Project report

Tang Ho Chun

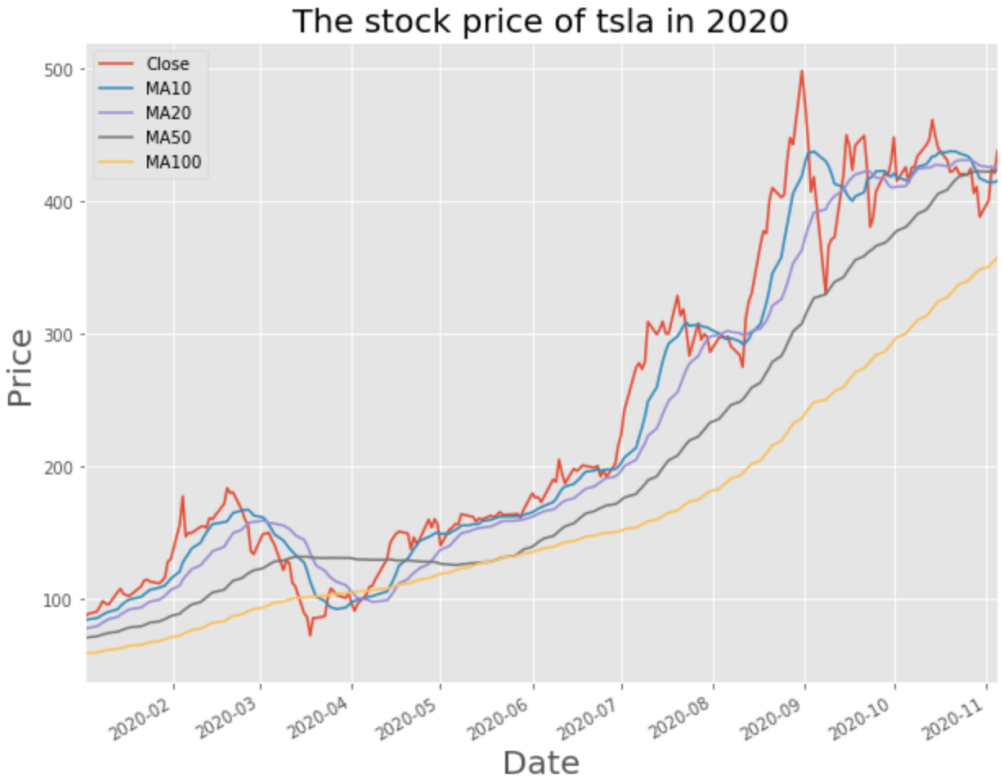
Objective

This project is to evaluate the performance of the stock using moving average strategy and machine learning.

Strategy

To start with, the strategy in this project is to buy the stock when the moving average (MA 10) > (MA 20) and (MA 50) > (MA 100), and sell the stock when the moving average (MA 10) < (MA 20) and (MA 50) < (MA 100). We would apply this strategy to the historical stock data of 'TSLA', a fast-growing electrical vehicle company in US. Our investigation period starts from 2010-11-17 to 2020-11-06.

We would use the library in python, pandas_datareader, and collect the historical data of 'TSLA' from 'Yahoo'.



Data preparation

```
def stock_info(stock):
    stock_ = web.DataReader(stock,'yahoo',start,end)
    stock_['Price'] = stock_['Close'].shift(-1)
    stock_['PriceDiff'] = stock_['Price'] - stock_['Close']
    stock_['Return'] = stock_['PriceDiff']/stock_['Close']
    stock_['Direction'] = [1 if stock_.loc[ei,'PriceDiff']>0
                  else -1 for ei in stock_.index]
    stock_['MA10'] = stock_['Close'].rolling(10).mean()
    stock_['MA20'] = stock_['Close'].rolling(20).mean()
    stock_['MA50'] = stock_['Close'].rolling(50).mean()
    stock_['MA100'] = stock_['Close'].rolling(100).mean()
    stock_['Strategy'] = [1 if (stock_.loc[ei,'MA10']> stock_.loc[ei,'MA20']
                    and stock_.loc[ei,'MA50']> stock_.loc[ei,'MA100'])
                  else 0 for ei in stock_.index]
    stock_['Profit'] = [stock_.loc[ei,'Price']-stock_.loc[ei,'Close'] if stock_.loc[ei,'Strategy']==1 else 0
            for ei in stock_.index]
    stock_['Wealth'] = stock_['Profit'].cumsum()
    stock_ = stock_.dropna()
    return stock_
    return stock_.shape

stock_info('tsla')
```

We first calculate the price difference for each day. That is the close price of the next day minus the close price of today. When the price difference greater than 0, we set the direction = 1. When the that smaller than 0, we set the direction = -1. Then, we calculate the return of the stock of each day by using the price difference divide the close price. After that, we calculate the strategy. We set the strategy = 1 when MA10 > MA20 and MA50 > MA100, and strategy = 0 when MA10 < MA20 and MA50 < MA100.

Stock return

Base on the strategy, we can calculate the cumulative sum for the whole investigation period. The total gain is USD 364.836. As seen below, the total gain increased a lot during 2020. This is due to the huge increase of stock price, increased from about $50 to $500 in 2020. This can show that our strategy is applicable when the stock price increase.

Model training:

```
def stock_info2(stock):
    stock_ = stock_info(stock)
    global future_day
    future_day = 100
    stock_['Prediction'] = stock_['Close'].shift(-future_day)
    new_df = stock_[['Close','Prediction']]
    return (new_df)

stock_info2('tsla')
```

Date	Close	Prediction
2010-11-17	5.898000	4.930
2010-11-18	5.978000	4.986
2010-11-19	6.198000	5.028
2010-11-22	6.680000	5.116
2010-11-23	6.914000	5.006
2010-11-24	7.094000	5.032
2010-11-26	7.064000	5.150
2010-11-29	6.866000	5.348
2010-11-30	7.066000	5.278
2010-12-01	6.870000	5.386

To train the model, we build a new function, stock_info2. Inside this function, we set the future_day = 100. This is used for the model to predict the stock price in the next 100 days. Then, we create a new dataframe, which contains the close price and the predicted close price in the next 100 days.

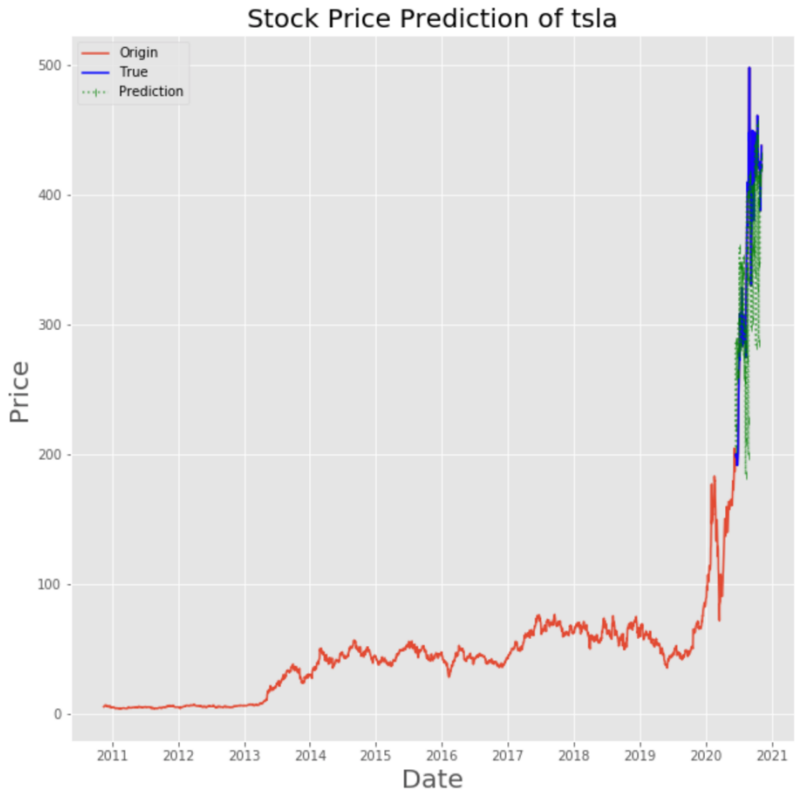
```
def model_prediction_plot(stock):

    stock_ = stock_info2(stock)
    feature = np.array(stock_.drop(['Prediction'],1))[:-future_day]
    target = np.array(stock_['Prediction'])[:-future_day]

    X_train, X_test, y_train, y_test = train_test_split(feature,target,test_size = 0.2, random_state = 0)
    x_future = stock_.drop(['Prediction'],1)[:-future_day]
    x_future = x_future.tail(future_day)
    x_future = np.array(x_future)

    forest = RandomForestRegressor(n_estimators = 100, random_state = 10)
    forest.fit(X_train,y_train)
    print(f'the training accuracy: {forest.score(X_train,y_train)}')
    print(f'the testing accuracy: {forest.score(X_test,y_test)}')
    tree_prediction = forest.predict(x_future)

    stock_ = stock_info2(stock)
    true = stock_[feature.shape[0]:]
    true['Prediction'] = tree_prediction
    plt.figure(figsize = (10,10))
    plt.plot(stock_['Close'])
    plt.plot(true['Close'], color = 'blue')
    plt.plot(true['Prediction'], color = 'green', linestyle=':', marker='|', alpha = 0.65)
    plt.title(f'Stock Price Prediction of {stock}',size = 20)
    plt.xlabel('Date',size = 20)
    plt.ylabel('Price',size = 20)
    plt.legend(['Origin', 'True', 'Prediction'], loc = 'best')
    plt.show()

model_prediction_plot('tsla')
```

```
training accuracy: 0.96996539766711
testing accuracy: 0.8136046945967637
```

After that, we create a new function for building the model and plotting the predicted stock. We first use the ‘feature’ variable to store the close price of the stock before the last 100 days and ‘target’ variable to store the predicted price of the stock before the last 100 days as well. This is because the remaining 100 days are ‘nan’ value so that we should not include it in ‘feature’ and ‘target’. The ‘x_future’ is the last 100 days of the ‘feature’ for the model prediction.

Then, we should split the data into training set and testing set using train_test_split. In this project, we used random forest regressor as the model to evaluate the dataset. The reason of choosing random forest is that the performance of this model is much better than decision tree regressor and produce a higher accuracy. We use the training set to fit and train the model. In this project, we set n_estimators to 100. This is to create 100 trees randomly and predict the highest probability. As we can see that, the training accuracy of the model is 97.0% and the testing accuracy is 81.4%

Finally, we use the model to predict the stock price of 'x_future' and store the predicted price of the last 100 days to 'tree_prediction'. And then we set the variable, 'true', to store the true stock price of the last 100 days.



As we can see that, the red line is the original stock price of 'TSLA'. The blue line is the true stock price of 'TSLA' in the last 100 days, while the dotted green line is the predicted stock price in the last 100 days. We can see that the model can predict the increasing trend of the stock in these 100 days. This shows that the model can be used to predict the trend of the stock in long run but there is some minor error in short run.

Comparison

We try to compare the return of different stock using this strategy. As seen below.

With the use of moving average strategy, the return is USD 364 for 'TSLA' and USD 1704.51 for 'AMZN'. However, the return is USD -78 and USD 7.84 for 'GOOG' and 'FB' respectively. This is because the stock price of 'TSLA' keep increasing in 2020 and that of 'AMZN' keep increasing from 2015 to 2020. However, the stock price of 'GOOG' and 'FB' are fluctuated and there is no significant long-term increase. This shows that this strategy performs well on some long-term increasing stock.

Conclusion

Finally, we make use all the data from 'feature' and 'target'. The accuracy of the final model is about 97%. In this report, we apply the stock data on random forest regressor to predict the stock price in long run. Since the strategy perform well on some long-term increasing stock, we can apply the strategy with the machine learning model together. We can utilize machine learning to predict the stock price in future and figure out some stock that can keep increasing. Then, we can apply the moving average strategy on those long-term increasing stock that we predicted using machine learning.