What is Exploratory Data Analysis?

Exploratory Data Analysis (EDA), also known as Data Exploration, is a step in the Data Analysis Process, where a number of techniques are used to better understand the dataset being used.

'Understanding the dataset' can refer to a number of things including but not limited to…

- Extracting important variables and leaving behind useless variables
- Identifying outliers, missing values, or human error
- Understanding the relationship(s), or lack of, between variables

Ultimately, maximizing your insights of a dataset and minimizing potential error that may occur later in the process

Here's why this is important.

Have you heard of the phrase, "garbage in, garbage out"?

With EDA, it's more like, "garbage in, perform EDA, possibly garbage out."

Exploratory Data Analysis does two main things:

- It helps clean up a dataset.
- It gives you a better understanding of the variables and the relationships between them.

Importing library

```
import numpy as np

import pandas as pd

import matplotlib.pylab as plt

import seaborn as sns   #Understanding my variables
```

As an example, I used the dataset of Used Car Dataset. First, I imported all of the libraries that I knew I'd need for my analysis and conducted some preliminary analyses.

Summery statistics

By using **df.shape**

.shape returns the number of rows by the number of columns for my dataset. My output was (525839, 22), meaning the dataset has 525839 rows and 22 columns.

.head() returns the first 5 rows of my dataset. This is useful if you want to see some example values for each variable.

.columns returns the name of all of your columns in the dataset.

.nunique(axis=0) returns the number of unique values for each variable.

.describe() summarizes the count, mean, standard deviation, min, and max for numeric variables. The code that follows this simply formats each row to the regular format and suppresses scientific notation

	price	year	odometer	lat	long
count	525839.000000	524399.000000	427248.000000	513618.000000	513618.000000
mean	61966.045503	2009.375184	101150.248338	38.470853	-94.259216
std	9949703.964406	8.975889	105525.184435	5.895637	17.742010
min	0.000000	1900.000000	0.000000	-83.668334	-176.748047
25%	3900.000000	2006.000000	49009.750000	34.669400	-108.386684
50%	8999.000000	2011.000000	94240.000000	39.213214	-88.544050
75%	17900.000000	2015.000000	138000.000000	42.445565	-80.990754
max	3048344231.000000	2020.000000	10000000.000000	78.928357	132.078349

Immediately, I noticed an issue with price, year, and odometer. For example, the minimum and maximum price are $0.00 and $3,048,344,231.00 respectively. You'll see how I dealt with this in the next section. I still wanted to get a better understanding of my discrete variables.

Using **.unique()**, I took a look at my discrete variables, including 'condition'. Also there is lots of missing value in that column so, I dropped that column.

2. Cleaning dataset

I now know how to reclassify discrete data if needed, but there are a number of things that still need to be looked at.

a. Removing Redundant variables

First I got rid of variables that I thought were redundant. This includes url, image_url, and city_url.

b. Variable Selection

Next, I wanted to get rid of any columns that had too many null values. I removed any columns that had 40% or more of its data as null values. Depending on the situation, I may want to increase or decrease the threshold. The remaining columns are shown bellow.

c. Removing Outliers

Revisiting the issue previously addressed, I set parameters for price, year, and odometer to remove any values outside of the set boundaries. In this case, I used my intuition to determine parameters — I'm sure there are methods to determine the optimal boundaries, but I haven't looked into it yet!

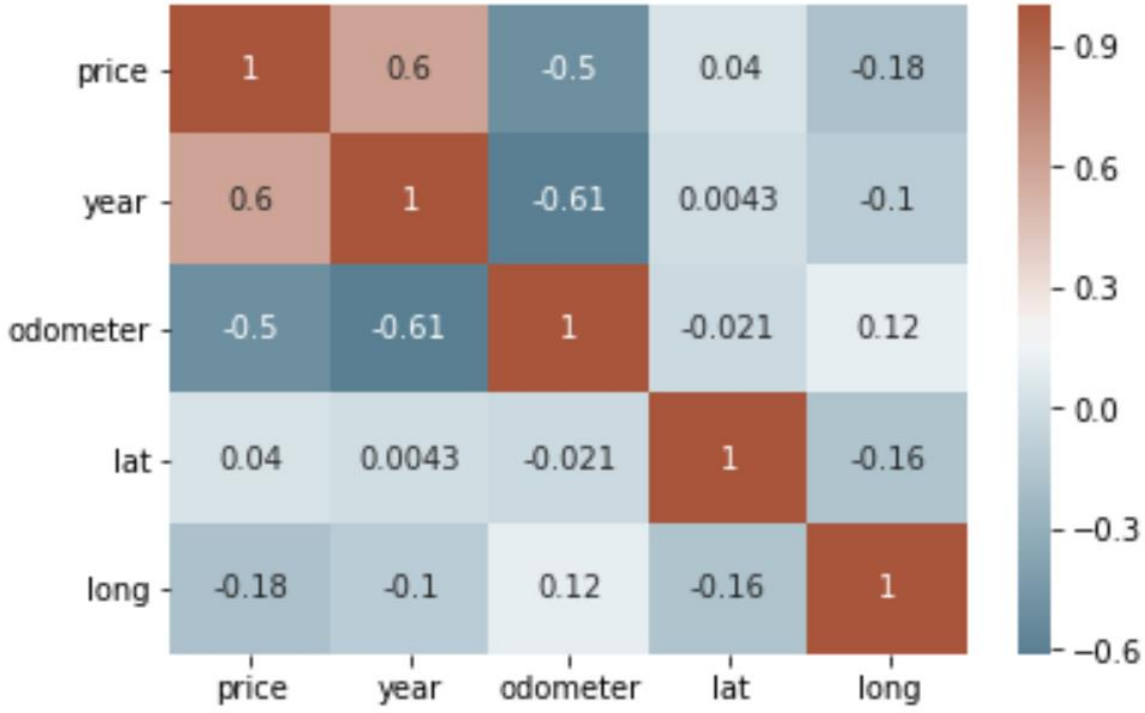
d. Removing Rows with Null Values

Lastly, I used .**dropna(axis=0)** to remove any rows with null values. After the code below, I went from 371982 to 208765 rows.

3. Analysing relationships between variables

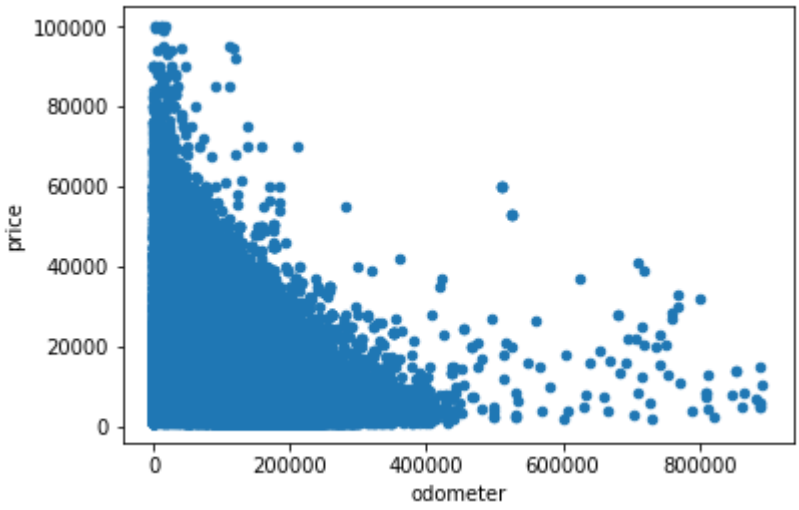
a. Correlation Matrix

The first thing I like to do when analysing my variables is visualizing it through a correlation matrix because it's the fastest way to develop a general understanding of all of my variables. To review, correlation is a measurement that describes the relationship between two variables. I used **sns.heatmap()** to plot a correlation matrix of all of the variables in the used car dataset. We can see that there is a positive correlation between price and year and a negative correlation between price and odometer. This makes sense as newer cars are generally more expensive, and cars with more mileage are relatively cheaper. We can also see that there is a negative correlation between year and odometer the newer a car the less number of miles on the car.



b. Scatterplot

It's pretty hard to beat correlation heatmaps when it comes to data visualizations, but scatterplots are arguably one of the most useful visualizations when it comes to data. A scatterplot is a type of graph which 'plots' the values of two variables along two axes, like age and height. Scatterplots are useful for many reasons: like correlation matrices, it allows you to quickly understand a relationship between two variables, it's useful for identifying outliers, and it's instrumental when polynomial multiple regression models. I used. **plot()** and set the 'kind' of graph as scatter. I also set the x-axis to 'odometer' and y-axis as 'price', since we want to see how different levels of mileage affects price.



This narrates the same story as a correlation matrix — there's a negative correlation between odometer and price. What's neat about scatterplots is that it communicates more information than just that. Another insight that you can assume is that mileage has a diminishing effect on price. In other words, the amount of mileage that a car accumulates early in its life impacts price much more than later on when a car is older. You can see this as the plots show a steep drop at first, but becomes less steep as more mileage is added. This is why people say that it's not a good investment to buy a brand new car!

c. Histogram

Correlation matrices and scatterplots are useful for exploring the relationship between two variables. But what if you only wanted to explore a single variable by itself? This is when histograms come into play. Histograms look like bar graphs but they show the distribution of a variable's set of values. We can quickly notice that the average car has an odometer from 0 to just over 200,000 km and a year of around 2000 to 2020. The difference between the two graphs is that the distribution of 'odometer' is positively skewed while the distribution of 'year' is negatively skewed. Skewness is important, especially in areas like finance, because a lot of models assume that all variables are normally distributed, which typically isn't the case.