Section 4: Risk Estimation

In this lesson, risk estimation for the stock of different companies is discussed.

We'll cover the following

Risk

For the scope of our project, the *risk* relates to the amount of capital we could lose on our investment on a daily basis.

The daily risk for the companies stock prices can be calculated by taking the standard deviation of the daily returns. The daily returns and risks can be visualized using a scatter plot, which can give us a better understanding of the return vs. risk ratio of each company's stock.

Let's first calculate the average daily return and the risk. Here the risk is the standard deviation of all the daily return values. As mentioned in the statistical features lesson, the *standard deviation* measures how far the values are from the mean or average value. In our case, this distance from the mean value is the value or amount that we are putting at risk by investing in a certain stock.

To perform the following steps, all the preprocessing of reading the companies as variables and setting `Time` as the index needs to be done again. So, for simplicity, these steps have been hidden in the following example.

```
ret = all_returns.dropna() # drop the null values

avg_daily_return = ret.mean() # Take mean of the daily return of all companies
print("Averge daily return of companies\n", avg_daily_return)

daily_risk = ret.std() # Take standard deviation of the daily return of all companies
print("\nDaily Risk or standard deviation of companies\n", daily_risk)
```

On **line 1**, the null values are dropped using the `dropna()` function.

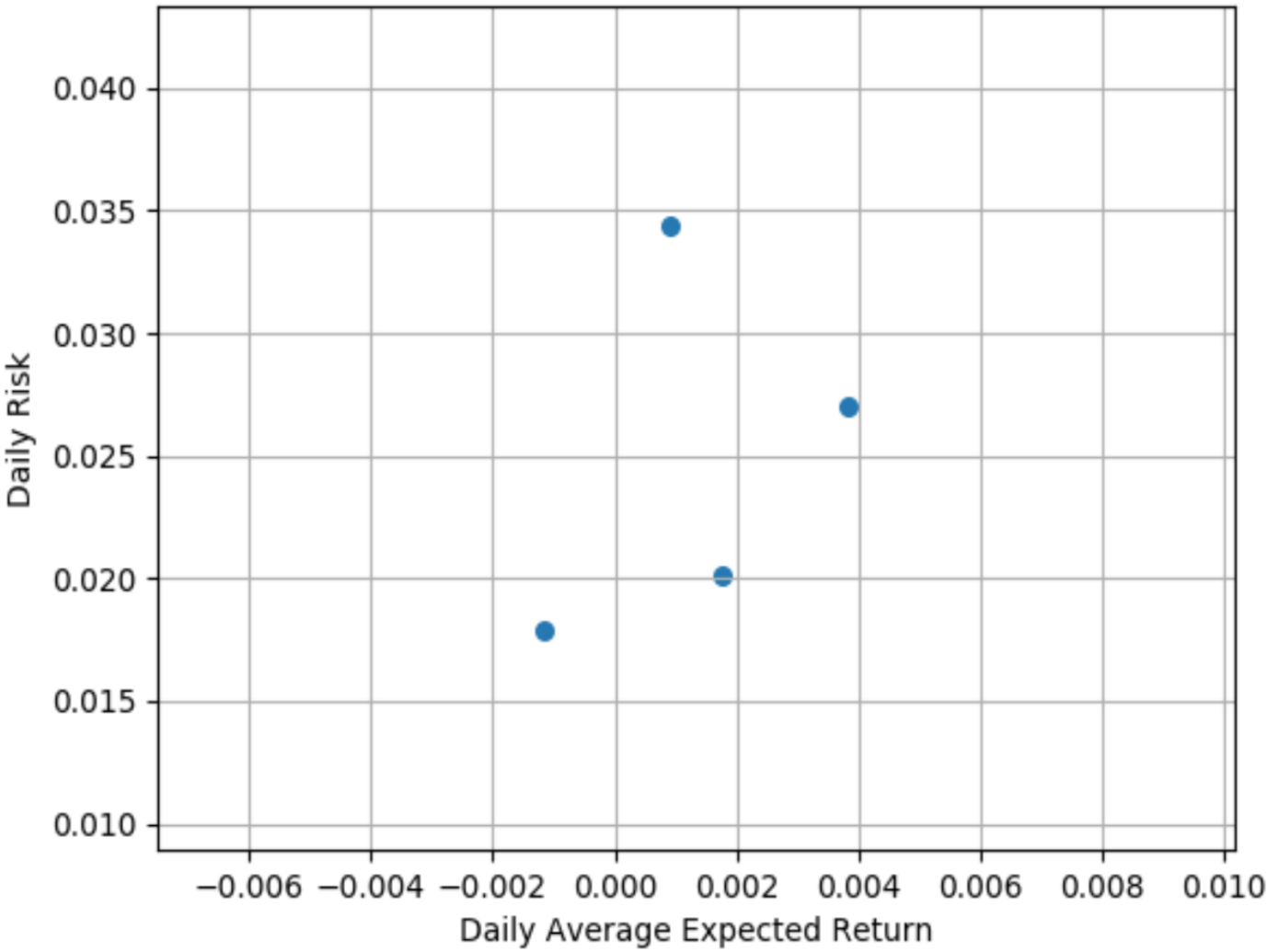
On **line 3**, the built-in `mean()` function is used to calculate the average of each column.

On **line 6**, the built-in `std()` function is used to calculate the standard deviation of each column.

Risk/Return plot

Now that *std* and *mean* are calculated, the scatter plot representing the risk and return can be plotted.

```
ret = all_returns.dropna() # drop the null values

avg_daily_return = ret.mean() # Take mean of the daily return of all companies
daily_risk = ret.std() # Take standard deviation of the daily return of all companies

plt.xlabel("Daily Average Expected Return") # Name the x-axis
plt.ylabel("Daily Risk") # Name the y-axis

plt.grid() # Add grid lines on the plot

plt.scatter(avg_daily_return, daily_risk, s = 30) # Plot the scatter plot for risk & return
```



The above plot is obtained after running the program.

On **lines 6 & 7**, the names of x-axis and y-axis are defined using the `xlabel()` and `ylabel()` functions, respectively.

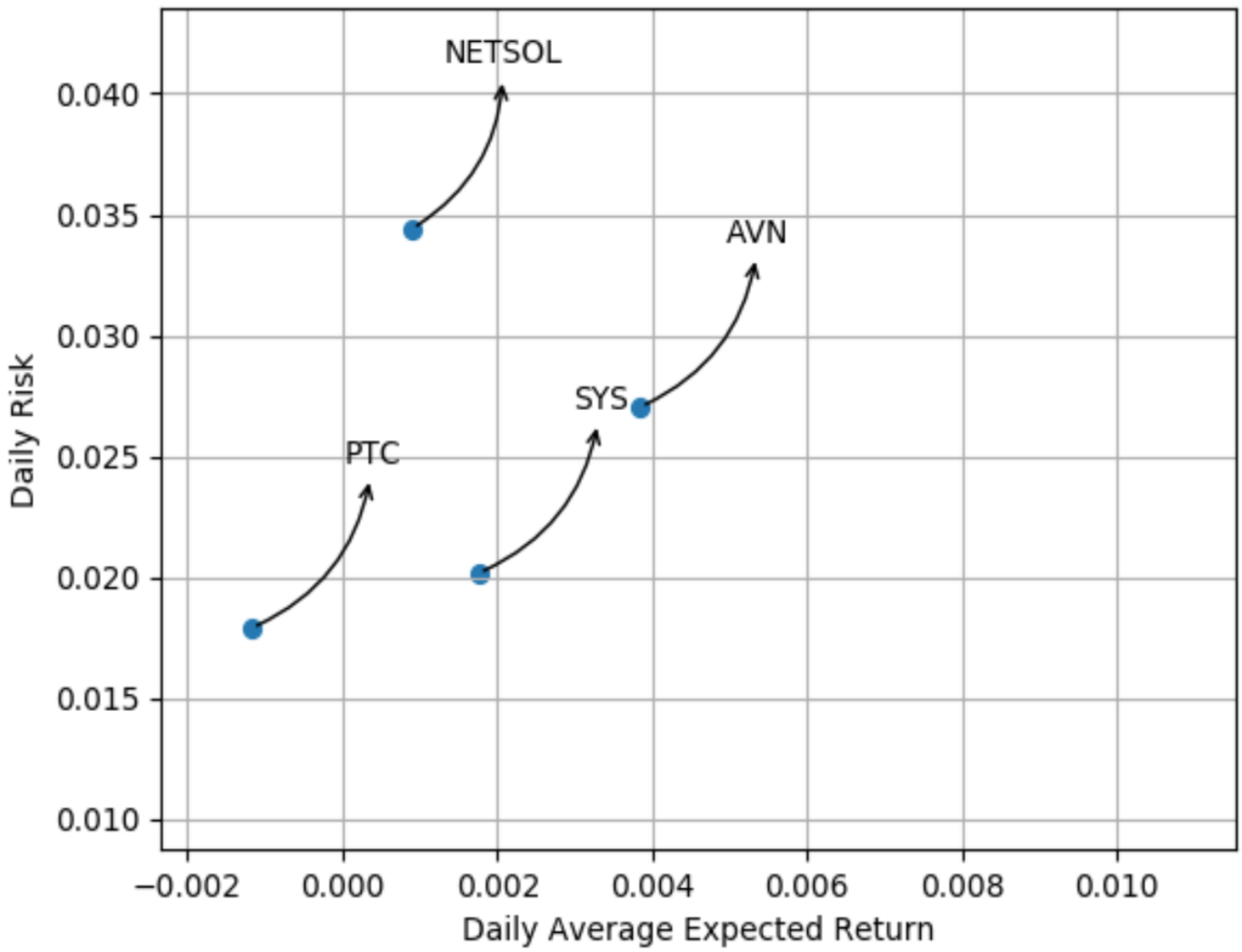
On **line 9**, grid lines are drawn on the graph to better assess the position of points. The `grid()` function is used for it.

On **line 11**, the `scatter()` function of matplotlib is used to draw a scatter plot with the average daily returns on the x-axis and standard deviation on the y-axis as the daily risk value. The `s` parameter defines the size of the dots in the plot.

It looks like the plot is missing something. Names of companies are not associated with the dots on the plot. Let's add some annotations to make it visually understandable.

```
ret = all_returns.dropna()

avg_daily_return = ret.mean()
```

```
daily_risk = ret.std()

plt.xlabel("Daily Average Expected Return")
plt.ylabel("Daily Risk")

plt.xlim(ret.mean().min() + ret.mean().min()*2, ret.mean().max() + ret.mean().max()*2)

for label, x, y in zip(ret.columns, ret.mean(), ret.std()):
    plt.annotate(
        label,
        xy = (x,y), xytext = (50, 50),
        textcoords = 'offset points', ha = 'right', va = 'bottom',
        arrowprops = dict(arrowstyle = '<-', connectionstyle = 'arc3,rad=-0.4'))

plt.grid()

plt.scatter(avg_daily_return, daily_risk, s = 30)
```



Now, the above plot can provide some relevant information about how much risk a particular stock holds for how much of an average return.

The code on **line 9** only extends the value limits of the x-axis.

The code on **lines 11-16** adds the relevant names and lines while structuring them. This part needs to be custom made for every instance of data so that if this code runs on more data or for different companies, this structure is not maintained. For information on how to make your own custom design to represent data, refer here.

Results

So, from the above graph, the following results can be inferred:

1. **NETSOL** stock has a little positive daily return and a very high risk, which means the loss is also high.
2. **Avanceon** has a high positive daily return, and the risk value is also less than **NETSOL**.
3. **Systems Ltd** has a positive daily return, but it is less than **Avanceon**. The risk value is less than **NETSOL** and **Avanceon** so it can be considered a good stock.
4. The **PTC** stock has the lowest risk value, but its average daily return value is negative. This indicates that even though our losses would be little, there would not be a positive return for our investment.

These results support the assumptions we made in the previous lesson that **Systems Ltd** and **Avanceon** stocks are better for investments than others.

Quantiles

This method can also be used to assess the risk associated with a stock. This method can determine the maximum loss over an investment with high certainty. The quantile method is explained here.

Fortunately, *pandas* `DataFrame` provides the `quantile(n)` function to calculate the value at a given quartile `n`. Let's discuss this using an example.

```
df = pd.DataFrame({'SYS': sys['Close'],
                   'NETSOL': ns['Close'],
                   'PTC': ptc['Close'],
                   'AVN': avn['Close']})

all_returns = df.pct_change()

investment = 100000

loss = (abs(all_returns.quantile(0.1))) * investment
```

```
loss = (abs(all_returns.quantile(0.1))) * investment
print(loss)
```

The new `DataFrame` is created with only the `Close` prices, and their daily returns are calculated using the `pct_change()` method.

Now, let's suppose we want to invest **100,000** in some stocks and want to know what our maximum loss for a day is. On **line 10**, the `quantile()` function is used with a parameter value of `0.1`. The absolute values returned from the `quantile()` function are then multiplied with our initial investment. The `0.1` value gives us the loss values with **90%** accuracy. More on quantiles are explained here.

In the output, four values are obtained for each company. Each number signifies that the total loss for a single day will not exceed this value. Considering our initial investment, this loss is not very much.

It can also be observed that if sorted, the number of values at risk is in the same order as detected by the scatter plot. **NETSOL** has the highest loss and **PTC** has the lowest loss value with other companies between them.

In the next lesson, you will try to predict future stock behavior.