Technical Indicators:
The TIs I use in this homework are MA and RSI of window sizes 5, 10, 20, and 50.
I divide each day's adj closing price with the MA to obtain a percentage indicating if the stock price is relatively high or low compared to the previous [window] days. I chose these TIs because we are going to predict prices for the next 50 days, so I think more recent data is more important. I also don't use other TIs because I think there we don't have a large amount of data, so using too many TIs may result in overfitting.

In this homework, I adopted the Random Forest Classifier decision model to predict the future stock price will increase or decrease. The "features" are the MA and RSI indicators, and the "target" is a binary value 1/0, which indicates if tomorrow's price went up or down. In this homework, I used the predict_prob() function and the model will return an array that represents [probability that price will go down, probability that price will go up]. Then, I use two thresholds to determine the final action. Hence, the final model will use the features to predict a set of actions (-1, 0, 1).

The hyperparameters of my models are:
1. Threshold_low: if the probability that price will go up < threshold_low, the model will return -1 (sell).
2. Threshold_high: if the probability that price will go up > threshold_high, the model will return 1 (buy). Hence, if the probability that price will go is between threshold_low and threshold_high, it will return 0 (hold).
3. N_estimators: the number of decision trees in the forest
4. Max_depth: the maximum depth of each decision tree
5. Min_samples_split: the min number of samples required to split an internal node.
6. Min_samples_leaf: the min number of samples required to be at a leaf node.

I use exhaustive search to optimize the hyperparameters. For every combination, I use the last 50 days (since 10/16-12/22 is also approximately 50 days) from the csv file as the test data, and calculate the return rate of these 50 days using myrrestimate.py. Then, I find the model with the best return rate and save the model to a pkl file.

myStrategy.py: In this file, I will load the model with the optimized hyperparameters and use it to predict today's actions given today's price.

Other things to optimize my strategy: I tried using the dp taught in class to obtain the optimized set of actions based on the historical stock price, and use them as the target to train the model. However, I found out that this model is usually worse when compared with the model that just predicts price will go up or go down, and it returns negative return rates in many cases. Hence, eventually I just adopted the latter approach.