**A Beginner’s Guide to Data Analysis in Python**

A step by step guide to get started with data analysis in Python

[[](https://natassha6789.medium.com/?source=post_page-----188706df5447--------------------------------)](https://natassha6789.medium.com/?source=post_page-----188706df5447--------------------------------)

[[Towards Data Science](https://towardsdatascience.com/?source=post_page-----188706df5447--------------------------------)](https://towardsdatascience.com/?source=post_page-----188706df5447--------------------------------)

[Natassha Selvaraj](https://natassha6789.medium.com/?source=post_page-----188706df5447--------------------------------)

·

Follow

Published in

[Towards Data Science](https://towardsdatascience.com/?source=post_page-----188706df5447--------------------------------)

·

9 min read

·

Oct 15, 2020

343

5



Photo by [Chris Liverani](https://unsplash.com/@chrisliverani?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/data-analysis?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

**The Role of a Data Analyst**

A data analyst uses programming tools to mine large amounts of complex data, and find relevant information from this data.

In short, an analyst is someone who derives meaning from messy data. A data analyst needs to have skills in the following areas, in order to be useful in the workplace:

* **Domain Expertise** — In order to mine data and come up with insights that are relevant to their workplace, an analyst needs to have domain expertise.
* **Programming Skills**—As a data analyst, you will need to know the right libraries to use in order to clean data, mine, and gain insights from it.
* **Statistics** — An analyst might need to use some statistical tools to derive meaning from data.
* **Visualization Skills** — A data analyst needs to have great data visualization skills, in order to summarize and present data to a third party.
* **Storytelling —**Finally, an analyst needs to communicate their findings to a stakeholder or client. This means that they will need to create a data story, and have the ability to narrate it.

In this article, I am going to walk you through the end-to-end data analysis process with Python.

**If you follow along to this tutorial and code everything out the way I did, you can then use these codes and tools for future data analytic projects.**

We will start with downloading and cleaning the dataset, and then move on to the analysis and visualization. Finally, we will tell a story around our data findings.

I will be using a dataset from Kaggle called [Pima Indian Diabetes Database](https://www.kaggle.com/uciml/pima-indians-diabetes-database), which you can download to perform the analysis.

**Pre-Requisites**

For this entire analysis, I will be using a Jupyter Notebook. You can use any Python IDE you like.

You will need to install libraries along the way, and I will provide links that will walk you through the installation process.

**The Analysis**



Photo by [Luke Chesser](https://unsplash.com/@lukechesser?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/data-analysis?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

After downloading the dataset, you will need to read the *.csv* file as a data frame in Python. You can do this using the Pandas library.

If you do not have it installed, you can do so with a simple “*pip install pandas”*in your terminal. If you face any difficulty with the installation or simply want to learn more about the Pandas library, you can check out their documentation [here](https://pandas.pydata.org/docs/).

**Read the Data**

To read the data frame into Python, you will need to import Pandas first. Then, you can read the file and create a data frame with the following lines of code:

import pandas as pd  
df = pd.read\_csv('diabetes.csv')

To check the head of the data frame, run:

df.head()

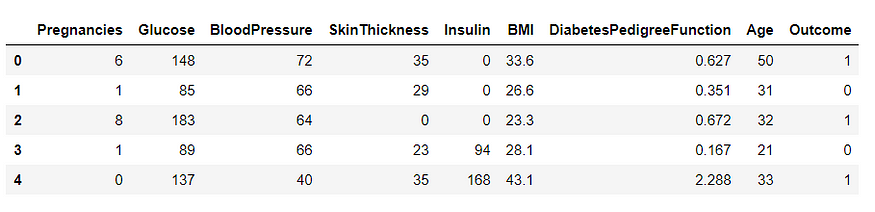


Image by Author

From the screenshot above, you can see 9 different variables related to a patient’s health.

As an analyst, you will need to have a basic understanding of these variables:

* **Pregnancies**: The number of pregnancies the patient had
* **Glucose**: The patient’s glucose level
* **Blood Pressure**
* **Skin Thickness**: The thickness of the patient’s skin in mm
* **Insulin**: Insulin level of the patient
* **BMI**: Body Mass Index of patient
* **DiabetesPedigreeFunction**: History of diabetes mellitus in relatives
* **Age**
* **Outcome**: Whether or not a patient has diabetes

As an analyst, you will need to know the difference between these variable types — Numeric and Categorical.

**Numeric variables**are variables that are a measure, and have some kind of numeric meaning. All the variables in this dataset except for “*outcome*” are numeric.

**Categorical variables** are also called nominal variables, and have two or more categories that can be classified.

The variable “*outcome*” is categorical — 0 represents the absence of diabetes, and 1 represents the presence of diabetes.

**A Quick Note**

Before continuing with the analysis, I would like to make a quick note:

***Analysts are humans, and we often come with preconceived notions of what we expect to see in the data.***

***For example, you would expect an older person to be more likely to have diabetes. You would want to see this correlation in the data, which might not always be the case.***

***Keep an open mind during the analysis process, and do not let your bias effect the decision making.***

**Pandas Profiling**

This is a very useful tool that can be used by analysts. It generates an analysis report on the data frame, and helps you better understand the correlation between variables.

To generate a Pandas Profiling report, run the following lines of code:

import pandas\_profiling as pp  
pp.ProfileReport(df)

This report will give you some overall statistical information on the dataset, which looks like this:

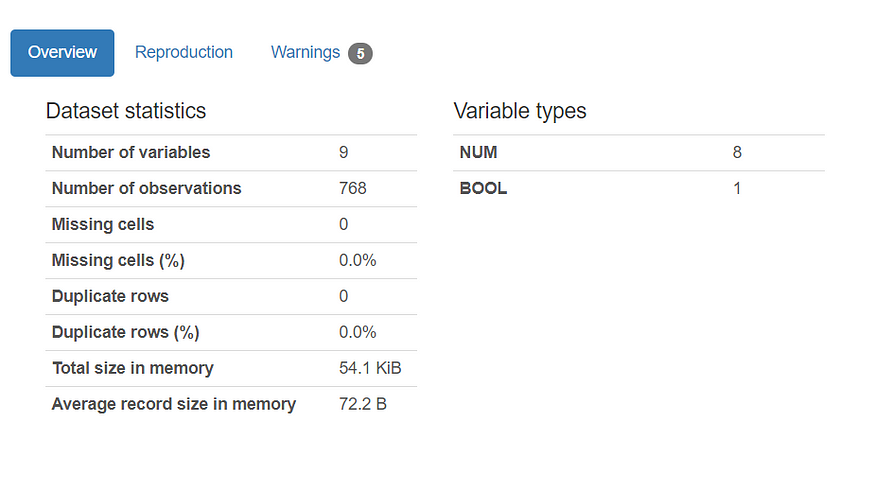


Image by Author

By just glancing at the dataset statistics, we can see that there are no missing or duplicate cells in our data frame.

The information provided above usually requires us to run a few lines of codes to find, but is generated a lot more easily with Pandas Profiling.

Pandas Profiling also provides more information on each variable. I will show you an example:

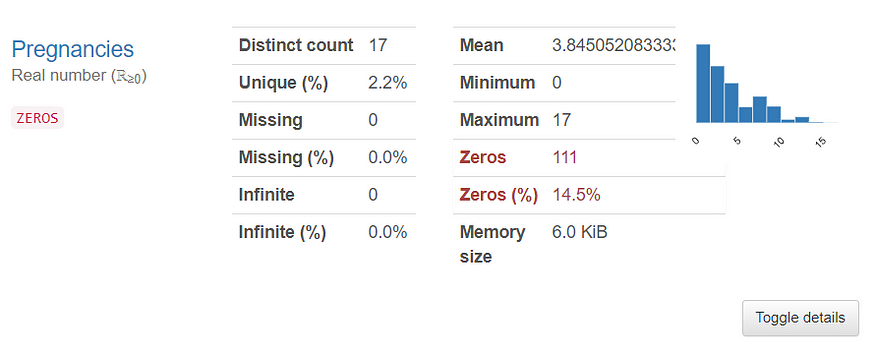


Image by Author

This is information generated for the variable called “*Pregnancies.”*

As an analyst, **this report saves a lot of time**, as we don’t have to go through each individual variable and run too many lines of code.

From here, we can see that:

* The variable “*Pregnancies”*has 17 distinct values.
* The minimum number of pregnancies a person has is 0, and the maximum is 17.
* The number of zero values in this column is pretty low (only 14.5%). This means that **above 80% of the patients in the dataset are pregnant**.

In the report, there is information like this provided for each variable. This helps us a lot in our understanding of the dataset and all the columns in it.

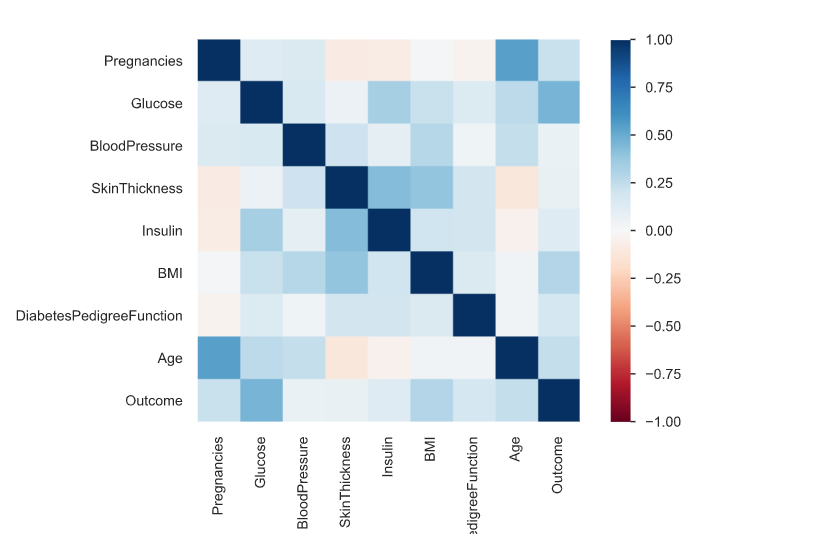


Image by Author

The plot above is a correlation matrix. It helps us gain a **better understanding of the correlation between the variables in the dataset**.

There is a slight positive correlation between the variables “*Age*” and “*Skin Thickness*”, which can be looked into further in the visualization section of the analysis.

Since there are **no missing or duplicate rows** in the data frame as seen above, we don’t need to do any additional data cleaning.

**Data Visualization**

Now that we have a basic understanding of each variable, we can try to find the relationship between them.

The simplest and fastest way to do this is by generating visualizations.

In this tutorial, we will be using three libraries to get the job done — Matplotlib, Seaborn, and Plotly.

If you are a complete beginner to Python, I suggest starting out and getting familiar with Matplotlib and Seaborn.

[Here](https://matplotlib.org/3.3.2/contents.html) is the documentation for Matplotlib, and [here](https://seaborn.pydata.org/) is the one for Seaborn. I strongly suggest spending some time reading the documentation, and doing tutorials using these two libraries in order to improve on your visualization skills.

Plotly is a library that allows you to create interactive charts, and requires slightly more familiarity with Python to master. You can find the installation guide and requirements [here](https://plotly.com/python/).

If you follow along to this tutorial exactly, you will be able to make beautiful charts with these three libraries. You can then use my code as a template for any future analysis or visualization tasks in the future.

**Visualizing the Outcome Variable**

First, run the following lines of code to import Matplotlib, Seaborn, Numpy, and Plotly after installation:

# Visualization Importsimport matplotlib.pyplot as plt  
import seaborn as sns  
color = sns.color\_palette()  
get\_ipython().run\_line\_magic('matplotlib', 'inline')  
import plotly.offline as py  
py.init\_notebook\_mode(connected=True)  
import plotly.graph\_objs as go  
import plotly.tools as tls  
import plotly.express as px  
import numpy as np

Next, run the following lines of code to create a pie chart visualizing the outcome variable:

dist = df['Outcome'].value\_counts()  
colors = ['mediumturquoise', 'darkorange']  
trace = go.Pie(values=(np.array(dist)),labels=dist.index)  
layout = go.Layout(title='Diabetes Outcome')  
data = [trace]  
fig = go.Figure(trace,layout)  
fig.update\_traces(marker=dict(colors=colors, line=dict(color='#000000', width=2)))  
fig.show()

This is done with the Plotly library, and you will get an interactive chart that looks like this:

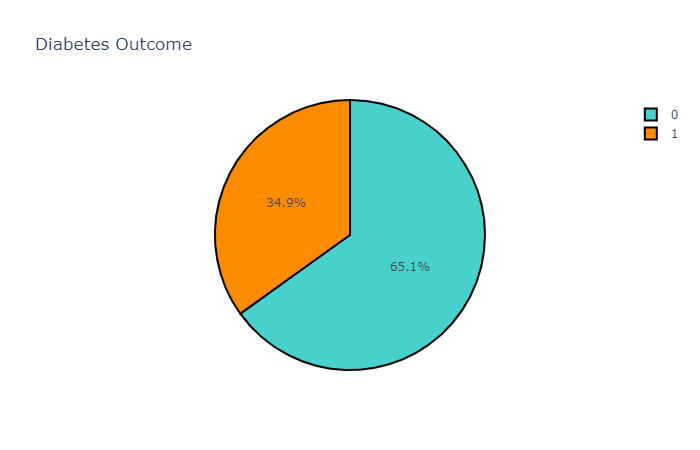


Image by Author

You can play around with the chart and choose to change the colors, labels, and legend.

From the chart above, however, we can see that most patients in the dataset are not diabetic. Less than half of them have an outcome of 1 (have diabetes).

**Correlation Matrix with Plotly**

Similar to the correlation matrix generated in Pandas Profiling, we can create one using Plotly:

def df\_to\_plotly(df):  
 return {'z': df.values.tolist(),  
 'x': df.columns.tolist(),  
 'y': df.index.tolist() }import plotly.graph\_objects as go  
dfNew = df.corr()  
fig = go.Figure(data=go.Heatmap(df\_to\_plotly(dfNew)))  
fig.show()

The codes above will generate a correlation matrix that is similar to the one above:

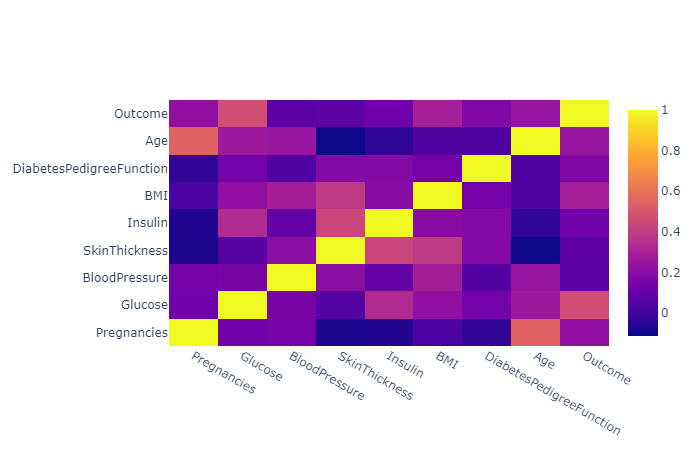


Image by Author

Again, similar to the matrix generated above, a positive correlation can be observed between the variables:

* Age and Pregnancies
* Glucose and Outcome
* SkinThickness and Insulin

To further understand the correlations between variables, we will create some plots:

**Visualize Glucose Levels and Insulin**

fig = px.scatter(df, x='Glucose', y='Insulin')  
fig.update\_traces(marker\_color="turquoise",marker\_line\_color='rgb(8,48,107)',  
 marker\_line\_width=1.5)  
fig.update\_layout(title\_text='Glucose and Insulin')  
fig.show()

Running the codes above should give you a plot that looks like this:

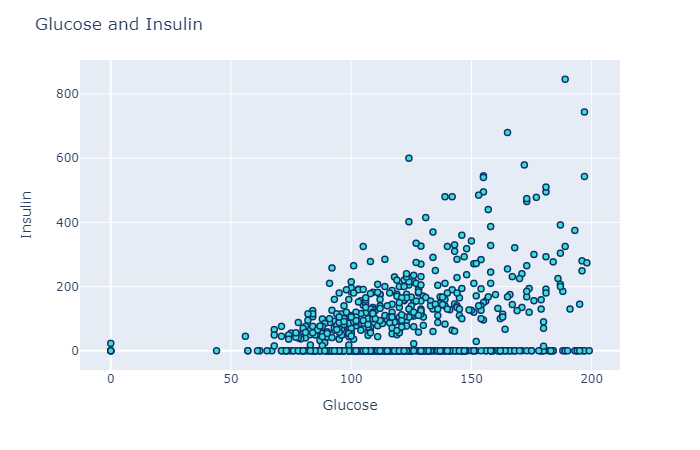


Image by Author

There is a positive correlation between the variables glucose and insulin. This makes sense, because a person with higher glucose levels would be expected to take more insulin.

**Visualize Outcome and Age**

Now, we will visualize the variables outcome and age. We will create a boxplot to do so, using the code below:

fig = px.box(df, x='Outcome', y='Age')  
fig.update\_traces(marker\_color="midnightblue",marker\_line\_color='rgb(8,48,107)',  
 marker\_line\_width=1.5)  
fig.update\_layout(title\_text='Age and Outcome')  
fig.show()

The resulting plot will look somewhat like this:

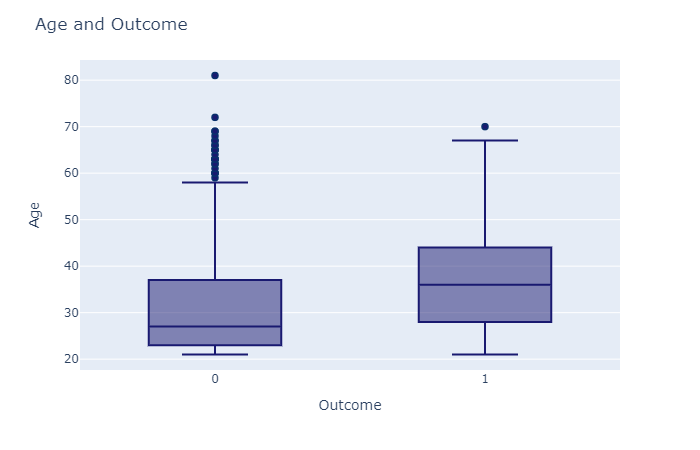


Image by Author

From the plot above, you can see that older people are more likely to have diabetes. The median age for adults with diabetes is around 35, while it is much lower for people without diabetes.

However, there are a lot of outliers.

There are a few elderly people without diabetes (one even over 80 years old), that can be observed in the boxplot.

**Visualizing BMI and Outcome**

Finally, we will visualize the variables “*BMI*” and “*Outcome*”, to see if there is any correlation between the two variables.

To do this, we will use the Seaborn library:

plot = sns.boxplot(x='Outcome',y="BMI",data=df)

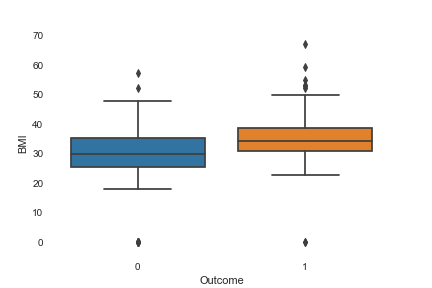


Image by Author

The boxplot created here is similar to the one created above using Plotly. However, Plotly is better at creating visualizations that are interactive, and the charts look prettier compared to the ones made in Seaborn.

From the box plot above, we can see that higher BMI correlates with a positive outcome. People with diabetes tend to have higher BMI’s than people without diabetes.

You can make more visualizations like the ones above, by simply changing the variable names and running the same lines of code.

I will leave that as an exercise for you to do, to get a better grasp on your visualization skills with Python.

**Data Storytelling**



Photo by [Blaz Photo](https://unsplash.com/@blazphoto?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/book?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

Finally, we can tell a story around the data we have analyzed and visualized. Our findings can be broken down as follows:

People with diabetes are highly likely to be older than people who don’t. They are also more likely to have higher BMI’s, or suffer from obesity. They are also more likely to have higher glucose levels in their blood. People with higher glucose levels also tend to take more insulin, and this positive correlation indicates that patients with diabetes could also have higher insulin levels (*this correlation can be checked by creating a scatter plot*).

That’s all for this article! I hope you found this tutorial helpful, and can use it as a future reference for projects you need to create. Good luck in your data science journey, and happy learning!