# CHAPTER 10: Analyzing a Dataset

## Introduction

In the previous chapter, which was about improving our machine learning model, tune its hyperparameters, and interpret its results and parameters to provide meaningful insights back to business. This chapter, we will learn how to explore a new dataset, prepare it for the modeling stage.

The most time-consuming aspect of any data science project is the transformation of data to a format that an analyst can use to build models. This is more critical for parametric models, which assume known distributions in the data. However, even before you begin to transform the data, you need to understand it.

What does it mean to “understand” data? The objectives of data understanding are:

* Understand the attributes of the data.
* Summarize the data by identifying key characteristics, such as data volume and total number of variables in the data.
* Understand the problems with the data, such as missing values, inaccuracies, and outliers.
* Visualize the data to validate the key characteristics of the data or unearth problems with the summary statistics.

## Exploring your data

Exploratory Data Analysis is an approach in analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods.  
EDA assists Data science professionals in various ways: -

* Getting a better understanding of data
* Identifying various data patterns
* Getting a better understanding of the problem statement

It is used to discover trends, patterns, or to check assumptions with the help of statistical and graphical representations.

For the rest of this section, we will be working with an automobile dataset.

The dataset is in GitHub repository:

https://github.com/fenago/DSBook/blob/main/Chapter%2010%20-%20Analyzing%20a%20Dataset/Datasets/Automobile\_data.csv

Our dataset is an csv file. The pandas package provides a method we can use to load this type of file: **read\_csv().**

Let’s read the data using the **.read\_csv()** method and store it in a pandas DataFrame, as shown in the following code snippet:





After loading the data into a DataFrame, we want to know the size of this dataset, that is, its number of rows and columns. To get this information, we just need to call the **.shape** attribute from **pandas**:

Graphical user interface, text, application

Description automatically generated

This attribute returns a tuple containing the number of rows as the first element and the number of columns as the second element. The loaded dataset contains 205 rows and 26 different columns.

Since this attribute returns a tuple, we can access each of its elements independently by providing the relevant index. Let's extract the number of rows (index 0):

Graphical user interface, application

Description automatically generated

Similarly, we can get the number of columns with the second index:

Graphical user interface

Description automatically generated

Usually, the first row of a dataset is the header. It contains the name of each column. By default, the **read\_excel()** method assumes that the first row of the file is the header. If the header is stored in a different row, you can specify a different index for the header with the parameter header from **read\_excel()**, such as **pd.read\_excel**(header=1) for specifying the header column is the second row.

To access the names of the columns for this DataFrame, we can call the **.columns** attributes:



Text, letter

Description automatically generated

Looking at these names, we can potentially guess what types of information are contained in these columns, however, to be sure, we can use the dtypes attribute, as shown in the following code snippet:



Table

Description automatically generated

From this output, we can see that the **highway-mpg** column is a data type (**int64**), and that **num-of-cylinders** and **engine-size** are (**object**).

The pandas package provides a single method that can display all the information we have seen so far, that is, the **info()** method:



Table

Description automatically generated

In just a few lines of code, we learned some high-level information about this dataset, such as its size, the column names, and their types.

In the next section, we will analyze the content of a dataset.

## Analyzing your dataset

Previously, we learned about the overall structure of a dataset and the kind of information it contains. Now, it is time to really dig into it and look at the values of each column.

The pandas package provides several methods so that you can display a snapshot of your dataset. The most popular ones are head(), tail(), and sample().

The head() method will show the top rows of your dataset. By default, pandas will display the first five rows:



A screenshot of a computer

Description automatically generated with low confidence

The output of the head method shows the top 5 rows of our dataset by default.

With pandas, you can specify the number of top rows to be displayed with the head() method by providing an integer as its parameter. Let's try this by displaying the first 10 rows:



Graphical user interface, application

Description automatically generated

Looking at the top 10 rows, we can assume that majority of the vehicles are “gas” fuel-type and also grouped by cars features like no-of-doors, body-style, engine-location, wheel-base etc. Let’s check whether this is really the case by looking at the last rows of the dataset. This can be achieved by calling the **tail()** method. Like **head()**, this method, by default, will display only five rows, but you can specify the number of rows you want as a parameter. Here, we will display the last eight rows:

Graphical user interface

Description automatically generated with low confidence

It seems that we were right in assuming that the majority of vehicles are of “gas” fuel-type. We can also use the **sample()** method to randomly pick a given number of rows from the dataset with the n parameter. You can also specify a seed in order to get reproducible results if you run the same code again with the **random\_state** parameter:



A screenshot of a computer

Description automatically generated with medium confidence

In this output, we can see an additional value in the fuel-type column: **diesel**

## Exercise 10.01:

**Exploring the Cristiano Ronaldo Dataset with Descriptive Statistics**

In this exercise, we will explore the Cristiano Ronaldo Dataset in order to get a good understanding of it by analyzing its structure and looking at some of its rows.

The dataset we will be using in this exercise is the Cristiano Ronaldo dataset, which can be found on our GitHub repository:

https://github.com/fenago/DSBook/blob/main/Chapter%2010%20-%20Analyzing%20a%20Dataset/Datasets/yds\_data.csv

This dataset contains some amazing facts and stats and the life and career of Cristiano Ronaldo. We look at the Portuguese’s career.

The following steps will help you to complete this exercise:

**Step1:** Import the packages

Text

Description automatically generated with low confidence

**Step2:** Loading the dataset



**Step3:** Print the number of rows and columns of the DataFrame using the shape attribute from the pandas package:





We can see that this dataset contains 30697 row and 27 different columns.

**Step4:** Print the names of the variables contained in this DataFrame using the columns attribute from the pandas packages:



Text, letter

Description automatically generated

We can infer the type of information contained in some of the variables by looking at their names, such as **power\_of\_shot**, **distance\_of\_shot**, and **is\_goal**.

**Step5:** Print out the type of each variable contained in this DataFrame using the dtypes attribute from the pandas package:

Graphical user interface, text, application

Description automatically generated with medium confidence

Table

Description automatically generated

**Step6:** Display the top rows of the dataframe using the head() method from pandas

****

**Graphical user interface

Description automatically generated with medium confidence**

**Step7:** Display the last five rows of the dataframe using the tail() method from pandas



Graphical user interface, application

Description automatically generated

**Step8:** Now, display 5 random smapled rows of the DataFrame using the sample() method from pandas and pass it a random\_state of 8

****

**A screenshot of a computer

Description automatically generated with medium confidence**

## Analyzing the content of a Categorical Variable

Now that we’ve got a good feel for the kind of information contained in the dataset. We want a dig a little deeper into each of its columns:

For instance, we would like to know how many different values are contained in each of the variables by calling the nunique() method. This is particularly useful for a categorical variable with a limited number of values, such as **body-style**.





We can see that there are 38 different body style vehicles in this dataset. It would be great if we could get a list of all values in this variable. Thankfully, the pandas provides a method to get these results: **unique():**



Graphical user interface, application

Description automatically generated

We can see that there are multiple body-style vehicles.

Another very useful method from pandas is value\_counts(). This method lists all the values from a given column but also their occurrence. By providing the dropna=False and normalise=True parameters, this method will include the missing value in the listing and calculate the number of occurrences as a ratio, respectively:



Text

Description automatically generated

From this output, we can see that the **sedan** value is totally dominating this column as it represents over 46% of the rows and that other values such as **hardtop** and **convertible** are quite rare as they represent less than 5% of this dataset.

## Exercise 10.02:

**Analyzing the Categorical Variables from the Cristiano Ronaldo Dataset**

In this exercise, we will continue our dataset exploration by analyzing the categorical variables of this dataset. To do so, we will implement our own describe functions.

The dataset we will be using in this exercise is the Cristiano Ronaldo Dataset, which can be found on our GitHub repository:

https://github.com/fenago/DSBook/blob/main/Chapter%2010%20-%20Analyzing%20a%20Dataset/Datasets/yds\_data.csv

Let’s get started:

**Step1:** Import library



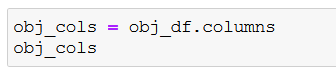
**Step2:** Loading the dataset

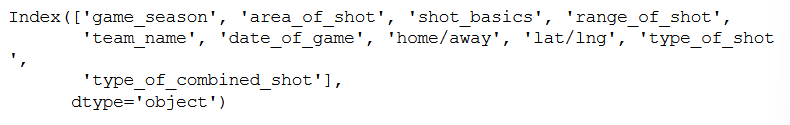


**Step3:** Create a new DataFrame called obj\_df with only the columns that are of numerical types using the select\_dtypes method from pandas package. Then, pass in the object value to the include parameter:

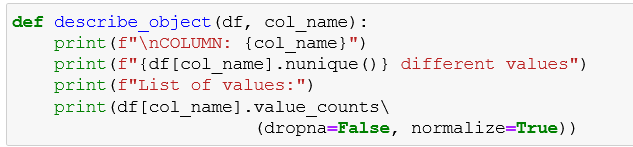
****

**Step4:** Using the columns attribute from pandas, extract the list of columns of this DataFrame, obj\_df, assign it to a new variable called obj\_cols, and print its content:





**Step5:** Create a function called describe\_object that takes a pandas DataFrame and a column name as input parameters. Then, inside the function, print out the name of the given column, its number of unique values using the nunique() method, and the list of values and their occurrence using the value\_counts() method, as shown in the following code snippet:



**Step6:** Test this function by providing the df DataFrame and the **area\_of\_shot**



Text

Description automatically generated with medium confidence

For the **area\_shot** column, the **center** value represents almost 41% of the values, while **Mid Ground** is only present in less than 1% of the rows.

**Step7:** Create a for loop that will call the created function for every element from the obj\_cols list:

Text

Description automatically generated

Text

Description automatically generated

We just analyzed all the categorical variables from this dataset. We saw how to look at the distribution of all the values contained in any feature. We also found that some of them are dominated by a single value and others have mainly missing values in them.

## Summarizing Numerical Variables

Now, let's have a look at a numerical column and get a good understanding of its content. We will use some statistical measures that summarize a variable. All of these measures are referred to as descriptive statistics. In this chapter, we will introduce you to the most popular ones.

With the pandas package, a lot of these measures have been implemented as methods. For instance, if we want to know what the highest value contained in the ‘curb\_weight' column is, we can use the .max() method:





We can see that the maximum weight of vehicle in this automobile dataset is 4066 pounds. We must see the distribution of the weight and confirm whether it fall inline with other car types. Now, let’s have a look at the lowest value for “curb\_weight’ using the .min() method:





The lowest value in this variable is extremely low.

Now, we are going to have a look at the central tendency of this column. Central tendency is a statistical term referring to the central point where the data will cluster around. The most famous central tendency measure is the average (or mean). The average is calculated by summing all the values of a column and dividing them by the number of values.

We can get the average value of a feature by using the mean() method from pandas:

Graphical user interface, text, application

Description automatically generated

In this dataset, the average weight is around 2555 pounds. The average measure is very sensitive to outliers.

We can use the median instead as another measure of central tendency. The median is calculated by splitting the column into two groups of equal lengths and getting the value of the middle point by separating these two groups.

In pandas, you can call the median() method to get this value:

Graphical user interface, application

Description automatically generated

The median value is 2414, which is quite close from the mean 2555. This tell us that there may be very little influence of outliers.

We can also evaluate the spread of this column (how much the data points vary from the central point). A common measure of spread is the standard deviation. The smaller this measure is, the closer the data is to its mean. On the other hand, if the standard deviation is high, this means there are some observations that are far from the average. We will use the std() method from pandas to calculate this measure:

Graphical user interface, text, application

Description automatically generated

The standard deviation for this column is 520 which is quite height, so the data is quite spread from the average, which is 2555 in this example.

In the pandas package, there is a method that can display most of these descriptive statistics with one single line of code: describe():

Text

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

We got the same values for the ‘curb\_weight’ column as we saw previously. This method has calculated the descriptive statistics for the numerical columns.

## Exercise 10.03:

**Analyzing Numerical Variables from the** **Cristiano Ronaldo Dataset**

In this exercise, we will continue our dataset exploration by analyzing the numerical variables of this dataset. To do so, we will implement our own describe functions.

The fist two steps are common, which includes importing the packages and loading the dataset.

**Step3:** Create a new DataFrame called num\_df with only the columns that are numerical using the select\_dtypes method from the pandas package and pass in the 'number' value to the include parameter:

****

**Step4:** Using the columns attribute from pandas, extract the list of columns of this DataFrame, num\_df, assign it to a new variable called num\_cols, and print its content:

A picture containing text

Description automatically generated

Text

Description automatically generated

**Step5:** Create a function called describe\_numeric that takes a pandas DataFrame and a column name as input parameters. Then, inside the function, print out the name of the given column, its minimum value using min(), its maximum value using max(), its average value using mean(), its standard deviation using std(), and its median using median():

Text

Description automatically generated

**Step6:** Now, test this function by providing the df DataFrame and the power\_of\_shot column:

Chart

Description automatically generated with low confidence

Text, letter

Description automatically generated

The power\_of\_shot ranges from 1 to 7 with an average of 2.5. The median is also most equal to the average, which tells us there are no outliers

**Step7:** Create a for loop that will call the created function for every element from the num\_cols list:

Text

Description automatically generated with medium confidence

Text

Description automatically generated

We saw how to explore a newly received dataset with just a few lines of code. This helped us to understand its structure, the type of information contained in each variable, and also helped us identify some potential data quality issues, such as missing values or incorrect values.

## Visualizing Your data

In the previous section, we saw how to explore a new dataset and calculate some simple descriptive statistics. These measures helped summarize the dataset into interpretable metrics, such as the average or maximum values. Now it is time to dive even deeper and get a more granular view of each column using data visualization.

In a data science project, data visualization can be used either for data analysis or communicating gained insights. Presenting results in a visual way that stakeholders can easily understand and interpret them in is a must-have skill for any good data scientist.

However, in this chapter, we will be focusing on using data visualization for analyzing data. Most people tend to interpret information more easily on a graph than reading written information. For example, when looking at the following descriptive statistics and the scatter plot for the same variable, which one do you think is easier to interpret? Let's look:

**How to use the Altair API**

We will be using a package called altair. There are quite a lot of Python packages for data visualization on the market, such as matplotlib, seaborn, or Bokeh, and compared to them, altair is relatively new, but its community of users is growing fast thanks to its simple API syntax.

Let's see how we can display a bar chart step by step on the online Cristiano Ronaldo Dataset.

**Step1:** importing the package

Graphical user interface, text

Description automatically generated with medium confidence

**Step2:** We will randomly sample 50 rows of this DataFrame using the sample() method (altair requires additional steps in order to display a larger dataset):

A picture containing text

Description automatically generated

Now instantiate a Chart object from altair with the pandas DataFrame as its input parameter:

Graphical user interface, text, application

Description automatically generated

Next, we call the mark\_circle() method to specify the type of graph we want to plot: a scatter plot:

Text

Description automatically generated with medium confidence

Finally, we specify the names of the columns that will be displayed on the x and y axes using the encode() method:

Chart

Description automatically generated with low confidence

Chart, scatter chart

Description automatically generated

Altair provides the option for combining its methods all together into one single line of code, like this:

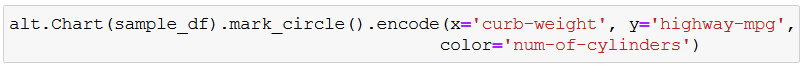


Chart, scatter chart

Description automatically generated

We can see that we got the exact same output as before. The graph shows us that there is a negative correlation between curb-weight and highwary-mpg. i.e., the increase in weight of the vehicle effects the milage in negatively.

Now, let's say we want to visualize the same plot while adding the num-of-cylinders column's information. One easy way to do this is to use the color parameter from the encode() method. This will color all the data points according to their value in the num-of-cylinders column:



Chart, scatter chart

Description automatically generated

We added the information from the num-of-cylinders column into the graph, but as we can see, there are too many vehicles with four cylinders.

## Histogram for Numerical Variables

Now that we are familiar with the altair API, let's have a look at some specific type of charts that will help us analyze and understand each variable. First, let's focus on numerical variables such as highway-mpg or curb-weight.

For this type of variable, a histogram is usually used to show the distribution of a given variable. The x axis of a histogram will show the possible values in this column and the y axis will plot the number of observations that fall under each value.

Let's look at this by using a real example. We will plot a histogram for highway-mpg using the mark\_bar() and encode() methods with the following parameters:

Text

Description automatically generated

Chart, histogram

Description automatically generated

By default, altair grouped the observations by bins of 5 steps: 20 to 25, then 25 to 30, and so on.

Let's plot the histogram for the Quantity column with a bin step size of 10:

Text

Description automatically generated

Chart, bar chart, histogram

Description automatically generated

## Bar Chart for Categorical Variables

Now, we are going to have a look at categorical variables. For such variables, there is no need to group the values into bins as, by definition, they have a limited number of potential values. We can still plot the distribution of such columns using a simple bar chart. In altair, this is very simple – it is similar to plotting a histogram but without the bin parameter. Let's try this on the num-of-cylinders column and look at the number of records for each of its values:



Chart, histogram

Description automatically generated

We can confirm that four cylinders are most used in cars.

## Boxplots

Now, we will have a look at another specific type of chart called a boxplot. This kind of graph is used to display the distribution of a variable based on its quartiles. Quartiles are the values that split a dataset into quarters. Each quarter contains exactly 25% of the observations. For example, in the following sample data, the quartiles will be as follows:

Diagram, schematic

Description automatically generated

So, the first quartile (usually referred to as Q1) is 4; the second one (Q2), which is also the median, is 5; and the third quartile (Q3) is 8.

A boxplot will show these quartiles but also additional information, such as the following:

* The interquartile range (or IQR), which corresponds to Q3 - Q1
* The lowest value, which corresponds to Q1 - (1.5 \* IQR)
* The highest value, which corresponds to Q3 + (1.5 \* IQR)
* Outliers, that is, any point outside of the lowest and highest points:

Diagram

Description automatically generated

With a boxplot, it is quite easy to see the central point (median), where 50% of the data falls under (IQR), and the outliers.

Another benefit of using a boxplot is to plot the distribution of categorical variables against a numerical variable and compare them. Let's try it with the num-of-cylinders and highway-mpg columns using the mark\_boxplot() method:

Text

Description automatically generated

Chart, box and whisker chart

Description automatically generated

This chart shows us how the **highway-mpg** variable is distributed across the different **num-of-cylinders** for this dataset. We can see four and six cylinders have outliers.

## Exercise 10.04:

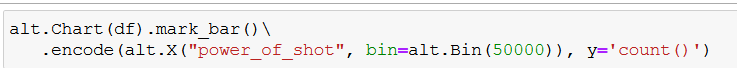
**Visualizing the Cristiano Ronaldo Dataset with Altair**

In this exercise, we will learn how to get a better understanding of a dataset and the relationship between variables using data visualization features such as histograms, scatter plots, or boxplots.

You will be using the same Cristiano Ronaldo Dataset that was used in the previous exercise.

The first two steps would be loading the package data.

**Step3:** Plot the histogram for the power\_of\_shot variable using the mark\_bar() and encode() methods from the altair package. Use the alt.X and alt.Bin APIs to specify the number of bin steps, that is, 50000:

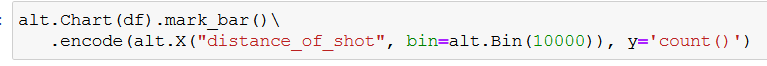


Chart, histogram

Description automatically generated

This chart shows that most number of records have power-of-shot around 3-4.

**Step4:** Now, let's plot the histogram for distance\_of\_shot but this time with a bin step size of 10000:



Chart, histogram

Description automatically generated

**Step5:** Now, plot a scatter plot with location\_y as the x axis and distance\_of\_shot as the y axis to understand the interactions between these two variables:

Graphical user interface, text

Description automatically generated with medium confidence

Chart, line chart

Description automatically generated

**Step6:** Build a boxplot with Power\_of\_shot:O (':O' is for specifying that this column is ordinal) on the x axis and distance\_of\_shot on the y axis using the mark\_boxplot() method, as shown in the following code snippet:

Text

Description automatically generated

Chart, box and whisker chart

Description automatically generated

## Activity 10.01:

**Analyzing Bikes Data Using Visual Data Analysis Techniques**

The dataset to be used in this activity can be found on our GitHub repository:

https://github.com/fenago/DSBook/blob/main/Chapter%2010%20-%20Analyzing%20a%20Dataset/Datasets/bikes.csv

The following steps will help you complete this activity:

1. Download and load the dataset into Python using .read\_csv().
2. Explore the structure and content of the dataset by using .shape, .dtypes, .head(), .tail(), or .sample().
3. Calculate and interpret descriptive statistics with .describe().
4. Analyze each variable using data visualization with bar charts, histograms, or boxplots.
5. Identify areas that need clarification from the marketing department and potential data quality issues.

## Summary

You just learned a lot regarding how to analyze a dataset. This a very critical step in any data science project. Getting a deep understanding of the dataset will help you to better assess the feasibility of achieving the requirements from the business.

Getting the right data in the right format at the right level of quality is key for getting good predictive performance for any machine learning algorithm. This is why it is so important to take the time analyzing the data before proceeding to the next stage.

You learned how to use descriptive statistics to summarize key attributes of the dataset such as the average value of a numerical column, its spread with standard deviation or its range (minimum and maximum values), the unique values of a categorical variable, and its most frequent values. You also saw how to use data visualization to get valuable insights for each variable. Now, you know how to use scatter plots, bar charts, histograms, and boxplots to understand the distribution of a column.

While analyzing the data, we came across additional questions that, in a normal project, need to be addressed with the business. We also spotted some potential data quality issues, such as missing values, outliers, or incorrect values that need to be fixed.