Introduction

The term Exploratory Data Analysis (EDA for short) refers to the process of **discovering patterns and relationships within a dataset**. EDA is about making sense of the data at hand and getting to know it, often before modeling or analysis.

It is important to note that **every dataset is different, and therefore will require different exploration. EDA is all about following the data, verifying your assumptions, and investigating anything that is unexpected**.

EDA is an important step prior to model building and can help you formulate further questions and areas for investigation.

Goals of EDA

Depending on your analysis or model-building plan, EDA can take many different forms; However, the main goals of EDA are generally:

- Uncover the data structure and determine how it is coded
- Inspect and “get to know” the data by summarizing and visualizing it
- Detect outliers, missing data, and other anomalies and decide how/whether to address these issues
- Find new avenues for analysis and further research
- Prepare for model building or analysis, including the following:
    - Check assumptions
    - Select features
    - Choose an appropriate method

EDA techniques

Just as the goals of EDA may vary, so do the techniques used to accomplish those goals. That said, the EDA process generally involves strategies that fall into the following three categories:

- Data inspection
- Numerical summarization
- Data visualization

Data inspection

Data inspection is an important first step of any analysis. This can help illuminate potential issues or avenues for further investigation.

For example, we might use the pandas **.head()** method to print out the first five rows of a dataset:

```
print(data.head())
```

Student Name	Favorite Class	Age	Hours of Sleep	Hours Spent Studying
John	Math	9	9	1
Sophie	Statistics	17	7.5	4
Alex	English	12	8	3
Liam	Statistics	18	nan	4.5
Colin	Math	18	6.5	6.5

Based on this output, we notice that hours of sleep is a quantitative variable. In order to summarize it, we'll need to make sure it is stored as an **int** or **float**.

We also notice that there is at least one instance of missing data, which appears to be stored as **nan**. As a next step, we could investigate further to determine how much missing data there is and what we want to do about it.

Numerical summarization

Once we've inspected our data and done some initial cleaning steps, numerical summaries are a great way to condense the information we have into a more reasonable amount of space. For numerical data, this allows us to get a sense of scale, spread, and central tendency. For categorical data, this gives us information about the number of categories and frequencies of each.

In pandas, we can get a quick collection of numerical summaries using the **.describe()** method:

```
data.describe(include = 'all')
```

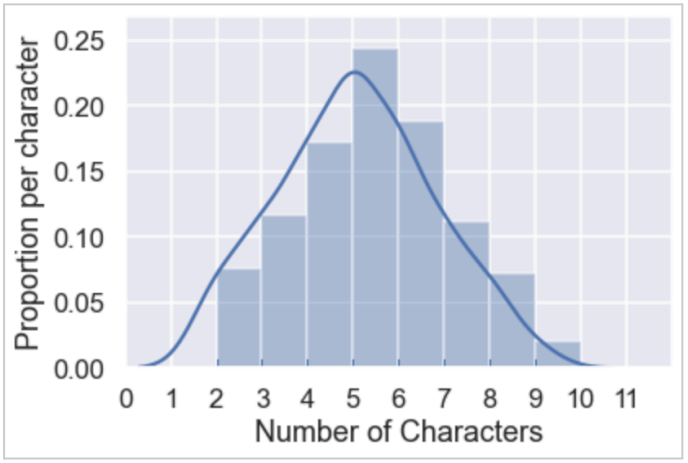
	Student Name	Favorite Class	Age	Hours of Sleep	Hours Spent Studying
count	250	250	250	250	250
unique	177	15	NaN	NaN	NaN
top	Kevin	Math	NaN	NaN	NaN
freq	12	23	NaN	NaN	NaN
mean	NaN	NaN	13.75	7.89	4.34
std	NaN	NaN	1.68	0.3	0.6
min	NaN	NaN	8	4.5	0.5
25%	NaN	NaN	10.3	5.6	1.47
50%	NaN	NaN	13.5	7.6	4.32
75%	NaN	NaN	17	9.7	6.5
max	NaN	NaN	23	11	10.5

Based on this table, we can see that there are 187 unique student names in our table, with Kevin being the most common. The average student age is 13.75 years with students as young as 8 years and as old as 23 years.

Data visualization

While numerical summaries are useful for condensing information, visual summaries can provide even more context and detail in a small amount of space.

There are many different types of visualizations that we might want to create as part of EDA. For example, **histograms allow us to inspect the distribution of a quantitative feature, providing information about central tendency, spread, and shape (eg., skew or multimodality)**. The histogram below shows the distribution of the number of characters in each student name. We see that the average name is about 5-6 characters long and up to 10 characters long.



Other kinds of visualizations are useful for investigating relationships between multiple features. For example, the **scatterplot below shows the relationship** between hours spent studying and hours spent sleeping.

EDA as a cyclical process

Though EDA is commonly performed at the start of a project — before any analysis or model building — you may find yourself revisiting EDA again and again. It is quite common for more questions and problems to emerge during an analysis (or even EDA itself!). EDA is also a great tool for tuning a predictive model to improve its accuracy. It is therefore useful to think of EDA as a cycle rather than a linear process in a data science workflow.

Conclusion

EDA is a crucial step before diving into any data project because it informs data cleaning, can illuminate new research questions, is helpful in choosing appropriate analysis and modeling techniques, and can be useful during model tuning.

Other info:
A common step is to address questions such as:

- How many (non-null) observations do we have?
- How many unique columns/features do we have?
- Which columns (if any) contain missing data?
- What is the data type of each column?

Using pandas, we can easily address these questions using the **.info()** method.

```
print(heart.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   age            303 non-null    float64
 1   sex            303 non-null    float64
 2   cp             303 non-null    float64
 3   trestbps       303 non-null    float64
 4   chol           303 non-null    float64
 5   fbs            303 non-null    float64
 6   restecg        303 non-null    float64
 7   thalach        303 non-null    float64
 8   exang          303 non-null    float64
 9   oldpeak        303 non-null    float64
 10  slope          303 non-null    float64
 11  ca             303 non-null    object 
 12  thal           303 non-null    object 
 13  heart_disease  303 non-null    int64  
dtypes: float64(11), int64(1), object(2)
memory usage: 33.3+ KB
```

Print out all of the rows that contain missing (null) values:

```
print(diabetes_data[diabetes_data.isnull().any(axis=1)])
```

To investigate the unexpected output here, we might want to take a look at the unique values in the **ca** column (i.e., example):

```
print(heart.ca.unique())
```