Hyperiondev

TASK

Exploratory Data Analysis on the Automobile Data Set

Visit our website

Introduction

In this data exploration task, we will be working with a dataset for automobiles, the dataset has 205 rows and 26 columns of data. The columns range from symboling which is the risk rating relative to a cars price, to the horsepower and engine-sizes.

symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size
3	138	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130
3	138	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130

The dataset is collected for twenty-two different car makes with Toyota being the most collected car make in the dataset. And the majority of the cars have a standard engine and use mostly gas. Before we can explore the dataset, it needs to be cleansed by removing any unwanted data and handling all missing data. All of this is done in the body of this report.

DATA CLEANING

For this part of the analysis any duplicate rows, empty rows, and anything else that needed to be removed or corrected was handled. Below are the steps taken in cleaning the dataset. There were no duplicate rows that can be found in the dataset as can be seen in the output visual below as the shape of the dataframe is still the same.

```
# removing duplicate rows
auto_df.drop_duplicates(keep='first')

auto_df.shape
```

```
(205, 26)
```

But from further inspection it seems that the empty dataset was not being recognised by python as the '?' were counted as data inputs and this also needed to be rectified by replacing all '?' occurrences with pythons' 'NaN'.

```
# replacing all ? with NaN
auto_df.replace('?', np.NaN, inplace = True)
```

```
auto_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  164 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       203 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               201 non-null    object
 19  stroke             201 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         203 non-null    object
 22  peak-rpm           203 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              201 non-null    object
dtypes: float64(5), int64(5), object(16)
```

From above we can see that the number of non-null values have dropped but still none of the columns are completely empty. Before we deal with missing data, we need to deal with the datatypes of some of these columns. The following columns bore, stroke, horsepower, peak-rpm, price and normalized-losses have numerical values, and the datatype should be a 'float' or 'int'.

Below I will correct the datatypes for all the columns from object to the appropriate datatype except and for normalized losses as I will deal with it after handling missing data.

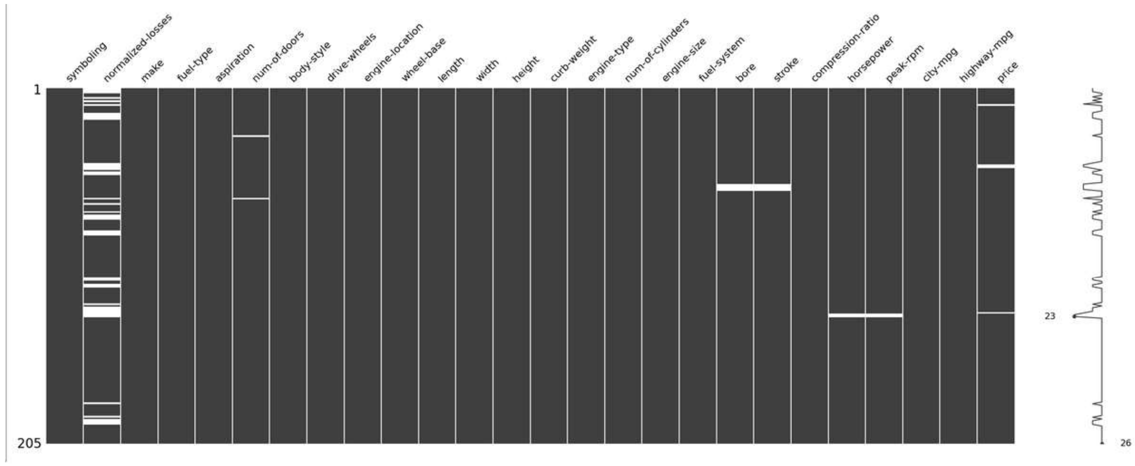
```
# columns to update datatype on
colum = ['bore','stroke','horsepower','peak-rpm','price']

auto_df[colum] = auto_df[colum].astype('float') # updating

auto_df.info() # viewing colum info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  164 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       203 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  205 non-null    float64
 21  horsepower         203 non-null    float64
 22  peak-rpm           203 non-null    float64
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              201 non-null    float64
dtypes: float64(10), int64(5), object(11)
memory usage: 41.8+ KB
```

The dataset had no empty rows or columns that needed to be removed and now that we have dealt with cleaning the data, we can move on to handling missing values with the suitable form of imputation. From above we can see that seven of the columns have missing values and this needs to be investigated.

MISSING DATA

First thing to do is to locate the missing data in the dataframe, a visualisation of the dataframe will make it is easier to get a quick sense of the spread of the missing data and which columns. Below is a visual representation of the dataset and the missing values.



In the plot above we can see that the missing data is in seven columns with most of the missing data is in the normalized-losses column. The breakdown of the amount of missing data points per column can be seen below.

```
normalized-losses    41
num-of-doors          2
bore                  4
stroke                4
horsepower            2
peak-rpm              2
price                 4
dtype: int64
```

With the code below a calculation was performed that could determine the percentage of missing data and from the results the missing data only makes 1.11% of the overall dataset.

```
# getting the total number of cells
total = np.product(auto_df.shape)

# gettng total number of missing data points
missing_data_total = missing_datapoints.sum()

# checking the percentage of missing data
percent = f"{round((missing_data_total/total)*100,2)}%"

print("Percentage of missing datapoints is",percent)
```

```
Percentage of missing datapoints is 1.11%
```

Now we need to decide on how to handle the missing data. for the columns below missing data will be handled with similar case imputation of the mean or median since these are numerical variables.

- normalized-losses
- bore
- stroke
- horsepower
- peak-rpm

- price

And for the 'num-of-doors' column I will do imputation with the mode since this is a categorical variable.

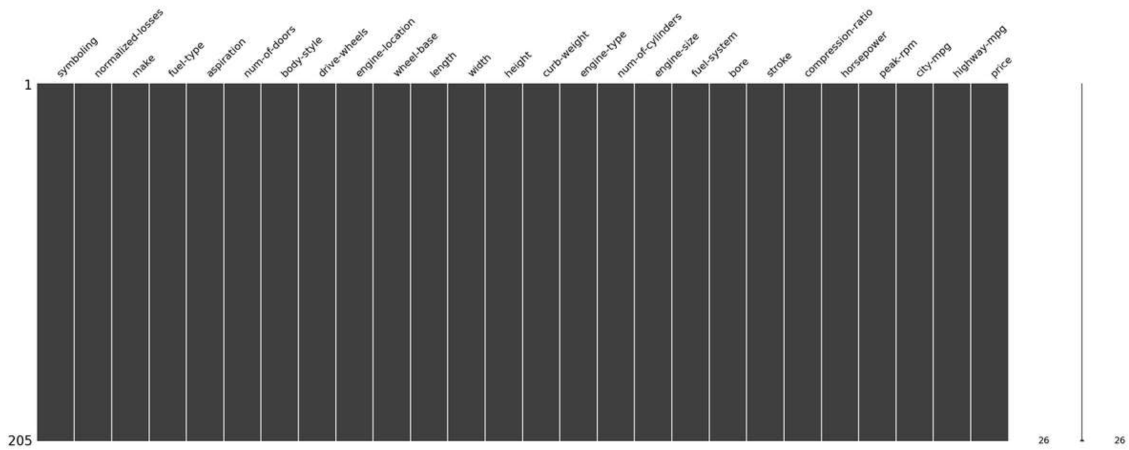
Imputation for columns with numerical values

```
# grouping price and getting the median price
price = auto_df.groupby('make')['price'].transform('median')

# similar case imputation of the median
auto_df['price'].fillna(price, inplace = True)

# grouping peak rpm by 'num-of-doors' and 'body-style'
rpm = auto_df.groupby(['num-of-doors','body-style'])['peak-rpm'].transform('median')

# similar case imputation of the median
auto_df['peak-rpm'].fillna(rpm, inplace = True)

# grouping horsepower by 'num-of-doors' and 'body-style'
power = auto_df.groupby(['num-of-doors','body-style'])['horsepower'].transform('median')

# similar case imputation of the median
auto_df['horsepower'].fillna(power, inplace = True)

# grouping bore and getting the median
bore = auto_df.groupby(['compression-ratio'])['bore'].transform('median')

# similar case imputation of the median
auto_df['bore'].fillna(bore, inplace = True)

# grouping stroke and getting the median
stroke = auto_df.groupby(['compression-ratio'])['stroke'].transform('median')

# similar case imputation of the median
auto_df['stroke'].fillna(stroke, inplace = True)

# grouping normalized-losses and getting the median
norm_loss = auto_df.groupby('body-style')['normalized-losses'].transform('median')

# similar case imputation of the median
auto_df['normalized-losses'].fillna(norm_loss, inplace = True)
```

Imputation for 'num-of-doors' column

```
# imputation for 'num-of-doors' column
auto_df['num-of-doors'].fillna(auto_df['num-of-doors'].mode()[0], inplace = True)
```

Now that we have performed imputation for missing values in the columns with the visualisation below we can confirm that there is no missing data.



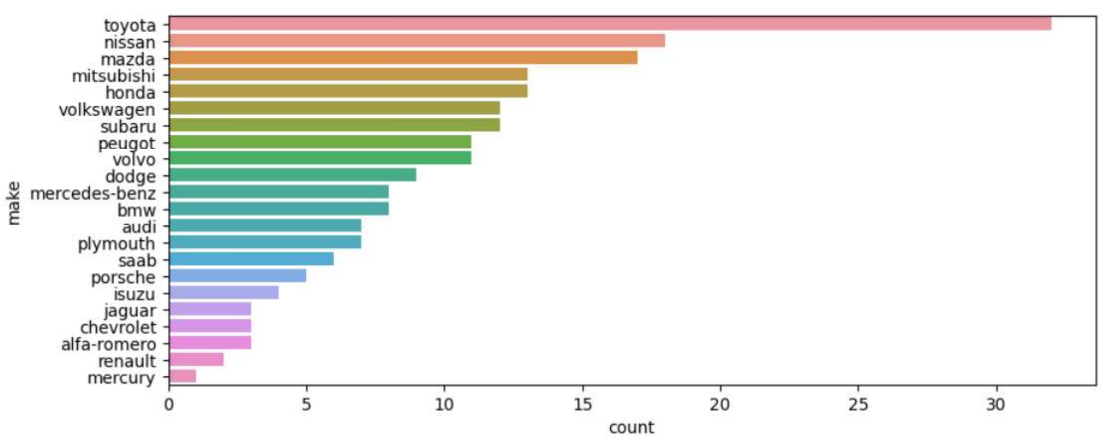
Now that the missing datapoints have been handled, we can move on to the next step which is the exploratory data analysis. From this we can hope to get some insights about this dataset.

DATA STORIES AND VISUALISATIONS

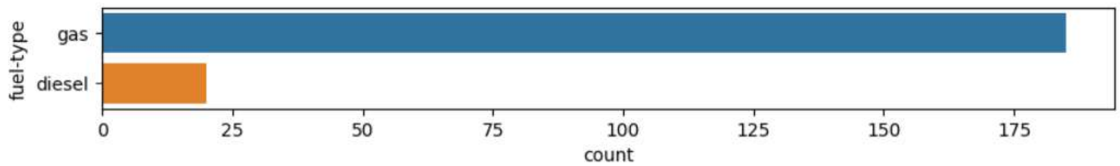
Before we begin with the exploration of the dataset, lets correct the datatype for the normalised-losses column in the dataset. Below is the python code that is used to convert the datatype from object to int.

```
# correcting datatype of 'normalized-losses'
auto_df[['normalized-losses']] = auto_df[['normalized-losses']].astype('int')
```

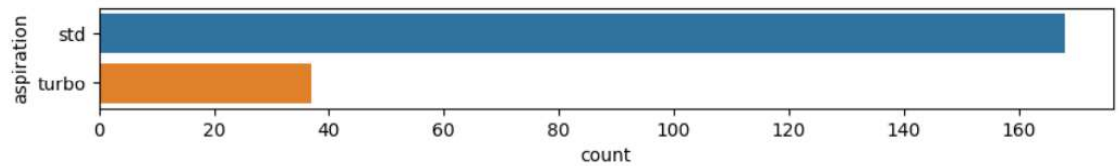
Now we can begin with the data exploration and see what insight we can exctract, and the first thing to do is to check the make distribution on the dataset, and we can see from the figure below that most of the data collected was from Toyota and the data was collected for twenty-two different types of car makes.



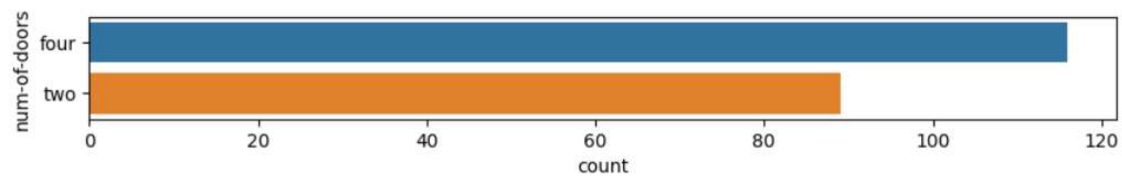
We can see in the figure below that for the cars that were collected a majority of them use gas for fuel while a very small number of them use diesel.



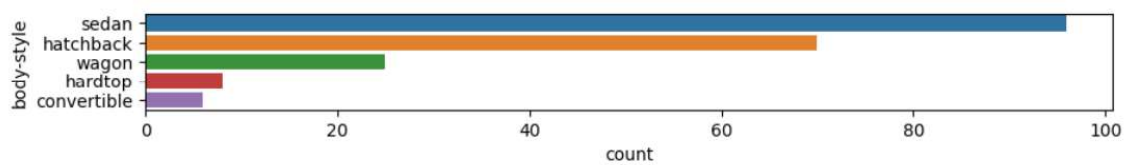
And a majority of the cars have a standard engine, and a small number uses turbo engines which gets additional compressed air from turbochargers for more power and enhanced efficiency.



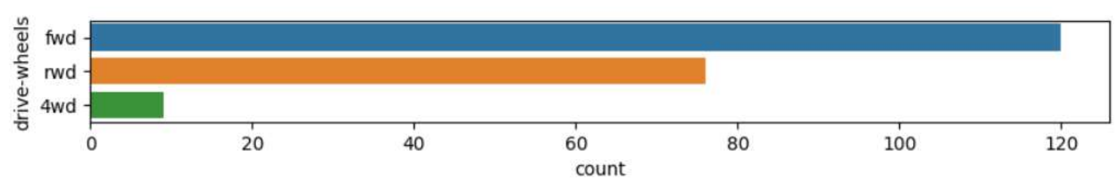
Majority of the cars have four doors but there isn't a significant difference in the number of four door and two door cars in the collected dataset.



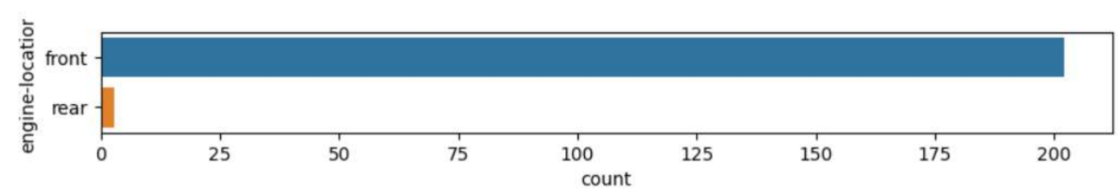
When it comes to the body types, we have five different body-style types for the cars with the sedan and hatchback making up majority of the cars.



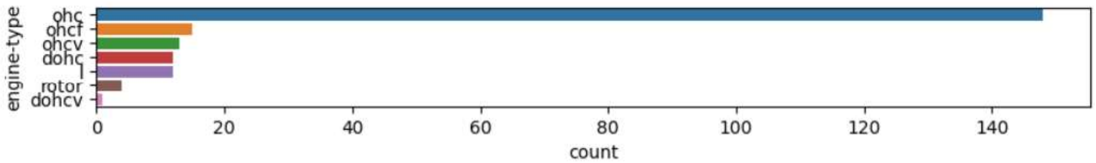
And we have three different types for drive wheel configurations with the majority of the cars using either a front-wheel drive or a rear-wheel drive configuration.



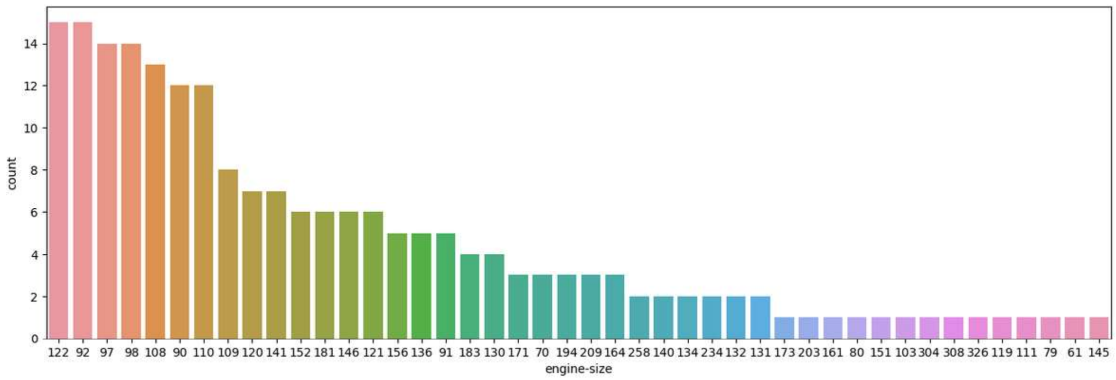
A significant number of the cars have a front located engine which are cars that are generally best for consumers while rear located engine cars offer unmatched acceleration.

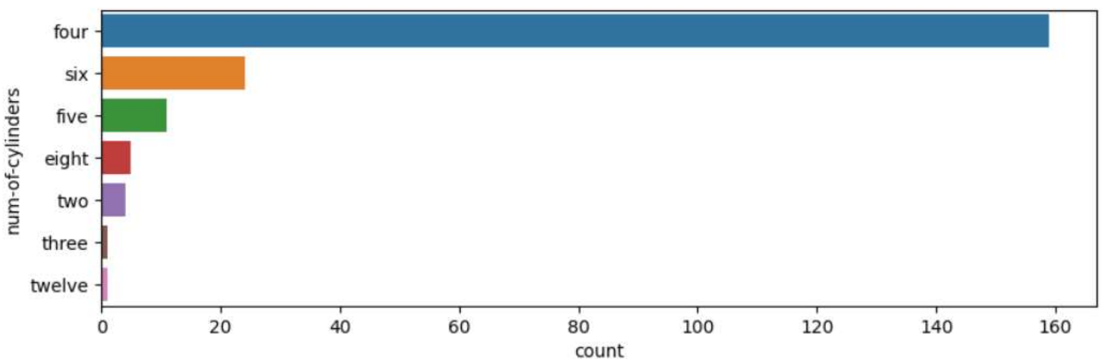


The cars have seven types of engines with the Overhead Camshaft being used in the majority of the cars.

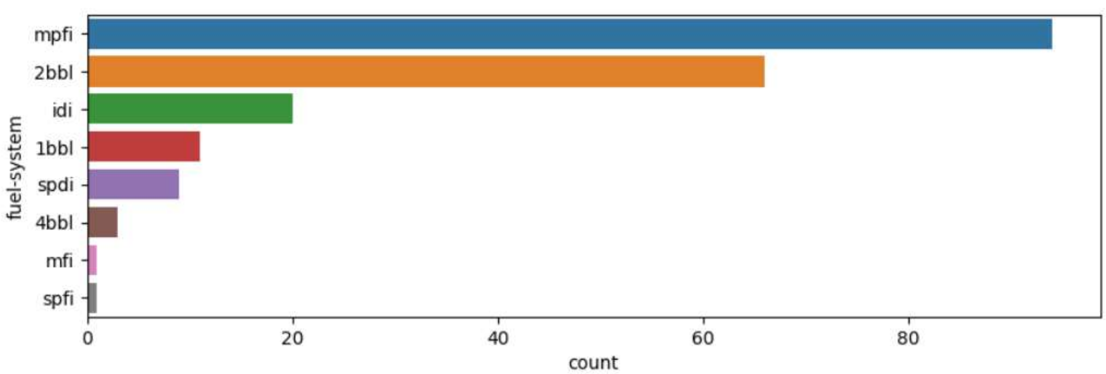


With the majority of the cars having engines of sizes 122, 98, 97, 92 and 108, and most of the cars have four-cylinder engines

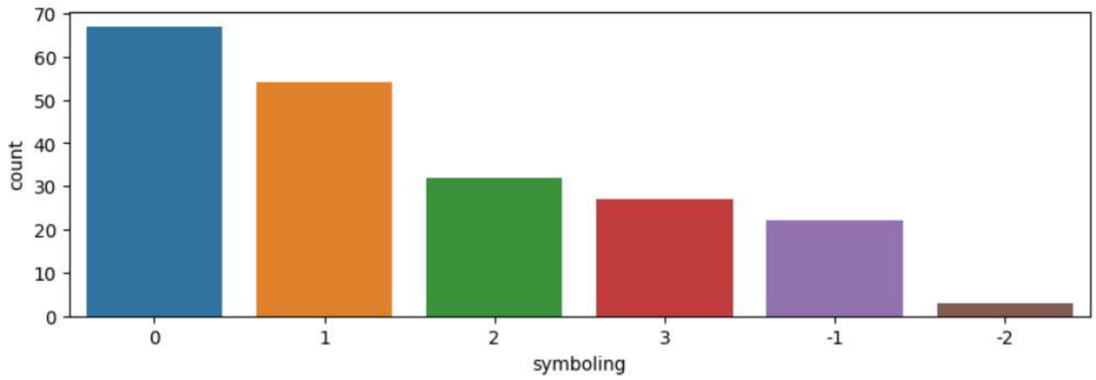




The cars have eight types of fuel systems with most cars having either a multi-point fuel injection, fuel system or a 2bbl fuel system.



Lastly let's look at symboling which is the indication of the cars risk relative to its price. A negative symboling value indicates that the car is less risky and most likely safe while the opposite is true for a positive symboling rate. Below we can see that only a few cars are considered pretty much safe while the majority have varying degrees of risk.



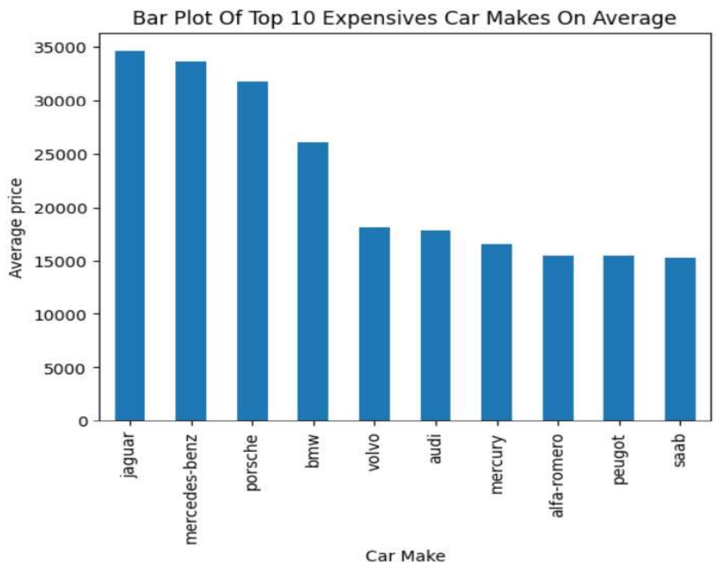
The above exploration was to give us an idea of what type of data we are working with. From here on I will try to analyse a few aspects of the columns in the dataset and the questions that I would like to answer are.

- Top 10 most expensive car makes on average?
- How does the engine size relate to price?
- price and horsepower relationship?
- How does peak-rpm affect horsepower?
- How does the engine size relate to horsepower and rpm?
- How does bore and stroke relate to compression ratio?
- Compression ratio and price relationship?
- Driving in the city vs the highway?

So, let's begin with the investigation of the above questions and see what insights and information we can be able to extract.

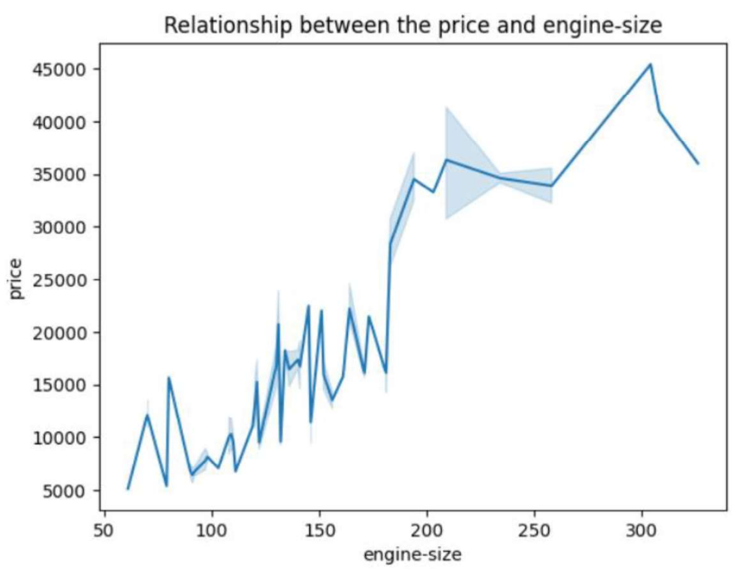
Top 10 most expensive car makes on average

From the figure below the Jaguar is the most expensive make on average. The top 5 most expensive makes on average include the Mercedes-benz, Porsche and BMW. The top also includes makes such as the Peugot, Alfa-romero and the Saab**.**



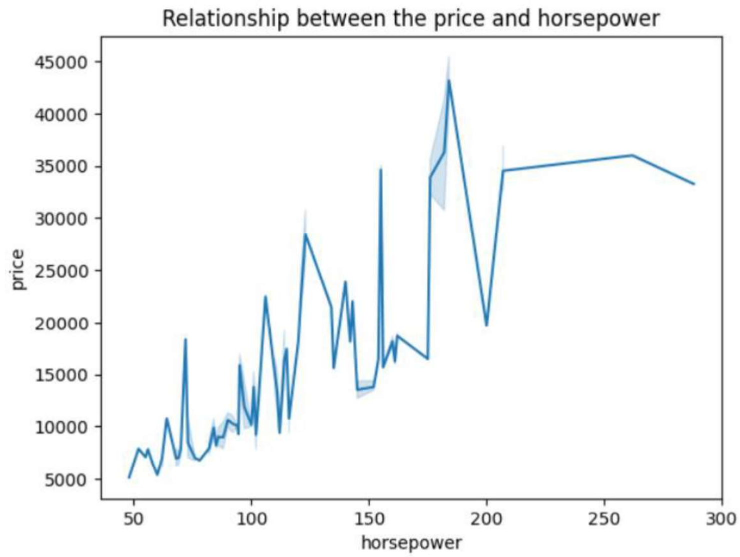
How does the engine-size relate to price?

From the lineplot figure below it can be concluded that the relationship has a positive increasing gradient, which means that in general the price of the car increases with the increase in the engine-size. The bigger the engine the more expensive the car.



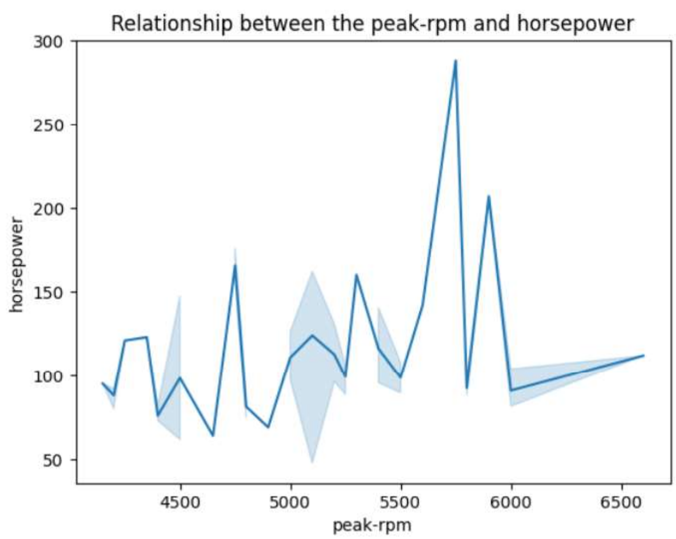
Price and horsepower relationship

From the figure below it can be concluded that a car with more horsepower will be a more expensive car . The relationship in the lineplot has positive gradient meaning more power results in a higher price.



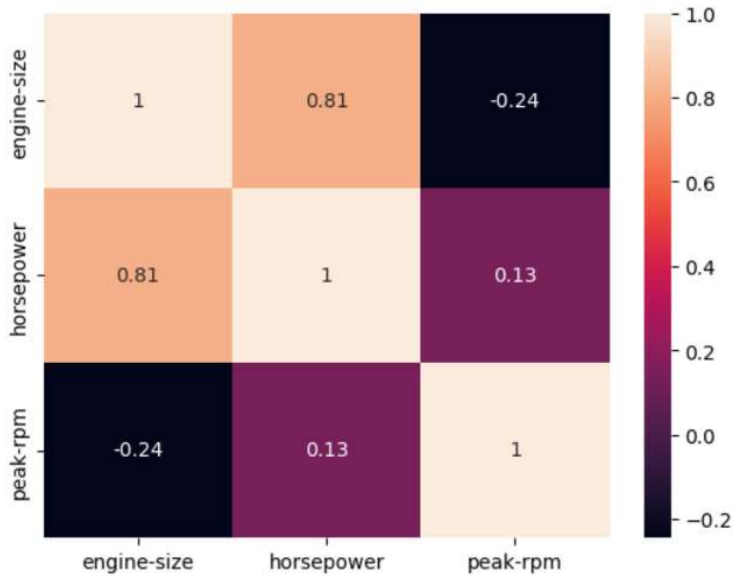
Peak-rpm and horsepower relationship

The relation between peak-rpm and the horsepower seems to have a random relationship (fluctuates), it cannot be concluded that with an increase in rpm that the results in horsepower will be favourable.



Relationship between engine-size, peak-rpm, and horsepower

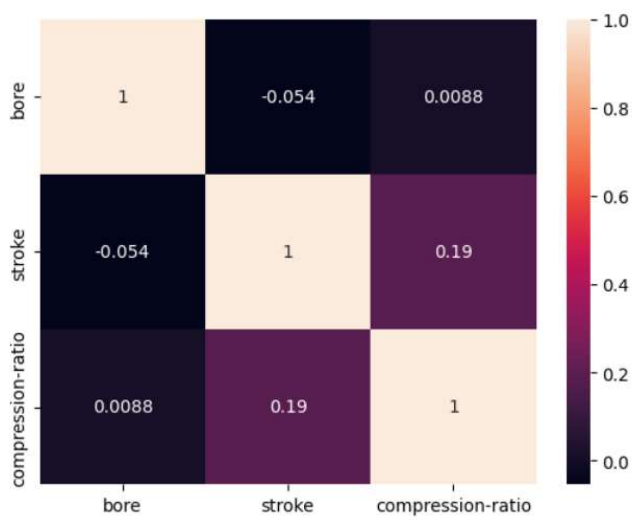
From the figure below it can be concluded that engine-size has a strong correlation with horsepower and thus we can conclude that with a bigger engine-size the horsepower will likely be more. But we can also conclude that the correlation between engine-size and peak-rpm is weak and in the negative direction.



Another thing to note is that the correlation between horsepower and peak-rpm is weak, which emphasises the finding from before, that with an increase in rpm the results in horsepower will be likely be unfavourable.

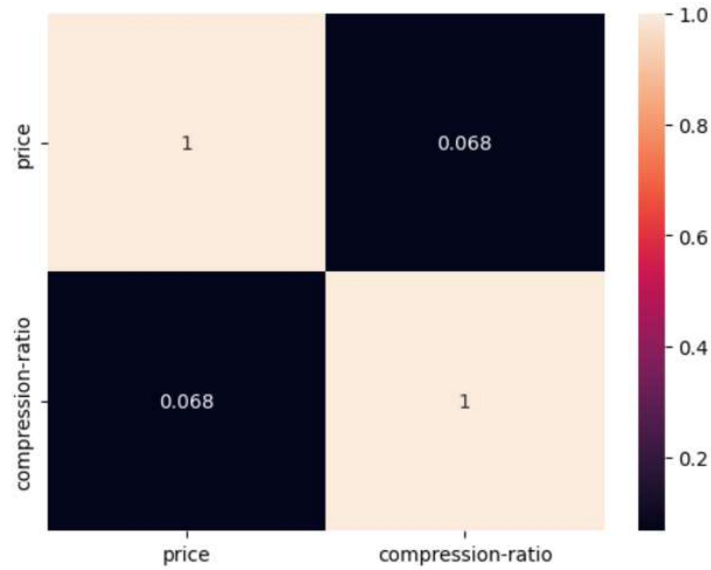
How does compression-ratio relate to bore and stroke

There is weak correlation between the compression-ratio and the stroke, and nearly no correlation between the compression-ratio and the bore of the car. The bore and stroke have a negative correlation which is also nearly zero, so these two variables also nearly have no relationship.



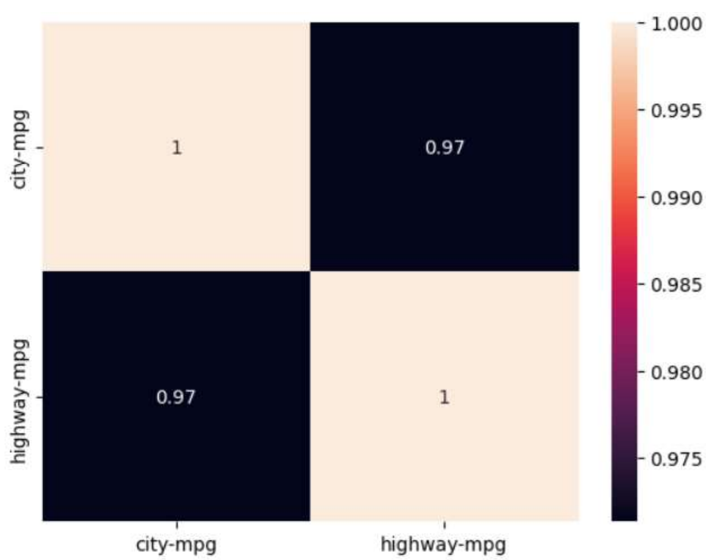
Relationship between price and compression-ratio

From the results below we can conclude that the compression-ratio of the car does not really affect the pricing. The correlation is positive, but we can also declare it as no correlation, as there is very little.



Driving in the city vs the highway?

From the figure below we can conclude that there is not much difference between driving in the city and driving in the highway. The correlation between the two variables is nearly perfect and hence the conclusion.



CONCLUSION

In conclusion, a lot of exploration can still be done on the dataset to extract more insights and information about the dataset. In the exploration and analysis that was concluded in this report, we learnt that with more horsepower the higher the price and the same is true for the engine-size of the car.

Driving in the city or the highway has little to an insignificant amount of difference, and that the compression-ratio of the car has little to nothing to do with the pricing. As stated before still a lot of exploration and analysis can be performed with more insights to be revealed.

THIS REPORT WAS WRITTEN BY : Thabiso Maqhajana