This Model helps planning to improve the existing algorithmic trading systems and maintain the firm's competitive advantage in the market. To do so, we will enhance the existing trading signals with machine learning algorithms that can adapt to new data.

Instructions:

Use the starter code file to complete the steps that the instructions outline. The steps for this Challenge are divided into the following sections:

- Establish a Baseline Performance
- Tune the Baseline Trading Algorithm
- Evaluate a New Machine Learning Classifier
- Create an Evaluation Report

Establish a Baseline Performance

In this section, we will run the provided starter code to establish a baseline performance for the trading algorithm. To do so, complete the following steps.

Open the Jupyter notebook. Restart the kernel, run the provided cells that correspond with the first three steps, and then proceed to step four.

1. Import the OHLCV dataset into a Pandas DataFrame.
2. Generate trading signals using short- and long-window SMA values.
3. Split the data into training and testing datasets.
4. Use the `SVC` classifier model from SKLearn's support vector machine (SVM) learning method to fit the training data and make predictions based on the testing data. Review the predictions.
5. Review the classification report associated with the `SVC` model predictions.
6. Create a predictions DataFrame that contains columns for "Predicted" values, "Actual Returns", and "Strategy Returns".
7. Create a cumulative return plot that shows the actual returns vs. the strategy returns. Save a PNG image of this plot. This will serve as a baseline against which to compare the effects of tuning the trading algorithm.
8. Write your conclusions about the performance of the baseline trading algorithm in the `README.md` file that's associated with your GitHub repository. Support your findings by using the PNG image that you saved in the previous step.

Tune the Baseline Trading Algorithm

In this section, we will tune, or adjust, the model's input features to find the parameters that result in the best trading outcomes. (We will choose the best by comparing the cumulative products of the strategy returns.) The adjusted size of the trading signals dataset SMA window

Short_Window = 4 Long_Window = 150 Ending Period = 4 months

To do so, complete the following steps:

1. Tune the training algorithm by adjusting the size of the training dataset. To do so, slice your data into different periods. Rerun the notebook with the updated parameters, and record the results in your README.md file. Answer the following question: What impact resulted from increasing or decreasing the training window?

Hint To adjust the size of the training dataset, you can use a different DateOffset value—for example, six months. Be aware that changing the size of the training dataset also affects the size of the testing dataset.

2. Tune the trading algorithm by adjusting the SMA input features. Adjust one or both of the windows for the algorithm. Rerun the notebook with the updated parameters, and record the results in your README.md file. Answer the following question: What impact resulted from increasing or decreasing either or both of the SMA windows?

3. Choose the set of parameters that best improved the trading algorithm returns. Save a PNG image of the cumulative product of the actual returns vs. the strategy returns, and document your conclusion in your README.md file.

Evaluate a New Machine Learning Classifier

In this section, you'll use the original parameters that the starter code provided. But, you'll apply them to the performance of a second machine learning model. To do so, complete the following steps:

1. Import a new classifier, such as AdaBoost, DecisionTreeClassifier, or LogisticRegression. (For the full list of classifiers, refer to the Supervised learning page in the scikit-learn documentation.)

2. Using the original training data as the baseline model, fit another model with the new classifier.

3. Backtest the new model to evaluate its performance. Save a PNG image of the cumulative product of the actual returns vs. the strategy returns for this updated trading algorithm, and write your conclusions in your README.md file. Answer the following questions: Did this new model perform better or worse than the provided baseline model? Did this new model perform better or worse than your tuned trading algorithm?

Create an Evaluation Report

In the previous sections, you updated your README.md file with your conclusions. To accomplish this section, you need to add a summary evaluation report at the end of the README.md file. For this report, express your final conclusions and analysis. Support your findings by using the PNG images that you created.

Summary/Answer

After changing the parameters found that the new model performed better than the provided baseline. The new model performed similar t0 the tuned trading algorithm.

Step 5: Review the classification report associated with the `SVC` model predictions.

[17]:
```
# Use a classification report to evaluate the model using the predictions and testing data
svm_testing_report = classification_report(training_signal_predictions, y_test)

# Print the classification report
print (svm_testing_report)
```

```
              precision    recall  f1-score   support

        -1.0       0.10      0.44      0.16       389
         1.0       0.90      0.56      0.69      3620

    accuracy                           0.55      4009
   macro avg       0.50      0.50      0.42      4009
weighted avg       0.82      0.55      0.64      4009
```

[19]:
```
# Plot the actual returns versus the strategy returns

(predictions_df[['Actual Returns', 'Strategy Returns']] + 1).cumprod().plot(figsize=(20,10));
```