Siddhardhan Follow

Dec 12, 2021 · 7 min read · Listen

Data Visualization in Python

Hands-on Data Visualization for interactive storytelling in Python

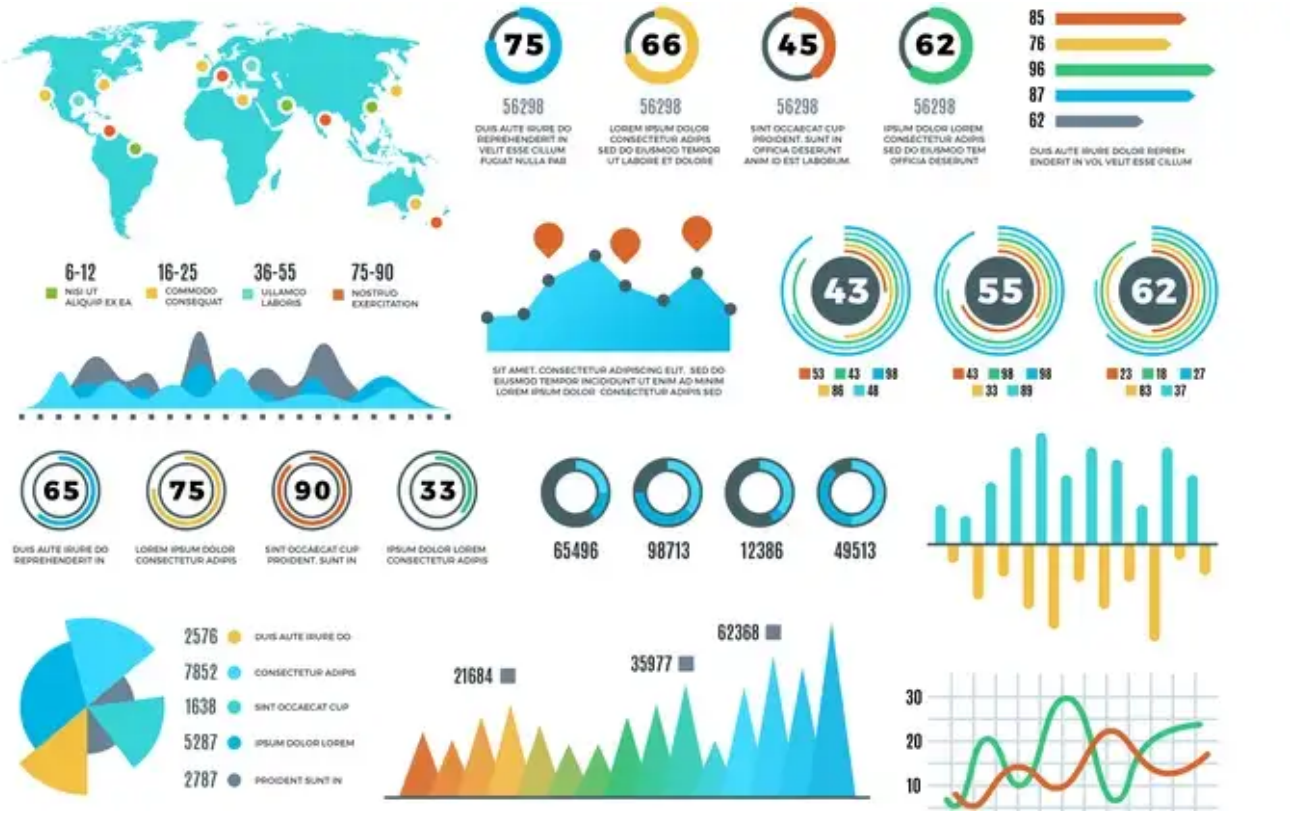


image source: Neil Patel

I would like to start this post with the good old quote **"A Picture is worth a Thousand Words"**. That is what Data Visualization is all about. Data Visualization is an integral part of Data Science & Data Analysis. It is not only useful for us to understand the data, but also to present the insights of the data in the form of a graphical representation

what's the actionable insight that we can derive from the data and how it can be used to solve the business problems. There is not a better way than Data Visualization in order to convey this information.

In this post, let's try to understand how we can carry out the Data Visualization tasks in Python.

Note: Data Visualization is a part of Exploratory Data Analysis(EDA). So I would suggest you to go through my post on EDA before learning about Data Visualization. Click here to read my Medium post on EDA.

Check out my Hands-on Machine Learning course on YouTube

Understanding Data Visualization with an interesting use case in Python:

> **Dataset:** *In order to understand EDA, we will be working on the Breast Cancer Wisconsin (Diagnostic) Data Set. Here, Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. You can find this dataset in Kaggle or UCI ML Repository. You can also download the dataset from here.*

Key Data Visualization tasks for this dataset:

1. Count plot for Categorical columns

2. Distribution plot for all columns

3. Pair plot

4. Checking for Outliers

5. Correlation matrix

6. Inference from EDA & Data Visualization


Matplotlib & Seaborn are the two main Data Visualization libraries in Python. There are also other libraries like Plotly and GGplot.

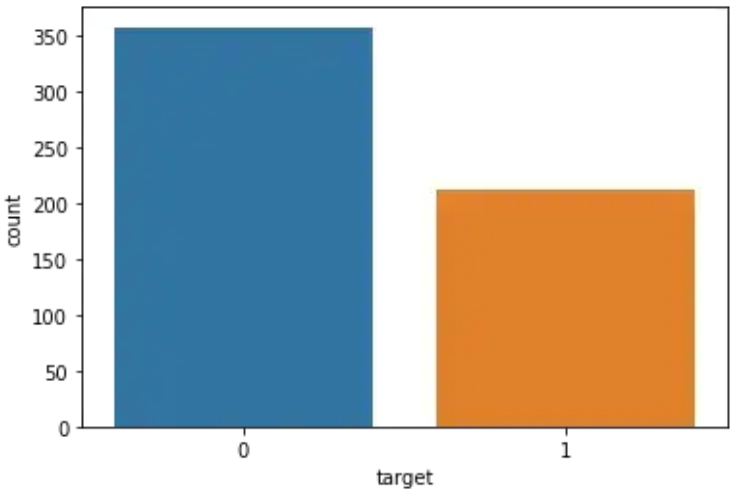
NOTE: I explained how to do some basic Exploratory Data Analysis & Data Processing on this dataset in the previous post. So this code will be a continuation of it.

1. Count plot for Categorical columns:

This dataset contains only one categorical variable("target") with two categories: 0 (Benign) and 1(Malignant). When we have categorical variable, we will plot it in a count plot and when we have a numerical variable, we will use a distribution plot.

(In this post I'll use the terms variables and columns interchangeably as they mean the same thing)

```
sns.countplot(x='target', data=breast_cancer_data
```

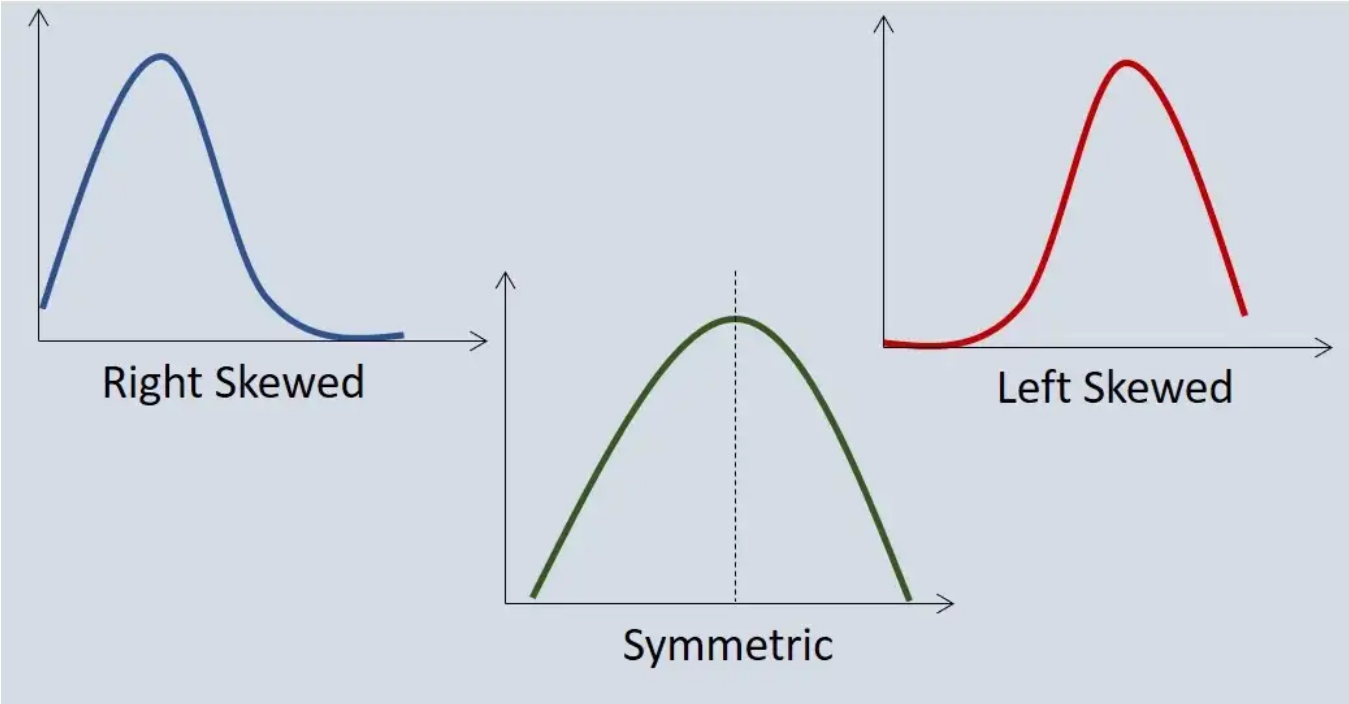


Output

> *As we can clearly see, the number of data points with label "0" is higher than label "1". This*

2. Distribution plot for all columns:

Now we can build distribution plot for all other columns as they contain numerical values. Distribution plot tells us whether the data is **Normally Distributed** or there is some **Skewness** in the data.

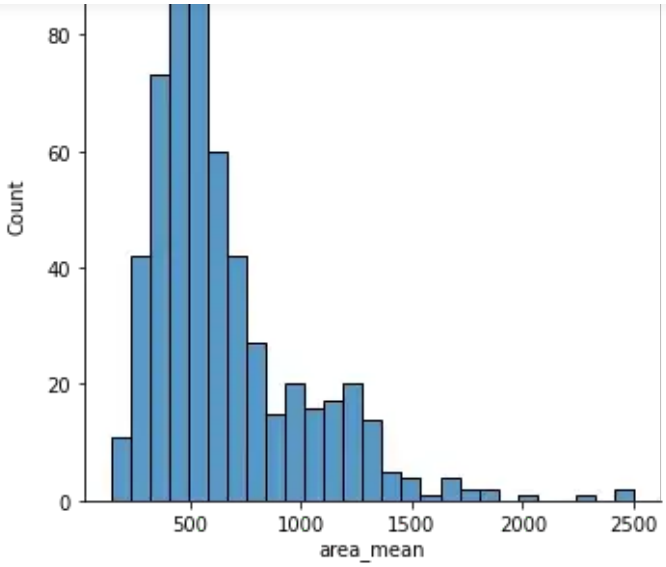


Symmetric distribution represents Normal Distribution

When the skewness in the data is large, we may need to do some transformations, in order to get better results from the Machine Learning models once we train them.

```
for column in breast_cancer_data:
    sns.displot(x=column, data=breast_cancer_data
```

Here, we are executing a for loop which will create distribution plot for all the columns in the dataset. I'll show the distribution plot for "area_mean" column.



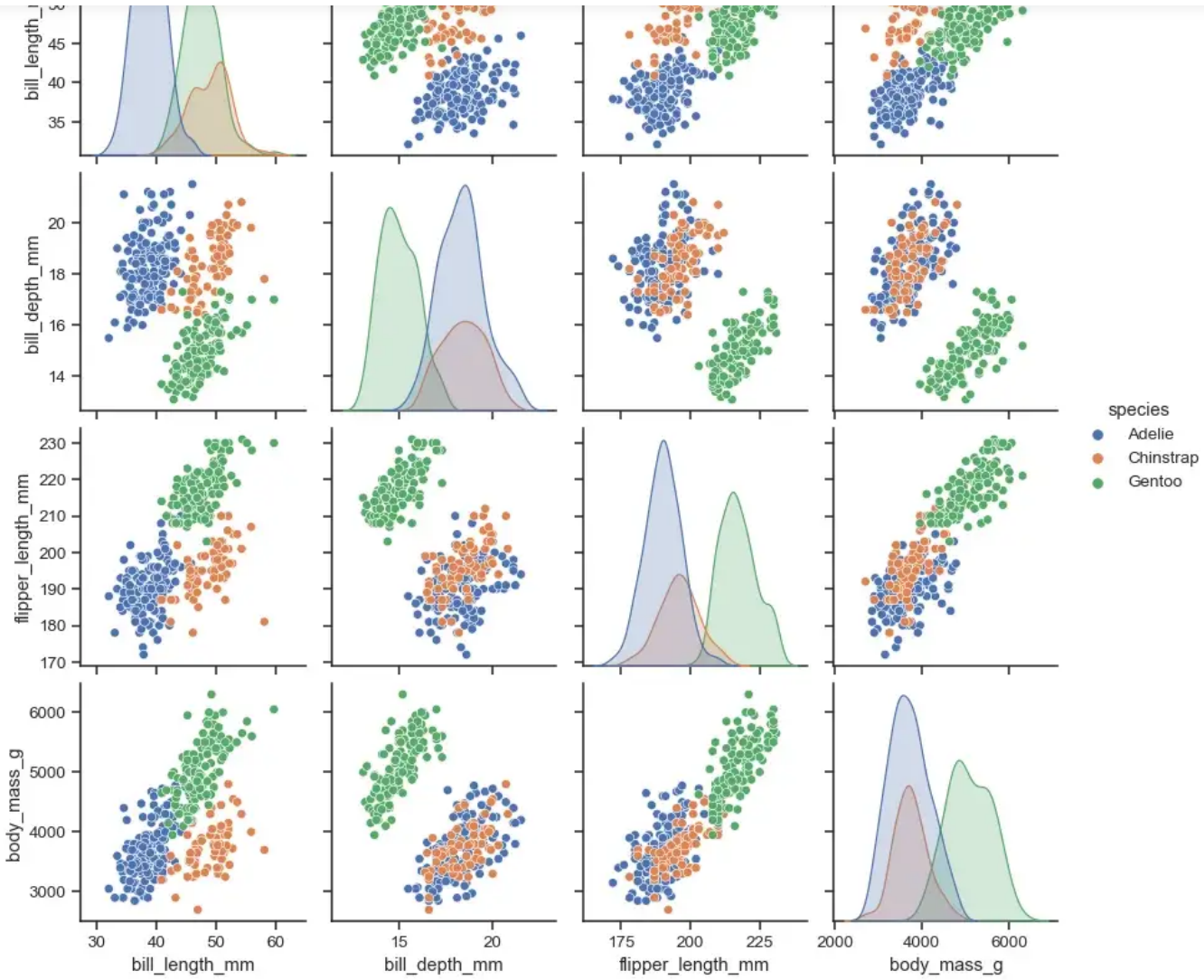
> *We can see that the data is right skewed for "area_mean" column. Similarly, most of the columns have right skewness in this dataset. We will deal with this in the Feature Engineering part.*

This is an what we call as **Univariate Analysis** where we are take one variable at a time and analyzing it. When we take two variables at a time and try to find the relationship between them, it is called **Bivariate Analysis.** When we have more than two variables, it is called **Multivariate Analysis.**

3. Pair plot:

A pair plot gives **pairwise relationships** in a dataset. Let's say that we have 10 variables in a dataset. When we implement pair plot with this data, it will create plots between those 10 variables. Say for example, the first variable will be taken as the x-axis value and other variables will be taken as y-axis value **individually**. As a result, you will have 10 plots for 1st variable alone. This will be repeated for the other variables as well. For this particular dataset, we won't be creating a pair plot as we have around 30 different variables and it will take a long time to plot it. So you can skip this step for datasets with many columns.

Pair plot example:



Pair plot of Penguin data. Source: Seaborn Documentation (When we have the same variable in both x-axis & y-axis, we will get the distribution of that variable. You can see this in the diagonal plots)

The idea behind pair plot is to understand the relationship between the variables present in the data. Alternatively, we can find this relationship using a **Correlation Matrix** which we will discuss later in this post.

If you want to create a pair plot, you can use the function,
sns.pairplot(dataframe_name)

4. Checking for Outliers:

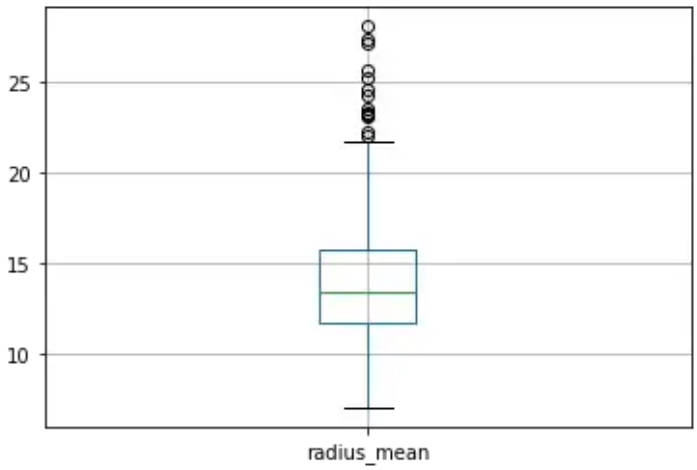
Outliers detection is one of the important tasks that we have to do. Most of the Machine

be carried out in the Feature Engineering part.

We will be creating a Box and Whisker plot in order to check the outliers.

```
for column in breast_cancer_data:
    plt.figure()
    breast_cancer_data.boxplot([column])
```

The above For loop will create box plot for all the columns in the dataset. I'll show the box plot for "radius_mean" variable alone.



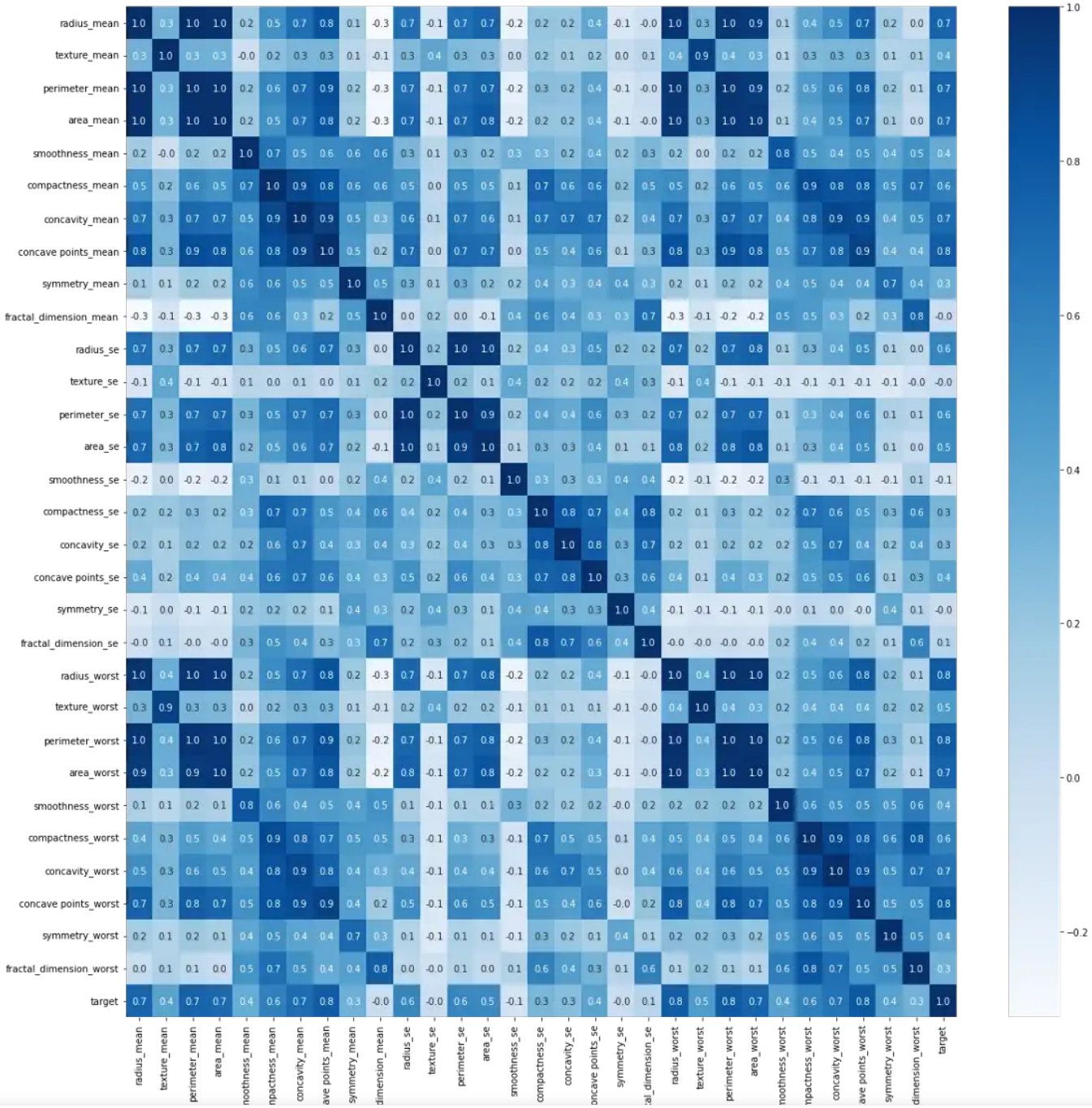
The circles above the top whisker and below the bottom whisker represents the Outliers. Here we have outliers in the upper range alone. I'll create a separate post to explain about outliers and what is the significance of the box plot. It is out of scope of this post.

5. Correlation matrix:

Building a Correlation Matrix is an important step in Data Visualization. The main purpose of a correlation matrix is to understand the **correlation** (in other words, **relationship**) between the variables present in a dataset. It is very helpful in **Feature**

```
plt.figure(figsize=(20,20))
sns.heatmap(correlation_matrix, cbar=True, fmt='.1f', annot=True,
cmap='Blues')
plt.savefig('Correlation Heat map')
```

We will create a Heat Map to visualize the correlation between the variables.



increases by 1 unit and vice versa. This is called **Positive Correlation**. If the correlation between two variables is -1, it means that the value of one variable decreases by 1 unit if the other variable increases by 1 unit and vice versa. This is called **Negative Correlation**.

You can understand Positive Correlation as direct proportionality & Negative Correlation as inverse proportionality.

Importance of determining correlation: As I mentioned earlier, correlation is very helpful in Feature Selection. When we have two independent variables that are very highly correlated, we **should remove** one of them because we run into the **multicollinearity** problem. In those cases, coefficients related to the two highly correlated variables will be unreliable.

We will discuss more of this in Feature Selection.

6. Inference from EDA & Data Visualization:

- No missing Values in the dataset.
- All variables have continuous numerical values except for Target column.
- Mean is slightly more than the median for most of the features. So it is right skewed. This is visible through the distribution plots.
- Slight imbalance in the dataset (Benign(0) cases are more than Malignant(1) cases). Refer Count Plot.
- Mean of most features are clearly larger for Malignant cases compared to the benign cases (Groupby).
- Most of the features have Outliers.
- Correlation Matrix reveal that most of the features are highly correlated. So we can remove certain features during Feature Selection.


derive from EDA & Data Visualization will be helpful for us when we move on to Feature Engineering & Model Training.

I suggest you to take a different dataset and try all these Exploratory Data Analysis & Data Visualization techniques to understand that dataset better.