SEABORN

Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with data frames and the Pandas library. The graphs created can also be customized easily.

How to use Seaborn for Data Visualization:

Data Visualization is the art of representing data in the form of graphs. It is a useful tool for professionals who work with data, i.e., financial analysts, business analysts, data analysts, data scientists, to name a few examples.

In this tutorial, we will be working with Seaborn, a Python Library.

Table of contents

1.Introduction

Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily

with data frames and the Pandas library. The graphs created can also be customized easily. Below are a few benefits of Data Visualization.

Graphs can help us find data trends that are useful in any machine learning or forecasting project.

- Graphs make it easier to explain your data to non-technical people.
- Visually attractive graphs can make presentations and reports much more appealing to the reader.

This tutorial can be divided into three main parts. The first part will talk about installing seaborn and loading our dataset. In the second part, we will discuss some common graphs in Seaborn.

2.Prerequisites

- A good understanding of Python.
- Some Experience working with the Pandas Library.
- Some Experience working with the Matplotlib Library.
- A basic understanding of Data Analysis.

3.Installing Seaborn

you will need to install Seaborn. I highly recommend creating a virtual environment to better manage your packages.

```
pip install pandas, matplotlib, seaborn
```

4.Import Seaborn and loading dataset

```
import seaborn as sns
import pandas
import matplotlib.pyplot as plt
```

Seaborn has 18 in-built datasets, that can be found using the following command.

```
sns.get_dataset_names()
```

We will be using the Titanic dataset for this tutorial.

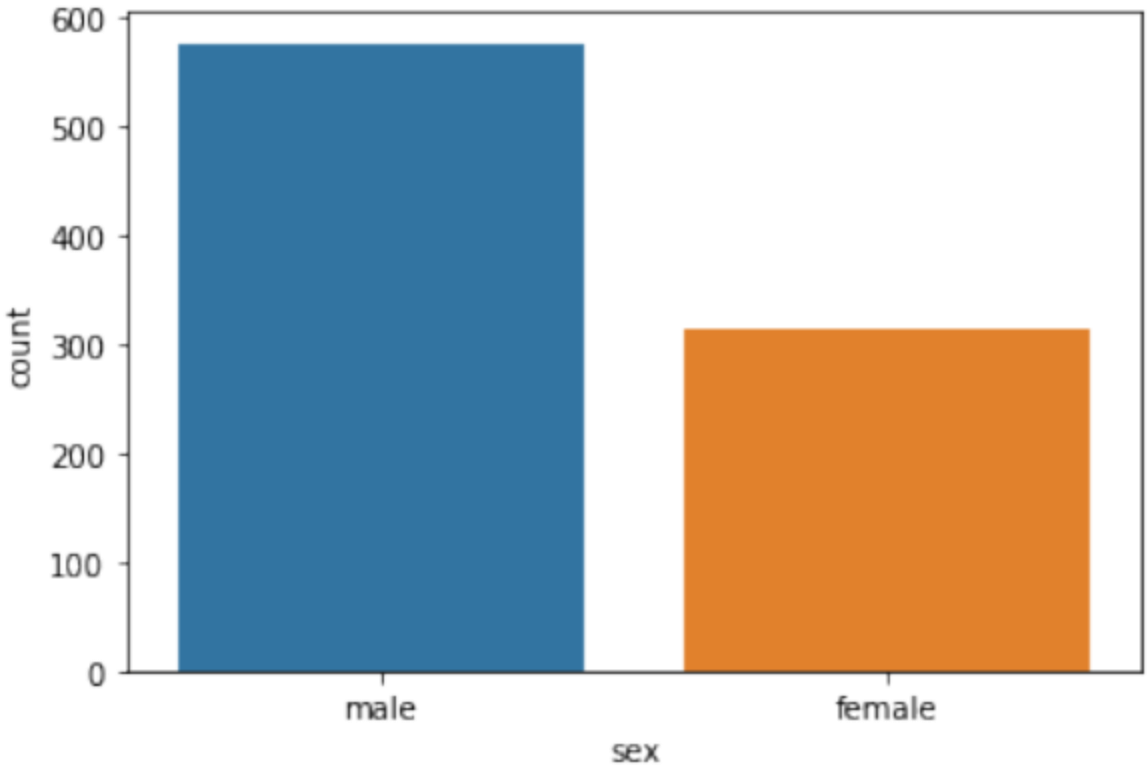
```
df = sns.load_dataset('titanic')
df.head()
```

5.Different types of graphs

Count plot

A count plot is helpful when dealing with categorical values. It is used to plot the frequency of the different categories. The column **sex** contains categorical data in the titanic data, i.e., male and female.

```
sns.countplot(x='sex',data=df)
```

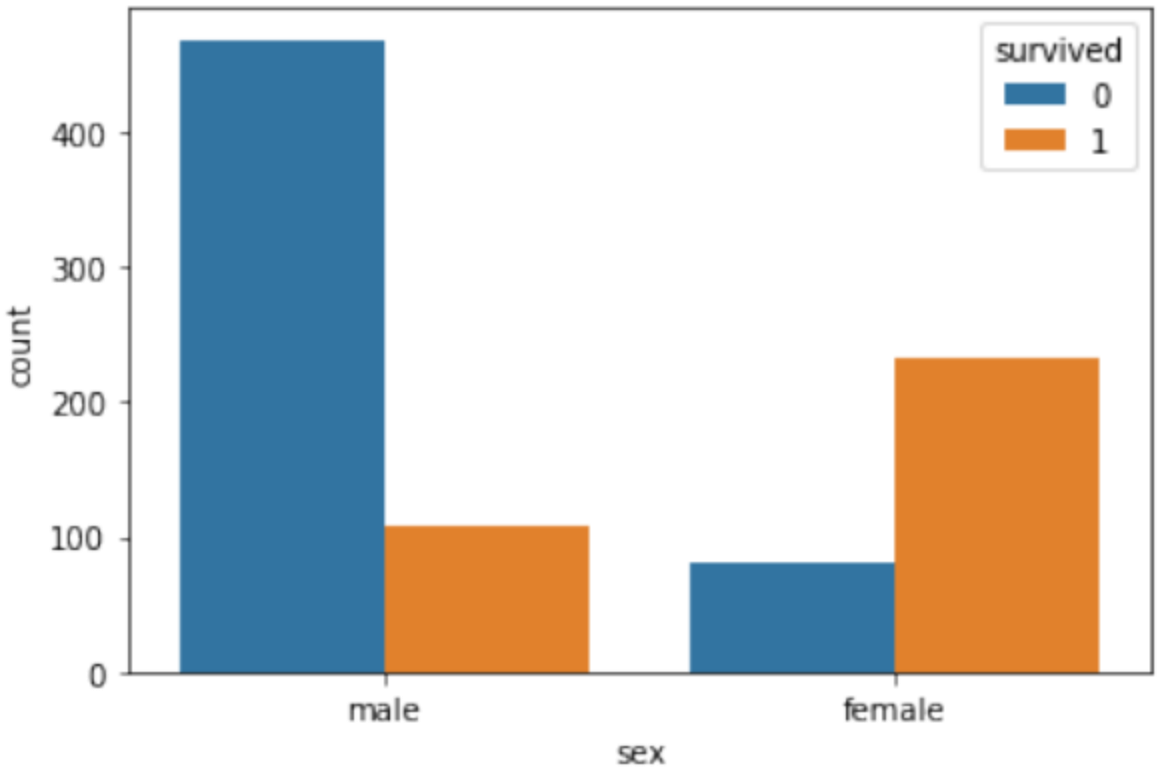


- **data** - The dataframe.
- **x** - The name of the column.

We can observe from the graph that the number of male passengers is significantly higher than the number of female passengers.

We can further break up the bars in the count plot based on another categorical variable. The color palette of the plot can also be customized.

```
sns.countplot(x='sex', hue = 'survived', data = df,
palette = 'Set1')
```

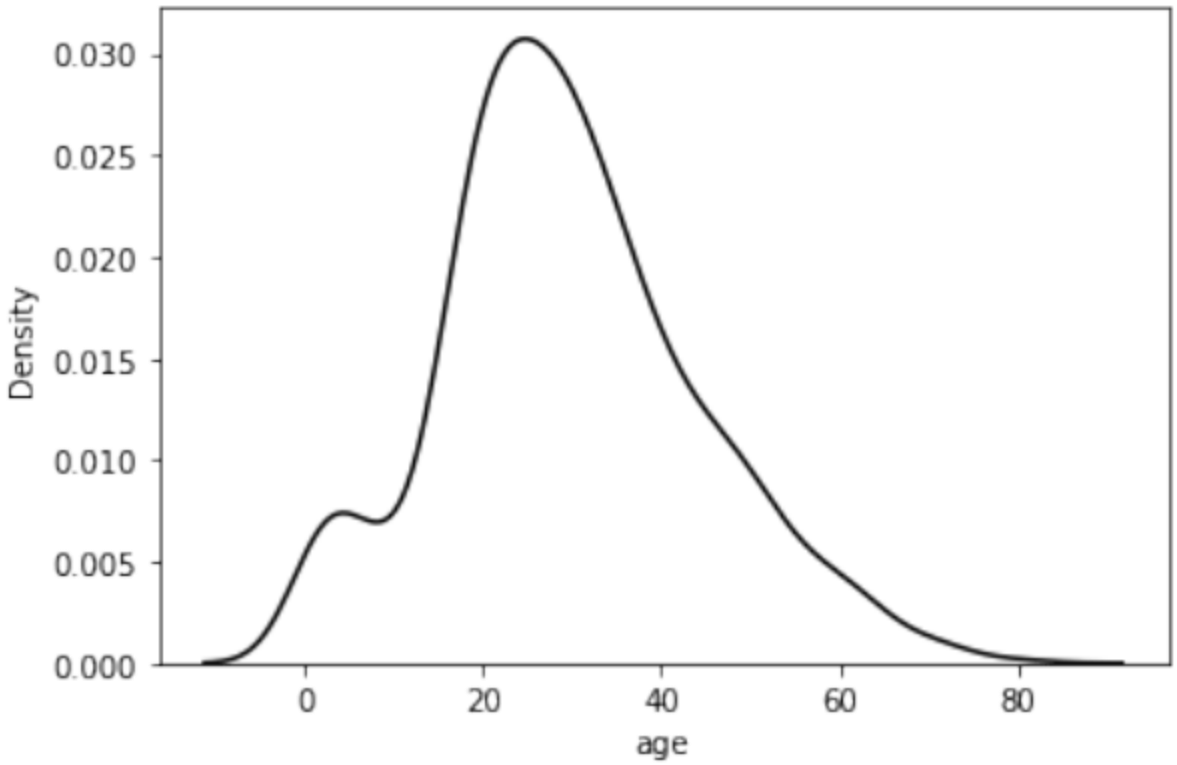


- **hue** - The name of the categorical column to split the bars.
- **palette** - The color palette to be used.

KDE Plot

A Kernel Density Estimate (KDE) Plot is used to plot the distribution of continuous data.

```
sns.kdeplot(x = 'age' , data = df , color = 'black')
```



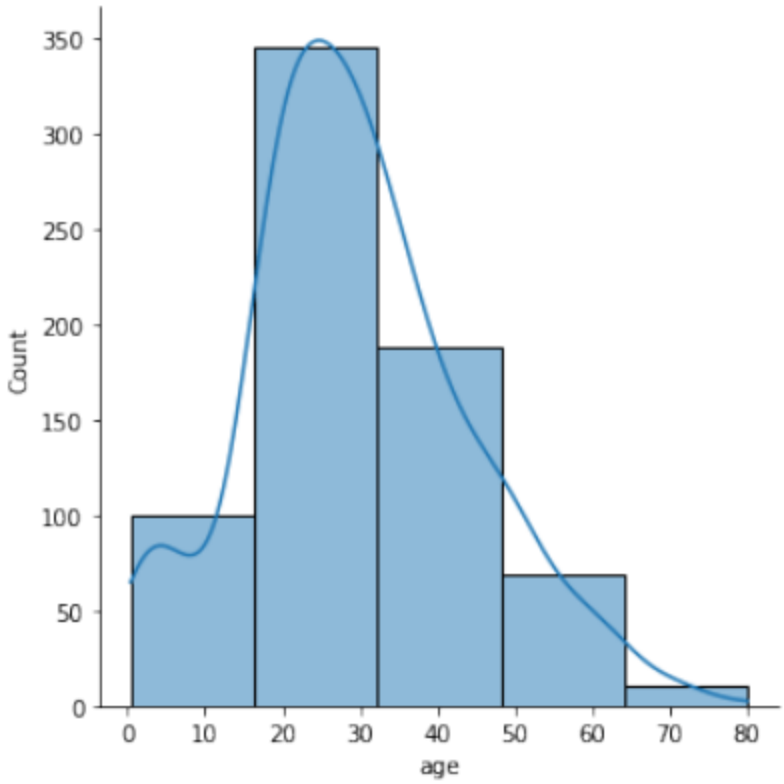
- **data** - The dataframe.
- **x** - The name of the column.
- **color** - The color of the graph.

The peak of the above graph is in between 20 and 40 so we can conclude that most passengers were between the ages of 20 and 40.

Distribution plot

A Distribution plot is similar to a KDE plot. It is used to plot the distribution of continuous data.

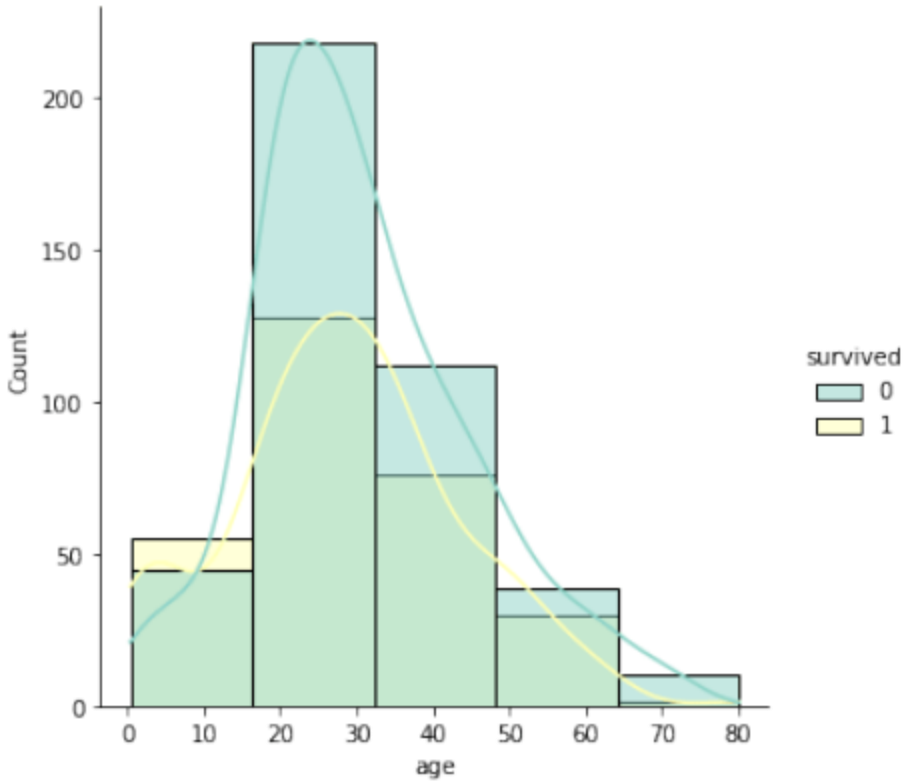
```
sns.displot(x = 'age',kde=True,bins = 5 , data =df)
```



- **kde** - It is set to False by default. However, if you wish to plot a KDE graph on top of the bars, you can set it to True.
- **bins** - The number of bins/bars. The lower the number, wider the bars and wider the intervals.

The plot above tells us that most people onboard the titanic were in their mid-twenties.

```
sns.displot(x ='age',kde=True,bins = 5 ,
hue = df['survived'] , palette = 'Set3', data=df)
```



You can also pass **hue** and **palette** as parameters to customize the graph.

Most of the surviving passengers were in their high-twenties.

Scatter plot

For this plot and the plots below, we will be working with the iris dataset. The iris dataset contains data related to flower's petal size (petal length and petal width) and sepal size (sepal length and sepal width).

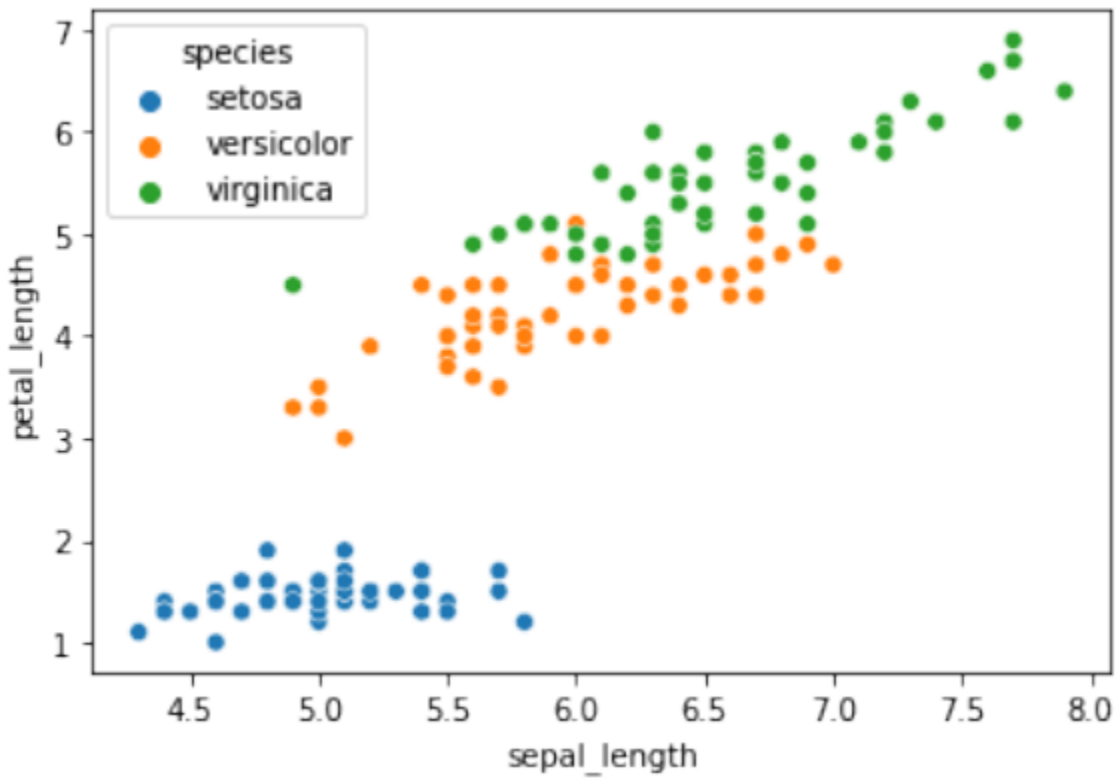
These features are used to classify the type of iris (Setosa, Versicolour, and Virginica). Below we will try to investigate the relation between the features.

First, we will need to load the iris dataset.

```
df = sns.load_dataset('iris')
df.head()
```

Scatter plots help understand co-relation between data,

```
sns.scatterplot(x='sepal_length', y ='petal_length' ,
data = df , hue = 'species')
```



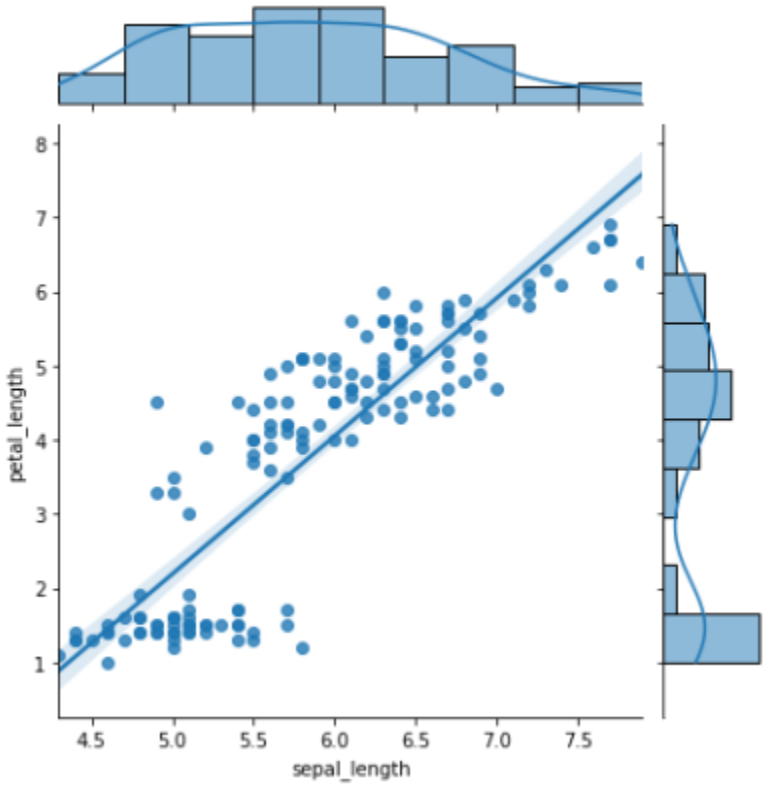
A scatterplot requires data for its **x-axis** and **y-axis**. We can also pass a value for the **hue** parameter to color the dots based on a categorical column.

In the plot above we can observe that an iris flower with a sepal length < 6cm and petal length > 2cm is most likely of type **setosa**. Although there is no distinct boundary present between the **versicolor** dots and **virginica** dots, an iris flower with petal length between 2cm and 5cm is most likely of type **versicolor**, while iris flowers with petal length > 5cm are most likely of type **virginica**.

Joint plot

A Joint Plot is also used to plot the correlation between data.

```
sns.jointplot(x='sepal_length' , y ='petal_length',
data = df , kind = 'reg')
```



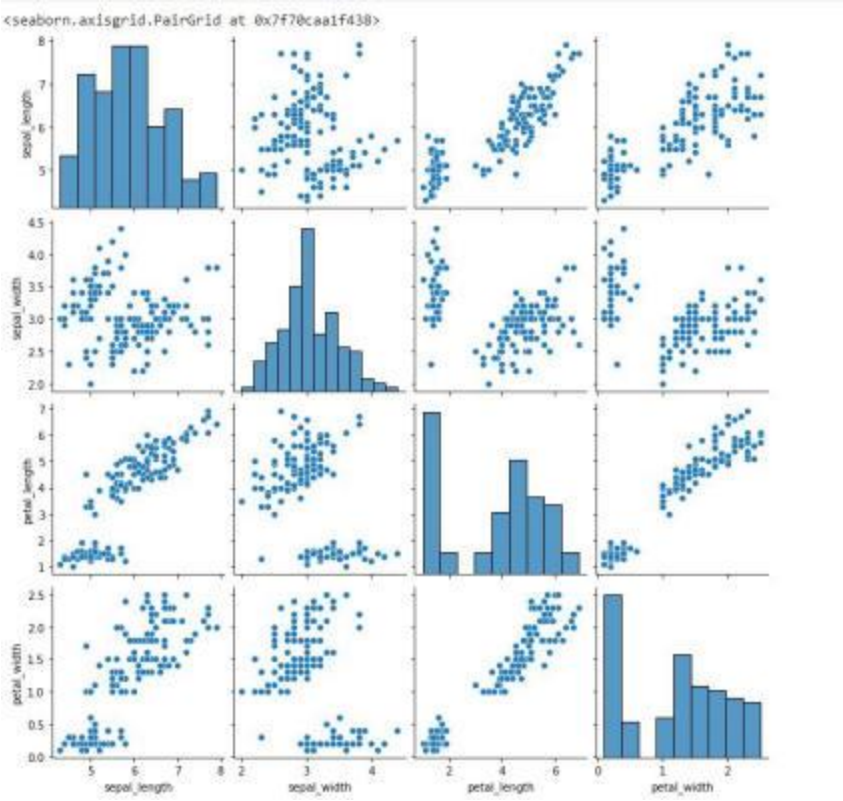
- **kind** - The kind of plot to be plotted. It can be one of the following.

```
'scatter', 'hist', 'hex', 'kde', 'reg', 'resid'
```

Pair plots

Seaborn lets us plot multiple scatter plots. It's a good option when you want to get a quick overview of your data.

```
sns.pairplot(df)
```



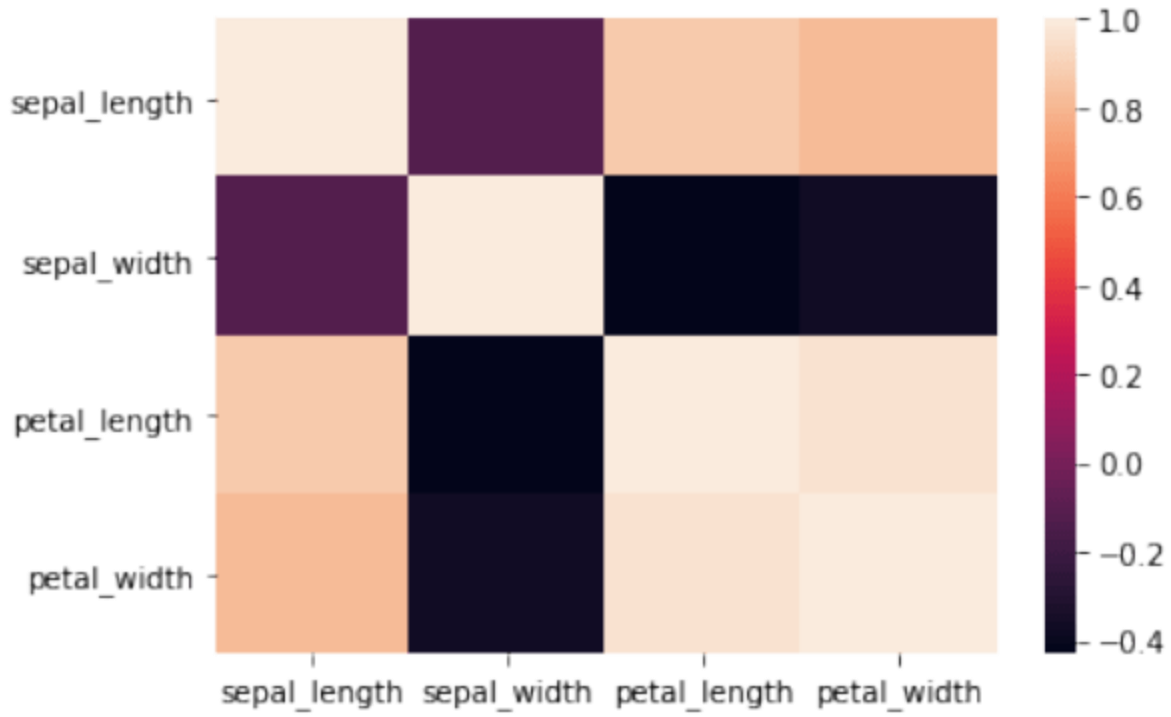
It pairs all the continuous data and plots their correlation. It also plots the distribution of the data.

If you do not wish to pair all the columns, you can pass in two more parameters **x_vars** and **y_vars.**

Heatmaps

A heat map can be used to visualize confusion, matrices, and correlation.

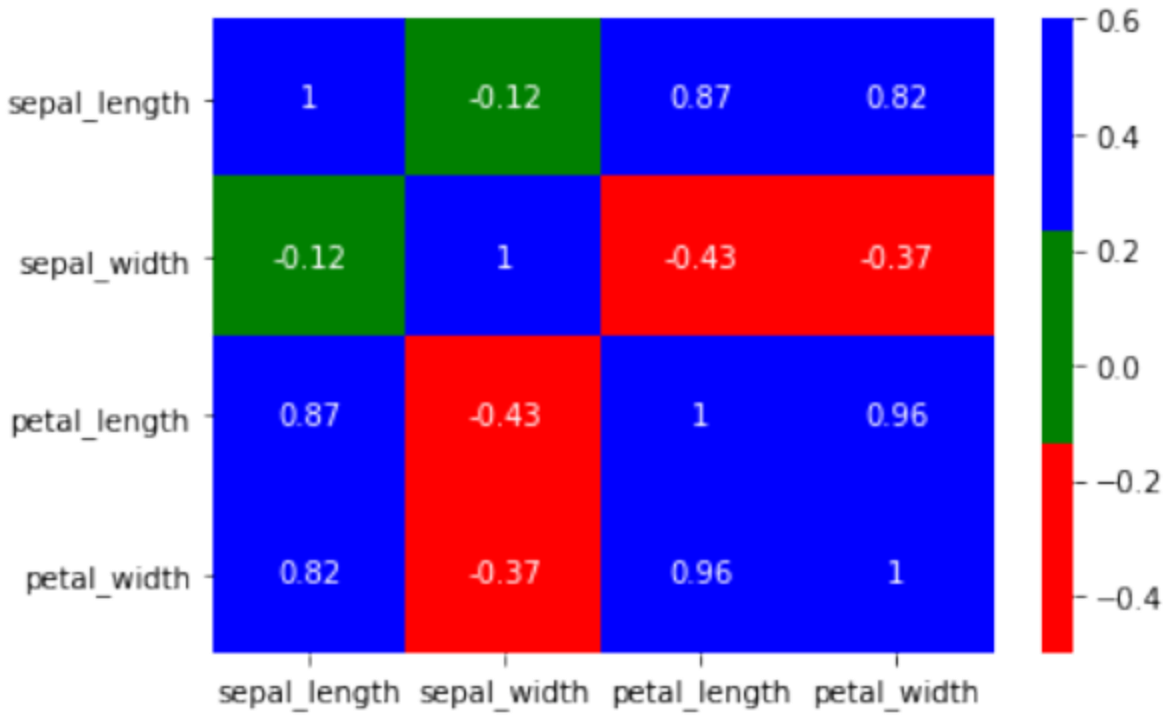
```
corr = df.corr()
sns.heatmap(corr)
```



We can customize the color scheme, the minimum/maximum values, and annotations.

Based on the heatmap we can conclude that sepal length has a high positive correlation with petal length and petal width while sepal width has negative correlation with petal length and petal width.

```
sns.heatmap(corr, cmap=['red','green','blue'],
vmin = -.5 , vmax = 0.6,annot = True)
```


sepal_length
sepal_width
petal_length
petal_width
1
-0.12
0.87
0.82
-0.12
1
-0.43
-0.37
0.87
-0.43
1
0.96
0.82
-0.37
0.96
1
sepal_length
sepal_width
petal_length
petal_width
0.6
0.4
0.2
0.0
−0.2
−0.4