Section 2: Daily Returns

In this lesson, the daily returns of different stocks are calculated.

We'll cover the following

Daily returns

The price of stock changes on a daily basis. Daily return calculations tell us how much the current day stock value is different from the previous day stock value. A positive change indicates a rise in the value of the stock while a negative change indicates a fall in the value of the stock. A stock with minimal positive or negative change is considered to be a stable and good stock.

As seen in the previous lesson, only **Systems Ltd** and **Avanceon** stocks provide promising results for the year *2018*. So, daily returns of **Systems Ltd** will be calculated and visualized in this lesson as this company had a higher value compared to **Avanceon**. Results for any company can be obtained by simply changing the name for the company file as discussed before.

Calculating daily returns

We don't need to figure out the formula or steps for calculating the daily returns because `pandas` already provides a built-in function for it. The `pct_change` function is called from a `Series` object, and it calculates the daily return for all rows based on the current and previous row value. This function returns a new `Series` with the calculated daily returns. The first row has no, or `NaN`, value as there is no previous value for it.

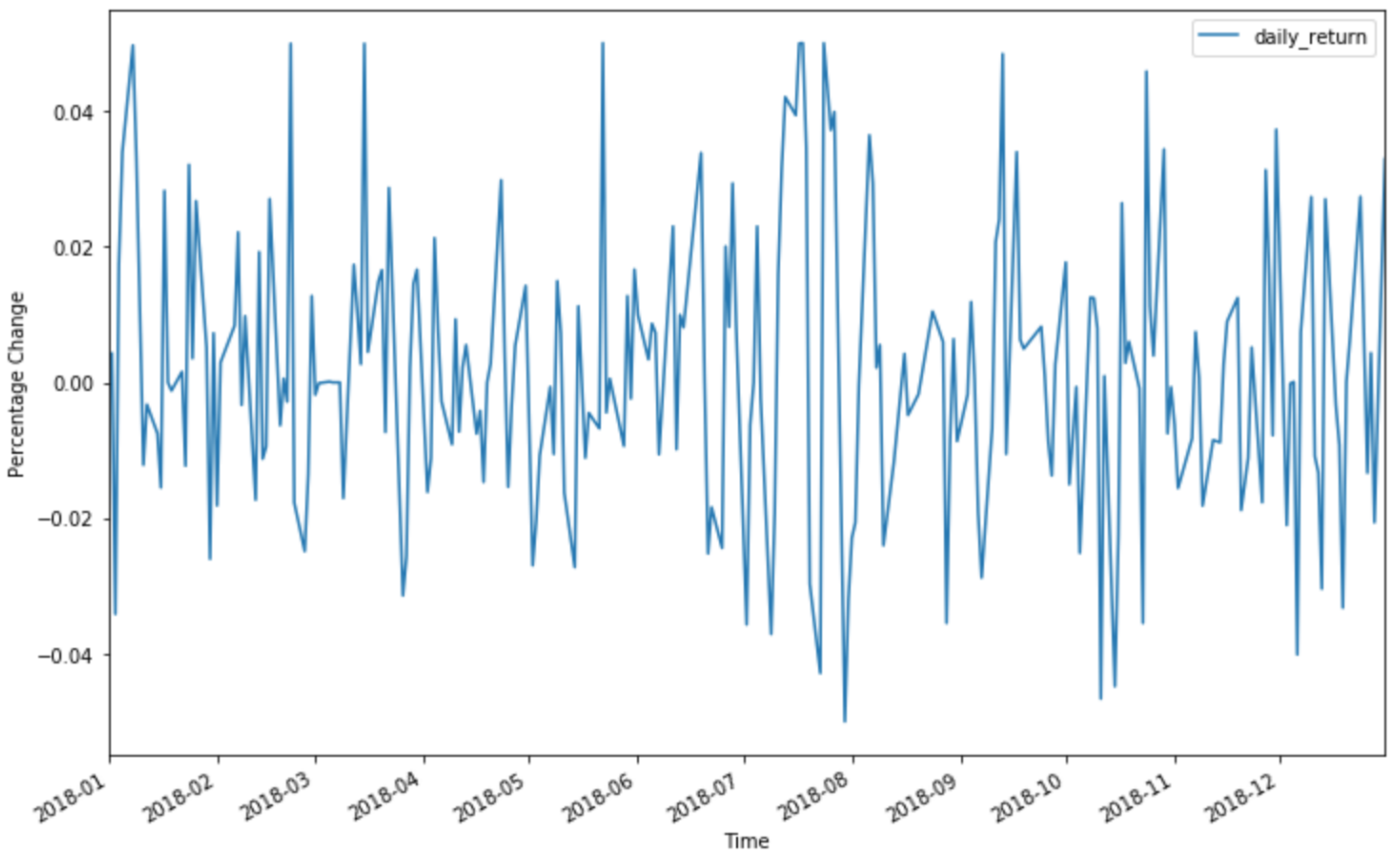
```
import pandas as pd
import matplotlib.pyplot as plt

sys = pd.read_csv('Year_2018/SYS.csv')
#The "SYS" file name can be changed to "NETSOL", "AVN" and "PTC",
sys['Time'] = pd.to_datetime(sys.Time) # correct the format of date
sys = sys.set_index('Time') # Set Time column as row index

daily_return = sys['Close'].pct_change() # Calculate the daily returns

sys['daily_return'] = daily_return # Create new column and assign daily return values to it

plt.ylabel('Percentage Change') # Assign a name to the y-axis of plot

sys['daily_return'].plot(legend = True, figsize=(15,10)) # plot the daily return values
```

On **line 9**, the `pct_change` function is called by the `Close` column. This returns a new `Series` in the `daily_returns` variable. The initial `DataFrame` values are sorted on the `Time`, i.e., the data is arranged in ascending order of days, so it calculates the correct daily return for each day.

On **line 11**, a new column named `daily_returns` is created in the `sys` dataframe and the `Series` variable `daily_returns` is assigned to it. Now, every date has the daily return of that day assigned to it.

On **line 13**, the function `plt.label` of *matplotlib* is used to assign a name to the y-axis.

On **line 15**, the new `daily_returns` column of the `sys` dataframe is plotted.



The output of the code snippet generates the above graph. The x-axis represents the time frames, and the y-axis represents the percentage change in daily stock values. It can be observed that the change in daily returns is between (-0.04%, 0.04%), which is not that big of a change.

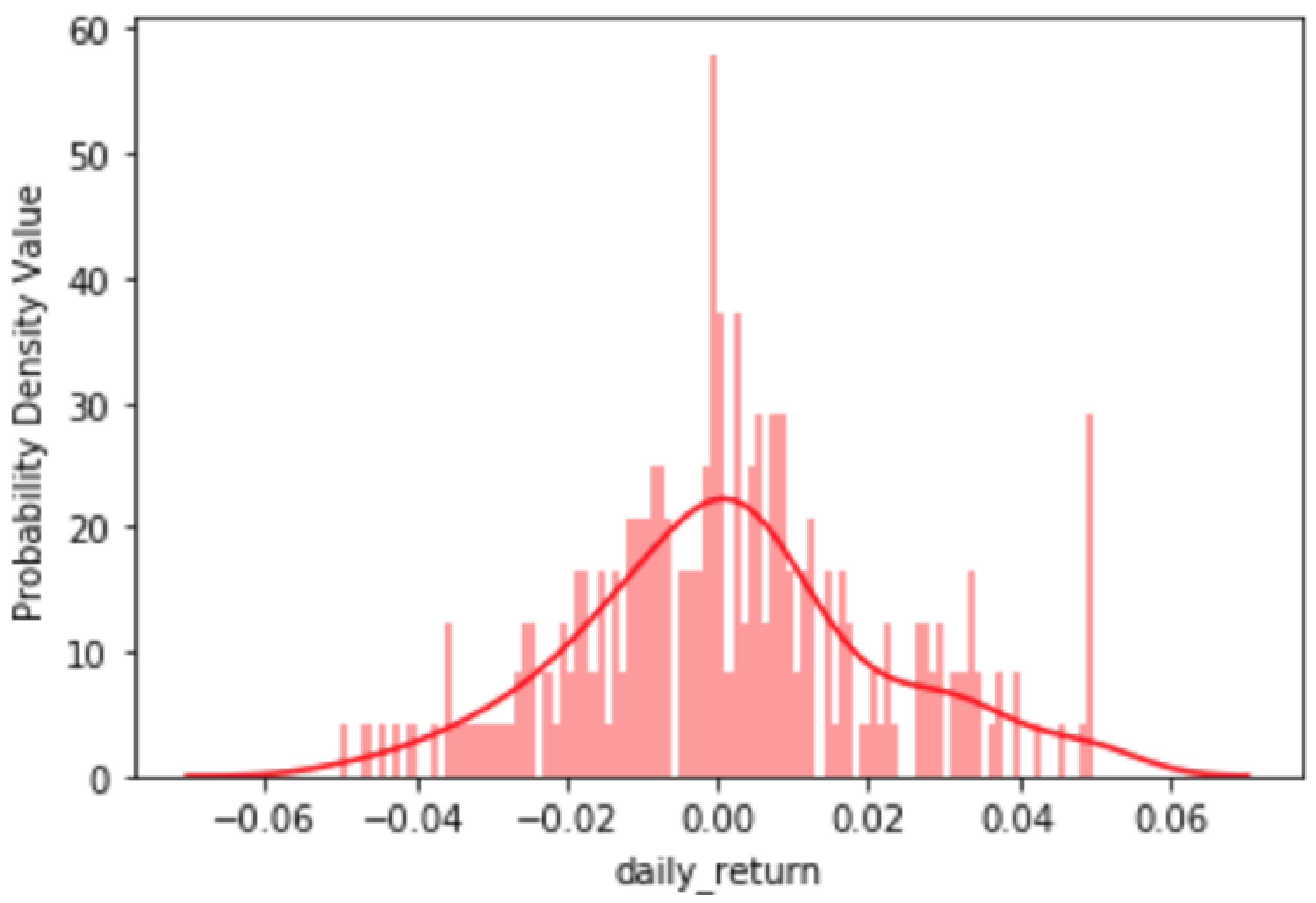
As mentioned above, the stock with minimal change is considered a stable stock. So, the **Systems Ltd** stock is a good stock to hold according to the daily returns.

Estimating daily return

Which daily return values are likely to occur using the probability density function can also be determined. This operation can easily be performed using the *KDE* plot, as mentioned in the previous lesson.

The *KDE* plot plots the probability density function values on the y-axis with respect to the values of the x-axis. The histogram can be used to determine the amount of those likely occurring values by dividing them into continuous intervals. Let’s visualize this with an example for better understanding.

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

sys = pd.read_csv('Year_2018/SYS.csv')
#The "SYS" file name can be changed to "NETSOL", "AVN" and "PTC".
sys['Time'] = pd.to_datetime(sys.Time) # correct the format of date
sys = sys.set_index('Time') # Set Time column as row index

daily_return = sys['Close'].pct_change() # Calculate the daily returns

sys['daily_return'] = daily_return # Create new column and assign daily return values to it

plt.ylabel('Probability Density Value') # Assign a name to the y-axis of plot

sns.distplot(sys['daily_return'].dropna(), bins = 100, color = 'red') # plots a distribution graph
```



On **line 16**, the `distplot()` function is used to plot a *KDE* over a histogram. The *KDE* is discussed extensively here.

The x-axis has daily return values. The y-axis values represent how likely the value on the x-axis occurs. The higher values mean more likelihood of a value occurring on the x-axis. In the coming lessons, we will determine the average daily return percentage. The histogram shows how many values each of the *one-hundred* bins contain. The y-axis here does not provide any information about the histogram.

The *KDE* plot informs us that most of the daily returns for **System Ltd** are close to **zero**. This tells us that the changes in price of this stock are not drastic and we can assume that **System Ltd** has a stable stock.

Try changing the file name to other company's names and observe how the plots change for them and what information can be extracted from them.

In the next lesson, we provide an explanation of how the stock behaviors of these different companies are related to each other using the correlation concepts.