INDEX

- Overview of the Project
- Getting the S&P ticker data
- Getting the stock price data
- ML model for Stock Price Prediction
- Conservative or Aggressive Strategy
- Putting Individual Pieces Together

Overview:

Before diving into any project, you should have a very clear sense of what the final output would look like, otherwise, your steps may not be according to what the project demands. So, here is **a video link** where I demonstrate this project visually so that you will have some motivation before proceeding with this tutorial.

Getting the S & P ticker data

The Standard and Poor's 500, aka S&P 500 is an American index that tracks the performance of the top 500 large American corporations.

Each corporation has been assigned a symbol (also known as ticker) which is usually made up of its name only. When we query the yahoo finance API to get the historical data of the prices of these corporations, we would need these tickers.

Getting the list of these tickers is quite easy and available in the public domain. There is also a Wikipedia page which list down all of these 500 corporations with its name, ticker name and other useful pieces of information.

We want to be able to import this data in python so that the process of getting the data pertaining to one ticker can be obtained from Yahoo finance API can be automated. To download this data free of cost, please visit this **link** and download the file name constituents.csv file.

If you have any difficulty downloading this data, you can refer to this **Github link.**

TASK FOR YOU

1. Try to get a similar list not for S&P companies but for NYSE Listed companies.
2. (EXTRA CHALLENGE) If part 1 was successful, try to execute all the below steps not with S&P but with NYSE.

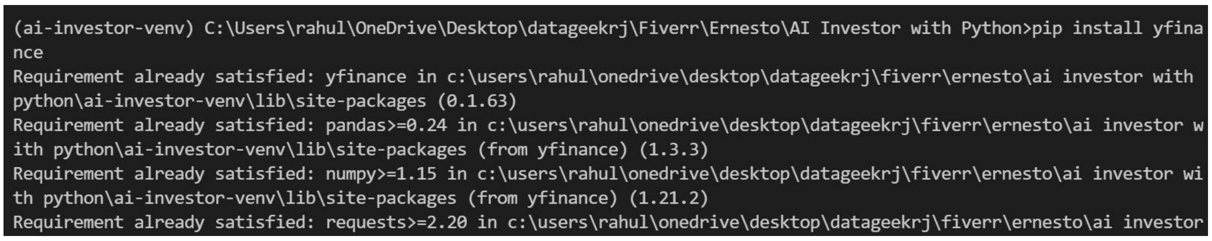
Getting the stock prices using yfinance

There was a time when Yahoo was officially supporting historical data API, life used to be easy for those who wanted to build applications that relied on financial data on stock prices but this was decommissioned and then people started looking at some paid alternatives.

Luckily many other people got together and create an unofficial python API called YFINANCE which almost does everything which the previous official version was capable of doing.

1. Installing the YFINANCE Library

We can install this library in many different ways. One of our simplest friends is obviously pip with which we can simply run the command “PIP INSTALL YFINANCE” on our command prompt and get this package in our system. But then there are other ways like building the package from the wheel from scratch. Here is a screenshot of the command which install the package:



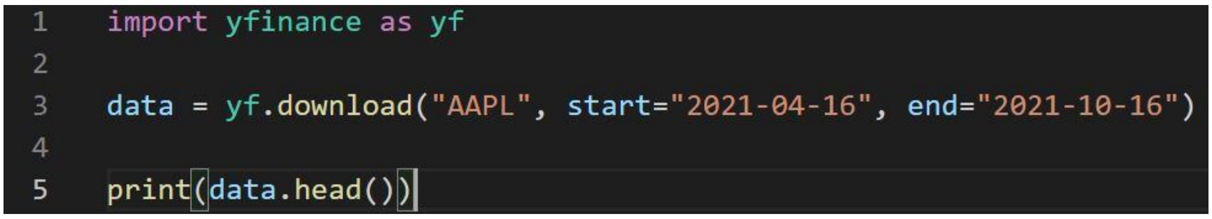
```
(ai-investor-venv) C:\Users\rahul\OneDrive\Desktop\datageekrj\Fiverr\Ernesto\AI Investor with Python>pip install yfina
nce
Requirement already satisfied: yfinance in c:\users\rahul\onedrive\desktop\datageekrj\fiverr\ernesto\ai investor with
python\ai-investor-venv\lib\site-packages (0.1.63)
Requirement already satisfied: pandas>=0.24 in c:\users\rahul\onedrive\desktop\datageekrj\fiverr\ernesto\ai investor w
ith python\ai-investor-venv\lib\site-packages (from yfinance) (1.3.3)
Requirement already satisfied: numpy>=1.15 in c:\users\rahul\onedrive\desktop\datageekrj\fiverr\ernesto\ai investor wi
th python\ai-investor-venv\lib\site-packages (from yfinance) (1.21.2)
Requirement already satisfied: requests>=2.20 in c:\users\rahul\onedrive\desktop\datageekrj\fiverr\ernesto\ai investor
```

2. GETTING THE STOCK PRICES DATA

Once we have the package installed in our system, we want to use this package to get the stock prices. To download this data, we need to have the following three pieces of information:

1. The symbol (ticker) of the corporation
2. The start date
3. The end date

Let us look at a program and then we will discuss what it does:



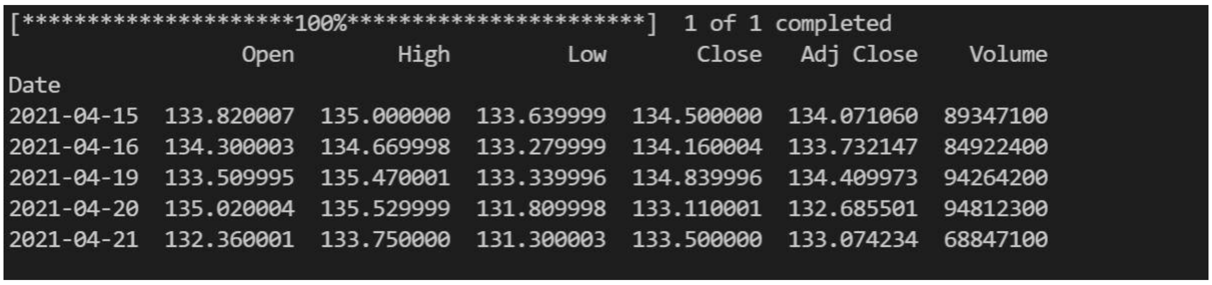
```python
import yfinance as yf

data = yf.download("AAPL", start="2021-04-16", end="2021-10-16")

print(data.head())
```

In the first line, we first import the library yfinance with an alias called “yf” and then we use a function download from this library. As already discussed, we have passed three pieces of information:

1. “AAP” ticker for getting the price data for Apple
2. “Start” is for starting date which is at 16th April 2021.
3. “end” is for the ending date which is 16th Oct’ 2021.

Notice here that we are getting the data for the past 6 months.

Finally, we are printing the result that we are getting from this query by calling the head method of data. This is what the output looks like:



```
[*********************100%***********************]  1 of 1 completed
                  Open        High         Low       Close   Adj Close    Volume
Date
2021-04-15  133.820007  135.000000  133.639999  134.500000  134.071060  89347100
2021-04-16  134.300003  134.669998  133.279999  134.160004  133.732147  84922400
2021-04-19  133.509995  135.470001  133.339996  134.839996  134.409973  94264200
2021-04-20  135.020004  135.529999  131.809998  133.110001  132.685501  94812300
2021-04-21  132.360001  133.750000  131.300003  133.500000  133.074234  68847100
```

It gives us day-wise day information on Open, High, Low and Close prices. We will be interested in only the close prices. The information on Volume is also given.

We will use this method of data collection in the final section when we put the whole project together where we will put the download method in a loop. Let’s move on to ML section.

TASK FOR YOU

1. Calculate the market capitalisation of apple using the above data.
2. (**EXTRA CHALLENGE**) Calculate the market capitalisation of all the S&P 500 companies and list the top 5 companies based on market capitalisation.

ML Model

ML model is required when we want to predict the price of the next day. So, the idea is that:

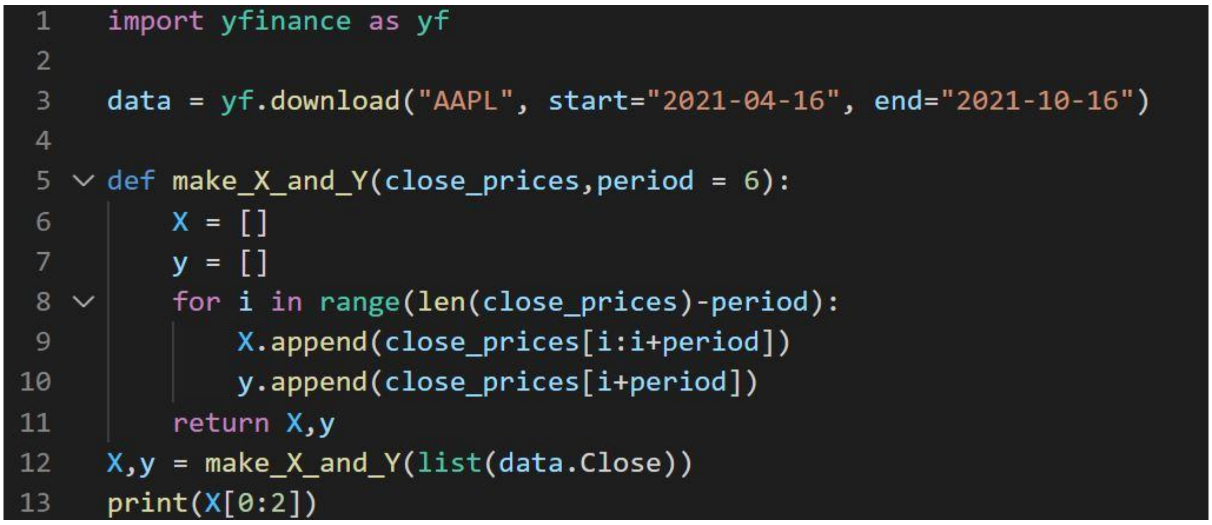
1. we collect the day-wise data of the stocks for the past 6 months and
2. based on that we try to predict what would be the price tomorrow.
3. If the price tomorrow is higher, which means today the stock can be purchased at a lower price and tomorrow it can be sold at a much higher price.

We understand that the stocks do not always work that simple but it gives us an overall idea.

1. Turing the Closing Price data into a Regression data

We will convert our time series data on close prices of stock into regression data which means that we take let say first 6 closing prices (let's call this 6 as n) and then call the 7th price as a dependent variable which we want to predict. Similarly, we take the prices from 2 to 7th day and predict the 8th day's price. In this manner, we will have data that will consists of n features (6 in this case) and against each row, we will have a value to predict.

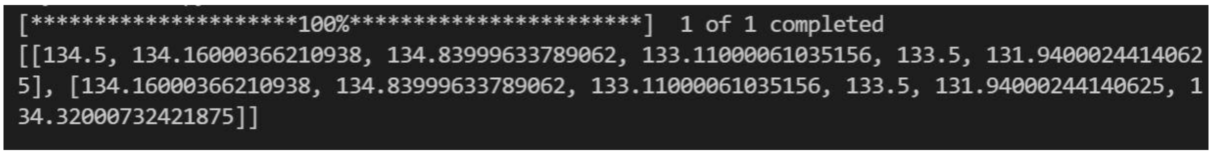
We do that using the following "make_X_and_Y" function :



```python
import yfinance as yf

data = yf.download("AAPL", start="2021-04-16", end="2021-10-16")

def make_X_and_Y(close_prices,period = 6):
    X = []
    y = []
    for i in range(len(close_prices)-period):
        X.append(close_prices[i:i+period])
        y.append(close_prices[i+period])
    return X,y
X,y = make_X_and_Y(list(data.Close))
print(X[0:2])
```

Notice one more argument of the function called "period" which essentially means "n" (=6) in our previous example.

Once we run the above code, we get X and y variables which represent our features and dependent variable respectively. We looked at the first two features and this is how they look like:



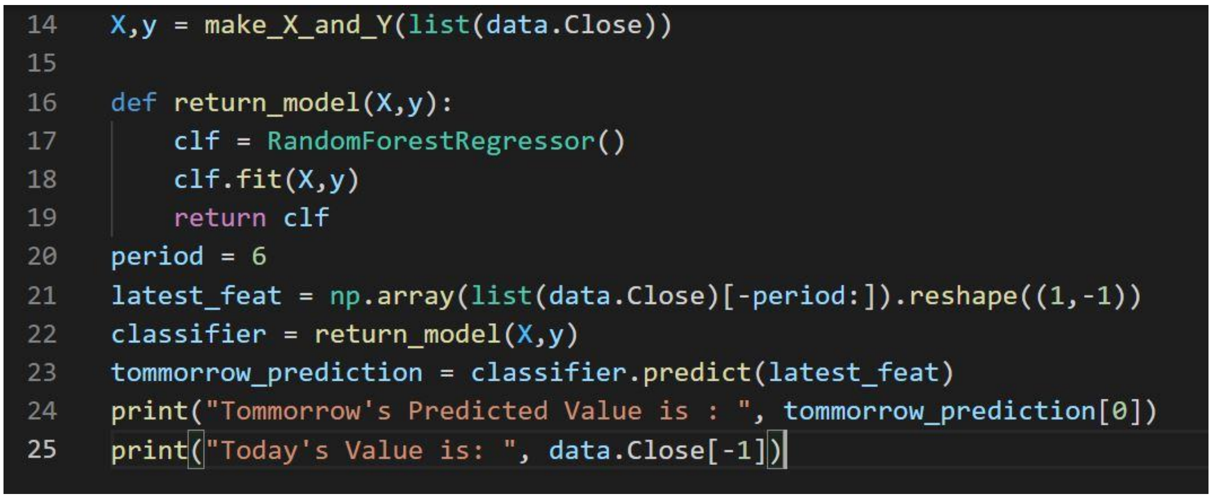
```
[*********************100%***********************]  1 of 1 completed
[[134.5, 134.16000366210938, 134.83999633789062, 133.11000061035156, 133.5, 131.9400024414062
5], [134.16000366210938, 134.83999633789062, 133.11000061035156, 133.5, 131.94000244140625, 1
34.32000732421875]]
```

2. Creating the Regressor

We can use any of the Regressors that is available with the Sklearn library. I am going to demonstrate the Random Forest Regressor because that was giving a good performance. You can experiment with other regressors as well.

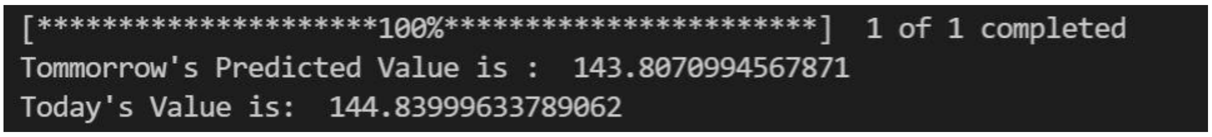
In the previous unit, we discussed how to create a feature matrix and the dependent variable vector. Once that is done, we can now create a classifier and use it to predict tomorrow's price which then can be compared with today's price and accordingly appropriate action can be taken.

The code for fitting the regressor and then using it to make predictions is given below:



```python
X,y = make_X_and_Y(list(data.Close))

def return_model(X,y):
    clf = RandomForestRegressor()
    clf.fit(X,y)
    return clf
period = 6
latest_feat = np.array(list(data.Close)[-period:]).reshape((1,-1))
classifier = return_model(X,y)
tommorrow_prediction = classifier.predict(latest_feat)
print("Tommorrow's Predicted Value is : ", tommorrow_prediction[0])
print("Today's Value is: ", data.Close[-1])
```

Once we run the above code, we can the following output:



```
[*********************100%***********************]  1 of 1 completed
Tommorrow's Predicted Value is :  143.8070994567871
Today's Value is:  144.83999633789062
```

From the output given above, it is clear that tomorrow's predicted price is less than today's price.

In the code, we are

1. First making a function to create and fit a classifier
2. Use that classifier to make the prediction on the latest data.
3. Then printing today's closing price and tomorrow's predicted price.

TASK FOR YOU

1. Run the other algorithms such as Decision Trees, Linear Regression and choose the best one based on the accuracy metric of the sklearn for the apple data.
2. (**EXTRA CHALLENGE**) Pass 200 estimators instead of 100 (default) in the Random Forest method.

Conservative vs Aggressive Strategy

There are two different styles of strategy in the investment world.

1. Conservative one: where the investor does not want to have many risks.
2. Aggressive One: The investor is looking for more risks.

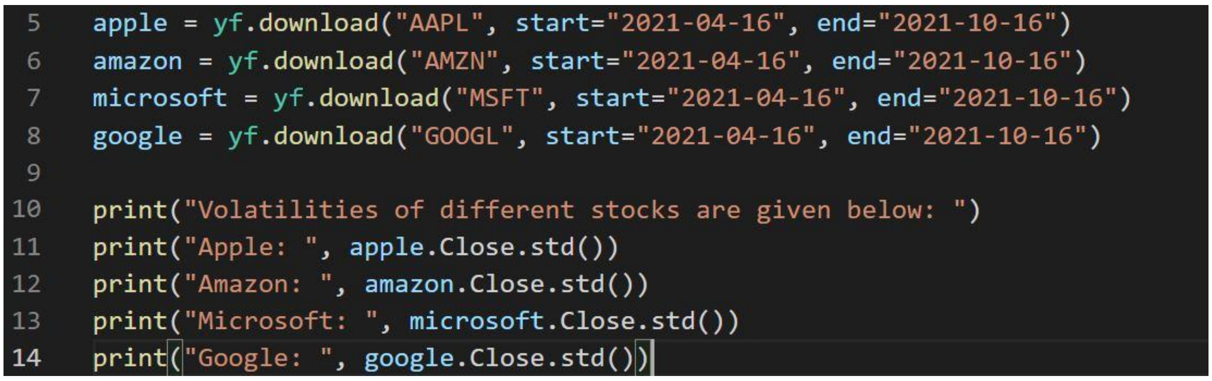
These two concepts are very broad in nature and can take many factors such as company profits, dividends and price data but as of now, we only have the price data with which we need to calculate the risk of all the stock.

Intuitively, we can understand one thing that if a stock's price changes very frequently and rapidly. Further, the amount of change is also quite large in absolute terms, them that stock can be said to have a high risk.

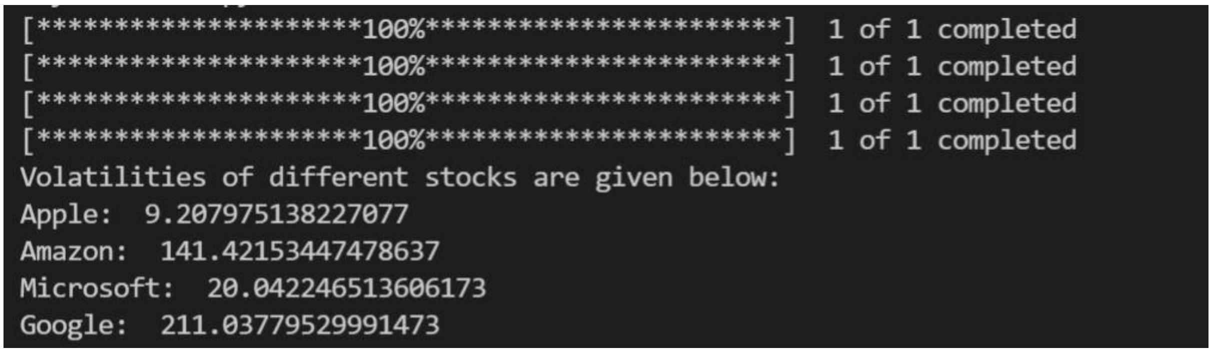
Conversely, if a stock's price remains constant around a fixed value, which means that the stock is having less risk.

With these definitions, we can compute the risk of stocks with the help of the standard deviation of the historical stock prices which will give us some idea of changing the behaviour of the stock prices.

The following images displayed the volatility of the stocks:



```python
apple = yf.download("AAPL", start="2021-04-16", end="2021-10-16")
amazon = yf.download("AMZN", start="2021-04-16", end="2021-10-16")
microsoft = yf.download("MSFT", start="2021-04-16", end="2021-10-16")
google = yf.download("GOOGL", start="2021-04-16", end="2021-10-16")

print("Volatilities of different stocks are given below: ")
print("Apple: ", apple.Close.std())
print("Amazon: ", amazon.Close.std())
print("Microsoft: ", microsoft.Close.std())
print("Google: ", google.Close.std())
```



```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
Volatilities of different stocks are given below:
Apple:  9.207975138227077
Amazon:  141.42153447478637
Microsoft:  20.042246513606173
Google:  211.03779529991473
```

In the above code, we first download the data for four big corporations called apple, amazon, Microsoft and Google. We then compute their volatility by computing the standard deviation by calling the function.

Based on the past 6 months of closing price data, we can conclude that

1. Apple is the least volatile stock among these 4
2. Google and Amazon are the most volatile stock among these 4.

Now that we have understood the individual pieces of our project, next we need to put everything together and make the full project.

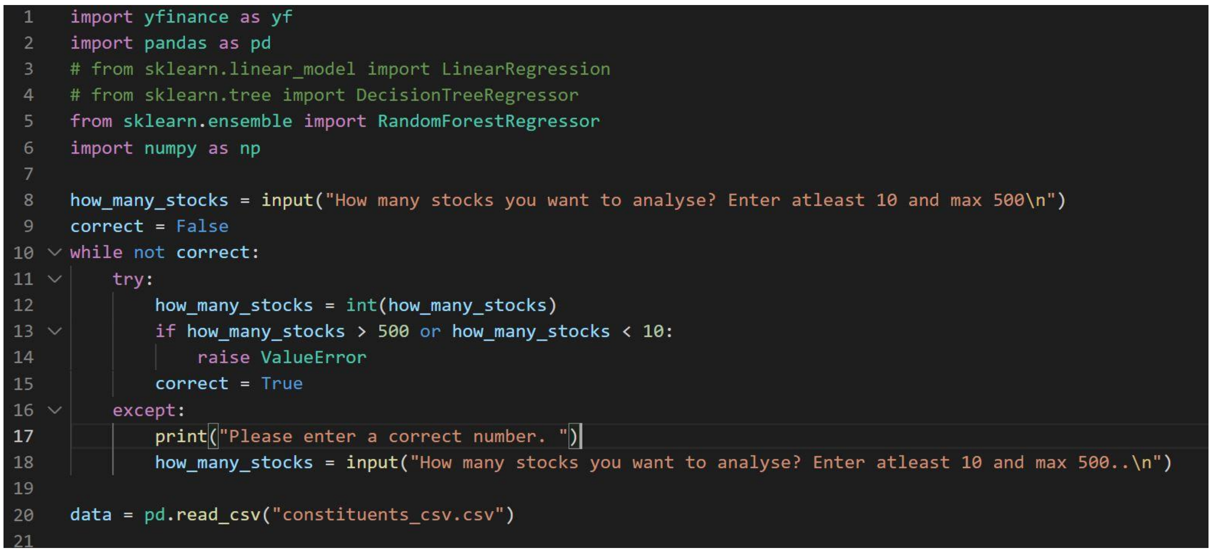
TASK FOR YOU

1. Calculate the volatility of all the S&P 500 companies and list the top 5 stocks for conservative and aggressive types of investors.
2. Try to compute volatility based on Open prices instead of Close prices which we used above. Do you see any change in the result of the above 4 stocks?

Putting Individual pieces together.

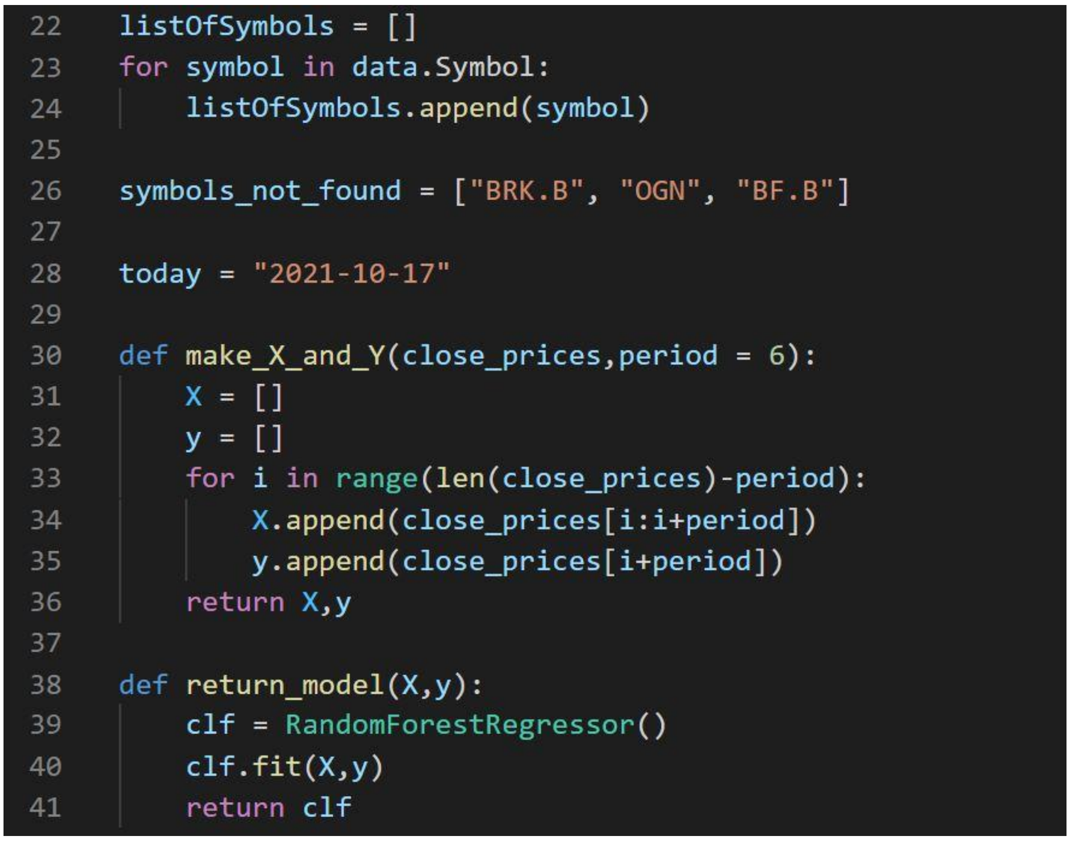
Let us put the individual pieces together.

PART.1 Importing the Packages, User Input of Number of Stocks and Importing Ticker data



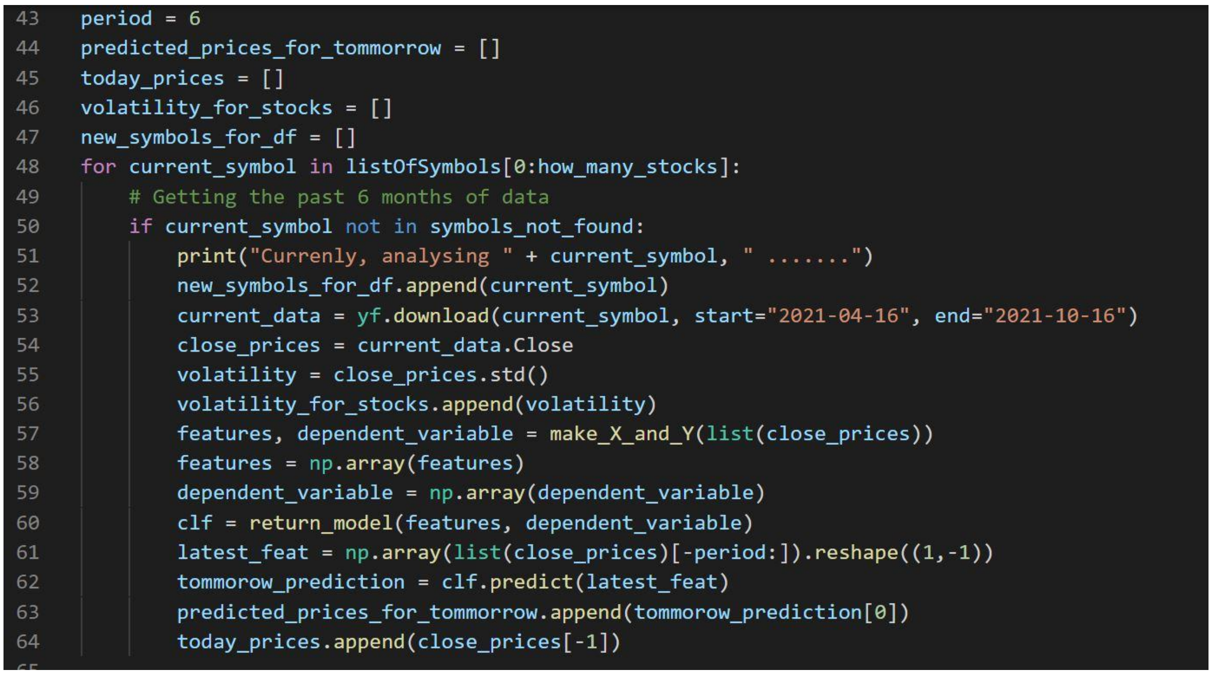
```python
import yfinance as yf
import pandas as pd
# from sklearn.linear_model import LinearRegression
# from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import numpy as np

how_many_stocks = input("How many stocks you want to analyse? Enter atleast 10 and max 500\n")
correct = False
while not correct:
    try:
        how_many_stocks = int(how_many_stocks)
        if how_many_stocks > 500 or how_many_stocks < 10:
            raise ValueError
        correct = True
    except:
        print("Please enter a correct number. ")
        how_many_stocks = input("How many stocks you want to analyse? Enter atleast 10 and max 500..\n")

data = pd.read_csv("constituents_csv.csv")
```

Part. 2 Defining functions for regression data and classifier



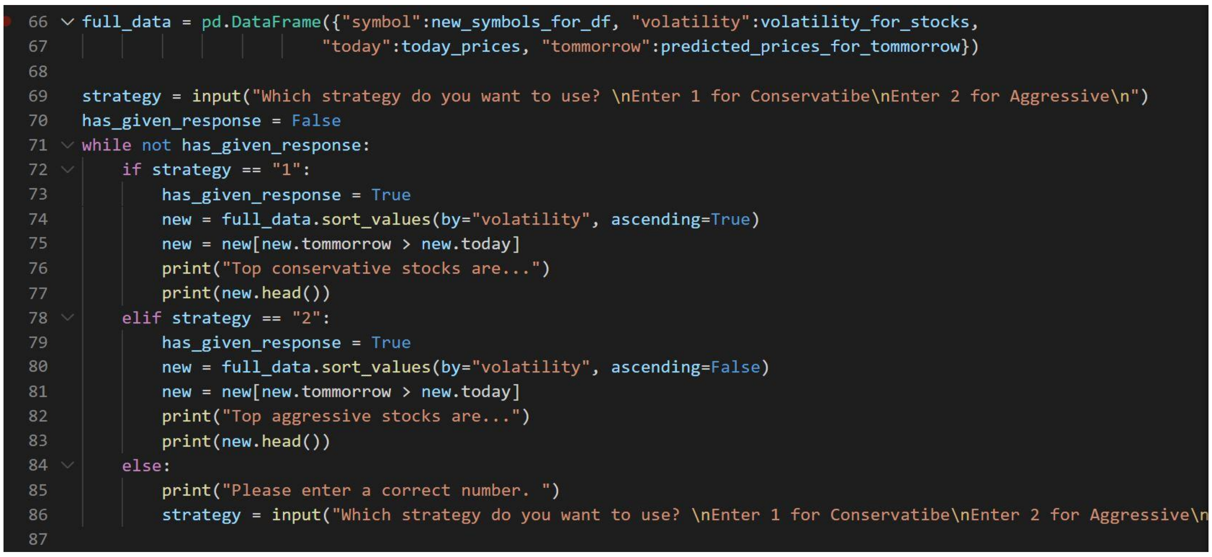
```python
listOfSymbols = []
for symbol in data.Symbol:
    listOfSymbols.append(symbol)

symbols_not_found = ["BRK.B", "OGN", "BF.B"]

today = "2021-10-17"

def make_X_and_Y(close_prices,period = 6):
    X = []
    y = []
    for i in range(len(close_prices)-period):
        X.append(close_prices[i:i+period])
        y.append(close_prices[i+period])
    return X,y

def return_model(X,y):
    clf = RandomForestRegressor()
    clf.fit(X,y)
    return clf
```

PART.3 Running a loop (it will be run whatever number of stock user has specified times)



```python
period = 6
predicted_prices_for_tommorrow = []
today_prices = []
volatility_for_stocks = []
new_symbols_for_df = []
for current_symbol in listOfSymbols[0:how_many_stocks]:
    # Getting the past 6 months of data
    if current_symbol not in symbols_not_found:
        print("Currenly, analysing " + current_symbol, " .......")
        new_symbols_for_df.append(current_symbol)
        current_data = yf.download(current_symbol, start="2021-04-16", end="2021-10-16")
        close_prices = current_data.Close
        volatility = close_prices.std()
        volatility_for_stocks.append(volatility)
        features, dependent_variable = make_X_and_Y(list(close_prices))
        features = np.array(features)
        dependent_variable = np.array(dependent_variable)
        clf = return_model(features, dependent_variable)
        latest_feat = np.array(list(close_prices)[-period:]).reshape((1,-1))
        tommorow_prediction = clf.predict(latest_feat)
        predicted_prices_for_tommorrow.append(tommorow_prediction[0])
        today_prices.append(close_prices[-1])
```

Part. 4 User Input of Strategy and Printing the top stocks based on their response



```python
full_data = pd.DataFrame({"symbol":new_symbols_for_df, "volatility":volatility_for_stocks,
                        "today":today_prices, "tommorrow":predicted_prices_for_tommorrow})

strategy = input("Which strategy do you want to use? \nEnter 1 for Conservatibe\nEnter 2 for Aggressive\n")
has_given_response = False
while not has_given_response:
    if strategy == "1":
        has_given_response = True
        new = full_data.sort_values(by="volatility", ascending=True)
        new = new[new.tommorrow > new.today]
        print("Top conservative stocks are...")
        print(new.head())
    elif strategy == "2":
        has_given_response = True
        new = full_data.sort_values(by="volatility", ascending=False)
        new = new[new.tommorrow > new.today]
        print("Top aggressive stocks are...")
        print(new.head())
    else:
        print("Please enter a correct number. ")
        strategy = input("Which strategy do you want to use? \nEnter 1 for Conservatibe\nEnter 2 for Aggressive\n
```

I hope you have been able to follow through with the tutorial. If you have any doubts in any steps, you can refer to [this code](#) file which is a working file and make it run on your system.

FINAL CHALLENGE FOR YOU

1. Build a similar project for the NYSE listed stocks.
2. Have a functionality of choosing between different algorithm types for doing ML.