**Data Analysis with Python**[**¶**](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/lab/tree/labs/DA0101EN/exploratory-data-analysis.ipynb#Data-Analysis-with-Python)

Estimated time needed: **30** minutes

**Objectives**

After completing this lab you will be able to:

* Explore features or charecteristics to predict price of car

**Table of Contents**

1. [Import Data from Module](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/DA0101EN/https%3A/%23import_data?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
2. [Analyzing Individual Feature Patterns using Visualization](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/DA0101EN/https%3A/%23pattern_visualization?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
3. [Descriptive Statistical Analysis](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/DA0101EN/https%3A/%23discriptive_statistics?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
4. [Basics of Grouping](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/DA0101EN/https%3A/%23basic_grouping?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
5. [Correlation and Causation](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/DA0101EN/https%3A/%23correlation_causation?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
6. [ANOVA](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/DA0101EN/https%3A/%23anova?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)

**What are the main characteristics that have the most impact on the car price?**

**1. Import Data from Module 2**

**Setup**

Import libraries:

[ ]:



**import** pandas **as** pd

**import** numpy **as** np

Load the data and store it in dataframe df:

This dataset was hosted on IBM Cloud object. Click [HERE](https://cocl.us/DA101EN_object_storage?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDA0101ENSkillsNetwork20235326-2021-01-01) for free storage.

[ ]:



path**=**'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/automobileEDA.csv'

df **=** pd.read\_csv(path)

df.head()

**2. Analyzing Individual Feature Patterns Using Visualization**

To install Seaborn we use pip, the Python package manager.

[ ]:



**%%**capture

**!** pip install seaborn

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

[ ]:



**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**%**matplotlib inline

**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 visualization method for that variable.

[ ]:



*# list the data types for each column*

print(df.dtypes)

**Question #1:**

**What is the data type of the column "peak-rpm"?**

[ ]:



*# Write your code below and press Shift+Enter to execute*

​

Click here for the solution

float64

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

[ ]:



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']]

[ ]:



*# Write your code below and press Shift+Enter to execute*

​

Click here for the solution

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

**Continuous Numerical Variables:**

Continuous numerical variables are variables that may contain any value within some range. They can be of 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".

[ ]:



*# 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.

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

[ ]:



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

Highway mpg is a potential predictor variable of price. Let's find the scatterplot of "highway-mpg" and "price".

[ ]:



sns.regplot(x**=**"highway-mpg", y**=**"price", data**=**df)

As 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.

[ ]:



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

**Weak Linear Relationship**

Let's see if "peak-rpm" is a predictor variable of "price".

[ ]:



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's not a reliable variable.

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

[ ]:



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

**Question 3 a):**

Find the correlation between x="stroke" and y="price".

Hint: if you would like to select those columns, use the following syntax: df[["stroke","price"]].

[ ]:



*# Write your code below and press Shift+Enter to execute*

​

Click here for the solution

*#The correlation is 0.0823, the non-diagonal elements of the table.*

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()".

[ ]:



*# Write your code below and press Shift+Enter to execute*

​

Click here for the solution

*#There is a weak correlation between the variable 'stroke' and 'price.' as such regression will not work well. We can see this using "regplot" to demonstrate this.*

*#Code:*

sns.regplot(x**=**"stroke", y**=**"price", data**=**df)

**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".

[ ]:



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, so body-style would not be a good predictor of price. Let's examine engine "engine-location" and "price":

[ ]:



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".

[ ]:



*# 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.

**3. Descriptive Statistical Analysis**

Let's first take a look at the variables by utilizing 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:

[ ]:



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:

[ ]:



df.describe(include**=**['object'])

**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 brackets df[['drive-wheels']].

[ ]:



df['drive-wheels'].value\_counts()

We can convert the series to a dataframe as follows:

[ ]:



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'.

[ ]:



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':

[ ]:



drive\_wheels\_counts.index.name **=** 'drive-wheels'

drive\_wheels\_counts

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

[ ]:



*# 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)

After examining the value counts of the engine location, we see that 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, so this result is skewed. Thus, we are not able to draw any conclusions about the engine location.

**4. Basics 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.

[ ]:



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".

[ ]:



df\_group\_one **=** df[['drive-wheels','body-style','price']]

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

[ ]:



*# 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 by multiple variables. For example, let's group by both 'drive-wheels' and 'body-style'. This groups the dataframe by the unique combination of 'drive-wheels' and 'body-style'. We can store the results in the variable 'grouped\_test1'.

[ ]:



*# 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-wheels variable as the rows of the table, and pivot body-style to become the columns of the table:

[ ]:



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.

[ ]:



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".

[ ]:



*# Write your code below and press Shift+Enter to execute*

​

Click here for the solution

*# grouping results*

df\_gptest2 **=** df[['body-style','price']]

grouped\_test\_bodystyle **=** df\_gptest2.groupby(['body-style'],as\_index**=** **False**).mean()

grouped\_test\_bodystyle

If you did not import "pyplot", let's do it again.

[ ]:



**import** matplotlib.pyplot **as** plt

**%**matplotlib inline

**Variables: Drive Wheels and Body Style vs. Price**

**Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in.**[**Learn more here.**](https://cocl.us/ibm_watson_studio_infobox)

Let's use a heat map to visualize the relationship between Body Style vs Price.

[ ]:



*#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' on 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:

[ ]:



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?

**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. Correlation does not imply causation. Determining correlation is much simpler the determining causation as causation may require independent experimentation.

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

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

[ ]:



df.corr()

Sometimes we would like to know the significant 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.
* the p-value is << 0.05: there is moderate evidence that the correlation is significant.
* the p-value is << 0.1: there is weak evidence that the correlation is significant.
* the p-value is >> 0.1: there is no evidence that the correlation is significant.

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

[ ]:



**from** scipy **import** stats

**Wheel-Base vs. Price**

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

[ ]:



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 =", p\_value)

**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).

**Horsepower vs. Price**

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

[ ]:



pearson\_coef, p\_value **=** stats.pearsonr(df['horsepower'], df['price'])

print("The Pearson Correlation Coefficient is", pearson\_coef, " with a P-value of P = ", p\_value)

**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).

**Length vs. Price**

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

[ ]:



pearson\_coef, p\_value **=** stats.pearsonr(df['length'], df['price'])

print("The Pearson Correlation Coefficient is", pearson\_coef, " with a P-value of P = ", p\_value)

**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).

**Width vs. Price**

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

[ ]:



pearson\_coef, p\_value **=** stats.pearsonr(df['width'], df['price'])

print("The Pearson Correlation Coefficient is", pearson\_coef, " with a P-value of P =", p\_value )

**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).

**Curb-Weight vs. Price**

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

[ ]:



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 = ", p\_value)

**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).

**Engine-Size vs. Price**

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

[ ]:



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 =", p\_value)

**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).

**Bore vs. Price**

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

[ ]:



pearson\_coef, p\_value **=** stats.pearsonr(df['bore'], df['price'])

print("The Pearson Correlation Coefficient is", pearson\_coef, " with a P-value of P = ", p\_value )

**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':

**City-mpg vs. Price**

[ ]:



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 = ", p\_value)

**Conclusion:**

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

**Highway-mpg vs. Price**

[ ]:



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 = ", p\_value )

**Conclusion:**

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

**6. ANOVA**

**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 how statistically significant our calculated score value is.

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

**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.

To see if different types of 'drive-wheels' impact 'price', we group the data.

[ ]:



grouped\_test2**=**df\_gptest[['drive-wheels', 'price']].groupby(['drive-wheels'])

grouped\_test2.head(2)

[ ]:



df\_gptest

We can obtain the values of the method group using the method "get\_group".

[ ]:



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**.

[ ]:



*# ANOVA*

f\_val, p\_val **=** stats.f\_oneway(grouped\_test2.get\_group('fwd')['price'], grouped\_test2.get\_group('rwd')['price'], grouped\_test2.get\_group('4wd')['price'])

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?

Let's examine them separately.

**fwd and rwd**

[ ]:



f\_val, p\_val **=** stats.f\_oneway(grouped\_test2.get\_group('fwd')['price'], grouped\_test2.get\_group('rwd')['price'])

print( "ANOVA results: F=", f\_val, ", P =", p\_val )

Let's examine the other groups.

**4wd and rwd**

[ ]:



f\_val, p\_val **=** stats.f\_oneway(grouped\_test2.get\_group('4wd')['price'], grouped\_test2.get\_group('rwd')['price'])

print( "ANOVA results: F=", f\_val, ", P =", p\_val)

**4wd and fwd**

[ ]:



f\_val, p\_val **=** stats.f\_oneway(grouped\_test2.get\_group('4wd')['price'], grouped\_test2.get\_group('fwd')['price'])

print("ANOVA results: F=", f\_val, ", P =", p\_val)

**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.