

Assignment1: SVM project

Prepare excel file for input data and identify the crash periods: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash

TASK1: Prepare excel files , do calculation and do backtest on result of calculation based on the SVM model.

Code: EF1.1.1

TA name:

Audit name:

Start date:

Finish date:

Approx. Time of Jobs Duration: 1 week(5days working time)

Input File ข้อมูล: backtest_dax_1day.xsl, backtest_u30_1day.xsl, backtest_u500_1day.xsl, ..etc.

อยู่ใน folder google drive: project_SVM

Output: matlab code program , backtest.m link to excel files for backtest and fill blank all columns.

แหล่งข้อมูล

1) Yahoo finance:

Source download: DJIA code U30

<https://finance.yahoo.com/quote/EURUSD%3DX/history/?period1=1070323200&period2=1722059238>

<https://finance.yahoo.com/quote/%5EDJI/history/>

<https://finance.yahoo.com/quote/%5EDJI/history/?p=%5EDJI&period1=694362600&period2=172205563>

Source download: S&P code U500

Source download: DAX code ^GDAXI

<https://finance.yahoo.com/quote/%5EGDAXI/history/?period1=1721972681&period2=1722059075>

2) Tickstory free download program:

<https://tickstory.com/download-tickstory/>

- 3) Eikon program at Econ TU server. ต้องใช้ login pass ด้านล่างทำการจองเวลาเข้าสู่ระบบ Eikon ล่วงหน้าก่อน
อย่างน้อยสองวันทำการ ใช้บริการได้ครั้งละ 2 hrs.

Data Description and Detail of information of the TASK1 in assigment1.

สถานที่ run คือ matlab บนเครื่อง server คณะและเวลาเริ่มเสร็จให้นำผลการรันไปไว้ใน google drive document ใน folder
ที่เตรียมให้ไว้

ข้อมูลรหัสผ่านเข้าเครื่อง server: login:RA990176

pass:ra0176

google drive for document: login:kanonoopapanovskybinboo@gmail.com

pass:Kabin2019##

google drive for backup: login:kakinskypapavanovsky@gmail.com

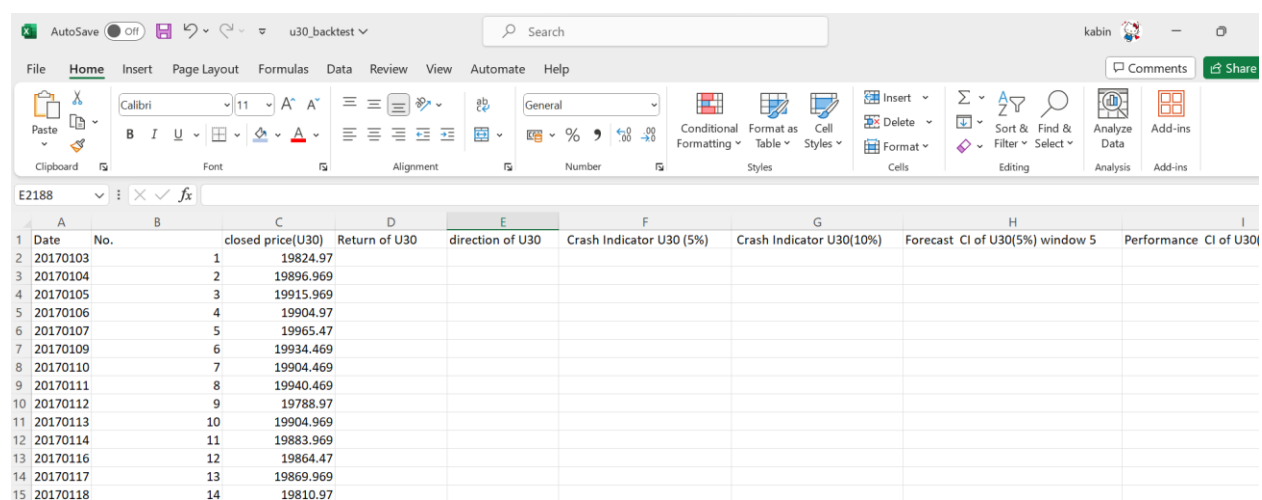
pass:Kabin2018##

ระยะเวลา	2019/1/1- 2020/3/31	2019/1/1- 2020/3/31	2019/1/1- 2022/12/30	2017/1/3- 2024/1/19 (u30,u500) 2016/09/21 (DAX)	2016/4/4 - 2024/4/3	2016/4/4- 2024/4/3
จำนวน ชนิดข้อมูล	14341 30min DAX DJIA(U30) S&P500 (U500)	7335 1hr DAX DJIA(U30) S&P500 (U500)	6,399 4hrs. DAX DJIA(U30) S&P500 (U500)	2190 (U50,U500) 2180 (DAX) 1day	1,947 1day SET50	1,947 1day 39Stocks
แบบจำลอง						
SVM	✓	✓	✓	✓		
LSTM	✓	✓	✓	✓		
GANs	✓	✓	✓	✓	✓	✓
Correlation Force					✓	

More Description of all datasets:

งานวิจัยนี้ใช้ข้อมูลรายวันสำหรับหุ้นรายตัวและดัชนีตลาดหุ้น (รวม SET 50) และ ข้อมูลระหว่างวันสำหรับดัชนีหุ้นทั่วโลก (ไม่รวม SET 50) โดยเลือกใช้เฉพาะข้อมูลที่สามารถเข้าถึงได้ และมีความยาวของระยะเวลาข้อมูล (duration) ตามความเหมาะสมกับระยะเวลาของโครงการ คือ 6 เดือน เหตุผลหลักเนื่องจากข้อมูลระหว่างวัน (เช่น 30 นาที, 1 ชั่วโมง, และ 4 ชั่วโมง) ของหุ้นแต่ละตัวใน SET50 บนฐานข้อมูล Eikon มีให้เข้าถึงย้อนหลังเพียง 3 เดือน ไม่ครอบคลุมช่วงเวลาที่เกิดภาวะล้มเหลวของตลาดปี 2563 (2020) ในขณะที่ข้อมูลระหว่างวันสำหรับดัชนีหุ้นทั่วโลก (ยกเว้น SET50) สามารถเข้าถึงได้จากฐานข้อมูล TickStory สำหรับข้อมูลราย 30 นาที, 1 ชั่วโมง, และ 4 ชั่วโมง อนึ่ง เราไม่สามารถใช้ราคาหุ้นครบทั้ง 50 ตัวที่ประกอบขึ้นเป็น SET50 เพื่อศึกษาการเคลื่อนไหวของหุ้น SET50 ได้ เนื่องจากตลาดหลักทรัพย์แห่งประเทศไทยมีการพิจารณาถอดหุ้นบางตัวเข้าและนำหุ้นบางตัวออกทุกปี โดยพิจารณาจากเกณฑ์ต่างๆ ดังนั้น จากการศึกษาข้อมูลสำหรับการวิเคราะห์สภาวะความล้มเหลวของตลาดในปี 2020 เราพบว่าสามารถคัดเลือกหุ้นใน SET50 มาวิเคราะห์ข้อมูลได้ครบถ้วนสมบูรณ์เพียง 39 ตัว (ดังแสดงในภาคผนวก) เพราะหุ้นบางตัวมีข้อมูลยาวไม่พอ เนื่องจากเป็นหุ้นใหม่หรือมีผลประกอบการไม่ผ่านเกณฑ์การคัดเลือก ในส่วนของการ Train data งานวิจัยนี้จะใช้ Sliding Window ที่มีขนาดแตกต่างกันและปรับเปลี่ยนปริมาณสัดส่วนของข้อมูลที่ใช้ Train และ Test เพื่อหาประสิทธิภาพสูงสุดในการประเมินแบบจำลองต่างๆ เช่น LSTM SVM และ GANs โดยมีขนาดของ window size เริ่มจาก 10, 30, และ 100 ของขนาดข้อมูล (อาจปรับตามความเหมาะสมขึ้นอยู่กับขนาดจำนวนข้อมูลทั้งหมด)

คำอธิบาย file excel ของข้อมูลรายวัน

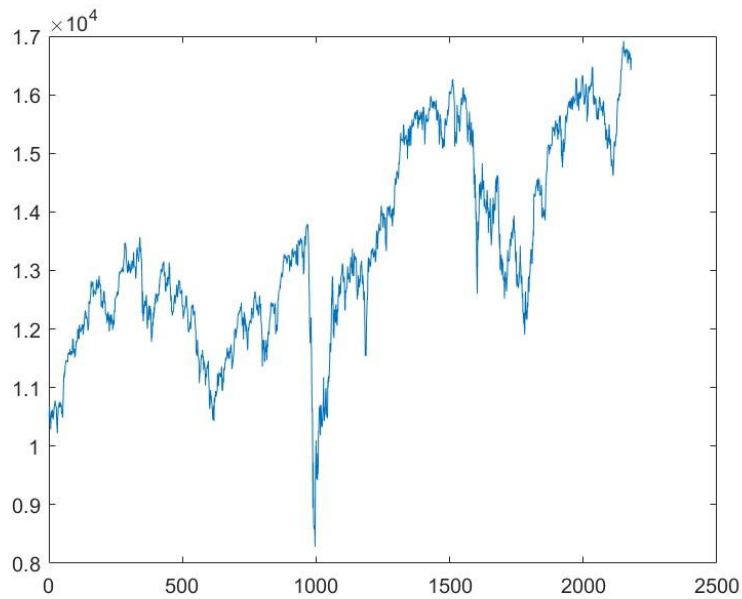


A	B	C	D	E	F	G	H	I
Date	No.	closed price(U30)	Return of U30	direction of U30	Crash Indicator U30 (5%)	Crash Indicator U30(10%)	Forecast CI of U30(5%) window 5	Performance CI of U30
20170103	1	19824.97						
20170104	2	19896.969						
20170105	3	19915.969						
20170106	4	19904.97						
20170107	5	19965.47						
20170109	6	19934.469						
20170110	7	19904.469						
20170111	8	19940.469						
20170112	9	19788.97						
20170113	10	19904.969						
20170114	11	19883.969						
20170116	12	19864.47						
20170117	13	19869.969						
20170118	14	19810.97						

ใน matlab คือ text file ของราคาปิดที่ใช้ในงานวิจัยนี้ซึ่งนำมาจาก column ของราคาปิดใน file excel.

ข้อมูลมาจากแหล่งข้อมูลใน tickstory ซึ่งบางครั้งอาจจะมีค่าไม่ตรงกับข้อมูลรายวันที่ได้จาก Yahoo finance และแหล่งข้อมูลอื่นๆ เพราะมีการบันทึกจำนวน tick size ไม่เท่ากันและช่วงเวลาไม่ตรงกัน ทั้งนี้งานวิจัยนี้เราได้ทำการคำนวณไปแล้วรอบหนึ่งโดยใช้ SVM กับข้อมูลรายวันชุดนี้ ขอให้ RA ยึดความถูกต้องของข้อมูลชุดนี้ตาม text file ของ matlab ที่เราใช้ในการคำนวณเป็นหลักซึ่งตรงกับข้อมูลที่ให้ใน file excel ที่เราจะทำ back test.

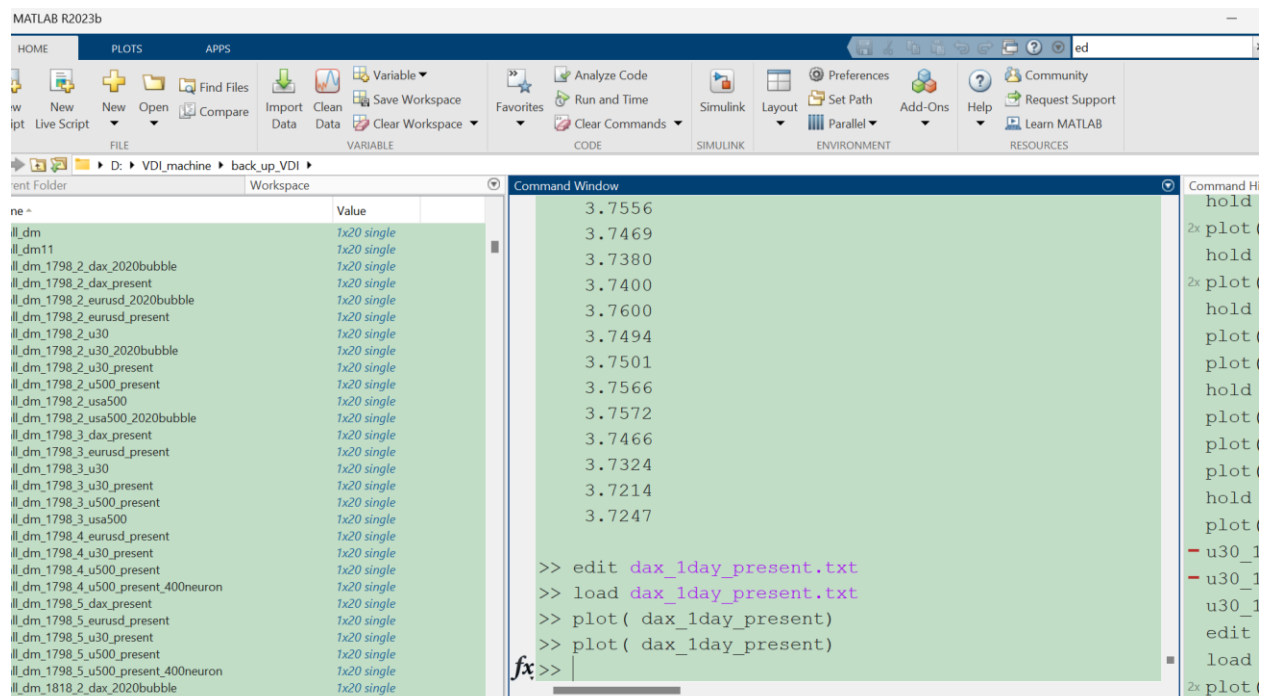
ชื่อ time series :DAX index (ดัชนีราคาหุ้นเยอรมัน) รหัสย่อที่ใช้ในงานวิจัยนี้ DAX 1 day



วิธีการดูกราฟข้อมูลด้วย matlab

```
>>load dax_1day_present.txt
```

```
>>plot(dax_1day_present)
```



จุดต่ำสุดคือช่วย covid market crash อยู่ที่ข้อมูลลำดับที่ 996 ซึ่งเราสามารถรู้วันที่ได้จากตาราง excel file ชื่อ dax_backtest_1day.xml อยู่ใน folder T:\RA\svm_project\backtest ในเครื่อง VDI ของ TU econ หรือใน google drive

ตัวอย่างข้อมูลเช่น

ชื่อ text file: dax_1day_present.txt

ความยาวข้อมูล: 2180

เวลาเริ่มต้น 2017/1/1

เวลาสิ้นสุด 2024/1/19

ชื่อ excel file: dax_backtest_1day.xml

หน้าที่ของ RA คือเขียนโปรแกรม matlab เชื่อมต่อกับ excel file ที่เราได้เตรียมให้เพื่อทำ back test โดยดึงข้อมูลที่เกิดจากการ forecast ใน file excel มาทำ backtest กับ column ราคาเพื่อหา profit โดยใช้ strategy buy and hold. คือ เราจะนำผลการ forecast ค่า market crash indicator ล่วงหน้าหนึ่งวันมาเทียบกับค่า market crash indicator ที่คำนวณได้ ถ้ามีค่าเป็น 0 หมายความว่า เป็นสภาวะปกติให้ hold ไว้ถ้า buy ไว้แล้วถ้ายังไม่ได้ buy ให้ buy หรือ long position แต่ถ้า มีค่าเป็น 0 ให้ sell หรือ short position.

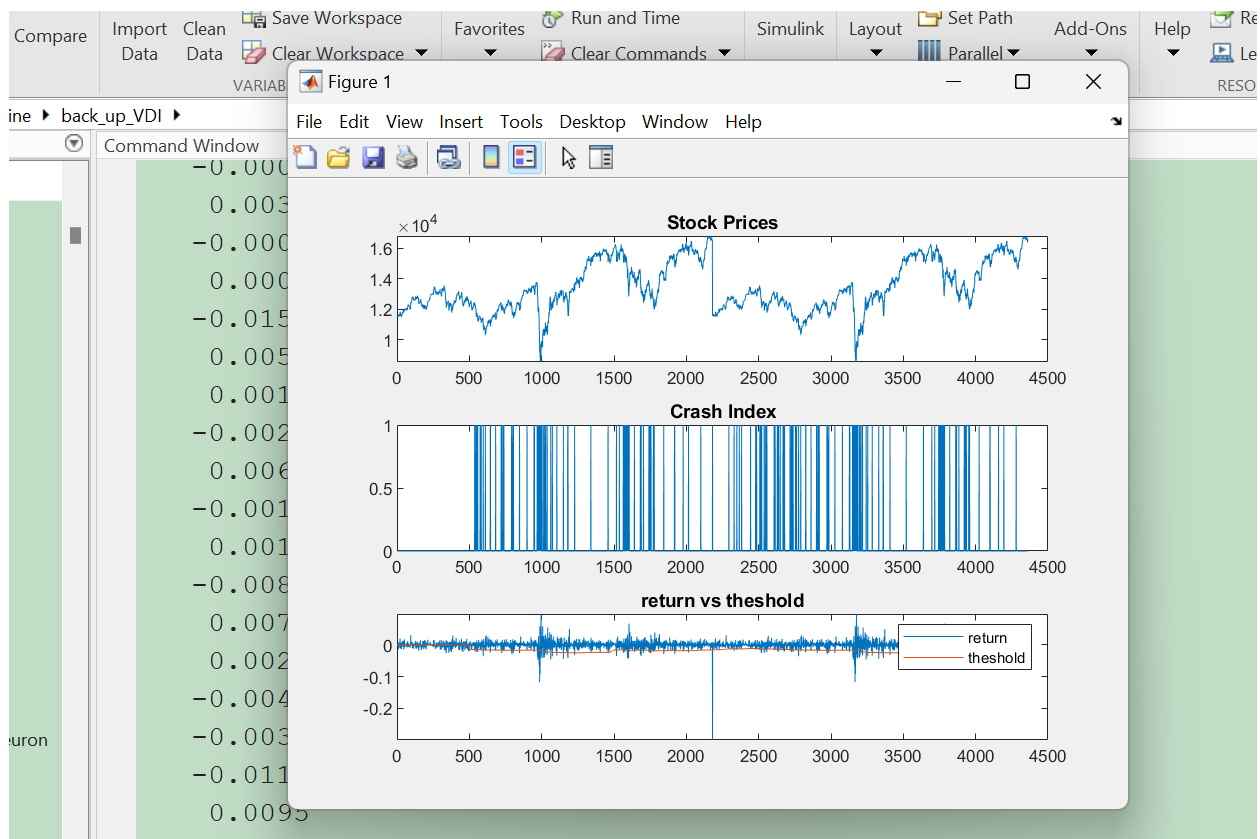
ขั้นตอนในการทำงาน

ขั้นที่ 1: นำข้อมูลราคาปิดจาก file excel ช่องราคาปิดเข้าไปคำนวณในโปรแกรม matlab ที่มี input คือ อนุกรมเวลาของ return ของราคาปิดและ target คือค่า market crash indicator.

```
3_dax_present      3.7466
3_eurusd_present   3.7324
3_u30              3.7214
3_u30_present      3.7214
3_u500_present     3.7247
3_usa500
4_eurusd_present
4_u30_present
4_u500_present
4_u500_present_400neuron
5_dax_present
5_eurusd_present
5_u30_present
5_u500_present
5_u500_present_400neuron

>> edit dax_1day_present.txt
>> load dax_1day_present.txt
>> plot( dax_1day_present)
>> plot( dax_1day_present)
fx>> [target_dax_1,HRT,ret_dax_1]=gen_market_crash_indicator(dax_1day_present_1,5)
```

ข้อมูลที่น่าออกมาคือ market crash indicator ที่เกิดจากการพยากรณ์ล่วงหน้าหนึ่งวันให้นำมาวางในช่องผลลัพธ์การคำนวณใน file excel.



You can save and edit copy and paste into excel file.

```
ent
ent_400neuron    0.0125
nt               0.0003
asent
nt
ent
ent_400neuron
fx>> writematrix(ret_dax_1, 'ret_dax1.txt', 'Delimiter', ' ');
fx>> edit ret_dax1.txt
```

สถานะของโปรแกรม matlab ในปัจจุบันคือมีวิธีการทำงานแบบ manual โดยการนำเข้าโดยใช้การ copy and paste:

เปิด file excel ขึ้นมาจะพบว่ามีราคาปิดและช่องว่างที่เป็นค่า return และ target ของการคำนวณที่เป็น 5% และ 10% คือ percentile ของ threshold ของ return ในช่วงที่เกิด market crash ที่ได้จากการคำนวณ โดยใช้ matlab program ชื่อ file

```
>>[target_dax,HRT,ret_dax]=gen_market_crash_indicator(dax_1day_present,10)
```

```
>>[target_dax,HRT,ret_dax]=gen_market_crash_indicator(dax_1day_present,5)
```

ในที่นี้ input คือ txt file ราคาปิด dax_1day_present.txt, 5 คือ percentile ที่ 5 และผลลัพธ์คือค่า return: ret_dax และค่า market crash indicator target_dax

หน้าที่ของ RA คือเขียนโปรแกรมเชื่อมต่อให้เหมือนกับการทำงานแบบ manual โดยอาศัย source code ตัวอย่างข้างล่างที่ได้จากการแนะนำของ chatGPT4 ถ้าไม่ทราบว่าจะเขียนอย่างไรให้สั่งให้ ChatGPT4 เขียนออกมาให้และทำการทดสอบรันดูว่าถูก

หรือไม่และค่อยๆแก้ไขให้ถูกต้อง ตัวอย่างคำแนะนำที่ได้จาก chatGPT4 เช่นตัวอย่างที่ตัดแปะมาให้ดูข้างล่าง ซึ่ง RA สามารถทำความเข้าใจและทดลองนำไปรันบน matlab และปรับใช้กับงานใน project นี้ได้

Reading with xlsread

xlsread is useful for reading numerical data from Excel files. If your data contains text, it's better to use readtable. However, I will show both methods for completeness.

Example Using xlsread

1. **Reading data:** Use xlsread to read data from the Excel file.
2. **Performing calculations or forecasting:** Perform your necessary calculations.
3. **Writing results back:** Use xlswrite to write the results back to the Excel file.

```
% Specify the Excel file path
```

```
filePath = 'path_to_your_excel_file.xlsx';
```

```
% Read the entire data range
```

```
[data, txt, raw] = xlsread(filePath);
```

```
% Display the first few rows of the numerical data
```

```
disp(data(1:5, :));
```

```
% Display the first few rows of the text data
```

```
disp(txt(1:5, :));
```

```
% Assuming 'ClosedPrice' is in the 3rd column
```

```
closedPrice = data(:, 3);
```

```
% Perform forecasting (using a placeholder function)
```

```
forecast_5_window = forecastFunction(closedPrice, 5);
```

```
forecast_10_window = forecastFunction(closedPrice, 10);
```

```
forecast_30_window = forecastFunction(closedPrice, 30);
```

```
forecast_100_window = forecastFunction(closedPrice, 100);
```

```
% Assuming you want to add the forecast results starting at column 10
```

```
startCol = 10;
```

```
data(:, startCol) = forecast_5_window;
```

```
data(:, startCol + 1) = forecast_10_window;
```

```

data(:, startCol + 2) = forecast_30_window;
data(:, startCol + 3) = forecast_100_window;

% Convert data back to raw cell array to include text headers if needed
raw(2:end, startCol:startCol + 3) = num2cell(data(:, startCol:startCol + 3));

% Write the updated data back to the Excel file
xlswrite(filePath, raw);

% Define the placeholder forecasting function (replace with your actual function)
function forecast = forecastFunction(prices, window)
    % This is a placeholder function. Replace it with your actual forecasting logic.
    forecast = prices * 0.01 * window; % Dummy calculation
end

```

Example Using readtable and writetable

Using readtable and writetable is more convenient for handling mixed data types (numerical and text) and retaining headers.[matlab](#)

```

% Specify the Excel file path
filePath = 'path_to_your_excel_file.xlsx';

% Read the Excel file into a table
data = readtable(filePath);

% Display the first few rows of the table to understand its structure
disp(data(1:5, :));

% Example: Assuming you have a forecasting function 'forecastFunction'
% that takes the 'ClosedPrice' as input and returns the forecasted values
forecast_5_window = forecastFunction(data.ClosedPrice, 5);
forecast_10_window = forecastFunction(data.ClosedPrice, 10);
forecast_30_window = forecastFunction(data.ClosedPrice, 30);
forecast_100_window = forecastFunction(data.ClosedPrice, 100);

```



```

% Add the forecasted values to the table
data.ForecastCI_U500_5_Window = forecast_5_window;
data.ForecastCI_U500_10_Window = forecast_10_window;
data.ForecastCI_U500_30_Window = forecast_30_window;
data.ForecastCI_U500_100_Window = forecast_100_window;

% Write the updated table back to the Excel file
writetable(data, filePath, 'Sheet', 1);

% Define the placeholder forecasting function (replace with your actual function)
function forecast = forecastFunction(prices, window)
    % This is a placeholder function. Replace it with your actual forecasting logic.
    forecast = prices * 0.01 * window; % Dummy calculation
end

```

Explanation

- **xlsread:**
 - `xlsread(filePath)` reads the numerical data from the Excel file.
 - `xlswrite(filePath, raw)` writes the modified data back to the Excel file.
- **readtable:**
 - `readtable(filePath)` reads the Excel file into a MATLAB table.
 - `writetable(data, filePath, 'Sheet', 1)` writes the updated table back to the Excel file.

Choose the method that best fits your data format and processing needs. If your data contains mixed types (numbers and text), `readtable` and `writetable` are generally more straightforward. If you only need numerical data, `xlsread` and `xlswrite` work well.

To integrate the results of your MATLAB forecasting program, which are stored in an array of variables (`cc_...`), into your Excel file, you need to:

1. Read the existing Excel data.
2. Perform any necessary calculations or forecasting (using the `cc_...` variables).
3. Write the backtest results into the appropriate columns of the Excel file.

Here's an example of how you might achieve this in MATLAB:

Step-by-Step MATLAB Code

```
% Specify the Excel file path
filePath = 'path_to_your_excel_file.xlsx';

% Read the Excel file into a table
data = readtable(filePath);

% Display the first few rows of the table to understand its structure
disp(data(1:5, :));

% Assume your forecast results are stored in variables cc_5, cc_10, cc_30, and cc_100
% Replace these with your actual forecast result arrays
cc_5 = rand(height(data), 1); % Example data, replace with actual forecast results
cc_10 = rand(height(data), 1); % Example data, replace with actual forecast results
cc_30 = rand(height(data), 1); % Example data, replace with actual forecast results
cc_100 = rand(height(data), 1); % Example data, replace with actual forecast results

% Add the forecast results to the table
data.ForecastCI_U500_5_Window = cc_5;
data.ForecastCI_U500_10_Window = cc_10;
data.ForecastCI_U500_30_Window = cc_30;
data.ForecastCI_U500_100_Window = cc_100;

% Example of performing backtest calculations
% This is a placeholder; replace with your actual backtest logic
data.Backtest_5_ProfitWindow = cc_5 * 0.05; % Example calculation
data.Backtest_10_ProfitWindow = cc_10 * 0.1; % Example calculation
data.Backtest_30_ProfitWindow = cc_30 * 0.3; % Example calculation
data.Backtest_100_ProfitWindow = cc_100 * 1; % Example calculation

% Write the updated table back to the Excel file
writetable(data, filePath, 'Sheet', 1);
```

Explanation:

1. Reading the Excel File:

- o readtable(filePath) reads the data from the Excel file into a MATLAB table.

2. Integrating Forecast Results:

- o Assuming your forecast results are stored in variables like cc_5, cc_10, cc_30, and cc_100, these are integrated into the table by adding them as new columns.

3. Performing Backtest Calculations:

- o This example includes placeholder calculations for backtesting. Replace these calculations with your actual backtesting logic.

4. Writing Results Back to the Excel File:

- o writetable(data, filePath, 'Sheet', 1) writes the updated table with the forecast and backtest results back to the Excel file.

Customizing the Backtest Logic

Replace the placeholder backtest calculations with your actual backtest logic. For instance, if your backtest logic involves computing profit based on certain conditions, implement those calculations in the code.

Task1.1: เขียน matlab program เชื่อมต่อกับ file excel เพื่อนำข้อมูลราคาปิดมาคำนวณค่าทางสถิติต่างๆแบบง่ายๆ เช่นค่า return ค่า drawdown ค่า market crash indicator เพื่อเตรียมข้อมูลสำหรับการพยากรณ์โดยวิธี svm

ตัวอย่าง1 source code matlab เชื่อมต่อกับ file excel

tic

% Specify the file name

fileName = 'D:\VDI_machine\back_up_VDI\u30_4hr_backtest.xlsx';

disp('Data has been successfully read to column B.');

% Read data from column A of the Excel file

sheet=1;

%data = xlsread(fileName,sheet, 'C2:C10');

data = readmatrix(fileName,'Sheet','backtest','Range', 'C1:C1000');

% Multiply the data by 2

modifiedData = data * 2;

% Write the modified data back to column B of the same Excel file

%xlswrite(fileName, modifiedData, 'F2:F10');

```

writematrix(modifiedData, fileName, 'Sheet','backtest','Range', 'F1:F1000');
disp('Data has been successfully modified and written to column B.');
```

% Start measuring time

% Your computational code goes here

```

pause(5); % Example code that takes some time to run (5 seconds in this case)
```


% Stop measuring time and store the elapsed time

```

elapsedTime = toc;
```


% Convert the elapsed time from seconds to hours

```

elapsedTimeInHours = elapsedTime / 3600;
```


% Display the elapsed time in hours

```

fprintf('Elapsed time: %.6f hours\n', elapsedTimeInHours);
```

ตัวอย่างที่ 2

คือตัวอย่างการเขียนดึงค่าจาก file excel มาคำนวณ forecast SVM ด้วย matlab และส่งค่ากลับเข้าไปที่ file excel

```

tic;
m=2;
```


%read data from Excel F1:F6000

% Specify the file name and sheet name (if applicable)


```

fileName = 'test.xlsx';
```


% Read data from column A (range A1:A100) of the Excel file

```

data = readmatrix(fileName,'Sheet','backtest','Range', 'D2:D4000');
```


% Multiply the data by 2

```

ret_eurusd = data;
```

```
disp('Data has been successfully modified and written to column B.');
```

```
for i=1:5
    [cc_crash_window10_eurusd_chinese4(i)]=run_test_dax2_new_alarm(ret_eurusd(1:2111-
i),target_eurusd(1:2111-i+1),200,m)
end
for i=1:5
    [cc_crash_window10_dax_chinese4(i)]=run_test_dax2_new_alarm(ret_dax(1:2113-i),target_dax(1:2113-
i+1),200,m)
end
for i=1:5
    [cc_crash_window10_u500_chinese4(i)]=run_test_dax2_new_alarm(ret_u500(1:2121-i),target_u500(1:2121-
i+1),200,m)
end
for i=1:5
    [cc_crash_window10_u30_chinese4(i)]=run_test_dax2_new_alarm(ret_u30(1:2121-i),target_u30(1:2121-
i+1),200,m)
end
```

```
disp('Data has been successfully modified and written to column B.');
```

```
% Write the modified data to column B (range B1:B100) of the same Excel file
writematrix(cc_crash_window10_eurusd_chinese4', fileName,'Sheet','backtest','Range', 'I2:I5');
writematrix(cc_crash_window10_dax_chinese4', fileName,'Sheet','backtest','Range', 'J2:J5');
writematrix(cc_crash_window10_u500_chinese4', fileName,'Sheet','backtest','Range', 'K2:K5');
writematrix(cc_crash_window10_u30_chinese4', fileName,'Sheet','backtest','Range', 'L2:L5');
```

```
elapsedTime = toc;
```

```
% Convert the elapsed time from seconds to hours
```

```
elapsedTimeInHours = elapsedTime / 3600;
```

```
% Display the elapsed time in hours
```

```
fprintf('Elapsed time: %.6f hours\n', elapsedTimeInHours);  
toc;
```

ตัวอย่างที่ 3

คือตัวอย่างการเขียนดึงค่าจาก file excel มาคำนวณ forecast SVM ด้วย matlab และส่งค่ากลับเข้าไปที่ file excel

```
tic;  
m=2;  
% Specify the file name and sheet name (if applicable)  
fileName = 'test.xlsx';  
% Read data from column A (range A1:A100) of the Excel file  
data = readmatrix(fileName,'Sheet','backtest','Range', 'E2:E4000');  
target=readmatrix(fileName,'Sheet','backtest','Range', 'H2:H4000');  
% Send read data to forecast module  
ret_eurusd = data;  
target_eurusd=target;  
disp('Data has been successfully modified and written to column B.');
```

```
    for i=1:100  
        [cc_crash_window10_eurusd_chinese4(i)]=run_test_dax2_new_alarm(ret_eurusd(1:2111-  
i),target_eurusd(1:2111-i+1),200,m)  
    end
```

```
%%chinese crash, we need to sort the output from backward order to forward order.  
%%start date of forecast 2121-100=2021  
%%end date of forecast 2121  =2121  
a=flip(cc_crash_window10_eurusd_chinese4);  
  
disp('Data has been successfully modified and written to column B.');
```

```
    % Write the modified data to column B (range B1:B100) of the same Excel file  
writematrix(a, fileName,'Sheet','backtest','Range', 'I2021:I2121');
```

```
elapsedTime = toc;  
  
% Convert the elapsed time from seconds to hours
```

```
elapsedTimeInHours = elapsedTime / 3600;
```

```
% Display the elapsed time in hours
```

```
fprintf('Elapsed time: %.6f hours\n', elapsedTimeInHours);
```

```
toc;
```

จากการคำนวณด้วยเครื่อง Cor.i7 RAM4 100 วัน ใช้เวลาประมาณ 3 hrs. แต่ถ้าคำนวณด้วย notebook ของโครงการใช้เวลา

Elapsed time: 0.557817 hours

ตัวอย่างที่ 4

คือตัวอย่างการเขียนดึงค่าจาก file excel มาคำนวณ market crash indicator ด้วย matlab และส่งค่ากลับเข้าไปที่ file excel

```
function [YY,HRT,ret]=gen_market_crash_indicator2(threshold)
```

```
% prices: Input array of stock prices
```

```
fileName = 'test.xlsx';
```

```
% Read data from column A (range A1:A100) of the Excel file
```

```
prices = readmatrix(fileName,'Sheet','backtest','Range', 'D2:D14339');
```

```
% Parameters
```

```
window_size = 500; % 10 years of monthly data
```

```
percentile_threshold = threshold; % 5th percentile
```

```
%reset=zeros(1>window_size);
```

```
% Calculate monthly returns
```

```
returns = diff(prices) ./ prices(1:end-1);
```

```
%returns = [reset'; returns(window_size+1:end)]; % Adding a zero for the first return value to keep indices  
consistent
```

```
returns=[0;returns];
```

```
ret=returns;
```

```
%%write return to excel file
```

```
writematrix(ret', fileName,'Sheet','backtest','Range', 'E2:E14339');
```

```
n = length(returns);
```

```
crash_index = zeros(n, 1); % Initialize crash index
```

```
% Calculate the historical return-based threshold (HRT)
```

```
HRT = zeros(n, 1); % Initialize HRT array
```

```

for t = window_size+1:n
    historical_returns = returns(t-window_size:t-1);
    HRT(t) = prctile(historical_returns, percentile_threshold);
end

% Calculate crash index based on HRT
for t = window_size+2:n
    if returns(t) - HRT(t) < 0
        crash_index(t) = 1;
    else
        crash_index(t) = 0;
    end
end
YY=crash_index;
%%write return to excel file
writematrix(YY, fileName,'Sheet','backtest','Range', 'H2:H14339');
% Prepare the data for SVM
X = returns(window_size+2:end-1); % Features (returns)
Y = crash_index(window_size+2:end-1); % Labels (crash index)

% Split the data into training and testing sets
train_ratio = 0.8;
train_size = floor(train_ratio * length(X));

X_train = X(1:train_size);
Y_train = Y(1:train_size);
X_test = X(train_size+1:end);
Y_test = Y(train_size+1:end);

% Plot the results
figure;
subplot(3, 1, 1);
plot(prices);
title('Stock Prices');

```



```

subplot(3, 1, 2);
plot(crash_index);
title('Crash Index');

subplot(3, 1, 3);
plot(ret);
hold on;
plot(HRT);
title('return vs theshold');
legend('return', 'theshold');
end

```

ตัวอย่างที่ 4

```

%%U500 30 min SVM start date:30/7/2567
%%EF1.1.3
%%forecast and do backtest
%%window5
tic;
m=1;
% Specify the file name and sheet name (if applicable)
fileName = 'test3.xlsx';
clear p;
clear cc;
clear cc_crash_window10_eurusd_chinese6;
clear target_eurusd;
clear ret_eurusd;
clear data;
clear target;
end_day=14339;
%%target =138691+1 since one day ahead!
% Read data from column A (range A1:A100) of the Excel file
data = readmatrix(fileName,'Sheet','backtest','Range', 'E2:E14340');
price=readmatrix(fileName,'Sheet','backtest','Range', 'D2:D14340');

target=readmatrix(fileName,'Sheet','backtest','Range', 'H2:H14340');
% Send read data to forecast module
ret_eurusd = data;
target_eurusd=target;

%%covid crash

disp('Data has been successfully modified and written to column B. ');
for i=1:100

[cc_crash_window5_u500_30min(i),pp_crash_window5_u500_30min(i),target2_crash_window5_u500_30min(i)]=run_sv
m_forecast_alarm3(ret_eurusd(1:end_day-i),target_eurusd(1:end_day-i+1),200,m)
end
%%forecast direction
for i=1:100

```

```

[cc_di_window5_u500_30min(i),pp_price_crash_window5_u500_30min(i),target2_price_crash_window5_u500_30min(i)
]=run_svm_forecast_direction3(price(1:end_day-i),price(1:end_day-i+1),200,m)
end

%%chinese crash, we need to sort the output from backward order to forward order.
%%start date of forecast 2121-100=2021
%%end date of forecast 2121 =2121
a=flip(cc_crash_window5_u500_30min);
b=flip(pp_crash_window5_u500_30min);
c=flip(target2_crash_window5_u500_30min);

%%%
% price
d=flip(cc_di_window5_u500_30min);
e=flip(pp_price_crash_window5_u500_30min);
f=flip(target2_price_crash_window5_u500_30min);
disp('Data has been successfully modified and written to column B.');
```

% Write the modified data to column B (range B1:B100) of the same Excel file

```

writematrix(b', fileName,'Sheet','backtest','Range','I14241:I14340');
writematrix(a', fileName,'Sheet','backtest','Range','J14241:J14340');
writematrix(c', fileName,'Sheet','backtest','Range','K14241:K14340');
writematrix(d', fileName,'Sheet','backtest','Range','L14241:L14340');
writematrix(e', fileName,'Sheet','backtest','Range','M14241:M14340');
writematrix(f', fileName,'Sheet','backtest','Range','N14241:N14340');
elapsedTime = toc;

% Convert the elapsed time from seconds to hours
elapsedTimeInHours = elapsedTime / 3600;
per_form1_dax30min_window5=sum(cc_di_window5_u500_30min);
per_form2_dax30min_window5=sum(cc_crash_window5_u500_30min)
% Display the elapsed time in hours
fprintf('Elapsed time: %.6f hours\n performance 1:=%2f performance 2:=%2f ', elapsedTimeInHours,
per_form1_dax30min_window5,per_form2_dax30min_window5);
toc;

%%window 10
%%window5
tic;
m=2;
% Specify the file name and sheet name (if applicable)
fileName = 'test3.xlsx';
clear p;
clear cc;
clear cc_crash_window10_eurusd_chinese6;
clear target_eurusd;
clear ret_eurusd;
clear data;
clear target;
end_day=14339;
%%target =138691+1 since one day ahead!
% Read data from column A (range A1:A100) of the Excel file
data = readmatrix(fileName,'Sheet','backtest','Range','E2:E14340');
price=readmatrix(fileName,'Sheet','backtest','Range','D2:D14340');

target=readmatrix(fileName,'Sheet','backtest','Range','H2:H14340');
% Send read data to forecast module
ret_eurusd = data;
target_eurusd=target;

%%%covid crash

disp('Data has been successfully modified and written to column B.');
```

for i=1:100

```

[cc_crash_window10_u500_30min(i),pp_crash_window10_u500_30min(i),target2_crash_window10_u500_30min(i)]=run
_svm_forecast_alarm3(ret_eurusd(1:end_day-i),target_eurusd(1:end_day-i+1),200,m)
end

```

```

%%chinese crash, we need to sort the output from backward order to forward order.
%%start date of forecast 2121-100=2021
%%end date of forecast 2121 =2121
aa=flip(cc_crash_window10_u500_30min);
bb=flip(pp_crash_window10_u500_30min);
cc=flip(target2_crash_window10_u500_30min);

disp('Data has been successfully modified and written to column B.');
```

% Write the modified data to column B (range B1:B100) of the same Excel file

```

writematrix(bb', fileName,'Sheet','backtest','Range', 'P14241:I14340');
writematrix(aa', fileName,'Sheet','backtest','Range', 'Q14241:J14340');
writematrix(cc', fileName,'Sheet','backtest','Range', 'R14241:K14340');
```

elapsedTime = toc;

% Convert the elapsed time from seconds to hours

```

elapsedTimeInHours = elapsedTime / 3600;
```

per_form1_dax30min_window10=sum(cc_crash_window10_u500_30min)

% Display the elapsed time in hours

```

fprintf('Elapsed time: %.6f hours\n performance 1:=%2f    ', elapsedTimeInHours,
per_form1_dax30min_window10 );
toc;
```

%%window 30

```

tic;
m=3;
% Specify the file name and sheet name (if applicable)
fileName = 'test3.xlsx';
clear p;
clear cc;
clear cc_crash_window10_eurusd_chinese6;
clear target_eurusd;
clear ret_eurusd;
clear data;
clear target;
end_day=14339;
%%target =138691+1 since one day ahead!
% Read data from column A (range A1:A100) of the Excel file
data = readmatrix(fileName,'Sheet','backtest','Range', 'E2:E14340');
price=readmatrix(fileName,'Sheet','backtest','Range', 'D2:D14340');
```

target=readmatrix(fileName,'Sheet','backtest','Range', 'H2:H14340');

% Send read data to forecast module

```

ret_eurusd = data;
target_eurusd=target;
```

%%covid crash

```

disp('Data has been successfully modified and written to column B.');
```

for i=1:100

```

[cc_crash_window30_u500_30min(i),pp_crash_window30_u500_30min(i),target2_crash_window30_u500_30min(i)]=run
_svm_forecast_alarm3(ret_eurusd(1:end_day-i),target_eurusd(1:end_day-i+1),200,m)
end
```

%%chinese crash, we need to sort the output from backward order to forward order.

%%start date of forecast 2121-100=2021

%%end date of forecast 2121 =2121

```

aaa=flip(cc_crash_window30_u500_30min);
bbb=flip(pp_crash_window30_u500_30min);
ccc=flip(target2_crash_window30_u500_30min);

disp('Data has been successfully modified and written to column B. ');
% Write the modified data to column B (range B1:B100) of the same Excel file
writematrix(bbb', fileName,'Sheet','backtest','Range', 'T14241:I14340');
writematrix(aaa', fileName,'Sheet','backtest','Range', 'U14241:J14340');
writematrix(ccc', fileName,'Sheet','backtest','Range', 'V14241:K14340');

elapsedTime = toc;

% Convert the elapsed time from seconds to hours
elapsedTimeInHours = elapsedTime / 3600;

per_form1_dax30min_window30=sum(cc_crash_window30_u500_30min)
% Display the elapsed time in hours
fprintf('Elapsed time: %.6f hours\n performance 1:=%2f    ', elapsedTimeInHours,
per_form1_dax30min_window30 );
toc;

```

ตัวอย่างที่ 5 คำนวณค่า F1 score

```
function [F1, TP, TN, FP, FN] = svm_forecast(prices)
```

```
% Calculate returns
```

```
returns = diff(prices) ./ prices(1:end-1);
```

```
% Generate labels: 1 for up, 0 for down
```

```
labels = returns > 0;
```

```
% Split data into training and testing sets (80% train, 20% test)
```

```
splitRatio = 0.8;
```

```
splitIndex = floor(splitRatio * length(labels));
```

```
trainData = returns(1:splitIndex);
```

```
trainLabels = labels(1:splitIndex);
```

```
testData = returns(splitIndex+1:end);
```

```
testLabels = labels(splitIndex+1:end);
```

```
% Train SVM model
```

```
svmModel = fitsvm(trainData, trainLabels, 'KernelFunction', 'linear', 'Standardize', true);
```

```
% Make predictions on the test set
```

```
predictions = predict(svmModel, testData);
```

```

% Calculate performance metrics
TP = sum((predictions == 1) & (testLabels == 1));
TN = sum((predictions == 0) & (testLabels == 0));
FP = sum((predictions == 1) & (testLabels == 0));
FN = sum((predictions == 0) & (testLabels == 1));

precision = TP / (TP + FP);
recall = TP / (TP + FN);
F1 = 2 * (precision * recall) / (precision + recall);

% Display the results
fprintf('True Positives (TP): %d\n', TP);
fprintf('True Negatives (TN): %d\n', TN);
fprintf('False Positives (FP): %d\n', FP);
fprintf('False Negatives (FN): %d\n', FN);
fprintf('F1 Score: %.2f\n', F1);

end

prices = [100, 101, 102, 101, 100, 99, 98, 99, 100, 101]; % Example prices
[logReturns, autocorrValues, responseFunction] = marketCrashPrediction(prices);
ตัวอย่างที่ 6 backtest
% Specify the file name and sheet name (if applicable)
fileName = 'test.xlsx';

% Read data from columns A (price), B (forecast direction), and C (position)
price = readmatrix(fileName, 'Range', 'A1:A10');
forecast = readmatrix(fileName, 'Range', 'B1:B10');
position = readmatrix(fileName, 'Range', 'C1:C10');

% Initialize return and cumulative return arrays
returns = zeros(length(price)-1, 1);
cumulative_returns = zeros(length(price)-1, 1);

% Perform the backtest
for i = 1:length(price)-1

```

```

% Check the forecast direction and determine the position
if forecast(i) == 1 % Forecast is up
    position(i) = 1; % Long position
elseif forecast(i) == -1 % Forecast is down
    position(i) = -1; % Short position
end

% Calculate the return
if position(i) == 1 % Long position
    returns(i) = (price(i+1) - price(i)) / price(i);
elseif position(i) == -1 % Short position
    returns(i) = (price(i) - price(i+1)) / price(i);
end

% Calculate the cumulative return
if i == 1
    cumulative_returns(i) = returns(i);
else
    cumulative_returns(i) = cumulative_returns(i-1) + returns(i);
end
end

% Write the results back to the Excel file
writematrix(returns, fileName, 'Range', 'D1:D10');
writematrix(cumulative_returns, fileName, 'Range', 'F1:F10');

disp('Backtest complete. Returns and cumulative returns have been written to columns D and F.');
```

ตัวอย่างที่ 7 backtest

```

%%backtest
% Specify the file name and sheet name (if applicable)
fileName = 'test3.xlsx';

% Read data from columns A (price), B (forecast direction), and C (position)
price = readmatrix(fileName, 'Sheet', 'backtest', 'Range', 'D14241:D14340');
forecast = readmatrix(fileName, 'Sheet', 'backtest', 'Range', 'L14241:L14340');
%position = readmatrix(fileName, 'Range', 'C1:C10');

% Initialize return and cumulative return arrays
returns = zeros(100, 1);
position = zeros(100, 1);
```

```

cumulative_returns = zeros(100, 1);

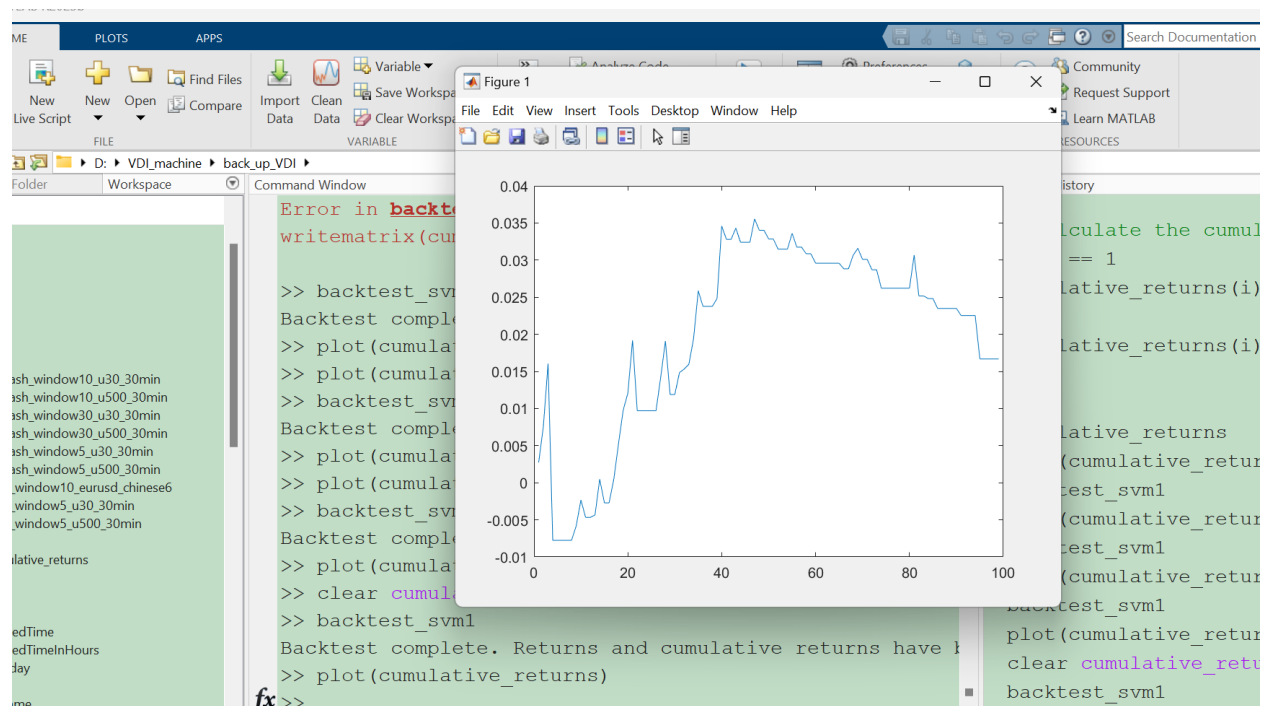
% Perform the backtest
for i = 1:99
    % Check the forecast direction and determine the position
    if forecast(i) == 1 % Forecast is up
        position(i) = 1; % Long position
    elseif forecast(i) == -1 % Forecast is down
        position(i) = -1; % Short position
    end

    % Calculate the return
    if position(i) == 1 % Long position
        returns(i) = (price(i+1) - price(i)) / price(i);
    elseif position(i) == -1 % Short position
        returns(i) = (price(i) - price(i+1)) / price(i);
    end

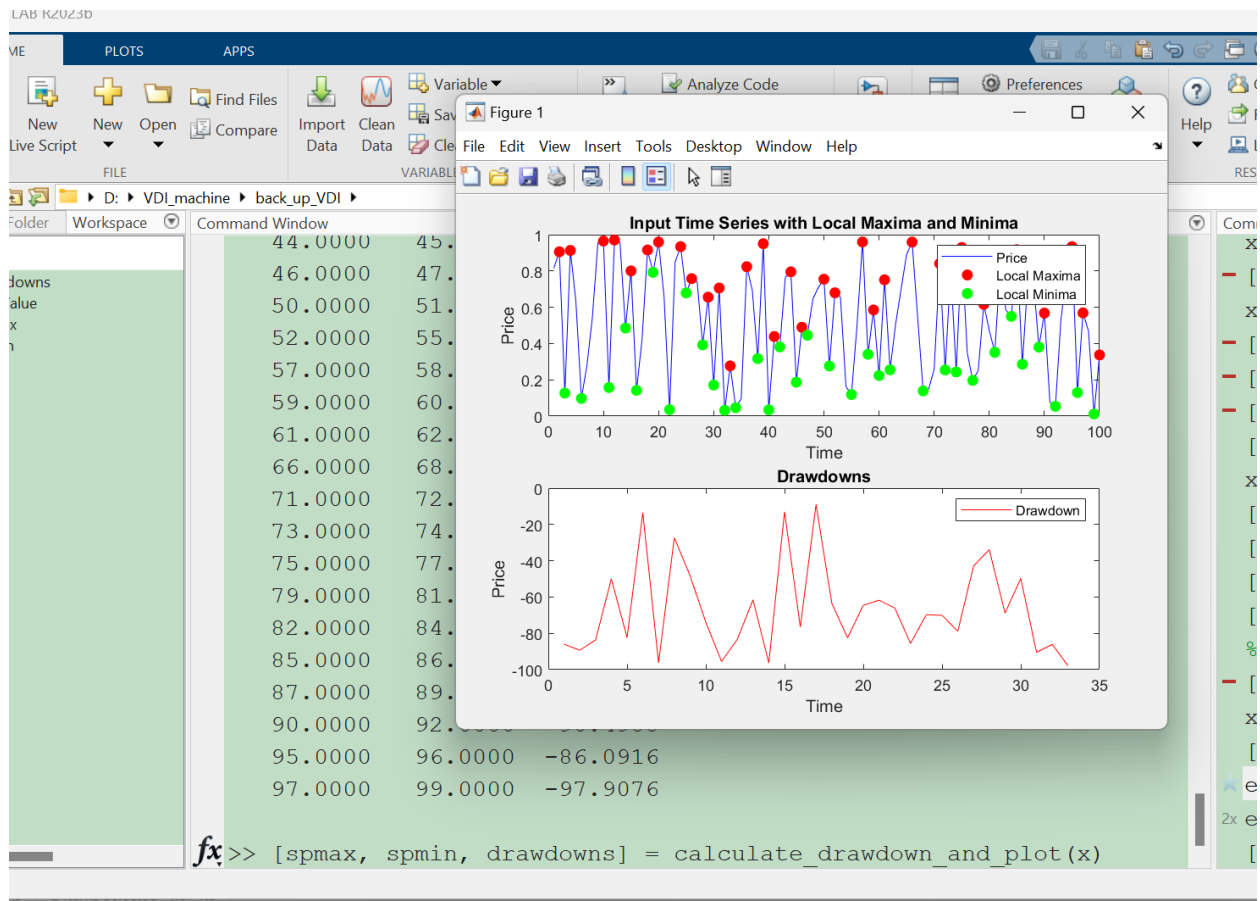
    % Calculate the cumulative return
    if i == 1
        cumulative_returns(i) = returns(i);
    else
        cumulative_returns(i) = cumulative_returns(i-1) + returns(i);
    end
end

% Write the results back to the Excel file
writematrix(returns, fileName, 'Range', 'D1:D10');
writematrix(cumulative_returns, fileName, 'Sheet', 'backtest', 'Range', 'O14241:O14340');

disp('Backtest complete. Returns and cumulative returns have been written to columns O.');
```



สิ่งที่ RA ต้องทำคือเขียนโปรแกรมดึงข้อมูลจาก column ที่เป็นราคาปิดในช่วงที่ forecast ได้ค่า market crash indicator มาทำ backtest และคำนวณค่า performance ของการ forecast ด้วย hit ratio



Assignment2: LSTM project

Matlab module:

```
[spmax, spmin, drawdowns] = calculate_drawdown_and_plot(x)
```

example_run_svm12.m

```
nextValue = test_lstm(timeSeries, numHiddenUnits)
```

```
p= test_lstm(target, 100);
```

```
run_svm_forecast_alarm3_lstm(input,ccc,n,choice)
```


Definition: Let $\{P_{t=1,T}\}$ be a time series of a financial index price at closure.

Let $P_{max} = P_k$ be a local maximum.

Let $P_{min} = P_{k+n}$ be the next following local minimum, where $k \geq 2$ and $n \geq 1$. Here n denotes the number of days between the local maximum and the local minimum.

The following time series will define the drawdown, $P_k > P_{k+1} > \dots > P_{k+n}$. The maximum, P_{max} , has to satisfy; $P_k \geq P_{k-1}$ and $P_k > P_{k+1}$. The minimum, P_{min} , has to satisfy; $P_{k+n} < P_{k+n-1}$ and $P_{k+n} \leq P_{k+n+1}$. The drawdown is calculated in percentage as;

$$D = \frac{P_{min} - P_{max}}{P_{max}} \quad (2)$$

For the 13 daily returns of the Dow Jones index, it means that the first drawdown lasted for 2 days, the second for 1 day and the third lasted for 5 days.

Prepare excel file for input data and identify the drawdown, crash size and crash duration: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash คำนวณค่า drawdown และ crash size, crash period เพื่อใช้ในการพยากรณ์โดย LSTM

Table 4: The largest negative ε -drawdowns, when $\varepsilon = 3\sigma$.

Rank	Size	Duration	Start date	Event
1	-0.3503	45*	1914-06-10	WW1 exogenous shock
2	-0.3480	13	1929-10-10	Bubble
3	-0.3416	11	1987-10-02	Bubble
4	-0.3109	25	1932-03-08	Great Depression

* The DJIA was closed from 1914-07-30 to 1914-12-12, due to the breakout of WW1.

Table 2

Summary of the Bitcoin crashes detected by LPPLS confidence indicator from 10/28/2017 21:00 to 6/30/2018 0:00 using the two levels of the adaptive multilevel time series detection methodology.

Crash no	Cluster of positive bubble		Cluster of negative bubble	
	Start time	End time	Start time	End time
1	10/31/2017 22:00	11/7/2017 7:00	11/10/2017 14:30	11/13/2017 9:00
2	11/25/2017 6:00	12/9/2017 11:00	12/9/2017 17:00	12/10/2017 16:30
3	12/12/2017 0:00	12/19/2017 10:00	12/19/2017 17:00	12/23/2017 1:30
4	12/26/2017 8:00	12/27/2017 13:30	12/28/2017 6:00	12/31/2017 18:30
5	1/5/2018 11:00	1/7/2018 23:30	1/8/2018 6:30	1/12/2018 13:30
6	1/13/2018 5:00	1/13/2018 22:30	1/16/2018 13:30	1/19/2018 10:00
7	1/27/2018 12:00	1/30/2018 12:30	2/2/2018 10:00	2/6/2018 18:00
8	2/14/2018 17:00	2/21/2018 12:30	2/25/2018 8:00	2/26/2018 12:00
9	3/3/2018 6:30	3/6/2018 7:00	3/6/2018 18:00	3/10/2018 5:30
10	3/20/2018 16:30	3/21/2018 16:00	3/26/2018 22:00	4/1/2018 22:00
11	4/20/2018 0:00	5/7/2018 3:00	5/22/2018 10:00	5/31/2018 3:00
12	6/6/2018 20:30	6/9/2018 22:00	6/10/2018 17:00	6/11/2018 12:30
13	6/19/2018 8:00	6/22/2018 8:00	6/26/2018 23:00	6/29/2018 20:00

No	Actual peak time	Actual peak price	Actual valley time	Actual valley price	Crash size
1	11/5/2017 13:00	7571	11/12/2017 22:00	5700	32.80%
2	12/8/2017 1:00	16626	12/10/2017 3:00	13110	21.10%
3	12/17/2017 12:00	19600	12/22/2017 14:00	12092	38.30%
4	12/27/2017 5:00	16320	12/30/2017 16:00	12300	24.60%
5	1/6/2018 23:00	17166	1/12/2018 0:00	12953	24.50%
6	1/13/2018 13:30	14492	1/17/2018 14:00	9500	34.40%
7	1/28/2018 6:00	11948	2/6/2018 4:00	6100	48.90%
8	2/20/2018 17:00	11670	2/25/2018 18:00	9354	19.80%
9	3/5/2018 18:00	11667	3/9/2018 9:00	8558	26.70%
10	3/21/2018 4:00	9122	4/1/2018 15:00	6429	29.50%
11	5/5/2018 10:00	9908	5/29/2018 2:00	7089	28.50%
12	6/7/2018 1:00	7738	6/10/2018 22:00	6694	13.50%
13	6/21/2018 2:00	6775	6/29/2018 13:00	5861	13.50%

Table 6: The largest negative drawdowns for OMXS30.

Rank	Size	Duration	Start date	Event
1	-0.2017	5	2008-10-03	Anti-Bubble
2	-0.1983	12	2001-08-24	Anti-Bubble*
3	-0.1729	7	1990-09-19	Swedish bank crisis, exogenous shock
4	-0.1586	5	2002-07-17	Anti-Bubble*

* The anti-bubble associated with the crash of the new economy.

Table 7: The largest negative ε -drawdowns, when $\varepsilon = \sigma$.

Rank	Size	Duration	Start date	Event
1	-0.2413	21	1990-08-30	Swedish bank crisis, exogenous shock
2	-0.2103	6	1987-10-21	Influence by the DJIA
3	-0.2062	16	1990-08-01	Swedish bank crisis, exogenous shock
4	-0.2017	5	2008-10-03	Anti-Bubble

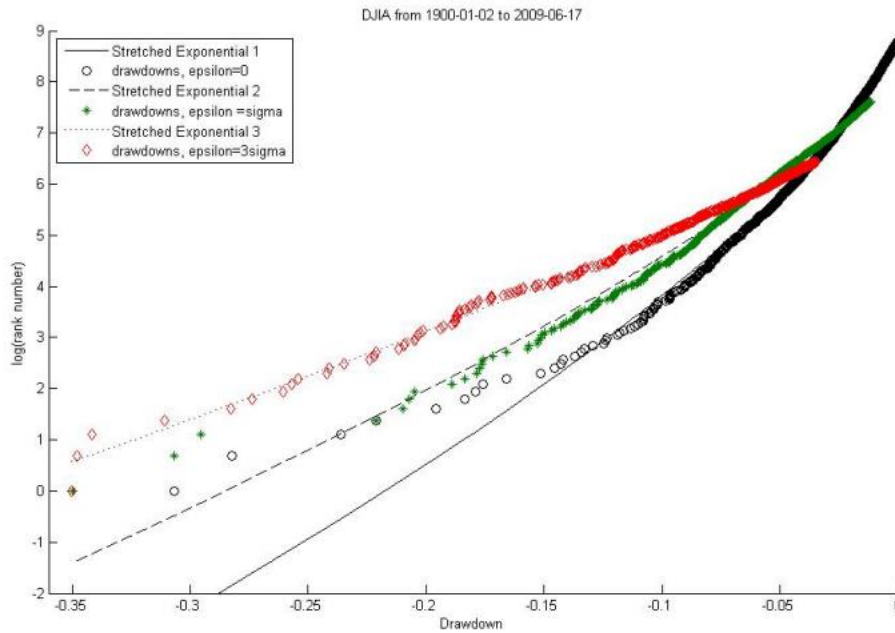


Figure 4: The stretched exponential fitted to the drawdowns of the DJIA when ε is 0, σ and 3σ . The outliers referred to by Johansen & Sornette [6] are visible over the fitted distributions in the lower left.

TASK2: Prepare excel files , do calculation and do backtest on result of calculation based on the SVM model.

Code: EF1.1.1

TA name:

Audit name:

Start date:

Finish date:

10

M. Shu and W. Zhu / Physica A 548 (2020) 124477

Table 1

Summary of the Bitcoin crashes with more than 15% sharp drop within 3 weeks from 9/13/2011 to 4/7/2019.

Number	Peak day	Peak price (\$)	Crash end day	Crash end price (\$)	Crash duration	Crash size
1	9/25/2011	6.1	10/9/2011	3.9	14	35.7%
2	10/10/2011	4.5	10/20/2011	2.2	10	50.3%
3	1/10/2012	7.1	1/28/2012	4.9	18	31.2%
4	2/13/2012	5.8	2/18/2012	4.2	5	26.9%
5	8/16/2012	13.4	8/19/2012	8.1	3	39.9%
6	10/10/2012	12.2	10/26/2012	10.0	16	18.3%
7	4/9/2013	229.0	4/16/2013	68.1	7	70.3%
8	4/29/2013	143.3	5/3/2013	98.1	4	31.6%
9	5/29/2013	130.4	6/14/2013	99.0	16	24.0%
10	6/19/2013	105.2	7/6/2013	66.3	17	37.0%
11	10/1/2013	127.3	10/2/2013	103.9	1	18.4%
12	11/18/2013	669.0	11/19/2013	536.0	1	19.9%
13	11/29/2013	1132.0	12/7/2013	693.3	8	38.8%
14	12/10/2013	979.2	12/18/2013	520.0	8	46.9%
15	1/6/2014	919.2	1/27/2014	752.0	21	18.2%
16	2/3/2014	808.5	2/24/2014	535.5	21	33.8%
17	3/5/2014	670.0	3/23/2014	561.0	18	16.3%
18	3/24/2014	586.0	4/10/2014	363.1	17	38.0%
19	4/16/2014	530.0	5/6/2014	428.0	20	19.2%
20	6/3/2014	670.1	6/14/2014	566.6	11	15.5%
21	8/10/2014	591.0	8/18/2014	475.2	8	19.6%
22	8/29/2014	509.3	9/19/2014	395.9	21	22.3%
23	9/23/2014	439.0	10/5/2014	323.5	12	26.3%
24	11/12/2014	426.6	11/21/2014	351.2	9	17.7%
25	12/7/2014	377.3	12/18/2014	312.7	11	17.1%
26	12/26/2014	328.3	1/14/2015	171.4	19	47.8%
27	3/11/2015	296.5	3/24/2015	245.0	13	17.4%
28	4/5/2015	260.5	4/14/2015	215.8	9	17.2%
29	7/28/2015	294.0	8/18/2015	224.8	21	23.5%
30	8/4/2015	285.0	8/24/2015	209.7	20	26.4%
31	11/4/2015	408.0	11/11/2015	309.9	7	24.0%
32	1/9/2016	448.8	1/15/2016	360.0	6	19.8%
33	6/16/2016	766.6	6/22/2016	601.3	6	21.6%
34	7/17/2016	679.5	8/2/2016	540.0	16	20.5%
35	1/4/2017	1114.9	1/11/2017	778.6	7	30.2%
36	3/3/2017	1285.3	3/24/2017	929.1	21	27.7%
37	6/11/2017	2954.2	6/15/2017	2424.9	4	17.9%
38	7/5/2017	2602.9	7/16/2017	1917.6	11	26.3%

Approx. Time of Jobs Duration: 1 week(5days working time)

Input File ข้อมูล: backtest_dax_1day.xsl, backtest_u30_1day.xsl, backtest_u500_1day.xsl,

Output: matlab code program , backtest.m link to excel files for backtest and fill blank all columns.

Data Description and Detail of information of the TASK1 in assignment2.

Generate Synthetic Signals Using Conditional GAN

This example shows how to generate synthetic pump signals using a conditional generative adversarial network.

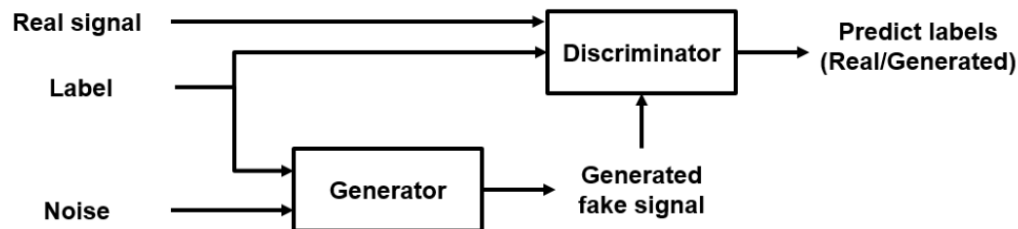
Generative adversarial networks (GANs) can be used to produce synthetic data that resembles real data input to the networks. GANs are useful when simulations are computationally expensive or experiments are costly. Conditional GANs (CGANs) can use data labels during the training process to generate data belonging to specific categories.

This example treats simulated signals obtained by a pump Simulink™ model as the "real" data that plays the role of training data set for a CGAN. The CGAN uses 1-D convolutional networks and is trained using a custom training loop and a deep learning array. In addition, this example uses principal component analysis (PCA) to visually compare the characteristics of generated and real signals.

CGAN for Signal Synthesis

CGANs consist of two networks that train together as adversaries:

1. **Generator network** — Given a label and random array as input, this network generates data with the same structure as the training data observed label. The objective of the generator is to generate labeled data that the discriminator classifies as "real."
2. **Discriminator network** — Given batches of labeled data containing observations from both training data and generated data from the generator, the discriminator classifies the observations as "real" or "generated." The objective of the discriminator is to not be "fooled" by the generator when given batches of both.



Ideally, these strategies result in a generator that generates convincingly realistic data corresponding to the input labels and a discriminator that can distinguish between real and generated data.

Assignment3: GANs project

Prepare excel file for input data and identify the crash periods: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash

TASK3: Prepare excel files , do calculation and do backtest on result of calculation based on the GANsmodel.

Code: EF3.1.1

TA name:

Audit name:

Start date:

Finish date:

by `tempdir`, change the directory name in the following code.

```
% Download the data
dataURL = 'https://ssd.mathworks.com/supportfiles/SPT/data/PumpSignalGAN.zip';
saveFolder = fullfile(tempdir,'PumpSignalGAN');
zipFile = fullfile(tempdir,'PumpSignalGAN.zip');
if ~exist(fullfile(saveFolder,'simulatedDataset.mat'),'file')
    websave(zipFile,dataURL);
    % Unzip the data
    unzip(zipFile,saveFolder)
end
```

The zip file contains the training data set and a pretrained CGAN:

- `simulatedDataset` — Simulated signals and their corresponding categorical labels
- `GANModel` — Generator and discriminator trained on the simulated data

Load the training data set and standardize the signals to have zero mean and unit variance.

```
load(fullfile(saveFolder,'simulatedDataset.mat')) % load data set
meanFlow = mean(flow,2);
flowNormalized = flow-meanFlow;
stdFlow = std(flowNormalized(:));
flowNormalized = flowNormalized/stdFlow;
```

Healthy signals are labeled as 1 and faulty signals are labeled as 2.

Define Generator Network

Approx. Time of Jobs Duration: 1 week(5days working time)

Input File ชื่อ: backtest_dax_1day.xsl, backtest_u30_1day.xsl, backtest_u500_1day.xsl,

Output: matlab code program , backtest.m link to excel files for backtest and fill blank all columns.

Data Description and Detail of information of the TASK3 in assignment3.

Prepare excel file for input data and identify the crash periods: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash

Assignment4: SET50 GANs SVM LTSM project

Prepare excel file for input data and identify the crash periods: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash

TASK4: Prepare excel files , do calculation and do backtest on result of calculation based on the SVM model.

Code: EF1.1.1

TA name:

Audit name:

Start date:

Finish date:

Approx. Time of Jobs Duration: 1 week(5days working time)

Input File ข้อมูล: backtest_dax_1day.xls, backtest_u30_1day.xls, backtest_u500_1day.xls,

Output: matlab code program , backtest.m link to excel files for backtest and fill blank all columns.

1)

Data Description and Detail of information of the TASK1 in assignment1.

Prepare excel file for input data and identify the crash periods: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash

Assignment5: SET50 GANs anomaly score project TanoGANs

Prepare excel file for input data and identify the crash periods: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash

Introducing x MATLAB Ex: x Yahoo x DAX PERFO x Dow Jones x FRED Data x (152) x TAnoGAN/T x

github.com/mdabashar/TAnoGAN/blob/master/TAnoGAN_Pytorch.ipynb

Product Solutions Resources Open Source Enterprise Pricing Search or jump to... Sign in Sign

mdabashar / TAnoGAN Public Notifications Fork 15 Star 59

Code Issues 5 Pull requests Actions Projects Security Insights

Files

master

Go to file

- NabDataset
- models
 - README.md
 - TAnoGAN_Pytorch.ipynb
 - nab_dataset.py

TAnoGAN / TAnoGAN_Pytorch.ipynb

mdabashar Add files via upload edfda54 · 3 years ago

Preview Code Blame 993 lines (993 loc) · 101 KB Raw

TAnoGAN: Time Series Anomaly Detection with Generative Adversarial Networks

Published in: 2020 IEEE Symposium Series on Computational Intelligence (SSCI)

Paper link: <https://ieeexplore.ieee.org/abstract/document/9308512>

Import required libraries

```
In [1]: import os
import random
```

om/mdabashar/IAnoGAN/blob/master/IAnoGAN_Pytorch.ipynb

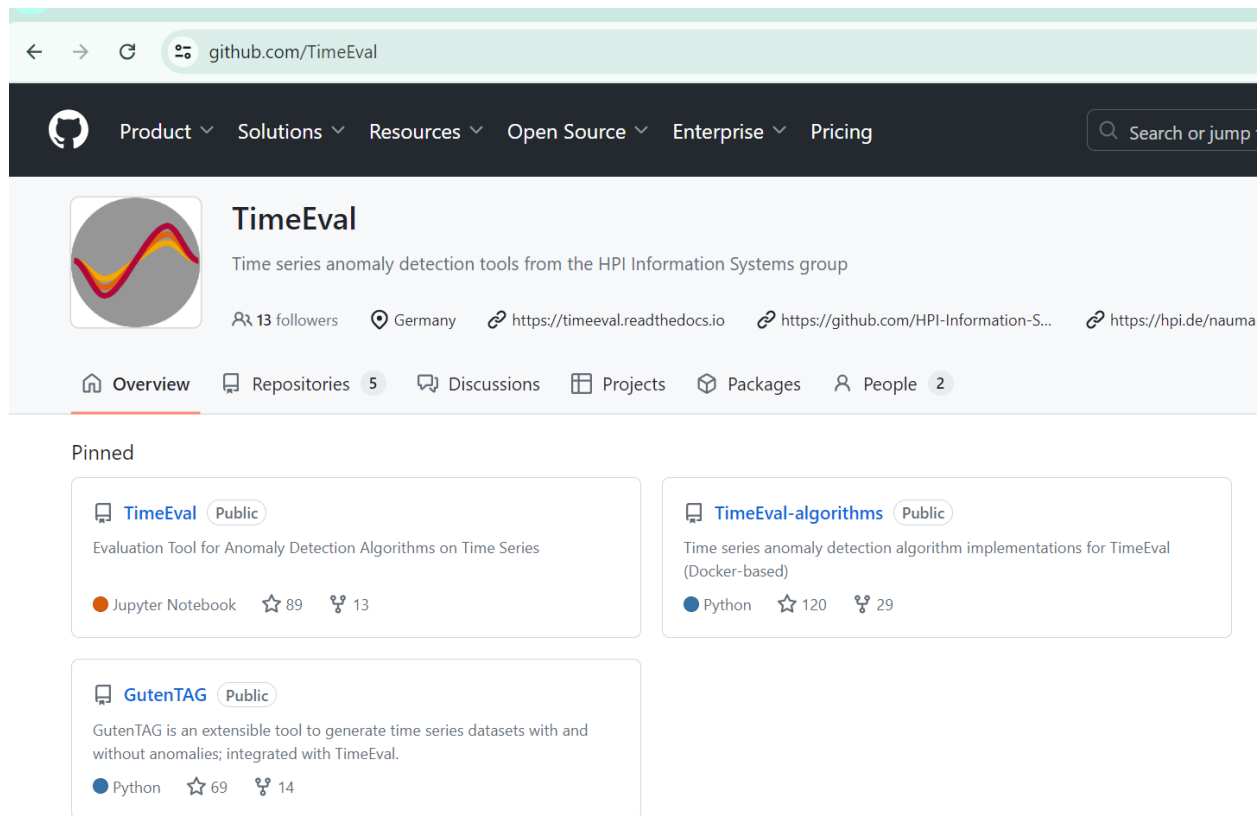
TAnoGAN / TAnoGAN_Pytorch.ipynb

Preview Code Blame 993 lines (993 loc) · 101 KB

warnings.warn(

Calculate the window-based anomalies

```
In [29]: import numpy as np
start_end = []
state = 0
for idx in test_score_df.index:
```



github.com/TimeEval

Product Solutions Resources Open Source Enterprise Pricing Search or jump

TimeEval
Time series anomaly detection tools from the HPI Information Systems group

13 followers Germany <https://timeeval.readthedocs.io> <https://github.com/HPI-Information-S...> <https://hpi.de/nauma>

Overview Repositories 5 Discussions Projects Packages People 2

Pinned

TimeEval Public

Evaluation Tool for Anomaly Detection Algorithms on Time Series

Jupyter Notebook 89 stars 13 forks

TimeEval-algorithms Public

Time series anomaly detection algorithm implementations for TimeEval (Docker-based)

Python 120 stars 29 forks

GutenTAG Public

GutenTAG is an extensible tool to generate time series datasets with and without anomalies; integrated with TimeEval.

Python 69 stars 14 forks

TASK5: Prepare excel files , do calculation and do backtest on result of calculation based on the SVM model.

Code: EF1.1.1

TA name:

Audit name:

Start date:

Finish date:

Approx. Time of Jobs Duration: 1 week(5days working time)

Input File ข้อมูล: backtest_dax_1day.xsl, backtest_u30_1day.xsl, backtest_u500_1day.xsl,

Output: matlab code program , backtest.m link to excel files for backtest and fill blank all columns.

Assignment6: SET50 Correlation force project

Prepare excel file for input data and identify the crash periods: Covid, Ukrain, Chinese market crash and do backtest for profit measurement

รวม files จาก folder set50 จากหุ้น 50 ตัวให้อยู่ใน file เดียวกัน

Job description: ตรวจสอบความถูกต้องของแหล่งที่มาของข้อมูลที่ใช้ในการคำนวณ ตามความถี่และวิธีการคำนวณในตาราง excel และทำการวัด performance และทำ back test เพื่อหา profit หรือ loss ในช่วงสาม periods ที่เกิด market crash คือ Covid, Ukrain, Chinese market crash

TASK6: Prepare excel files , do calculation and do backtest on result of calculation based on the SVM model.

Code: EF1.1.1

TA name:

Audit name:

Start date:

Finish date:

Approx. Time of Jobs Duration: 1 week(5days working time)

Input File ข้อมูล: backtest_dax_1day.xsl, backtest_u30_1day.xsl, backtest_u500_1day.xsl,

Output: matlab code program , backtest.m link to excel files for backtest and fill blank all columns.

Data Description and Detail of information of the TASK1 in assignment1.

Appendix **Number Job Codes of Effort on all TASKs for TA to monitor running on server and record to database.**

EF1. SVM

EF1.1 preprocessing SVM,

Matlab module: [ret, HR,target]=market_crash_indicator(price,theshold)

[drawdown, date]=market_crash_drawdown(price)

Requirement: compute return, market crash indicator, crash probability, drawdown, event, identify date start, crash size, crash duration.

daily data

DAX EF1.1.1

U30 EF1.1.2

U500 EF1.1.3

30min

DAX EF1.1.4

U30	EF1.1.5
U500	EF1.1.6
1hr	
DAX	EF1.1.7
U30	EF1.1.8
U500	EF1.1.9
4hr	
DAX	EF1.1.10
U30	EF1.1.11
U500	EF1.1.12

EF1.2 Data analysis :forecast SVM

Matlab module: `[ret, HR,target]=run_all_svm_forecast(price,theshold)`

Requirement: Compute prediction of market crash indicator with SVM

3 crash regions for daily data:

EF1.2.1 Covid19 crash for 100 days backward

daily data

DAX	EF1.2.1.1
U30	EF1.2.1.2
U500	EF1.2.1.3
30min	
DAX	EF1.2.1.4
U30	EF1.2.1.5
U500	EF1.2.1.6
1hr	
DAX	EF1.2.1.7
U30	EF1.2.1.8
U500	EF1.2.1.9
4hr	
DAX	EF1.2.1.10
U30	EF1.2.1.11
U500	EF1.2.1.12

EF1.2.2 Ukrainian crash for 100 days backward

daily data

DAX EF1.2.2.1

U30 EF1.2.2.2

U500 EF1.2.2.3

30min

DAX EF1.2.2.4

U30 EF1.2.2.5

U500 EF1.2.2.6

1hr

DAX EF1.2.2.7

U30 EF1.2.2.8

U500 EF1.2.2.9

4hr

DAX EF1.2.2.10

U30 EF1.2.2.11

U500 EF1.2.2.12

EF1.2.3 Chinese crash for 100 days backward

daily data

dax EF1.2.3.1

U30 EF1.2.3.2

U500 EF1.2.3.3

30min

DAX EF1.2.3.4

U30 EF1.2.3.5

U500 EF1.2.3.6

1hr

DAX EF1.2.3.7

U30 EF1.2.3.8

U500 EF1.2.3.9

4hr

DAX EF1.2.3.10

U30 EF1.2.3.11
U500 EF1.2.3.12

EF1.3 performance test of SVM for 5% and 10% threshold

Matlab module: `[TP,NP,TF,NF,F1,hit_ratio]=performance_svm(price,result_forecast)`

daily data

DAX EF1.3.1
U30 EF1.3.2
U500 EF1.3.3

30min

DAX EF1.3.4
U30 EF1.3.5
U500 EF1.3.6

1hr

DAX EF1.3.7
U30 EF1.3.8
U500 EF1.3.9

4hr

DAX EF1.3.10
U30 EF1.3.11
U500 EF1.3.12

EF1.4 Data analysis :backtest SVM

Matlab module: `[ret, HR,target]=backtest_svm(price,result_forecast)`

daily data

DAX EF1.4.1
U30 EF1.4.2
U500 EF1.4.3

30min

DAX EF1.4.4
U30 EF1.4.5
U500 EF1.4.6

1hr

DAX EF1.4.7

U30 EF1.4.8

U500 EF1.4.9

4hr

DAX EF1.4.10

U30 EF1.4.11

U500 EF1.4.12

EF2. LTSM

EF2.1 preprocessing LTSM,

Matlab module: [ret, HR,target]=market_crash_indicator(price,theshold)

daily data

DAX EF2.1.1

U30 EF2.1.2

U500 EF2.1.3

30min

DAX EF2.1.4

U30 EF2.1.5

U500 EF2.1.6

1hr

DAX EF2.1.7

U30 EF2.1.8

U500 EF2.1.9

4hr

DAX EF2.1.10

U30 EF2.1.11

U500 EF2.1.12

EF2.2 Data analysis :forecast LSTM

Matlab module: [ret, HR,target]=run_all_lstm_forecast(price,theshold)

3 crash regions for daily data:

EF2.2.1 Covid crash for 100 days backward

daily data

DAX EF2.2.1.1

U30 EF2.2.1.2

U500 EF2.2.1.3

30min

DAX EF2.2.1.4

U30 EF2.2.1.5

U500 EF2.2.1.6

1hr

DAX EF2.2.1.7

U30 EF2.2.1.8

U500 EF2.2.1.9

4hr

DAX EF2.2.1.11

U30 EF2.2.1.12

U500 EF2.2.1.13

EF2.2.2 Ukrainian crash for 100 days backward

daily data

DAX EF2.2.2.1

U30 EF2.2.2.2

U500 EF2.2.2.3

30min

DAX EF2.2.2.4

U30 EF2.2.2.5

U500 EF2.2.2.6

1hr

DAX EF2.2.2.7

U30 EF2.2.2.8

U500 EF2.2.2.9

4hr

DAX EF2.2.2.10

U30 EF2.2.2.11
U500 EF2.2.2.12

EF2.2.3 Chinese crash for 100 days backward

daily data

DAX EF2.2.3.1
U30 EF2.2.3.2
U500 EF2.2.3.3

30min

DAX EF2.2.3.4
U30 EF2.2.3.5
U500 EF2.2.3.6

1hr

DAX EF2.2.3.7
U30 EF2.2.3.8
U500 EF2.2.3.9

4hr

DAX EF2.2.3.10
U30 EF2.2.3.11
U500 EF2.2.3.12

EF2.3 Data analysis :performance of SVM for 5% and 10% threshold

Matlab module: [TP,NP,TF,NF,F1,hit_ratio]=performance_svm(price,result_forecast)

daily data

DAX EF2.3.1
U30 EF2.3.2
U500 EF2.3.3

30min

DAX EF2.3.4
U30 EF2.3.4
U500 EF2.3.6

1hr

DAX EF2.3.7

U30 EF2.3.8

U500 EF2.3.9

4hr

DAX EF2.3.10

U30 EF2.3.11

U500 EF2.3.12

EF2.4 Data analysis :backtest LSTM

Matlab module: [ret, HR,target]=backtest_svm(price,result_forecast)

daily data

DAX EF2.4.1

U30 EF2.4.2

U500 EF2.4.3

30min

DAX EF2.4.4

U30 EF2.4.5

U500 EF2.4.6

1hr

DAX EF2.4.7

U30 EF2.4.8

U500 EF2.4.9

4hr

DAX EF2.4.10

U30 EF2.4.11

U500 EF2.4.12

EF3. GANS

EF3.1 preprocessing GANs,

Matlab module: [ret, HR,target]=market_crash_indicator(price,theshold)

daily data

DAX EF3.1.1

U30 EF3.1.2

U500 EF3.1.3

30min

DAX EF3.1.4

U30 EF3.1.5

U500 EF3.1.6

1hr

DAX EF3.1.7

U30 EF3.1.8

U500 EF3.1.9

4hr

DAX EF3.1.10

U30 EF3.1.11

U500 EF3.1.12

EF3.2 Data analysis :forecast GANs

Matlab module: [ret, HR,target]=run_all_svm_forecast(price,theshold)

3 crash regions for daily data:

EF3.2.1 COVID crash for 100 days backward

daily data

DAX EF3.2.1.1

U30 EF3.2.1.2

U500 EF3.2.1.3

30min

DAX EF3.2.1.4

U30 EF3.2.1.5

U500 EF3.2.1.6

1hr

DAX EF3.2.1.7

U30 EF3.2.1.8

U500 EF3.2.1.9

4hr

DAX EF3.2.1.11

U30 EF3.2.1.12

U500 EF3.2.1.13

EF3.2.2 Ukrainian crash for 100 days backward

daily data

DAX EF3.2.2.1

U30 EF3.2.2.2

U500 EF3.2.2.3

30min

DAX EF3.2.2.4

U30 EF3.2.2.5

U500 EF3.2.2.6

1hr

DAX EF3.2.2.7

U30 EF3.2.2.8

U500 EF3.2.2.9

4hr

DAX EF3.2.2.10

U30 EF3.2.2.11

U500 EF3.2.2.12

EF3.2.3 Chinese crash for 100 days backward

daily data

dax EF3.2.3.1

U30 EF3.2.3.2

U500 EF3.2.3.3

30min

DAX EF3.2.3.4

U30 EF3.2.3.5

U500 EF3.2.3.6

1hr

DAX EF3.2.3.7

U30 EF3.2.3.8

U500 EF3.2.3.9

4hr

DAX EF3.2.3.10

U30 EF3.2.3.11
U500 EF3.2.3.12

EF3.3 Data analysis :performance of GANs for 5% and 10% threshold

Matlab module: [TP,NP,TF,NF,F1,hit_ratio]=performance_svm(price,result_forecast)

daily data

DAX EF3.3.1
U30 EF3.3.2
U500 EF3.3.3

30min

DAX EF3.3.4
U30 EF3.3.5
U500 EF3.3.6

1hr

DAX EF3.3.7
U30 EF3.3.8
U500 EF3.3.9

4hr

DAX EF3.3.10
U30 EF3.3.11
U500 EF3.3.12

EF3.4 Data analysis :backtest GANs

Matlab module: [ret, HR,target]=backtest_svm(price,result_forecast)

daily data

DAX EF3.4.1
U30 EF3.4.2
U500 EF3.4.3

30min

DAX EF3.4.4
U30 EF3.4.5
U500 EF3.4.6

1hr

DAX EF3.4.7

U30 EF3.4.8

U500 EF3.4.9

4hr

DAX EF3.4.10

U30 EF3.4.11

U500 EF3.4.12

EF4:SVM, LTSM and GANs for SET50

EF4.1 preprocessing SET50,

Matlab module: [ret, HR,target]=preprocessing(price,theshold)

daily data

SET50 EF1.1.1

EF4.2 forecast SET50,

Matlab module: [ret, HR,target]=forecast_set50(price,theshold)

3 crash regions for daily data:

EF1.2.1 Chinese crash for 100 days backward start from xxx end at xxx

daily data

SVM SET50 EF1.2.1.1

LSTM SET50 EF1.2.1.1

GANs SET50 EF1.2.1.1

EF1.2.2 Covid crash for 100 days backward start from xxx end at xxx
daily data

SM SET50 EF1.2.1.1 LSTM SET50 EF1.2.1.1
GANs SET50 EF1.2.1.1

EF1.2.2 Covid crash for 100 days backward start from xxx end at xxx
daily data

SVM SET50 EF1.2.1.1
LSTM SET50 EF1.2.1.1
GANs SET50 EF1.2.1.1

EF1.3 Data analysis :performance of SVM for 5% and 10% threshold

Matlab module: [TP,NP,TF,NF,F1,hit_ratio]=performance_svm(price,result_forecast)
daily data
SET50 EF1.1.1

EF1.4 Data analysis :backtest SVM

Matlab module: [ret, HR,target]=backtest_svm(price,result_forecast)
daily data
SET50 EF1.1.1

EF5:TanoGANs, for 39 Stocks

EF4.1 preprocessing 39stocks,

Matlab module: [ret, HR,target]=preprocessing(price,theshold)
daily data
39stocks EF1.1.1

EF4.2 forecast SET50,

Matlab module: [ret, HR,target]=forecast_set50(price,theshold)

tanoGANs 39stocks EF1.2.1.1

EF1.3 Data analysis :performance of SVM for 5% and 10% threshold

Matlab module: [TP,NP,TF,NF,F1,hit_ratio]=performance_svm(price,result_forecast)

daily data

SET50 EF1.1.1

EF1.4 Data analysis :backtest SVM

Matlab module: [ret, HR,target]=backtest_svm(price,result_forecast)

daily data

SET50 EF1.1.1

EF6. Correlation Force for SET50

EF4.1 preprocessing Correlation force

Matlab module: [ret, HR,target]=preprocessing(price,theshold)

daily data

Compute imf EF1.1.1

Compute correlation EF1.1.2

Do minimum spanning tree EF1.1.3

Visualized Tensor Network EF1.1.4

Detect Centrality EF1.1.5

Forecast Centrality EF1.1.6

Performance test EF1.1.7

Backtest EF1.1.8