Histogram Plots

In this lesson, histogram plots with matplotlib and seaborn are discussed.

We'll cover the following

Histogram

This visualization is used to display the distribution of numerical data as accurately as possible. This plot divides the data into a specified number of bins, thus separating the data into a range of values. Then, it sketches bars to show the number of data points that each bin contains.

Using matplotlib

Matplotlib provides the `hist()` function to plot a *histogram*. The following example implements a histogram of *two-hundred* random values.

```
import numpy as np
import matplotlib.pyplot as plt

set1 = np.random.randn(200) # Generating random data

plt.hist(set1, edgecolor='black', color='red', bins=20) # plotting histogram
```

Two additional parameters are passed to the `hist()` function. The `color` parameter defines the color of the plot. The `edgecolor` parameter defines an outline around each bin. The `bins` parameter defines how many sections the data is distributed into. By default, the number of bins is set to **ten**. The output can be zoomed in, revealing that the random *two-hundred* values are divided into exactly

twenty sections.

Multiple histograms

Matplotlib provides functionality to visualize one histogram on top of another to compare different sets of data. The overlapping points of the two datasets can be clearly displayed using this technique.

```
import numpy as np
import matplotlib.pyplot as plt
# Generating random data
set1 = np.random.randn(200)
set2 = np.random.randn(150)
# ploting histograms
plt.hist(set1, density=True, color='red', alpha=0.6,bins=20)
plt.hist(set2, density=True, alpha=0.6,bins=20)
```

On **lines 7 & 8**, two `hist()` functions are used with the two datasets defined on **lines 4 & 5**. The `alpha` parameter introduced here makes the respective plots transparent to a specific percentage assigned to it. This can be set according to the needs of the user.

For more information on the `hist()` function, refer here.

Using seaborn

As *seaborn* is an extension of *matplotlib*, it provides a better way to visualize multiple sets of histogram data separately. The `jointplot()` function is used for this task. By default, it takes the two datasets as parameters. The following example uses this approach.

```
import numpy as np
import seaborn as sns
# Generating random data
set1 = np.random.randn(500)
set2 = np.random.randn(500)

sns1 = sns.jointplot(set1, set2)

sns2 = sns.jointplot(set1, set2, kind='hex')
```

Two plots are generated using the *seaborn* `jointplot()` function. Press the right arrow on the output to view the other plot.

arrow on the output to view the other plot.

On **line** 7, the simple `jointplot` function is called with the two datasets, and it creates a *scatter plot* by default. The two histograms are at the top and right, and the data is scattered in the center.

On **line 9**, a parameter `kind` is passed. It specifies what kind of plot we want to be displayed. By default, it displays the scatter plot. This time the `hex` type, which distributes the central data in a hexagonal form to better differentiate between data, is defined. The hex graph displays the distribution of data with dark color as high density and light colors as low density. The other types that can be assigned to the `kind` parameter are `scatter`, `reg`, `resid`, or `kde`.

For more information on the `jointplot()` function, refer here.

In the next lesson, box plots are discussed.