Machine Learning

Setup

As my data is time-dependent, I can not split into train and test set using the scikit-learn module. Instead, I use:

- Train data: dataset from the beginning of 2000 to the end of 2019
- Test data: dataset from the beginning of 2020 to present

Even though at the moment the test dataset is relatively small compared to the train dataset, it will be updated continuously.

In order to start, I use 3 different features (Volume, Effective Federal Interest Rates, and EUR/USD Exchange Rates), which can be expanded later. These features are assigned to the X_train variable.

The second variable that needs to be set up is the y_train variable. Since the goal of this algorithm is to beat the market (in this case the SP500), I copied a function from robertmartin8's github page.

```
def status_calc(stock, sp500, outperformance):
    """A simple function to classify whether a stock outperformed the S&P500
    :param stock: stock price
    :param sp500: S&P500 price
    :param outperformance: stock is classified 1 if stock price > S&P500 price + outperformance
    :return: true/false
    """
    if outperformance < 0:
        raise ValueError("outperformance must be positive")
    return stock - sp500 >= outperformance
```

The function checks if the stock performance on any given day is higher than the market on the same day by the outperformance constant (any number >0 where 1% is 0.01). Thus, setting up the y_train with the following line code.

```
y_train = status_calc(disney_.Change, sp500_.Change, 0.01)
```

Method

For this project, I use the RandomForestClassifier method from scikit learn library. The challenge is to determine how many clusters are appropriate for the model. I created a function to test different n clusters to see which has the best accuracy score. The function does:

1. Initialize the RandomForestClassifier module
2. Fit the training data
3. Calculate the accuracy score
4. Predict the decision

Below are the results of the function with estimator of 1, 2, 3, 4, 5, 10, 50, 100, 200 and 500.

Estimator	Accuracy	Transactions
1	0.829787	12
2	0.808511	5
3	0.787234	8
4	0.787234	6
5	0.787234	6
10	0.808511	9
50	0.765957	11
100	0.765957	11
200	0.765957	11
500	0.765957	11

As we can see, the highest accuracy is the one with estimator 1, but it will overestimate the number of transactions which may lead to wrong decision to purchase the stock. 2 clusters will also give a good accuracy number, but it underestimates the number of transactions. In this case, however, underestimating is better than overestimating. In my opinion, the best method will be to use 10 clusters using the 3 features mentioned above. If additional features are added, I will then check again if 10 are still the best clusters.

Results

By using 10 clusters, we can see that in 2020 (47 trading days), the algorithm predicts that 9 of those days are good to buy Disney stock and sell it the next day. The dates are as follow:

1. January 14, 2020
2. February 4, 2020
3. February 24, 2020
4. February 25, 2020
5. March 2, 2020
6. March 3, 2020
7. March 5, 2020
8. March 6, 2020
9. March 10, 2020

Reference

https://github.com/robertmartin8/MachineLearningStocks