financial burden that is relayed to customers in the form of price increases to vehicles.

Introduction

Due to rising inflation rates caused by the aftermath of Covid-19 and various political stressors, several automotive manufacturers have faced worldwide supply chain disruptions and must re-evaluate their forecasting methods to determine ideal production levels.

As such, the chosen dataset for our analysis is as follows: https://www.epa.gov/automotive-trends/explore-automotive-trends-data#DetailedData which contains 5,171 observations that were obtained by the Environmental Protection Agency in the United States for car models produced between the years 1975-2020.

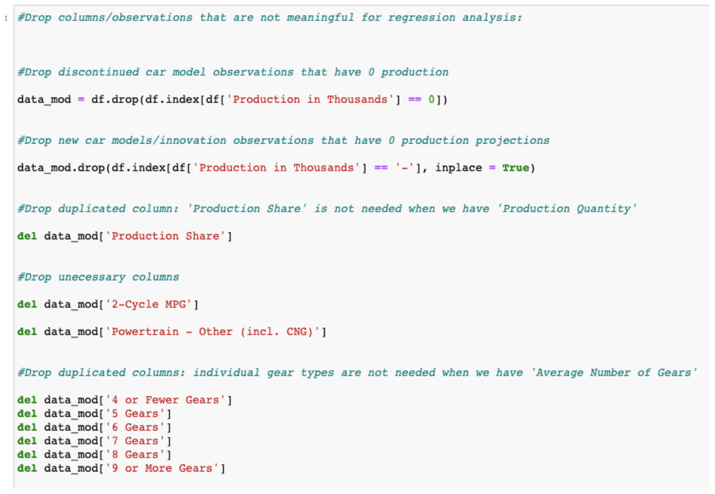
Problem Summary

A detailed analysis was conducted on the *Cars and Fuel Economy Dataset* in order to uncover which attributes are significant in determining production quantity for certain types of cars? The findings have been used to build a data model that determines the suggested quantity of cars that will need to be produced whenever a new car model is developed by a manufacturer.

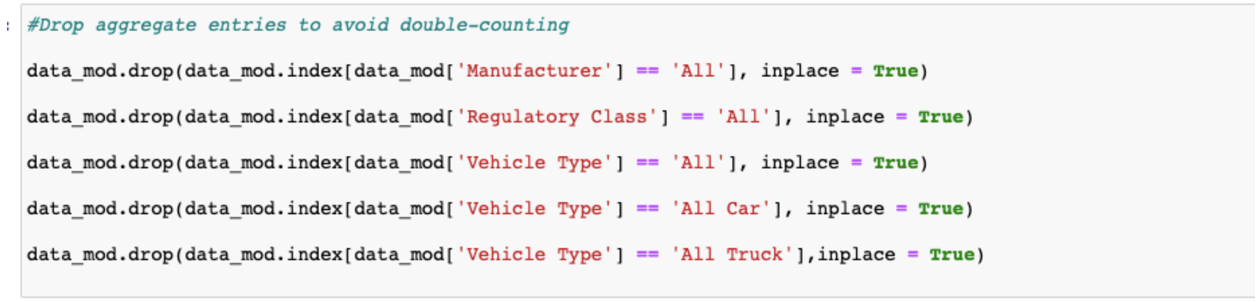
Data Exploration & Cleaning Approach

1) Exploration & Data Cleaning

After importing all the required libraries and reading the dataset, it was clear that data cleaning and transformation steps would need to be conducted, given that some columns had null values, unsuitable headings, and the default data types were incorrect. The dataset also included columns that were duplicated, which overstated certain features, in addition to having columns that were not meaningful for regression analysis, which was dropped from the dataset altogether. For instance, having 6 additional columns for individual gear types was not meaningful, when the "average number of gear types" was given in another column. Furthermore, having "production share" as a percentage was not meaningful when "production in thousands" was given to specify manufacturing quantities. Since our target variable is also related to production quantity, it was logical to drop the production share variable.

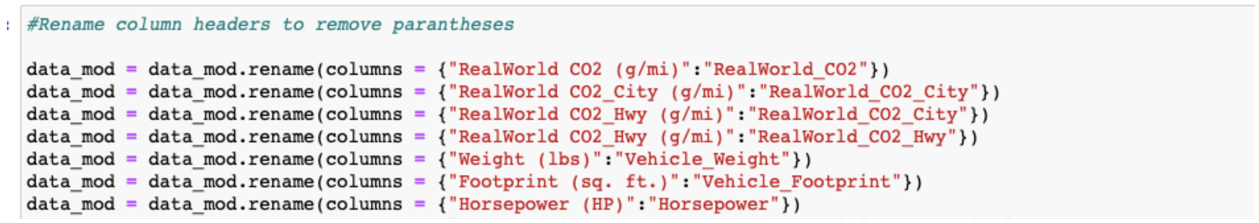


```
#Drop columns/observations that are not meaningful for regression analysis:


#Drop discontinued car model observations that have 0 production

data_mod = df.drop(df.index[df['Production in Thousands'] == 0])

#Drop new car models/innovation observations that have 0 production projections

data_mod.drop(df.index[df['Production in Thousands'] == '-'], inplace = True)

#Drop duplicated column: 'Production Share' is not needed when we have 'Production Quantity'

del data_mod['Production Share']

#Drop unecessary columns

del data_mod['2-Cycle MPG']

del data_mod['Powertrain - Other (incl. CNG)']

#Drop duplicated columns: individual gear types are not needed when we have 'Average Number of Gears'

del data_mod['4 or Fewer Gears']
del data_mod['5 Gears']
del data_mod['6 Gears']
del data_mod['7 Gears']
del data_mod['8 Gears']
del data_mod['9 or More Gears']
```



```
#Drop aggregate entries to avoid double-counting

data_mod.drop(data_mod.index[data_mod['Manufacturer'] == 'All'], inplace = True)

data_mod.drop(data_mod.index[data_mod['Regulatory Class'] == 'All'], inplace = True)

data_mod.drop(data_mod.index[data_mod['Vehicle Type'] == 'All'], inplace = True)

data_mod.drop(data_mod.index[data_mod['Vehicle Type'] == 'All Car'], inplace = True)

data_mod.drop(data_mod.index[data_mod['Vehicle Type'] == 'All Truck'],inplace = True)
```

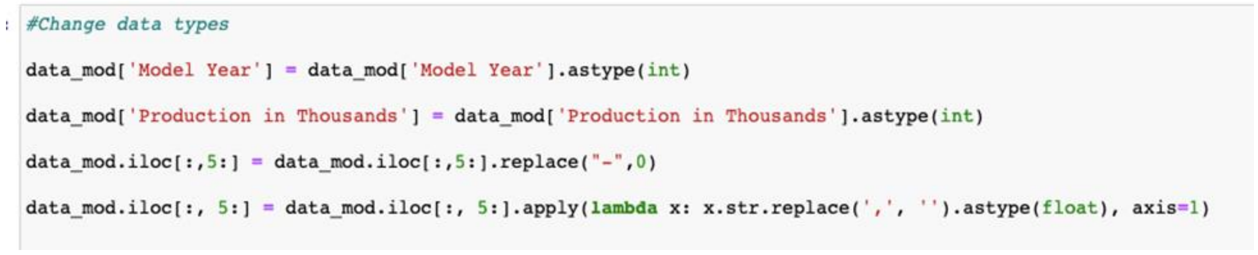
Furthermore, many of the columns in the dataset had parentheses within headers, which was difficult to work with in Python. To remedy this, the names of multiple columns were changed to remove all brackets and dashes.



```
#Rename column headers to remove parantheses

data_mod = data_mod.rename(columns = {"RealWorld CO2 (g/mi)":"RealWorld_CO2"})
data_mod = data_mod.rename(columns = {"RealWorld CO2_City (g/mi)":"RealWorld_CO2_City"})
data_mod = data_mod.rename(columns = {"RealWorld CO2_Hwy (g/mi)":"RealWorld_CO2_City"})
data_mod = data_mod.rename(columns = {"RealWorld CO2_Hwy (g/mi)":"RealWorld_CO2_Hwy"})
data_mod = data_mod.rename(columns = {"Weight (lbs)":"Vehicle_Weight"})
data_mod = data_mod.rename(columns = {"Footprint (sq. ft.)":"Vehicle_Footprint"})
data_mod = data_mod.rename(columns = {"Horsepower (HP)":"Horsepower"})
```

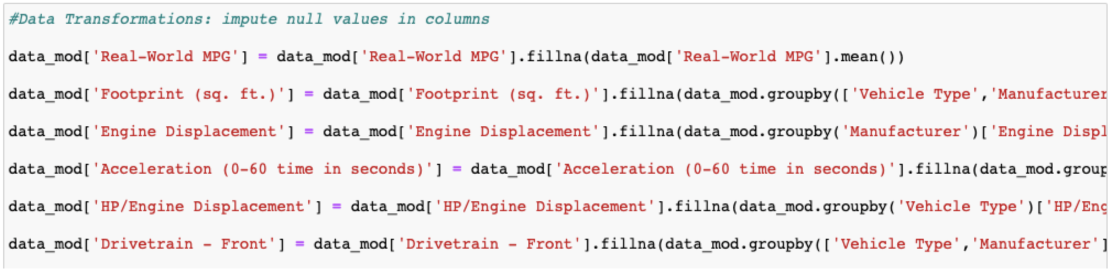
2) Data Transformations

Once the columns were finalized, it was found that most of the data was stored as an object data type, which is not ideal when predicting a continuous target variable. As such, all of the variables were transformed to their appropriate type.



```
#Change data types
data_mod['Model Year'] = data_mod['Model Year'].astype(int)
data_mod['Production in Thousands'] = data_mod['Production in Thousands'].astype(int)
data_mod.iloc[:,5:] = data_mod.iloc[:,5:].replace("-",0)
data_mod.iloc[:, 5:] = data_mod.iloc[:, 5:].apply(lambda x: x.str.replace(',', '').astype(float), axis=1)
```

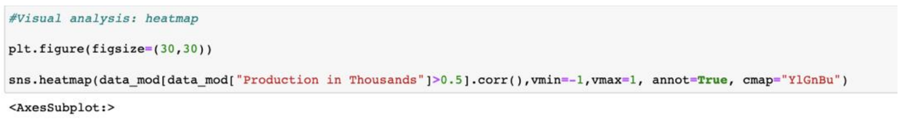
In order to impute all the null values, a case-by-case approach was taken to determine all the columns that were impacted and decide on whether to impute them using the mean or median. Given that the Automotive dataset is not normally distributed, it was decided to use the median for replacing most of the null values as follows:



```
#Data Transformations: impute null values in columns
data_mod['Real-World MPG'] = data_mod['Real-World MPG'].fillna(data_mod['Real-World MPG'].mean())
data_mod['Footprint (sq. ft.)'] = data_mod['Footprint (sq. ft.)'].fillna(data_mod.groupby(['Vehicle Type','Manufacturer
data_mod['Engine Displacement'] = data_mod['Engine Displacement'].fillna(data_mod.groupby('Manufacturer')['Engine Displ
data_mod['Acceleration (0-60 time in seconds)'] = data_mod['Acceleration (0-60 time in seconds)'].fillna(data_mod.group
data_mod['HP/Engine Displacement'] = data_mod['HP/Engine Displacement'].fillna(data_mod.groupby('Vehicle Type')['HP/Eng
data_mod['Drivetrain - Front'] = data_mod['Drivetrain - Front'].fillna(data_mod.groupby(['Vehicle Type','Manufacturer']
```

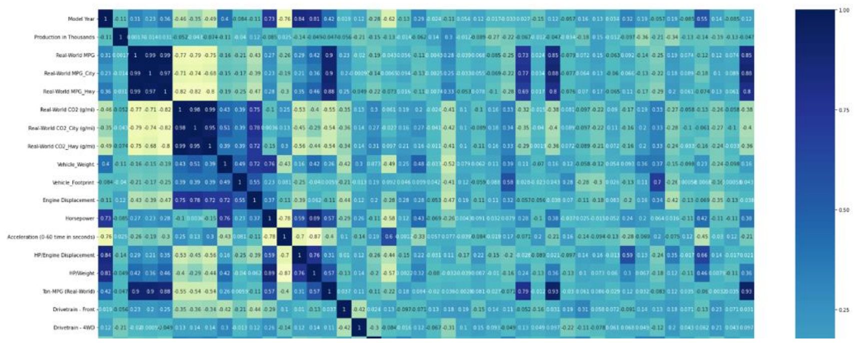
3) Visual Analysis

In order to determine which variables may have relationships that are worth exploring with regression analysis, a heatmap was created to display the strength of relationships between features. Given that “Production in Thousands” is the y-variable (the “target” variable), the data was grouped to focus on relationships between production and other factors that might correlate to it.



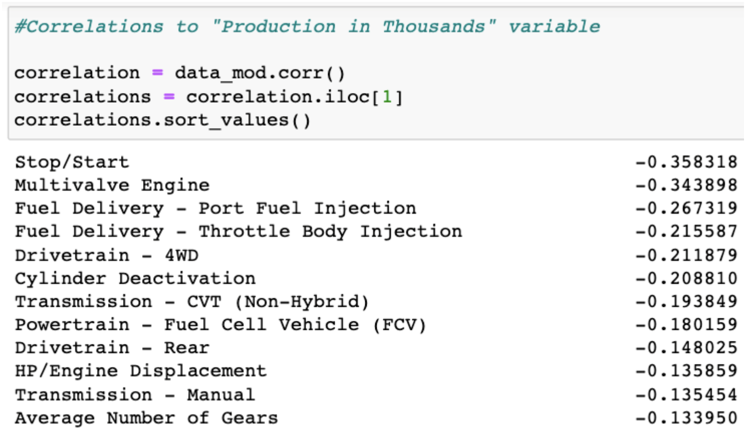
```
#Visual analysis: heatmap

plt.figure(figsize=(30,30))

sns.heatmap(data_mod[data_mod["Production in Thousands"]>0.5].corr(),vmin=-1,vmax=1, annot=True, cmap="YlGnBu")
```

```
<AxesSubplot:>
```



4) Correlation Analysis

Given the large number of variables in the Automotive dataset, the heatmap was visually difficult to interpret as it was cross-referencing every single feature and not just those related to production quantity. As such, the correlation was presented in array format as follows. The analysis shows that there is a very low correlation between production and any of the variables in our dataset, meaning that further transformations need to be conducted to understand data distributions.



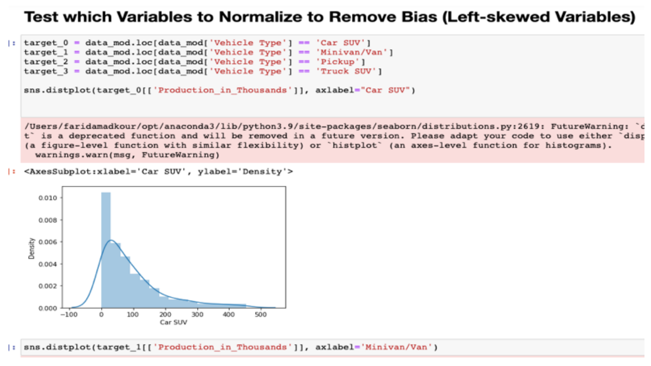
```
#Correlations to "Production in Thousands" variable

correlation = data_mod.corr()
correlations = correlation.iloc[1]
correlations.sort_values()
```

```
Stop/Start                                          -0.358318
Multivalve Engine                                   -0.343898
Fuel Delivery - Port Fuel Injection                 -0.267319
Fuel Delivery - Throttle Body Injection             -0.215587
Drivetrain - 4WD                                    -0.211879
Cylinder Deactivation                               -0.208810
Transmission - CVT (Non-Hybrid)                     -0.193849
Powertrain - Fuel Cell Vehicle (FCV)                -0.180159
Drivetrain - Rear                                   -0.148025
HP/Engine Displacement                              -0.135859
Transmission - Manual                               -0.135454
Average Number of Gears                             -0.133950
```

Further Data Transformations

1) **Data Distribution Testing:** After implementing further visualization methods to show data distributions, it was clear that the some of the dataset was not normally distributed and multiple variables were skewed to the left, in the same direction as the target variable "Production in Thousands," which over-fit the data feeding the model in our initial regression tests. Following distribution testing on various variables, it was evident that the

"Powertrain" and "Transmission" x-variables needed to be transformed to remove bias from the model.

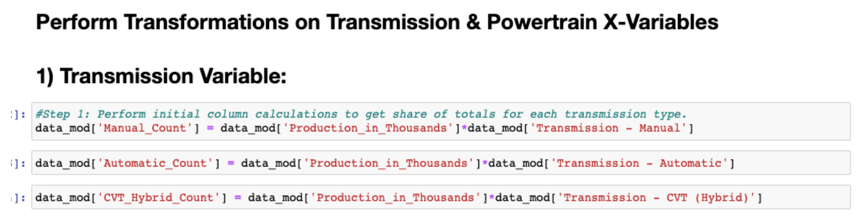

Test which Variables to Normalize to Remove Bias (Left-skewed Variables)

```
target_0 = data_mod.loc[data_mod['Vehicle Type'] == 'Car SUV']
target_1 = data_mod.loc[data_mod['Vehicle Type'] == 'Minivan/Van']
target_2 = data_mod.loc[data_mod['Vehicle Type'] == 'Pickup']
target_3 = data_mod.loc[data_mod['Vehicle Type'] == 'Truck SUV']

sns.distplot(target_0[['Production_in_Thousands']], axlabel="Car SUV")
```

```
/Users/faridamadkour/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `d
t` is a deprecated function and will be removed in a future version. Please adapt your code to use either `disp
(a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='Car SUV', ylabel='Density'>
```

```
sns.distplot(target_1[['Production_in_Thousands']], axlabel='Minivan/Van')
```

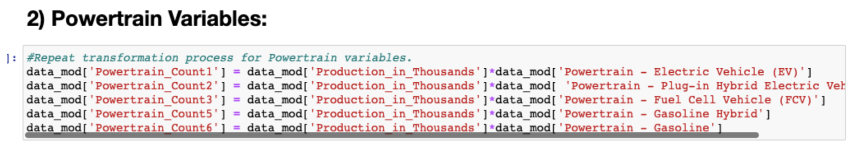

Perform Transformations on Transmission & Powertrain X-Variables

1) Transmission Variable:

```
#Step 1: Perform initial column calculations to get share of totals for each transmission type.
data_mod['Manual_Count'] = data_mod['Production_in_Thousands']*data_mod['Transmission - Manual']
```

```
data_mod['Automatic_Count'] = data_mod['Production_in_Thousands']*data_mod['Transmission - Automatic']
```

```
data_mod['CVT_Hybrid_Count'] = data_mod['Production_in_Thousands']*data_mod['Transmission - CVT (Hybrid)']
```


2) Powertrain Variables:

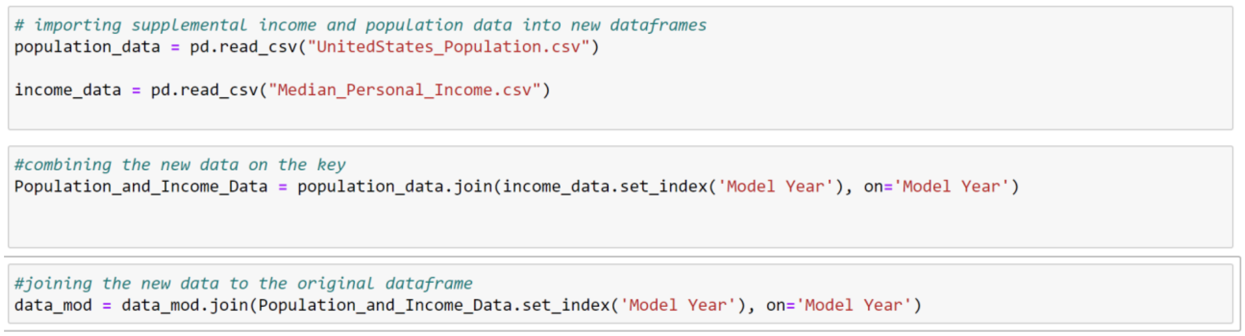
```
#Repeat transformation process for Powertrain variables.
data_mod['Powertrain_Count1'] = data_mod['Production_in_Thousands']*data_mod['Powertrain - Electric Vehicle (EV)']
data_mod['Powertrain_Count2'] = data_mod['Production_in_Thousands']*data_mod[ 'Powertrain - Plug-in Hybrid Electric Veh
data_mod['Powertrain_Count3'] = data_mod['Production_in_Thousands']*data_mod['Powertrain - Fuel Cell Vehicle (FCV)']
data_mod['Powertrain_Count5'] = data_mod['Production_in_Thousands']*data_mod['Powertrain - Gasoline Hybrid']
data_mod['Powertrain_Count6'] = data_mod['Production_in_Thousands']*data_mod['Powertrain - Gasoline']
```

Following these transformations, data correlation scores improved as we identified 7 variables that were strongly correlated to "Production in Thousands," as follows:

```
manual_total_share                                 0.601234
Powertrain_Electric                                0.829799
Powertrain_Gasoline_Hybrid                         0.844538
lockup_total_share                                 0.920635
other_total_share                                  0.939838
Powertrain_Plugin_Hybrid                           0.941358
Powertrain_Gasoline                                0.970684
```

2) **Merging data:** From analyzing the dataset, we had determined that there was no data relating to population or income, which can be valuable predictors of production quantity

for vehicles. Therefore, to supplement the dataset, USA population and median personal income data were merged to the data frame to strengthen the x-variables in the model.

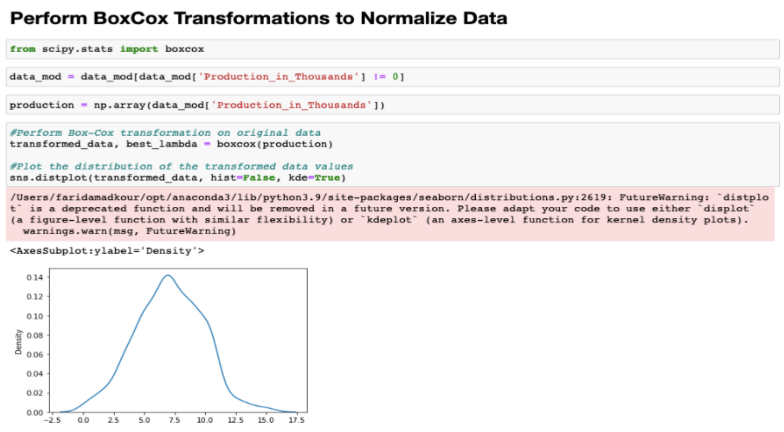


```
# importing supplemental income and population data into new dataframes
population_data = pd.read_csv("UnitedStates_Population.csv")

income_data = pd.read_csv("Median_Personal_Income.csv")
```

```
#combining the new data on the key
Population_and_Income_Data = population_data.join(income_data.set_index('Model Year'), on='Model Year')
```

```
#joining the new data to the original dataframe
data_mod = data_mod.join(Population_and_Income_Data.set_index('Model Year'), on='Model Year')
```

3) **Perform BoxCox transformations:** In order to normalize the data, a BoxCox transformations was applied to the production as it was skewed to the left, prior to conducting final regressions.



Perform BoxCox Transformations to Normalize Data

```
from scipy.stats import boxcox
```

```
data_mod = data_mod[data_mod['Production_in_Thousands'] != 0]
```

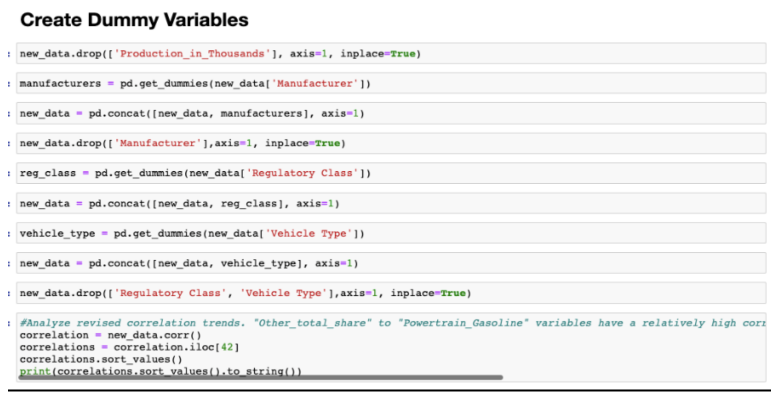
```
production = np.array(data_mod['Production_in_Thousands'])
```

```
#Perform Box-Cox transformation on original data
transformed_data, best_lambda = boxcox(production)

#Plot the distribution of the transformed data values
sns.distplot(transformed_data, hist=False, kde=True)
```

```
/Users/faridamadkour/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplo
t` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

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

4) **Hot Encoding:** Dummy variables were created to preprocess categorical features within the dataset, as the final preparation step for our machine learning models.



Create Dummy Variables

```
: new_data.drop(['Production_in_Thousands'], axis=1, inplace=True)

: manufacturers = pd.get_dummies(new_data['Manufacturer'])

: new_data = pd.concat([new_data, manufacturers], axis=1)

: new_data.drop(['Manufacturer'],axis=1, inplace=True)

: reg_class = pd.get_dummies(new_data['Regulatory Class'])

: new_data = pd.concat([new_data, reg_class], axis=1)

: vehicle_type = pd.get_dummies(new_data['Vehicle Type'])

: new_data = pd.concat([new_data, vehicle_type], axis=1)

: new_data.drop(['Regulatory Class', 'Vehicle Type'],axis=1, inplace=True)

: #Analyze revised correlation trends. "Other_total_share" to "Powertrain_Gasoline" variables have a relatively high corr
  correlation = new_data.corr()
  correlations = correlation.iloc[42]
  correlations.sort_values()
  print(correlations.sort_values().to_string())
```

Methodology

Overall, 5 models were tested including: 1) Linear Regression, 2) Ridge Regression, 3) Lasso Regression, 4) Elastic-Net and 5) Random Forest using the same x and y variables for a comparison of the R2 output.

Based on the analysis, it is evident that the x-variables that are most meaningful for prediction of car production quantities and have the strongest correlation are:

y = Transformed Production

x = Acceleration (0-60 time in seconds), Powertrain Gasoline, Manual Total Share, Sedan/Wagon, GM, Stop Start, Mazda, Car SUV, Drivetrain – Rear, Real World MPG

To translate this information into y = mx + b equation format, the data can be modelled in the form of:

$$Production = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

Transformed Production = Acceleration (0-60 time in seconds) x1 + Powertrain Gasoline x2 + Manual Total Share x3 + Sedan/Wagon x4 + GM x5 + Stop Start x6 + Mazda x7 + Car SUV x8 + Drivetrain – Rear x8 + Real World MPG x9.

Comparing Model Outputs

Coefficients

Linear Regression:

Transformed Production = (-0.17) Acceleration (0-60 time in seconds) +
(0.058) Powertrain Gasoline + (26.162) Manual Total Share + (0.436) Sedan/Wagon +
(1.1) GM + (-1.229) Stop Start + (-.938) Mazda + (-0.412) Car SUV +
(-1.248) Drivetrain – Rear + (-0.016) Real World MPG

Ridge Regression

Transformed Production = (-0.17) Acceleration (0-60 time in seconds) +
(0.058) Powertrain Gasoline + (25.886) Manual Total Share + (0.438) Sedan/Wagon +
(1.1) GM + (-1.236) Stop Start + (-.94) Mazda + (-0.413) Car SUV +
(-1.248) Drivetrain – Rear + (-0.016) Real World MPG

Lasso Regression

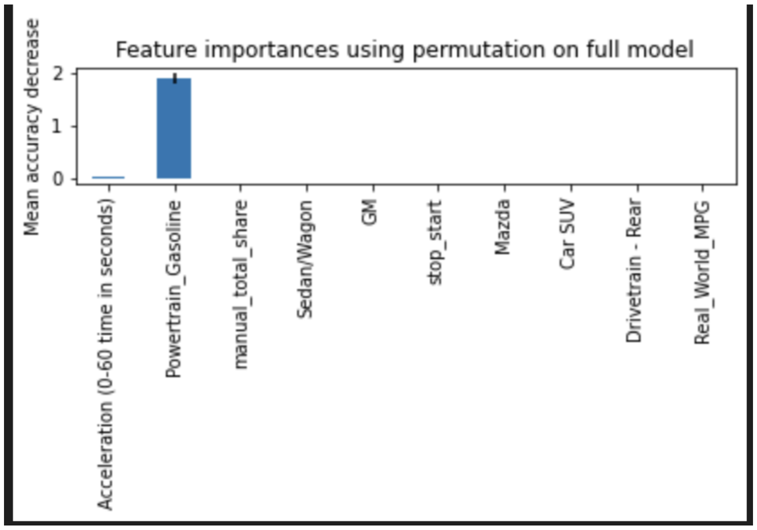
Transformed Production = (-0.173) Acceleration (0-60 time in seconds) +
(0.072) Powertrain Gasoline + (12.806) Manual Total Share + (0.479) Sedan/Wagon +
(0.933) GM + (-1.44) Stop Start + (-.906) Mazda + (-0.43) Car SUV +
(-1.226) Drivetrain – Rear + (-0.016) Real World MPG

Elastic Net Regression:

Transformed Production = (-0.177) Acceleration (0-60 time in seconds) +
(0.081) Powertrain Gasoline + (2.494) Manual Total Share + (0.579) Sedan/Wagon +
(0.912) GM + (-1.577) Stop Start + (-.981) Mazda + (-0.518) Car SUV +

(-1.212) Drivetrain – Rear + (-0.017) Real World MPG

Feature Importance for Forest Regressor:



In our initial linear regression, we found that the variables were all statistically significant and the largest coefficients were Manual Total Share, GM, Stop Start, and Drivetrain – Rear. The coefficients remained quite similar between all the regressions with only the Manual Total Share variable varying significantly. In the forest regressor we found a curious result of the largest feature importance by far belonged to the Powertrain – Gasoline variable, which was quite a small coefficient within the other variables. The only other variable with registerable feature importance in the graph above is the Acceleration variable, which was also a small coefficient in the other regressions.

Goodness of Fit

```
#R2 Values:

#Random Forest Regression = 99%
#Linear Regression = 64%
#Ridge Regression = 64%
#Lasso Regression = 62%
#Elastic Net = 58%
```

In summary, the Forest Regressor model performed the best with an R2 value of 99%, and Elastic Net performed the worst with an R2 of 58%.

Model Comparison

Despite a high R2 value, the Random Forest Regressor could prove to be misleading given the current choice of x-variables being used to train and test the model. Given the lack of a dictionary, reducing variables to only those that have a clear definition reduced the r2 value in a non-meaningful way because removing variables from the model is bound to worsen the results, similar to how adding variables improves the R2 value – without necessarily indicating a good model. However, we can still conclude that it produced the best algorithm, given that it reduced our initial over-fitting problem when the distribution plots indicated that the x-variable data was skewed to the left, in the same direction as the target variable, which yielded a misleadingly high accuracy score in the preliminary regression tests. Even when new datapoints such as Income and Population were introduced in the dataset, the overall algorithm was not affected and it handled non-linear parameters efficiently, causing it to outperform the other four machine learning models.

Despite Random Forest working well to solve our automotive classification and regression problem with high accuracy, a limitation of this model includes requiring a longer training period given that it generates multiple decision trees and makes decisions on the basis of majority votes. In this case, our x-variables would need further investigating and training to validate a score of 99% through additional feature importance measuring. As such, adding external variables to the dataset may generate a larger number of trees that can make the algorithm slow and ineffective for real-time predictions. A similar comparison of the advantages and disadvantages of the remaining four models is outlined below:

Advantages	**Disadvantages**
Linear Regression	
• Easy to implement with a simple algorithm.	• Over simplifies real world problems.

• Very efficient to train even on systems with low computational power. • Easy to interpret the results and understand model outputs.	• Extremely sensitive to outliers. • Prone to noise and over-fitting: feature-engineering required. • Prone to multicollinearity: dimensionality reduction techniques were used to reduce these effects.
Ridge Regression	
• Decreases complexity of a model. • Performs well with large datasets. • Can be used to resolve multicollinearity issues when independent variables are highly correlated.	• Does not reduce the number of variables in a model and is not ideal for feature reduction. • Trades variance for bias. • Never sets the value of coefficients to zero unlike Lasso regression.
Lasso Regression	
• Shrinks unnecessary variables and performs feature selection for better prediction. • Helps to reduce the risk of over-fitting a model. • Works best if there is a small number of significant parameters.	• Is limited when the number of predictors are greater than the number of observations. • If there are two or more highly collinear variables, one is selected at random which can create bias in the data.
Elastic-Net	
• Combines the regularization of both Lasso and Ridge regression: shrinkage & sparsity. • Can handle correlations between predictors better than Lasso.	• Computationally more expensive than Lasso & Ridge. • If the main objective is feature selection then Lasso is better to use.

<u>**Recommendations & Conclusion**</u>

Given that a data dictionary was not provided to accompany the U.S Automotive dataset, one recommendation would be to form clear definitions of what each of the variables represent, in order to further test which are most appropriate for regression model building. The current x-variables have p-values that are < 0.005, indicating a strong statistical significance. However, it would be beneficial to make clear distinctions between variables in order to gain a better understanding of the business context and make more informed decisions regarding which variables to drop from the model.

In order to strengthen the model, it is recommended to integrate external variables that are not currently in the dataset that would be meaningful for prediction of production levels. For instance, technology features are a factor that customers consider when purchasing a vehicle, therefore it could be worth exploring if there is a relationship between safety features such as automatic parking, adaptive headlights, backup cameras and sensors in relation to production quantity. Furthermore, vehicle pricing information would also be another variable to consider adding to the model given that potential buyers will make price comparisons relative to their budget guidelines, which potentially may impact production levels and could be worth exploring. In relation to financing options, having interest rate data can be an indicator of ideal production quantities given that the automobile industry is extremely sensitive towards increases in interest. An increase in the interest rate makes servicing automobile debt more expensive, which increases costs for an auto-manufacturer and creates pressure to drive more sales (i.e. production) to remain profitable.

In summary, the Forest Regressor model yielded the best results, however, an R2 of 99% may be misleading if the model requires further testing with more appropriate x-variables that need to be integrated from an external data-source. Overall, global automotive supply chains have been disrupted as a result of the Coronavirus outbreak, significantly hindering production and leaving markets drastically undersupplied. Applying a Forest Regressor forecasting technique will support in reducing the ripple effects of production shortages, and alleviate the financial burden that is relayed to customers in the form of price increases to vehicles.

References

Median Personal Income Data

USA Population Data

EPA Automotive Trends Data