exploratory_data_analysis1

December 20, 2018

Link
Data Analysis with Python

1 Module 3: Exploratory Data Analysis

1.0.1 Welcome!

In this section, we will explore several methods to see if certain characteristics or features can be used to predict price.

1.0.2 What are the main characteristics which have the most impact on the car price?

1.1 1. Import Data from Module 2

Setup Import libraries:

```
In [1]: import pandas as pd
        import numpy as np
```

Load data and store in dataframe df:

```
In [2]: path='https://ibm.box.com/shared/static/q6iiqb1pd7wo8r3q28jvgsrprzezjqk3.csv'

        df = pd.read_csv(path)
        print('\n')
        print('file read')
        print('\n')
        df.head()
```

```
file read
```

```
Out[2]:    Unnamed: 0  symboling  normalized-losses         make aspiration  \
        0           0          3                122  alfa-romero        std
```

```
1           1          3                122  alfa-romero        std
2           2          1                122  alfa-romero        std
3           3          2                164         audi        std
4           4          2                164         audi        std

  num-of-doors   body-style drive-wheels engine-location  wheel-base ...   \
0          two  convertible          rwd           front        88.6 ...
1          two  convertible          rwd           front        88.6 ...
2          two    hatchback          rwd           front        94.5 ...
3         four        sedan          fwd           front        99.8 ...
4         four        sedan          4wd           front        99.4 ...

   compression-ratio  horsepower  peak-rpm  city-mpg highway-mpg    price  \
0                9.0       111.0    5000.0        21          27  13495.0
1                9.0       111.0    5000.0        21          27  16500.0
2                9.0       154.0    5000.0        19          26  16500.0
3               10.0       102.0    5500.0        24          30  13950.0
4                8.0       115.0    5500.0        18          22  17450.0

   city-L/100km horsepower-binned  diesel  gas
0     11.190476            Medium       0    1
1     11.190476            Medium       0    1
2     12.368421            Medium       0    1
3      9.791667            Medium       0    1
4     13.055556            Medium       0    1

[5 rows x 30 columns]
```

1.2 2. Analyzing Individual Feature Patterns using Visualization

Import visualization packages "Matplotlib" and "Seaborn". Don't forget about "%matplotlib inline" to plot in a Jupyter notebook:

```
In [3]: import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline


        ---------------------------------------------------------------------------

        ModuleNotFoundError                       Traceback (most recent call last)

        <ipython-input-3-bae5756cecfc> in <module>
          1 import matplotlib.pyplot as plt
    ----> 2 import seaborn as sns
          3 get_ipython().run_line_magic('matplotlib', 'inline ')
```

```
ModuleNotFoundError: No module named 'seaborn'
```

1.2.1 How to choose the right visualization method:

When visualizing individual variables, it is important to first understand what type of variable you are dealing with. This will help us find the right visualisation method for that variable.

```
In [ ]: # list the data types for each column
        df.dtypes
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Question #1:
What is the data type of the column "peak-rpm"?
Click here for the solution

```
float64
```

For example, we can calculate the correlation between variables of type "int64" or "float64" using the method "corr":

```
In [ ]: float64
```

```
In [ ]: df.corr()
```

The diagonal elements are always one. We will study correlation, more precisely Pearson correlation, in-depth at the end of the notebook.

Question #2:

Find the correlation between the following columns: bore, stroke, compression-ratio, and horsepower. Hint: if you would like to select those columns use the following syntax: df[['bore','stroke' ,'compression-ratio','horsepower']]:

```
In [ ]: df[['bore', 'stroke','compression-ratio', 'horsepower']].corr()
```

Click here for the solution

```
df[['bore','stroke' ,'compression-ratio','horsepower']].corr()
```

1.3 Continuous numerical variables:

Continuous numerical variables are variables that may contain any value within some range. Continuous numerical variables can have the type "int64" or "float64". A great way to visualize these variables is by using scatterplots with fitted lines.

In order to start understanding the (linear) relationship between an individual variable and the price, we can use "regplot", which plots the scatterplot plus the fitted regression line for the data.

Let's see several examples of different linear relationships:

Positive linear relationship Let's find the scatterplot of "engine-size" and "price":

```
In [ ]: # Engine size as potential predictor variable of price
        sns.regplot(x="engine-size", y="price", data=df)
        plt.ylim(0,)
```

As the engine-size goes up, the price goes up: this indicates a positive direct correlation between these two variables. Engine size seems like a pretty good predictor of price since the regression line is almost a perfect diagonal line. E

We can examine the correlation between 'engine-size' and 'price' and see it's approximately 0.87:

```
In [ ]: df[["engine-size", "price"]].corr()
```

1.3.1 Negative linear relationship

Highway mpg is a potential predictor variable of price:

```
In [ ]: sns.regplot(x="highway-mpg", y="price", data=df)
```

As the highway-mpg goes up, the price goes down: this indicates an inverse/negative relationship between these two variables. Highway mpg could potentially be a predictor of price.

We can examine the correlation between 'highway-mpg' and 'price' and see it's approximately -0.704:

```
In [ ]: df[['highway-mpg', 'price']].corr()
```

1.3.2 Weak Linear Relationship

Let's see if "Peak-rpm" as a predictor variable of "price":

```
In [ ]: sns.regplot(x="peak-rpm", y="price", data=df)
```

Peak rpm does not seem like a good predictor of the price at all since the regression line is close to horizontal. Also, the data points are very scattered and far from the fitted line, showing lots of variability. Therefore it is not a reliable variable.

We can examine the correlation between 'peak-rpm' and 'price'and see it is approximately -0.101616:

```
In [ ]: df[['peak-rpm','price']].corr()
```

Question 3 a):

Find the correlation between x="stroke", y="price". Hint: if you would like to select those columns use the following syntax: df[["stroke","price"]]:

```
In [ ]: df[['stroke', 'price']].corr()
```

Click here for the solution

```
The correlation is 0.0823, the non-diagonal elements of the table.
code:df[["stroke","price"]].corr()
```

Question 3 b):

Given the correlation results between "price" and "stroke", do you expect a linear relationship? Verify your results using the function "regplot()":

```
In [ ]: sns.regplot(x='stroke', y='price', data=df)
```

Click here for the solution

```
There is a weak correlation between the variable 'stroke' and 'price.' as such regression will n

Code: sns.regplot(x="stroke", y="price", data=df)
```

1.4 Categorical variables

These are variables that describe a 'characteristic' of a data unit, and are selected from a small group of categories. The categorical variables can have the type "object" or "int64". A good way to visualize categorical variables is by using boxplots.

Let's look at the relationship between "body-style" and "price":

```
In [ ]: sns.boxplot(x="body-style", y="price", data=df)
```

We see that the distributions of price between the different body-style categories have a significant overlap, and so body-style would not be a good predictor of price. Let's examine engine "engine-location" and "price" :

```
In [ ]: sns.boxplot(x="engine-location", y="price", data=df)
```

Here we see that the distribution of price between these two engine-location categories, front and rear, are distinct enough to take engine-location as a potential good predictor of price.

Let's examine "drive-wheels" and "price":

```
In [ ]: # drive-wheels
        sns.boxplot(x="drive-wheels", y="price", data=df)
```

Here we see that the distribution of price between the different drive-wheels categories differs. As such, drive-wheels could potentially be a predictor of price.

1.5 3. Descriptive Statistical Analysis

Let's first take a look at the variables by utilising a description method.

The **describe** function automatically computes basic statistics for all continuous variables. Any NaN values are automatically skipped in these statistics.

This will show: - the count of that variable - the mean - the standard deviation (std) - the minimum value - the IQR (Interquartile Range: 25%, 50% and 75%) - the maximum value

We can apply the method "describe" as follows:

```
In [ ]: df.describe()
```

The default setting of "describe" skips variables of type object. We can apply the method "describe" on the variables of type 'object' as follows:

```
In [ ]: df.describe(include=['object'])
```

1.5.1 Value Counts

Value-counts is a good way of understanding how many units of each characteristic/variable we have. We can apply the "value_counts" method on the column 'drive-wheels'. Don't forget the method "value_counts" only works on Pandas series, not Pandas Dataframes. As a result, we only include one bracket "df['drive-wheels']", not two "df[['drive-wheels']]".

```
In [ ]: df['drive-wheels'].value_counts()
```

We can convert the series to a Dataframe as follows :

```
In [ ]: df['drive-wheels'].value_counts().to_frame()
```

Let's repeat the above steps but save the results to the dataframe "drive_wheels_counts" and rename the column 'drive-wheels' to 'value_counts':

```
In [ ]: drive_wheels_counts = df['drive-wheels'].value_counts().to_frame()
        drive_wheels_counts.rename(columns={'drive-wheels': 'value_counts'}, inplace=True)
        drive_wheels_counts
```

Now let's rename the index to 'drive-wheels':

```
In [ ]: drive_wheels_counts.index.name = 'drive-wheels'
        drive_wheels_counts
```

We can repeat the above process for the variable 'engine-location':

```
In [ ]: # engine-location as variable
        engine_loc_counts = df['engine-location'].value_counts().to_frame()
        engine_loc_counts.rename(columns={'engine-location': 'value_counts'}, inplace=True)
        engine_loc_counts.index.name = 'engine-location'
        engine_loc_counts.head(10)
```

Examining the value counts of the engine location would not be a good predictor variable for the price. This is because we only have three cars with a rear engine and 198 with an engine in the front, creating a skewed result. Thus, we are not able to draw any conclusions about the engine location.

1.6 4. Basic of Grouping

The "groupby" method groups data by different categories. The data is grouped based on one or several variables, and analysis is performed on the individual groups.

For example, let's group by the variable "drive-wheels". We see that there are 3 different categories of drive wheels:

```
In [ ]: df['drive-wheels'].unique()
```

If we want to know, on average, which type of drive wheel is most valuable, we can group "drive-wheels" and then average them.

We can select the columns 'drive-wheels','body-style', and 'price', then assign it to the variable "df_group_one".

```
In [ ]: df_group_one=df[['drive-wheels','body-style','price']]
```

We can then calculate the average price for each of the different categories of data:

```
In [ ]: # grouping results

        df_group_one=df_group_one.groupby(['drive-wheels'],as_index= False).mean()
        df_group_one
```

From our data, it seems rear-wheel drive vehicles are, on average, the most expensive, while 4-wheel and front-wheel are approximately the same in price.

You can also group with multiple variables. For example, let's group by both 'drive-wheels' and 'body-style'. This groups the dataframe by the unique combinations 'drive-wheels' and 'body-style'. We can store the results in the variable 'grouped_test1':

```
In [ ]: # grouping results
        df_gptest=df[['drive-wheels','body-style','price']]
        grouped_test1=df_gptest.groupby(['drive-wheels','body-style'],as_index= False).mean()
        grouped_test1
```

This grouped data is much easier to visualize when it is made into a pivot table. A pivot table is like an Excel spreadsheet, with one variable along the column and another along the row. We can convert the dataframe to a pivot table using the method "pivot " to create a pivot table from the groups.

In this case, we will leave the drive-wheel variable as the rows of the table, and pivot body-style to become the columns of the table:

```
In [ ]: grouped_pivot=grouped_test1.pivot(index='drive-wheels',columns='body-style')
        grouped_pivot
```

Often, we won't have data for some of the pivot cells. We can fill these missing cells with the value 0, but any other value could potentially be used as well. It should be mentioned that missing data is quite a complex subject and is an entire course on its own.

```
In [ ]: grouped_pivot=grouped_pivot.fillna(0) #fill missing values with 0
        grouped_pivot
```

Question 4 :

Use the "groupby" function to find the average "price" of each car based on "body-style":

```
In [ ]: df_body=df[['body-style','price']]
        grouped_test2=df_body.groupby(['body-style'],as_index= False).mean()
        grouped_test2
```

If you didn't import "pyplot", let's do it again:

```
In [ ]: import matplotlib.pyplot as plt
        % matplotlib inline
```

Variables: Drive Wheels and Body Style vs Price Let's use a heat map to visualize the relationship between Body Style vs Price:

```
In [ ]: #use the grouped results
        plt.pcolor(grouped_pivot, cmap='RdBu')
        plt.colorbar()
        plt.show()
```

The heatmap plots the target variable (price) proportional to colour with respect to the variables 'drive-wheel' and 'body-style' in the vertical and horizontal axis, respectively. This allows us to visualize how the price is related to 'drive-wheel' and 'body-style'. The default labels convey no useful information to us. Let's change that:

```
In [ ]: fig, ax=plt.subplots()
        im=ax.pcolor(grouped_pivot, cmap='RdBu')

        #label names
        row_labels=grouped_pivot.columns.levels[1]
        col_labels=grouped_pivot.index
        #move ticks and labels to the center
        ax.set_xticks(np.arange(grouped_pivot.shape[1])+0.5, minor=False)
        ax.set_yticks(np.arange(grouped_pivot.shape[0])+0.5, minor=False)
        #insert labels
        ax.set_xticklabels(row_labels, minor=False)
        ax.set_yticklabels(col_labels, minor=False)
        #rotate label if too long
        plt.xticks(rotation=90)

        fig.colorbar(im)
        plt.show()
```

Visualization is very important in data science, and Python visualization packages provide great freedom. We will go more in-depth in a separate Python Visualizations course.

The main question we want to answer in this module is, "What are the main characteristics which have the most impact on the car price?".

To get a better measure of the important characteristics, we look at the correlation of these variables with the car price. In other words, how is the car price dependent on this variable?

1.7 5. Correlation and Causation

Correlation: a measure of the extent of interdependence between variables.

Causation: the relationship between cause and effect between two variables.

It is important to know the difference between these two and that correlation does not imply causation. Determining correlation is much simpler than determining causation, as causation may require independent experimentation.

1.8 Pearson Correlation

The Pearson Correlation measures the linear dependence between two variables, X and Y. The resulting coefficient is a value between -1 and 1 inclusive, where: - **1**: total positive linear correlation,

- **0**: no linear correlation, the two variables most likely do not affect each other - **-1**: total negative linear correlation.

Pearson Correlation is the default method of the function "corr". As before, we can calculate the Pearson correlation of the of the 'int64' or 'float64' variables:

```
In [ ]: df.corr()
```

Sometimes we would like to know the significance of the correlation estimate.

P-value: What is this P-value? The P-value is the probability value that the correlation between these two variables is statistically significant. Normally, we choose a significance level of 0.05, which means that we are 95% confident that the correlation between the variables is significant.

By convention, when the p-value is: - < 0.001 we say there is strong evidence that the correlation is significant, - < 0.05; there is moderate evidence that the correlation is significant, - < 0.1; there is weak evidence that the correlation is significant, and - is > 0.1; there is no evidence that the correlation is significant.

We can obtain this information using "stats" module in the "scipy" library:

```
In [ ]: from scipy import stats
```

1.8.1 Wheel-base vs Price

Let's calculate the Pearson Correlation Coefficient and P-value of 'wheel-base' and 'price':

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])
        print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between wheel-base and price is statistically significant, although the linear relationship isn't extremely strong (~0.585).

1.8.2 Horsepower vs Price

Let's calculate the Pearson Correlation Coefficient and P-value of 'horsepower' and 'price':

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
        print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between horsepower and price is statistically significant, and the linear relationship is quite strong (~0.809, close to 1).

1.8.3 Length vs Price

Let's calculate the Pearson Correlation Coefficient and P-value of 'length' and 'price':

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['length'], df['price'])
        print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between length and price is statistically significant, and the linear relationship is moderately strong (~0.691).

1.8.4 Width vs Price

Let's calculate the Pearson Correlation Coefficient and P-value of 'width' and 'price':

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['width'], df['price'])
        print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between width and price is statistically significant, and the linear relationship is quite strong (~0.751).

1.8.5 Curb-weight vs Price

Let's calculate the Pearson Correlation Coefficient and P-value of 'curb-weight' and 'price':

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['curb-weight'], df['price'])
        print( "The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between curb-weight and price is statistically significant, and the linear relationship is quite strong (~0.834).

1.8.6 Engine-size vs Price

Let's calculate the Pearson Correlation Coefficient and P-value of 'engine-size' and 'price':

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['engine-size'], df['price'])
        print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between engine-size and price is statistically significant, and the linear relationship is very strong (~0.872).

1.8.7 Bore vs Price

Let's calculate the Pearson Correlation Coefficient and P-value of 'bore' and 'price':

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['bore'], df['price'])
        print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between bore and price is statistically significant, but the linear relationship is only moderate (~0.521).

We can relate the process for each 'City-mpg' and 'Highway-mpg':

1.8.8 City-mpg vs Price

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['city-mpg'], df['price'])
        print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between city-mpg and price is statistically significant, and the coefficient of ~ -0.687 shows that the relationship is negative and moderately strong.

1.8.9 Highway-mpg vs Price

```
In [ ]: pearson_coef, p_value = stats.pearsonr(df['highway-mpg'], df['price'])
        print( "The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =",
```

Conclusion: Since the p-value is < 0.001, the correlation between highway-mpg and price is statistically significant, and the coefficient of ~ -0.705 shows that the relationship is negative and moderately strong.

1.9 6. ANOVA

1.9.1 ANOVA: Analysis of Variance

The Analysis of Variance (ANOVA) is a statistical method used to test whether there are significant differences between the means of two or more groups. ANOVA returns two parameters:

F-test score: ANOVA assumes the means of all groups are the same, calculates how much the actual means deviate from the assumption, and reports it as the F-test score. A larger score means there is a larger difference between the means.

P-value: P-value tells us the statistical significance of our calculated score value.

If our price variable is strongly correlated with the variable we are analyzing, expect ANOVA to return a sizeable F-test score and a small p-value.

1.9.2 Drive Wheels

Since ANOVA analyzes the difference between different groups of the same variable, the groupby function will come in handy. Because the ANOVA algorithm averages the data automatically, we do not need to take the average before-hand.

Let's see if different types 'drive-wheels' impact 'price'. We group the data:

```
In [ ]: grouped_test2=df_gptest[['drive-wheels','price']].groupby(['drive-wheels'])
        grouped_test2.head(2)
```

We can obtain the values of the method group using the method "get_group":

```
In [ ]: grouped_test2.get_group('4wd')['price']
```

We can use the function 'f_oneway' in the module 'stats' to obtain the **F-test score** and **P-value**:

```
In [ ]: # ANOVA
        f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'], grouped_test2.get

        print( "ANOVA results: F=", f_val, ", P =", p_val)
```

This is a great result, with a large F test score showing a strong correlation and a P value of almost 0, implying almost certain statistical significance. But does this mean all three tested groups are all this highly correlated?

Separately: fwd and rwd:

```
In [ ]: f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'], grouped_test2.get

        print( "ANOVA results: F=", f_val, ", P =", p_val )
```

Let's examine the other groups

4wd and rwd:

```
In [ ]: f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'], grouped_test2.get

        print( "ANOVA results: F=", f_val, ", P =", p_val)
```

4wd and fwd:

```
In [ ]: f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'], grouped_test2.get

        print("ANOVA results: F=", f_val, ", P =", p_val)
```

1.10 Conclusion: Important Variables

We now have a better idea of what our data looks like and which variables are important to take into account when predicting the car price. We have narrowed it down to the following variables:

Continuous numerical variables: - Length - Width - Curb-weight - Engine-size - Horsepower - City-mpg - Highway-mpg - Wheel-base - Bore

Categorical variables: - Drive-wheels

As we now move into building machine learning models to automate our analysis, feeding the model with variables that meaningfully affect our target variable will improve our model's prediction performance.

2 About the Authors:

This notebook written by Mahdi Noorian PhD ,Joseph Santarcangelo PhD, Bahare Talayian, Eric Xiao, Steven Dong, Parizad , Hima Vsudevan and Fiorella Wenver.

Link

```
In [ ]:
```