User Manual

ส่วนประกอบของโปรแกรม (ภาพรวม)

ในภาพรวมโปรแกรมจะประกอบไปด้วย 3 ส่วนหลักดังนี้ Frontend, Backend และ Main algorithm

- File.Frontend

  Frontend มีหน้าที่รับคำสั่งจากผู้ใช้งานและแสดงผลลัพธ์ที่ได้จาก Main algorithm โดยไฟล์และโปรแกรมที่ใช้ทั้งหมดจะอยู่ในโฟลเดอร์ cmdf-frontend ก่อนการเปิดใช้งานนั้นจำเป็นที่จะต้องติดตั้ง nodejs ก่อน โดยวิธีการติดตั้งจะอยู่ในเอกสารหัวข้อ “วิธีการใช้งานโปรแกรม”

- File.Backend

  หน้าที่หลักของ Backend คือการเป็นตัวเชื่อมระหว่าง Frontend กับ Main algorithm โดยจะรับคำสั่งของผู้ใช้จาก Frontend ส่งไปยัง Main algorithm และส่งผลลัพธ์จาก Main algorithm ไปยัง Frontend

  ไฟล์ที่ใช้ในการทำงานของ backend จะมีดังนี้

  1. backend_server.py เป็นตัวเชื่อมระหว่าง Frontend กับ Main algorithm โดย backend_server.py จะมีหน้าที่ดังนี้
     - รับคำสั่ง ซื้อ-ขาย จาก Frontend แล้วส่งต่อให้ Main algorithm
     - รับผลลัพธ์จาก Main algorithm และปรับรูปแบบข้อมูลของผลลัพธ์(jsonify) ก่อนส่งไปยัง Frontend
     - อ่านและส่งข้อมูลที่ Frontend ต้องการ ยกตัวอย่างเช่น ชื่อหุ้นทั้งหมดของวันที่ Frontend ต้องการ
  2. utils.DataProcessor.py ใช้ทำการปรับรูปแบบข้อมูลต่างๆ

ข้อมูลที่แต่ละไฟล์ต้องการ

backend_server.py

Variable name	FilePath	หน้าที่
data (อยู่ใน ticker(working_date))	”./date_ticker.txt”	บ่งบอกว่าแต่ละวันมีหุ้นชื่ออะไรบ้าง

date_ticker.txt เมื่อถูกโปรแกรมเรียกใช้งานจะได้ข้อมูลอยู่ในรูปแบบ dictionary โดยมี keys เป็นวันที่ในรูปแบบ “YYYYMMDD” และข้อมูลข้างในจะเป็น list ที่ประกอบไปด้วยชื่อของหุ้นที่มีอยู่ในวันนั้นๆ

- File.Main algorithm

ไฟล์ที่ใช้ในการทำงานของ main algorithm จะมีดังนี้

1. Predictor.py เป็นตัวเชื่อมระหว่าง Backend กับ Main algorithm โดย Predictor.py จะรับข้อมูลคำสั่ง ซื้อ-ขาย จาก Backend แล้วส่งต่อให้ Main algorithm(model.py) แล้วส่งผลลัพธ์กลับไปยัง Backend
2. model.py เป็นไฟล์หลักที่ใช้ในการทำ Order Execution เมื่อทำงานเสร็จจะส่งข้อมูลกลับไปให้ Predictor.py
3. simulation.py ใช้ทำการจำลองการ ซื้อ-ขาย ของตลาดสำหรับการทำ Order Execution
4. Volonline.py ใช้สร้างแผนในการ ซื้อ-ขาย โดยจะกำหนด Volume และอัตราส่วน Market Order : Limit Order ของการทำ Order Execution
5. SVM_lib.py ใช้สร้างหรือเรียกข้อมูลที่มีการทำนายทิศทางราคาระยะสั้น (Price direction prediction) เพื่อนำมาใช้ในการปรับอัตราส่วน Market Order : Limit Order ของการทำ Order Execution

ข้อมูลที่แต่ละไฟล์ต้องการ

model.py

Variable name	**FilePath**	**หน้าที่**
orderbook	f"cmdf/Predictor/Data/{DATE}/pickle/orderbook.dat"	เรียก Event,Trade,Auction
Event	f"cmdf/Predictor/Data/{DATE}/pickle/{symbol}_{OrderbookID}_event.dat"	สร้าง simulation
Trade	f"cmdf/Predictor/Data/{DATE}/pickle/{symbol}_{OrderbookID}_trade2.dat"	สร้าง simulation
Auction	f"cmdf/Predictor/Data/{DATE}/pickle/{symbol}_{OrderbookID}_auction.dat"	สร้าง simulation
volpred	f"cmdf/Predictor/VolPredDir/volpred_{symbol}.obj"	สร้าง VolumeProfile (ถ้าไม่มีโค้ดจะสั่งให้สร้างขึ้นมาเอง)

Volonline.py

Variable name	FilePath	หน้าที่
Trade	f"cmdf/Predictor/Data/{DATE}/pickle/{symbol}_{OrderbookID}_trade2.dat"	บ่งบอกปริมาณการซื้อขาย

SVM_lib.py

Variable name	FilePath	หน้าที่
Data	f"cmdf/Predictor/BidOfferVolume/{quote}.dat"	สร้าง feature
df	f"cmdf/Predictor/SVM_Data/Data/{quote}_PRICE.csv"	สร้าง SVM (ถ้าไม่มีโค้ดจะสั่งให้สร้างขึ้นมาเอง)
orderbook	f"cmdf/Predictor/Data/{DATE}/pickle/orderbook.dat"	นำเลข OrderbookID ของหุ้นมาใช้เรียกข้อมูล
trade_D	f"cmdf/Predictor/Data/{d}/pickle/{symbol}_{OrderbookID}_trade2.dat"	สร้าง VWAP เพื่อทำ feature
SVM_df	f"cmdf/Predictor/SVM_Data/SVM_result/{quote}_SVM.csv"	output ของ SVM_lib.py ที่ถูกเก็บไว้(ถ้าไม่มีโค้ดจะสั่งให้สร้างขึ้นมาเอง)

คำอธิบายข้อมูล(สำหรับ model.py และ Volonline.py)

orderbook

Column's name	Type	Description
Persistent Name	String	ชื่อของหุ้นเมื่อเข้าตลาด
OrderbookID	String	Id ของหุ้น

Event

เก็บข้อมูลเหตุการณ์ต่างๆ รวมถึงเหตุการณ์ที่เกิดขึ้นกับ Limit Orederbook ของ bid และ offer โดยมีความลึกของราคา 5 ระดับ

Column's name	Type	Description
Timestamp	pandas._libs.tslibs.timestamps.Timestamp	เวลาที่เกิดเหตุการณ์
Price	Float	ราคาของการ ซื้อ-ขาย
Change	Float	ปริมาณที่เปลี่ยนแปลงใน Limit Orderbook
Flag1	String	ชนิดของเหตุการณ์

		• INSERT = เพิ่ม • UPDATE = เปลี่ยนแปลง • DELETE = ยกเลิก
Flag2	String	ชนิดของ Order
Mark	Integer	ตัวแปรที่ใช้ในการจัดข้อมูล Limit Orderbook • 0 = มีการเพิ่มปริมาณและราคาใน bid offer • 1 = เป็นการ INSERT ที่เกิดจากการเปลี่ยนแปลงช่วงราคา bid offer • 2 = เป็นการสร้าง order เทียมที่เกิดจากการจัดการข้อมูล iceberg
SeqNumber	Float	เลขลำดับเหตุการณ์

Trade

ไฟล์ที่เก็บข้อมูลเมื่อมีการ ซื้อ-ขาย เกิดขึ้น

Column's name	**Type**	**Description**
SeqNumber	Float	เลขลำดับเหตุการณ์
TimeStamp	pandas._libs.tslibs.timestamps.Timestamp	เวลาที่เกิดเหตุการณ์
EventType2	String	ชนิดของการ ซื้อ-ขาย • AUCTION = การซื้อขายแบบ ATO ATC • T_TO_T การซื้อขายที่มีการแสดงผลใน bid offer

		• TRADE_REPORT การซื้อขายที่ไม่มีการแสดงผลใน bid offer
Volume	Float	จำนวนหุ้นที่ ซื้อ-ขาย แต่ละรายการ
Price	Float	ราคาที่ ซื้อ-ขาย
OrderbookRef	Float	SeqNumber ที่เกิดจากการเปลี่ยนแปลงของ order trade นี้
OrderbookRef2	Float	SeqNumber ของ event จำลองในกรณีที่เกิด Iceburg
Side	String	ชนิดของรายการ • B = ซื้อ • S = ขาย • A = อื่นๆ

Auction

ไฟล์ข้อมูลในช่วงก่อนตลาดเปิด (ATO) และหลังตลาดปิด (ATC) ซึ่งใช้สำหรับการคำนวณเวลาเปิดการซื้อขาย และใช้ในการหา volume ในช่วง ATO และ ATC

Column's name	**Type**	**Description**
SeqNumber	Float	เลขลำดับเหตุการณ์
TimeStamp	pandas._libs.tslibs.timestamps.Timestamp	เวลาที่เกิดเหตุการณ์
Imbalance	Float	ความแตกต่างของปริมาณระหว่างฝั่ง bid กับ offer
AuctionPrice	Float	ราคาที่จับคู่กันระหว่างฝั่ง bid กับ offer
MatchQuantity	Float	ปริมาณที่จับคู่กันระหว่างฝั่ง bid กับ offer

IsFinal	String	• T เมื่อเป็นเหตุการณ์แบบ auction • F เมื่อไม่ใช่ auction

volpred

เป็น object ที่ได้จาก Volonline.py และถูกเก็บไว้ด้วย pickle และ volpred จะถูก Main Algorithm เรียกใช้งานทุกครั้ง โดยหุ้นแต่ละตัวจะต้องใช้ volpred ของหุ้นตัวนั้นๆ หาก volpred ยังไม่เคยถูกสร้างขึ้นทาง Main Algorithm จะสั่งให้สร้างขึ้นมาใหม่แล้วเก็บไว้เพื่อการใช้งานครั้งหน้า

คำอธิบายข้อมูล(สำหรับ SVM_lib.py)

Data

เป็นข้อมูลที่ถูกเก็บอยู่ในรูปแบบ Dictionary โดยมี keys เป็นวันที่(“YYYMMDD”) ข้อมูลแต่ละวันจะเป็น list ของ pandas.DataFrame ข้อมูลที่ถูกใช้จะมีเฉพาะ Data[date][0] และ Data[date][1] ซึ่งเป็นข้อมูลที่เก็บปริมาณ Limit Orderbook ของฝั่ง bid และ offer ดังภาพ

Time	57.25	58.50	60.00	63.00	63.25	64.00	64.25	64.75	65.00	65.25	.
2020-07-07 10:00:00.049	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
2020-07-07 10:00:00.136	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
2020-07-07 10:00:00.137	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
2020-07-07 10:00:00.148	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
2020-07-07 10:00:00.223	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.
2020-07-07 10:00:00.294	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.

รูปที่ 1 : ตัวอย่างข้อมูล Limit Orderbook ของฝั่ง bid หรือ offer (ผู้เขียนตั้งใจเปลี่ยนข้อมูลให้เป็น 0 เพื่อป้องกันการเผยแพร่ข้อมูล)

df

pandas.DataFrame ที่รวมข้อมูลที่จำเป็นในการสร้าง Support Vector Machine(SVM) ของหุ้นแต่ละตัวและมีความถี่ของข้อมูลเท่ากับ 5 นาที หาก df ยังไม่เคยถูกสร้างขึ้นทาง Main Algorithm จะสั่งให้สร้างขึ้นมาใหม่แล้วเก็บไว้เพื่อการใช้งานครั้งหน้า

Column's name	**Type**	**Description**
BestBid	Float	ราคา BestBid
BestOffer	Float	ราคา BestOffer
BestBidVolume	Float	จำนวนหุ้นที่อยู่ใน Limit Orderbook ที่มีราคาเท่ากับ BestBid
BestOfferVolume	Float	จำนวนหุ้นที่อยู่ใน Limit Orderbook ที่มีราคาเท่ากับ BestOffer
Price	Float	ราคา BestBid
return	Float	ผลตอบแทน 5 นาทีล่วงหน้า
rolling_std	String	ค่า Standard Deviation ของ Price 10 นาทีย้อนหลัง
local_minimum	Float	Price ที่ถูกที่สุด 5 นาทีย้อนหลัง
min-Price	Float	ค่า local_minimum – Price
FirstPrice	Float	ค่าเฉลี่ยระหว่าง BestBid และ BestOffer
OrderImbalance(at the touch)	Float	ผลต่างระหว่าง BestBidVolume และ BestVOfferVolume และปรับให้อยู่ในช่วง [-1,1]

orderbook

เหมือนกับ orderbook ที่ใช้บน model.py

trade_D

เหมือนกับ Trade ที่ใช้บน model.py แต่จะไม่มีคอลัมน์ side

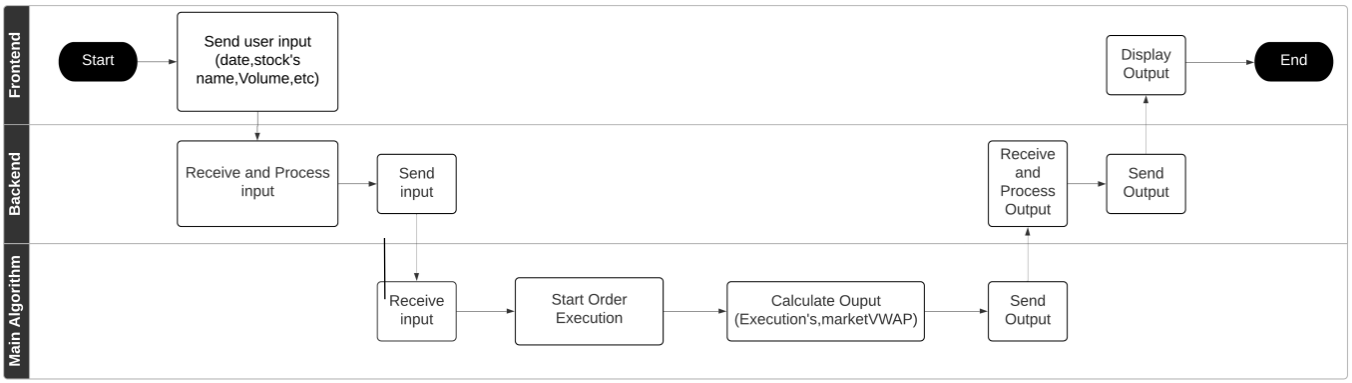
SVM_df

ข้อมูลเหมือนกับ df แต่จะมีคอลัมน์เพิ่มขึ้นมาดังนี้

Column's name	Type	Description
VWAP	Float	VWAP คำนวนจากรายการ ซื้อ-ขาย ย้อนหลัง 5 นาที
Cost	Float	คำนวนจาก ln(VWAP – Price) * 1000
Class	Float	Target Class ที่ SVM ต้องทำนาย คำนวนจาก sign(return) (ไม่ได้ใช้ข้อมูลนี้มาคิดการทำนาย ใช้เป็นเป้าหมายเท่านั้น)
pred	Integer	Class ที่ได้จากการทำนายของ SVM

Swimlane diagram

การทำงานของแต่ละส่วนของโปรแกรมสามารถสรุปวาดเป็น Swimlane diagram ได้ดังภาพ



รูปที่ 2 : Swimlane diagram ของการทำงานของโปรแกรม

คำอธิบายแต่ละส่วนของโปรแกรม

ไฟล์ main algorithm แบ่งเป็น 4 ส่วนหลักคือ simulation,volume profile, LO:MO และ price direction prediction

- Simulation
    - ใน Simulation จะแบ่งออกเป็น 3 class ได้แก่ class Queue , class Simulation และ class tracker
    - ใน class Queue มีหน้าที่ทำการจำลอง Queue เพื่อที่จะใช้ใน Simulation
        - queue เป็น List โดยแต่ละ element เป็น list ที่มีสมาชิก 2 ตัวได้แก่ id ของ order และจำนวนหุ้นที่เหลืออยู่
        - isLock ทำการกำหนดว่า queue ตอนนี้ถูก lock อยู่หรือไม่ ซึ่งจะถูก lock ในกรณีที่ queue นี้ไม่อยู่ใน 5 Bid 5 Offer
        - isPseudoLock ทำการกำหนดว่า queue นี้ถูก lock อยู่หรือไม่ ซึ่งจะถูก lock ในกรณีที่ตลาดอยู่ในช่วงที่ไม่สามารถซื้อขายได้
        - add (value , id) ทำการเพิ่ม order จำนวนเท่ากับ value ลงใน queue และตั้ง id ของ order ตามค่า id ของ input
        - remove_last (N) ทำการนำหุ้นออกจาก queue จำนวน N หุ้นนับจากท้าย queue และนำออก เฉพาะ order ที่มี id มากกว่า 0
        - remove_first (N) ทำการนำหุ้นออกจาก queue จำนวน N หุ้นนับจากหัว queue และนำออกเฉพาะ order ที่มี id มากกว่า 0
        - search_id (id) ทำการส่งกลับปริมาณหุ้นใน order ที่มี id เท่ากับ id ใน input และ return เป็น False ถ้าหากไม่พบ
        - delete_id (id) ทำการลบ order ที่มี id เท่ากับ id ใน input และ return เป็น False ถ้าหากไม่พบ และ True ถ้าหากลบสำเร็จ
        - lock (pseudo) ถ้าหาก pseudo = False จะทำการตั้งค่า isLocked เป็น True และถ้าหาก pseudo = True จะทำการตั้งค่า isPseudoLocked เป็น True
        - unlock (pseudo) ถ้าหาก pseudo = False จะทำการตั้งค่า isLocked เป็น False และถ้าหาก pseudo = True จะทำการตั้งค่า isPseudoLocked เป็น False

- get_sum(alien) ถ้าหาก alien = True จะคืนค่าจำนวนหุ้นทั้งหมดที่อยู่ใน queue และถ้าหากalien = False จะคืนค่าจำนวนหุ้นทั้งหมดที่อยู่ในqueue ที่มี id มากกว่า 0
- organize( ) ถ้าหากไม่มี order ที่มี id มากกว่า 0 เลยจะทำการนำทุก order ออกจาก queue

- ใน class Simulation มีหน้าที่ทำการจำลองการ simulation โดยจะทำการแปลงไฟล์ Trade และ Event ให้เป็น message และทำการประมวลผลทีละ message
  - __init__(Event , Trade , Auction , start_time , end_time) กำหนดค่าเริ่มต้นของตัวแปรต่างๆ มี input เป็นไฟล์ Event , Trade และ Auction โดยมี start_time และ end_time เป็นเวลาเริ่มต้นของการซื้อขาย และเวลาสิ้นสุดของการซื้อขาย อยู่ในรูปของ tuple โดยมีค่าพื้นฐาน start_time ที่ (10 , 5) และend_time ที่ (16 , 29)
  - __feed (msg) ใช้ในการประมวลผล message
  - feed_until (time) ประมวลผล message จนถึงเวลา time โดยเวลา time จะอยู่ในรูป datetime.datetime
  - forward ( ) ทำการเลื่อนเวลาไปอีก 1 นาทีถัดไป
  - do ( ) ทำการประมวลผล message จนถึงเวลาปัจจุบัน
  - update ( ) ทำการอัพเดทสถานะต่างๆ เช่น BestBid BestOffer isFloor isCeiling
  - isClosed ( ) ทำการคืนค่าว่าตอนนี้เลยเวลาในการ trade ที่เราต้องการแล้วหรือยัง
  - __transit ( ) ใช้ในกรณีที่เมื่อเลื่อนเวลาถัดไป 1 นาทีแล้วอยู่ในช่วงพักกลางวัน และจะทำการปรับเวลาให้เป็นเวลาซื้อขายตอนบ่าย
  - addLO (Price , amount) ทำการเพิ่ม limit order จำนวน amount หุ้น ที่ราคา Price บาท ซึ่ง Price สามารถเป็น string ‘BestBid’ หรือ ‘BestOffer’ ได้
  - addMO (amount , side) ทำการเพิ่ม market order ไปยังฝั่ง side จำนวน amount หุ้น ซึ่ง side เป็น string ‘Bid’ เมื่อทำการขาย , ‘Offer’ เมื่อทำการซื้ออ
  - cancleLO (id) ทำการ cancel LO ใน order ที่มี id เป็น id ที่ได้จาก input
  - myVWAP ( ) ทำการคำนวณว่าปัจจุบัน VWAP ของเราอยู่ที่เท่าไร
  - marketVWAP ( ) ทำการคำนวณ VWAP ของตลาดทั้งวัน

- ใน class tracker มีหน้าที่ทำการติดตามสถานะของ order ของเราที่ได้วางลงไป
  - __init__ (queue , Mapper) ทำการกำหนดค่าเริ่มต้นของตัวแปร โดย queue เป็น dict ของ object simulation.Queue ซึ่งเก็บข้อมูล queue ในแต่ละระดับราคา และ Mapper เป็น dict ของ integer ซึ่งเก็บว่า order id ต่างๆถูกวางไว้ที่ระดับราคาใดบ้าง
  - add(id , Time , Type , Side , Level , InitValue) ทำการเพิ่มข้อมูลลงไปในการติดตาม โดย id คือ id ของ order , Time คือเวลาที่วาง order , Type คือประเภทของ order สามารถเป็นได้ 2 แบบคือ ‘MO’ หรือ ‘LO’ , Side เป็นฝั่งของ order สามารถเป็นได้ 2 แบบคือ ‘Bid’ หรือ ‘Offer’ , Level เป็นระดับราคาของ order และ InitValue เป็นจำนวนหุ้นที่ถูกเพิ่มลงมา
  - update ( ) ทำการอัพเดทสถานะของแต่ละ order
  - lock (id) ทำการ lock order ใน tracker ที่มี id เท่ากับ id ใน input โดยแสดงถึงว่า order นั้นไม่มีอยู่บนตลาดแล้ว ซึ่งอาจจะเกิดจากการ execute จนหมด หรือ cancel order
  - get ( ) ทำการเรียก tracker ออกมาดู ซึ่งมี column ตาม input ใน __init__ ซึ่งเพิ่ม Column CurrentValue คือจำนวนหุ้นในขณะนี้ที่มีอยู่ในตลาด และ Column status แสดงว่าตอนนี้ถูก lock หรือไม่

- การใช้งานใน Main algorithm
  - จะมีการเรียกใช้งานเพียง class simulation ซึ่งจะไปเรียกใช้งานและจัดการ class Queue และ class tracker โดยอัตโนมัติ
  - ในการเรียกใช้งานจะเริ่มต้นด้วยการสร้าง class simulation ขึ้นมาจากนั้นทำการ loop ในแต่ละเวลา
    - เริ่มต้นแต่ละ loop จะทำการเรียก simulation.do เพื่อทำการจำลองตลาดจนถึงเวลาปัจจุบัน
    - ก่อนสิ้นสุด loop จะทำการเรียก simulation.forward เพื่อทำการเลื่อนเวลาไปอีก 1 นาที

```
sim  = simulation.Simulation( Event , Trade , Auction )
while sim.isClosed() == False:
    sim.do()
    sim.addLO ( 'BestBid' , 100 )
    sim.addMO ( 100 , 'Offer' )
    sim.forward()
```

รูปที่ 3 : ตัวอย่างการแสดงการใช้งาน simulation อย่างง่าย โดยจะวาง LO ที่ best bid จำนวน 100 หุ้นทุกๆ 1 นาที และทำการซื้อหุ้นจากฝั่ง Offer จำนวน 100 หุ้นทุกๆ 1 นาที

```
sim.tracker.get()
```

ID	Time	Type	Side	Level	InitValue	CurrentValue	Status
-1	2020-11-23 10:05:00	LO	Bid	63.00	100	0	False
0	2020-11-23 10:05:00	MO	Offer	63.25	100	0	False
-2	2020-11-23 10:06:00	LO	Bid	63.00	100	0	False
0	2020-11-23 10:06:00	MO	Offer	63.25	100	0	False
-3	2020-11-23 10:07:00	LO	Bid	63.00	100	0	False
0	2020-11-23 10:07:00	MO	Offer	63.25	100	0	False
-4	2020-11-23 10:08:00	LO	Bid	63.00	100	0	False
0	2020-11-23 10:08:00	MO	Offer	63.25	100	0	False
-5	2020-11-23 10:09:00	LO	Bid	63.00	100	0	False
0	2020-11-23 10:09:00	MO	Offer	63.25	100	0	False
-6	2020-11-23 10:10:00	LO	Bid	63.00	100	0	False
0	2020-11-23 10:10:00	MO	Offer	63.25	100	0	False
-7	2020-11-23 10:11:00	LO	Bid	63.00	100	100	True
0	2020-11-23 10:11:00	MO	Offer	63.25	100	0	False
-8	2020-11-23 10:12:00	LO	Bid	63.00	100	100	True

รูปที่ 4 : ตัวอย่างการแสดงการเรียก tracker จากการทำงานของโค้ดในรูปที่ 3 แสดงให้เห็นว่า limit order ไม่ถูก execute ในช่วงเวลา 10.11 และ 10.12

- Volume profile และ LO:MO
    - การทำงานของ Volume profile และ อัตราส่วน LO:MO จะถูกแบ่งออกเป็นสองส่วนได้แก่ การทำนาย Volume การคำนวณ Volume profile หรือ อัตราส่วน LO:MO
    - โปรแกรมที่ใช้ทำนาย Volume นั้นจะอยู่ในไฟล์ VolOnline.py class VolPred

- __init__(quote , number) สร้าง model สำหรับการทำนาย Volume ของหุ้นที่มีชื่อเท่ากับ quote และมีหมายเลขเท่ากับ number โดยโปรแกรมจะใช้ข้อมูลที่มีอยู่ทั้งหมดยกเว้น 100 วันสุดท้ายในการ train มี features เป็น Volume ของตลาดและปริมาณ ATO และมี output อยู่ในรูปของ dict ที่มี key เป็นเวลาที่ต้องการ ยกตัวอย่างเช่น models[(10,30)][(16,30)] คือ models สำหรับการทำนาย volume ทั้งหมดในช่วงเวลา 10.30 – 16.30

```
VolPred = VolOnline.VolPred('CPALL','1860')
```

```
VolPred.models[ ( 10 , 30 ) ][ ( 16 , 30 )]
```

```
LinearRegression()
```

รูปที่ 5 : ตัวอย่างการแสดงการทำงานของ VolPred.__init__

- predict(start , end , volume , ATO) ทำนาย Volume ตั้งแต่เวลา start จนถึงเวลา end โดยใช้ features เป็นปริมาณ Volume และปริมาณ ATO ยกตัวอย่างเช่น predict((10 , 30) , (16 , 30) , 100000 , 10000) หมายถึงทำนาย Volume ในช่วงเวลา 10.30น – 16.30น โดยมี volume ตลาด ณ เวลา 10.30น ที่ 100000 หุ้น และมีปริมาณ ATO 10000 หุ้น

```
VolPred = VolOnline.VolPred('CPALL','1860')
```

```
VolPred.predict( ( 10 , 30 ) , ( 16 , 30 ) , 100000 , 10000 )
```

```
array([10915441.18843938])
```

รูปที่ 6 : ตัวอย่างการแสดงการทำงานของ VolPred.predict

- โปรแกรมที่ใช้คำนวณ Volume profile และอัตราส่วน LO:MO นั้นจะอยู่ในไฟล์ VolOnline.py class VolOnline
  - __init__(volpred , sim , start , end) ใช้ในการกำหนดค่าเริ่มต้นของโปรแกรม โดย volpred เป็น object class VolOnline.VolPred , sim เป็น object class simulation.Simulation มีเวลา start และ end เป็นเวลาเริ่มต้นการซื้อขายและสิ้นสุดการซื้อขาย อยู่ในรูป tuple เช่น start = (10 , 30) , end = (16 , 30) หมายถึงซื้อขายตั้งแต่เวลา 10.30น. – 16.30น.

- get_plan( ) ทำการคำนวณ Volume profile และอัตราส่วน LO:MO แล้วทำการคืนค่าออกมาในรูปของ เปอร์เซ็นของ volume ที่เหลือ และอัตราส่วน LO:MO โดย function นี้จะทราบเวลาและ features ที่ใช้ในการ predict Volume จาก simulation ที่ถูกใส่เข้าไปใน __init__

```
VolPred    = VolOnline.VolPred('CPALL','1860')
```

```
sim        = simulation.Simulation( Event , Trade , Auction )
sim.do()
```

```
plan       = VolOnline.VolumeOnline( VolPred , sim )
```

```
plan.getPlan()
```

```
(0.0029593844337917606, 0.75)
```

รูปที่ 7 : ตัวอย่างการแสดงการทำงานของ VolumeOnline ที่เวลาแรกของการซื้อขาย โดยให้ซื้อ 0.296 % ของ Volume ที่เหลือที่ต้องซื้อ และมีอัตราส่วน MO 75%

- การใช้งานใน main algorithm
  - การเริ่มต้นของ algorithm จะเริ่มจากการสร้าง VolOnline.VolPred เสมอ ทั้งนี้จะบันทึก object นี้ไว้เป็นไฟล์ถ้าหากยังไม่เคยคำนวณ และถ้าหากเคยคำนวณแล้วให้อ่านไฟล์ที่ถูกบันทึกขึ้นมาแทน
  - หลังจากนั้นจึงมีการเรียกใช้ VolOnline.VolumeOnline เพื่อทำการกำหนดค่าเริ่มต้นก่อนที่จะทำการซื้อขาย
  - ในทุกๆเวลา จะมีการเรียกใช้ VolOnline.VolumeOnline.predict เพื่อที่จะทำการคำนวณหา Volume ที่ต้องซื้อตอนนั้น และ อัตราส่วน LO:MO ที่ต้องซื้อตอนนั้น

- Price direction prediction
  - ตัวโปรแกรมจะใช้ Support Vector Machine(SVM) ในการแบ่งประเภททิศทางของราคาที่จะเกิดขึ้น 5 นาทีข้างหน้า
  - โปรแกรมที่ใช้ในการสร้าง Price direction prediction นั้นจะอยู่ในไฟล์ SVM_lib.py โดยจะมีฟังก์ชันดังนี้
    - get_data(*quote*) ใช้เรียกหรือสร้างข้อมูลที่จำเป็นต้องใช้ในการสร้าง SVM ของหุ้นที่มีเครื่องหมายเท่ากับ *quote* ข้อมูลที่ได้จะเป็นดังภาพ

Time	BestBid	BestOffer	BestBidVolume	BestOfferVolume	Prices	return	rolling_std	local_minimum	min - Prices	FirstPrice	OrderImbalance(at the touch)
2017-12-01 10:05:00	73.50	73.75	56700.0	296000.0	73.50	-0.003407	0.000000	73.50	0.000000	73.625	-0.678480
2017-12-01 10:10:00	73.25	73.50	426900.0	278500.0	73.25	0.000000	0.176777	73.50	0.003401	73.375	0.210377
2017-12-01 10:15:00	73.25	73.50	316900.0	330000.0	73.25	0.000000	0.000000	73.25	0.000000	73.375	-0.020250
2017-12-01 10:20:00	73.25	73.50	535300.0	149000.0	73.25	0.000000	0.000000	73.25	0.000000	73.375	0.564518
2017-12-01 10:25:00	73.25	73.50	531500.0	105900.0	73.25	0.000000	0.000000	73.25	0.000000	73.375	0.667713
...	...	...	...	...	...	...	...	...	...	...	...
2020-11-23 16:05:00	63.00	63.25	1490800.0	1625500.0	63.00	0.003960	0.000000	63.00	0.000000	63.125	-0.043224
2020-11-23 16:10:00	63.25	63.50	736500.0	776100.0	63.25	-0.003960	0.176777	63.00	-0.003968	63.375	-0.026180
2020-11-23 16:15:00	63.00	63.25	1338900.0	870300.0	63.00	0.000000	0.176777	63.25	0.003953	63.125	0.212113
2020-11-23 16:20:00	63.00	63.25	1765200.0	830500.0	63.00	0.000000	0.000000	63.00	0.000000	63.125	0.360096
2020-11-23 16:25:00	63.00	63.25	1833800.0	1597500.0	63.00	0.000000	0.000000	63.00	0.000000	63.125	0.068866

รูปที่ 8 : ตัวอย่างข้อมูลที่ได้จาก get_data(*quote*)

- get_VWAP(*quote,df*) ใช้สร้าง VWAP จากข้อมูลการ ซื้อ-ขาย ย้อนหลัง 5 นาทีของหุ้นที่มีเครื่องหมายเท่ากับ *quote* และจำเป็นต้องใช้ข้อมูลในรูปแบบ pandas.DataFrame ที่ได้จาก get_data(*quote*) หรือเรียกว่า *df* เพื่อให้ VWAP ที่ได้สอดคล้องกับเวลาใน *df*
- get_SVM_df(*quote*) สร้าง pandas.DataFrame ที่มีข้อมูลการทำนายทิศทางราคา (SVM_df) ของหุ้นที่มีเครื่องหมายเท่ากับ *quote* อยู่ในรูปแบบ pandas.DataFrame โดยค่าการทำนายทิศทางราคาจะอยู่ในคอลัมน์ที่มีชื่อว่า pred ดังภาพ

Time	OrderImbalance(at the touch)	min - Prices	rolling_std	Cost	pred
2019-11-15 15:40:00	-0.265936	0.000000	0.000000	21.551732	0
2019-11-15 15:45:00	-0.302695	0.000000	0.000000	5.560751	-1
2019-11-15 15:50:00	-0.122021	0.000000	0.000000	13.546319	0
2019-11-15 15:55:00	0.002747	0.000000	0.000000	10.133479	0
2019-11-15 16:00:00	0.058883	0.000000	0.000000	6.053362	0
...	...	...	...	...	...
2020-11-23 16:05:00	-0.043224	0.000000	0.000000	18.817684	0
2020-11-23 16:10:00	-0.026180	-0.003968	0.176777	0.272633	-1
2020-11-23 16:15:00	0.212113	0.003953	0.176777	39.679776	1
2020-11-23 16:20:00	0.360096	0.000000	0.000000	32.893209	1
2020-11-23 16:25:00	0.068866	0.000000	0.000000	29.147849	0

รูปที่ 9 : ตัวอย่างข้อมูลที่ได้จาก get_SVM_df(*quote*)
(ในภาพนี้ไม่ได้แสดง column ของข้อมูลครบทุกแถว)

ในการสร้าง SVM นั้นข้อมูลของแต่ละคลาสจะมีจำนวนที่แตกต่างกันอยู่มาก เราจำเป็นต้องทำการ oversampling ข้อมูลก่อนที่จะนำข้อมูลมาสร้าง SVM

- load_SVM_df(*quote*) จะเรียกหรือสั่งให้สร้าง SVM_df ของหุ้นที่มีเครื่องหมายเท่ากับ *quote*

- การใช้งานใน Main algorithm
  - ใน Main algorithm นั้นจะมีเพียง load_SVM_df(*quote*) ที่ถูกเรียกใช้งานเท่านั้น หากตัวฟังก์ชันหา SVM_df ไม่เจอจะเรียก get_SVM_df(*quote*) เพื่อสร้าง SVM_df ขึ้นมา โดยในตัวฟังก์ชัน get_SVM_df(*quote*) จะเรียก get_data(*quote*) และ get_VWAP(*quote,df*) เพื่อนำข้อมูลมาสร้าง SVM ต่อไป
  - get_SVM_df(*quote*) และ get_data(*quote*) เมื่อทำงานเสร็จจะเก็บข้อมูลที่สร้างขึ้นไว้ในโฟลเดอร์ที่เรากำหนดสำหรับการใช้งานในครั้งถัดไป
  - การ oversampling ใน get_SVM_df(*quote*) จะส่งผลให้ใช้เวลาในการสร้าง SVM นาน เราจึงเปลี่ยนเป็นการใช้ class_weight = 'balanced' แทนแต่เรายังคงเก็บโค้ดส่วน oversampling ในรูปแบบ comment แทนดังภาพ

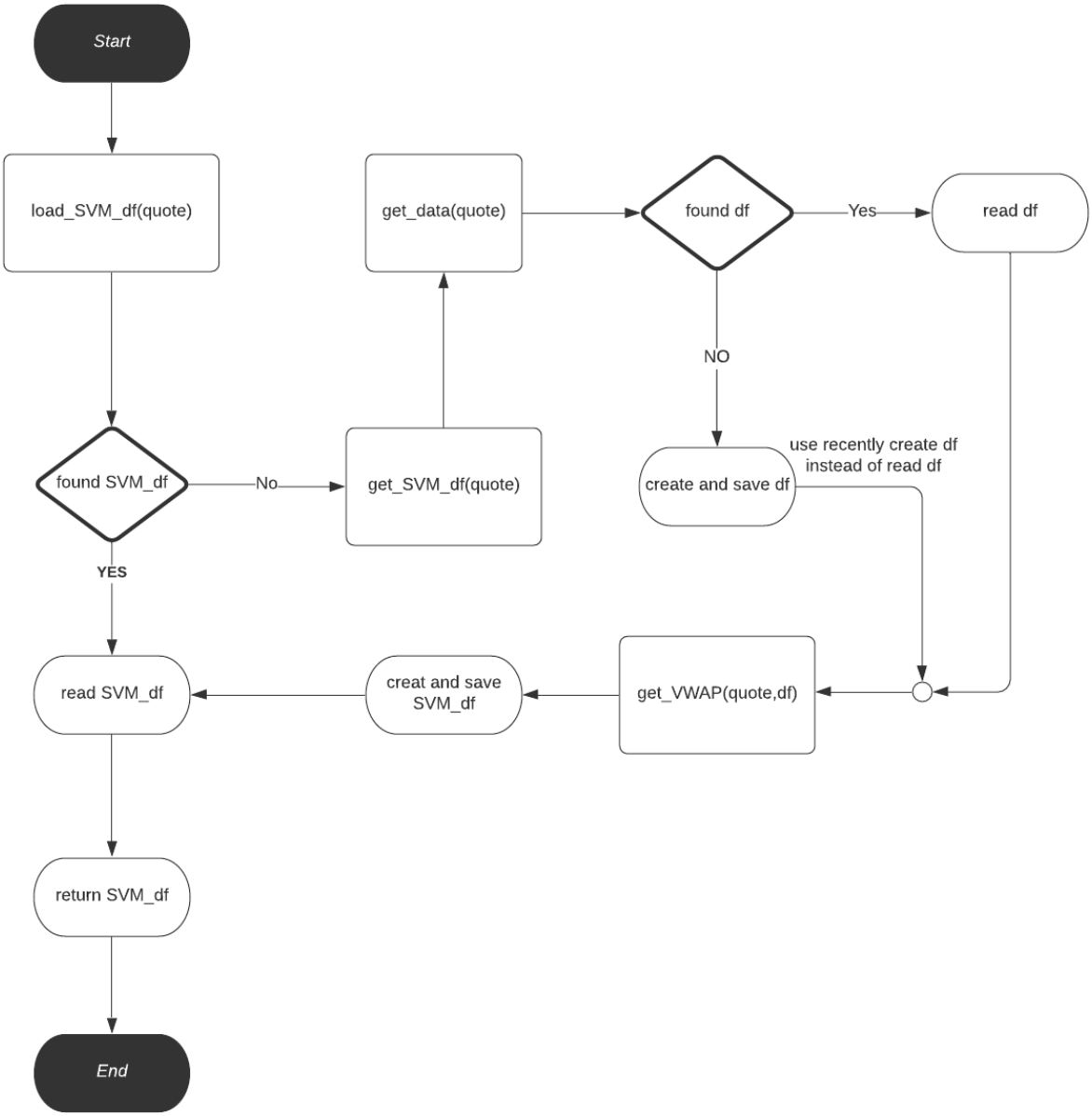
```
163         #oversample data using smote to prevent model from predict class 0 only
164         oversample = SMOTE()
165         '''try:
166             X_train, y_train = oversample.fit_resample(X_train, y_train)
167         except:
168             pass'''
169         clf = make_pipeline(StandardScaler(), SVC(kernel="linear",class_weight='balanced'))
```

รูปที่ 10 : แสดงโค้ดส่วน oversampling ข้อมูลใน SVM_lib.py

หากผู้ใช้งานต้องการใช้การ oversampling แทนผู้ใช้งานสามารถนำ Triple Quotes ออกได้

load_SVM_df(*quote*)'s flowchart

การเรียก load_SVM_df(*quote*) สามารถเขียนเป็น flowchart ได้ดังภาพ



รูปที่ 11 : flowchart การเรียก load_SVM_df(*quote*)

วิธีการใช้งานโปรแกรม

Software ที่จำเป็นต้องใช้และ Version จะมีดังนี้

- Python version 3.8.5
- Frontend
  - React version 17.02
- Backend
  - Python's Library
    - Flask version 1.1.2
    - flask_cors version 3.0.10
- Algorithm
  - Python's Library
    - Numpy version 1.19.2
    - Pandas version 1.1.3
    - Imblearn version 0.8.0
    - Sklearn version 0.24.2

1. ขั้นตอนการ install nodejs และ npm

   1.1. เปิดใช้งาน terminal แล้วพิมพ์คำสั่ง sudo apt install nodejs

   1.2. ไปยังโฟลเดอร์ ./cmdf-frontend ผ่าน terminal

   1.3. ใช้คำสั่ง npm install ดังภาพ

รูปที่ 12 : วิธีการว install npm

2. ขั้นตอนเปิด frontend

   2.1. ใช้คำสั่ง npm start ดังภาพ

รูปที่ 13 : วิธีการเปิดใช้งาน frontend

   2.2. frontend จะทำงานและสามารถเข้าถึงได้ที่ port 3000

3. ขั้นตอนการเปิด backend

   3.1. ไปยังโฟลเดอร์ ./cmdf ผ่าน terminal

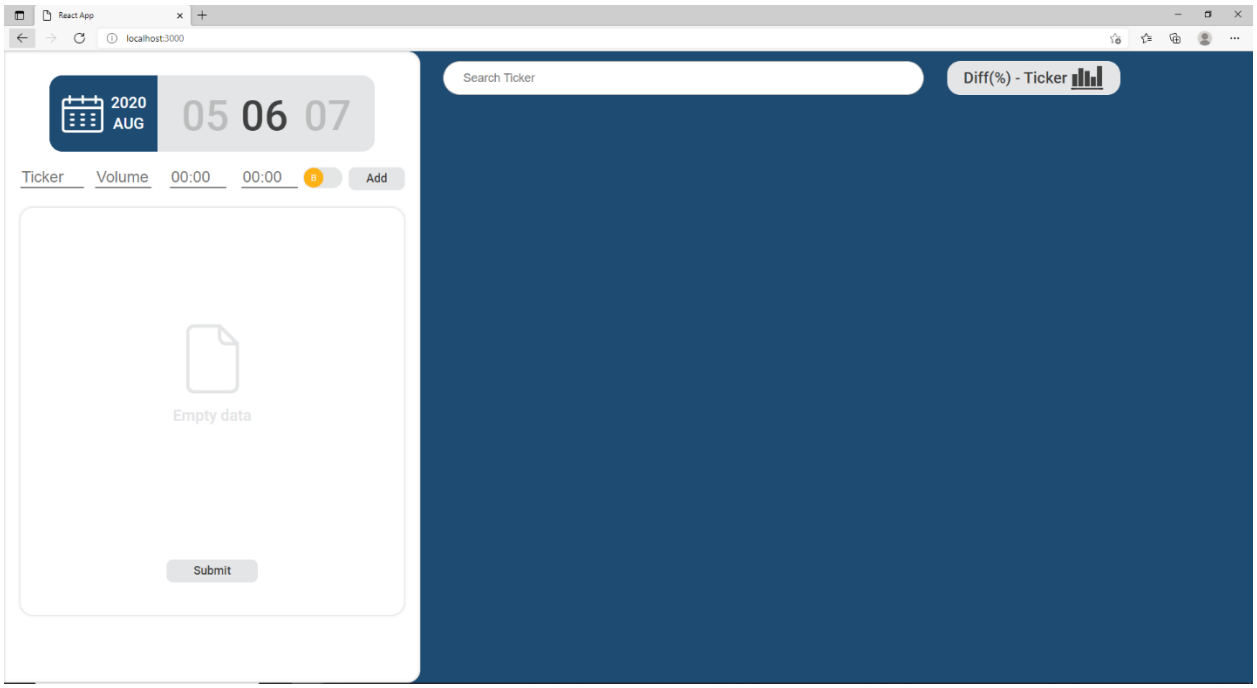
   3.2. ใช้คำสั่ง python backend_server.py ดังภาพ

รูปที่ 14 : วิธีการเปิดใช้งาน backend

   3.3. backend จะทำงานและสามารถเข้าถึงได้ที่ port 1111

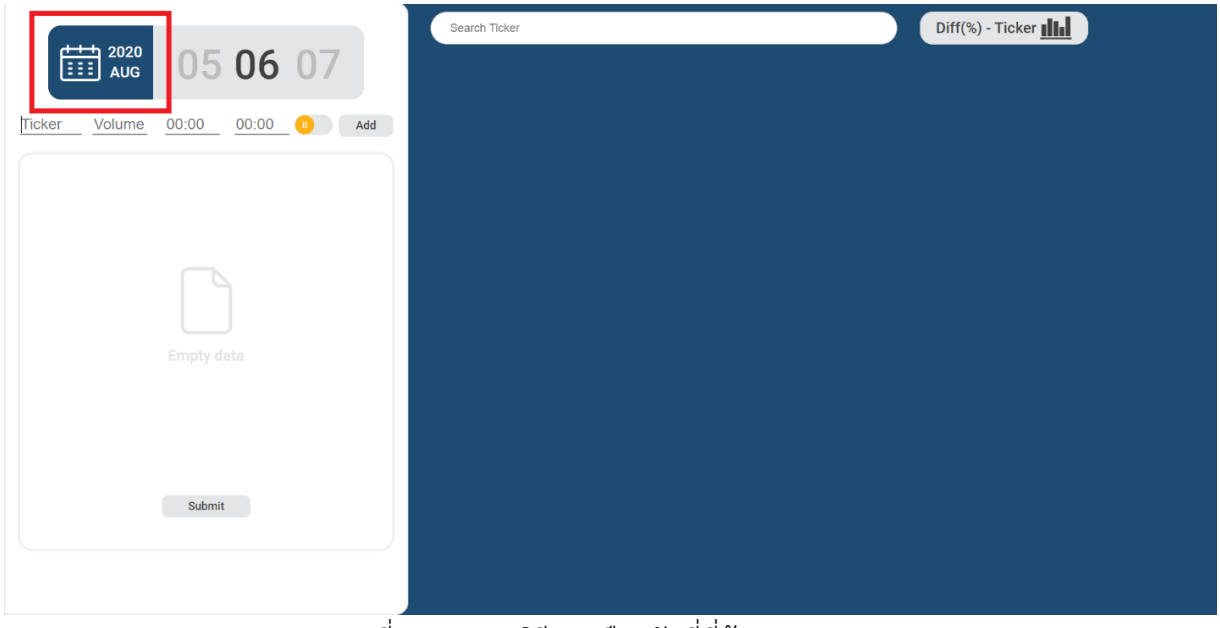
4. การใช้งานเว็บไซต์

4.1. เมื่อเปิดใช้งานทั้ง frontend และ backend เสร็จสิ้นจะสามารถเข้าใช้งานโปรแกรมได้ผ่านทางเบราว์เซอร์โดยใช้ url เป็น address ของเครื่องที่เปิดใช้งาน port 3000 โดยตัวอย่างที่ใช้จะให้ url ของเบราว์เซอร์เป็น localhost:3000

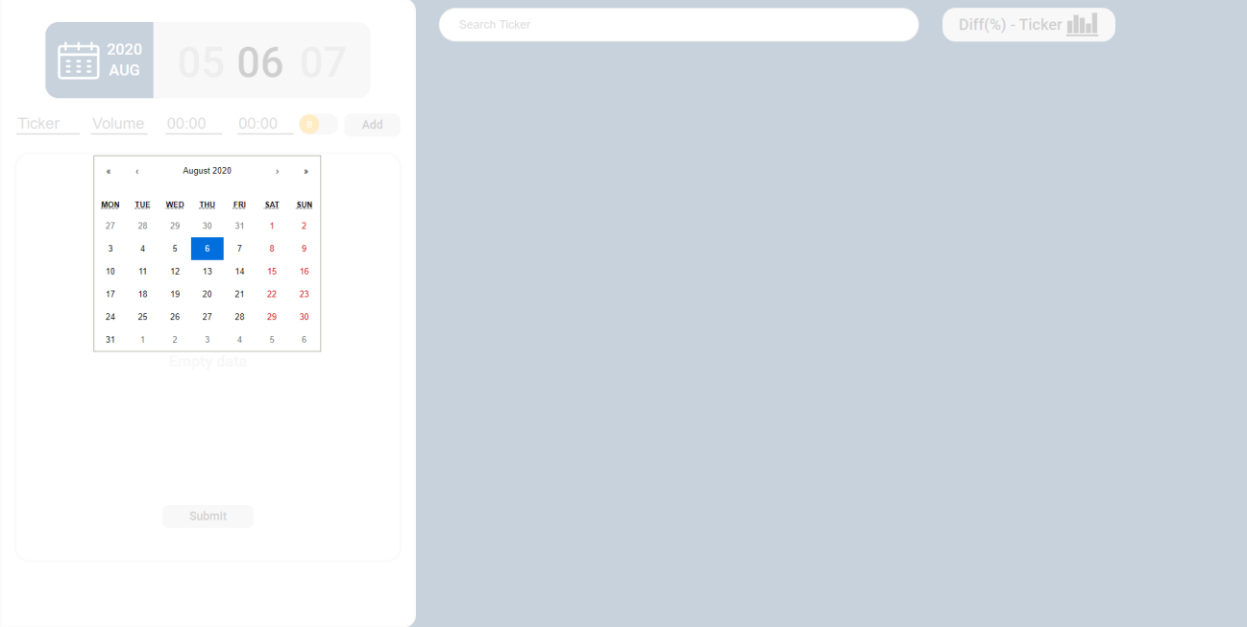


รูปที่ 15 : หน้าหลักของเว็บไซต์

4.2. เลือกวันที่ที่ต้องการโดยการคลิกที่รูปปฏิทิน

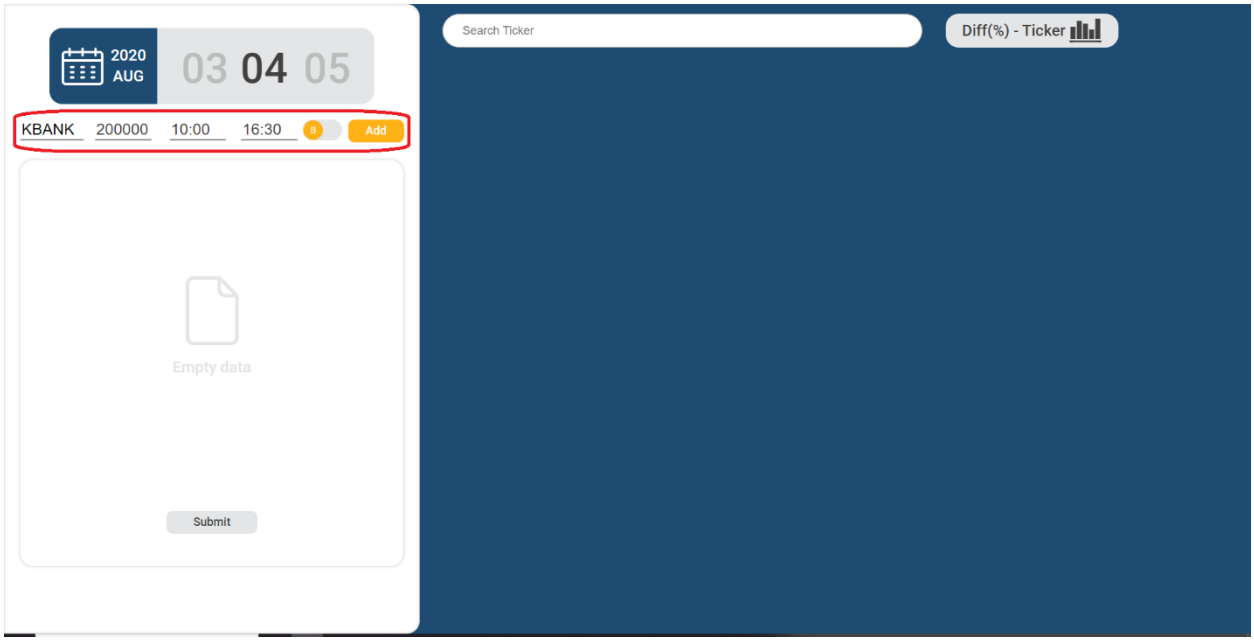


รูปที่ 16 : แสดงวิธีการเลือกวันที่ที่ต้องการ



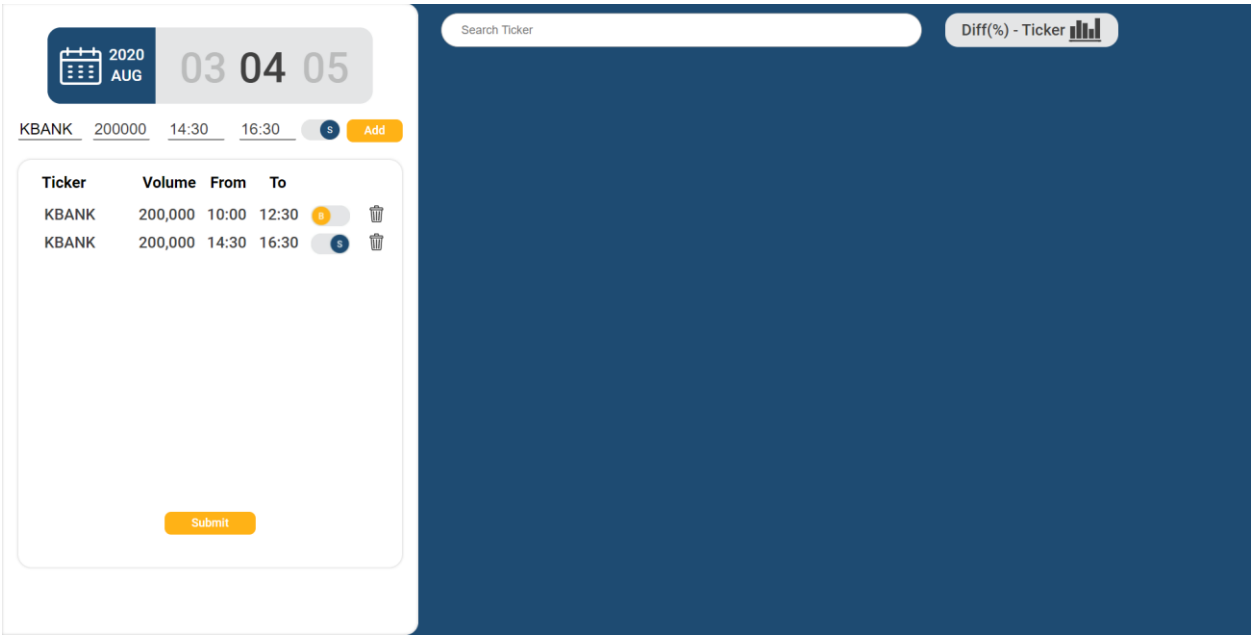
รูปที่ 17 : แสดงวิธีการเลือกวันที่ที่ต้องการ

4.3. กรอกข้อมูลของคำสั่ง ซื้อ-ขาย ที่ต้องการโดยสามารถกรอก ชื่อหุ้น, จำนวนหุ้นที่ต้องการ ซื้อ-ขาย, เวลาที่ต้องการ ซื้อ-ขาย และ เลือกที่จะซื้อหรือขาย



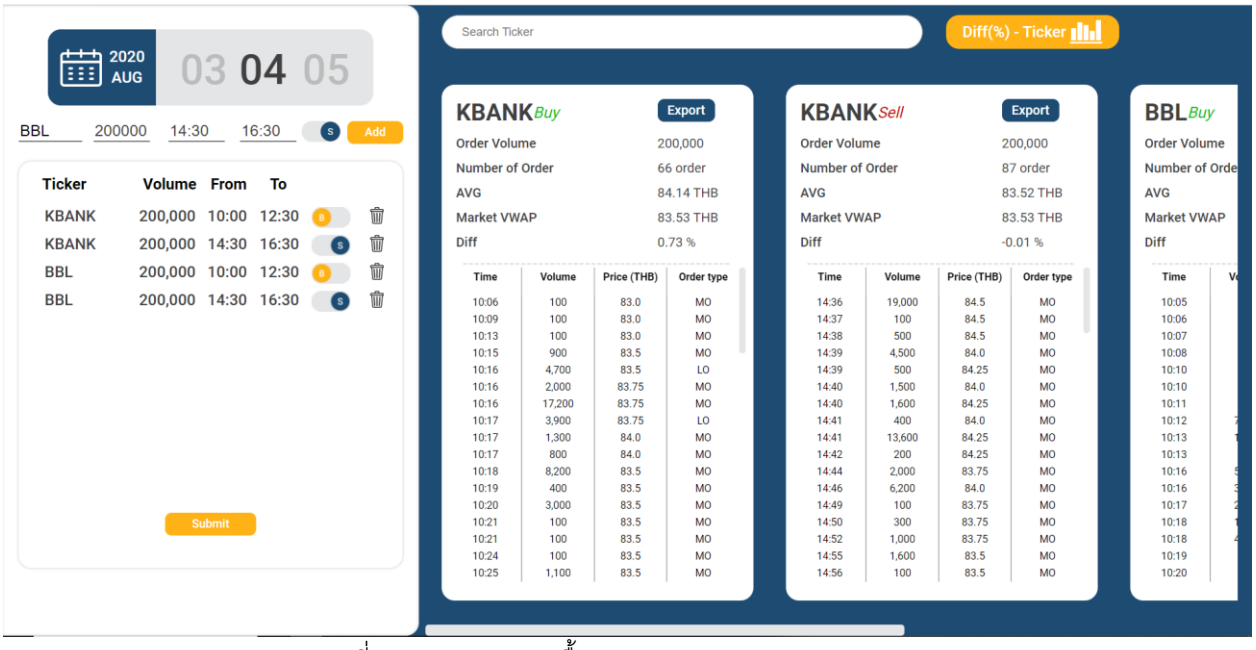
รูปที่ 18 : แสดงวิธีการกรอกข้อมูลคำสั่ง ซื้อ-ขาย

4.4. เมื่อกรอกคำสั่ง ซื้อ-ขาย ที่ต้องการเสร็จสิ้น หน้าเว็บไซต์จะแสดงรายการดังภาพ



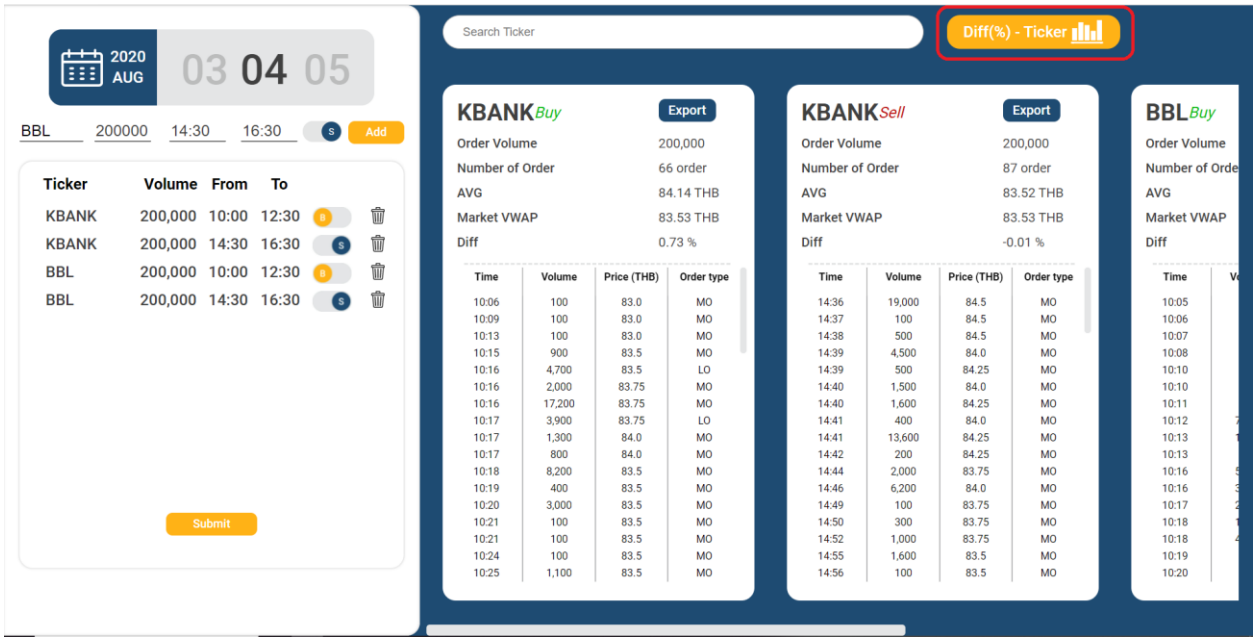
รูปที่ 19 : แสดงรายการคำสั่ง ซื้อ-ขาย

4.5. เมื่อกดปุ่ม Submit ระบบจะทำการจำลองการ ซื้อ-ขาย และส่งคำสั่ง ซื้อ-ขาย ที่ตัดสินใจกลับมาโดยจะระบุเวลา, จำนวนหุ้น, ราคา, และ ชนิดของคำสั่ง ดังภาพ

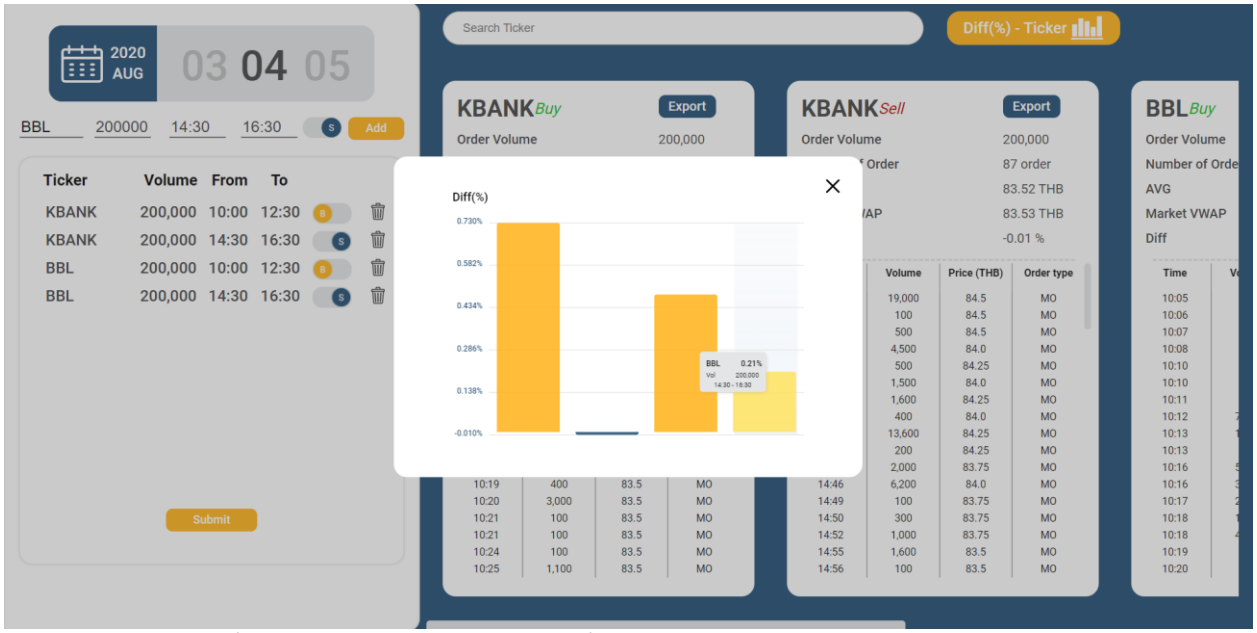


รูปที่ 20 : แสดงผลการ ซื้อ-ขาย ของแต่ละรายการ

4.6. สามารถดูผลส่วนต่าง VWAP ของระบบเมื่อเทียบกับตลาดในรูปแบบกราฟแท่งได้เมื่อกดปุ่มดังภาพ



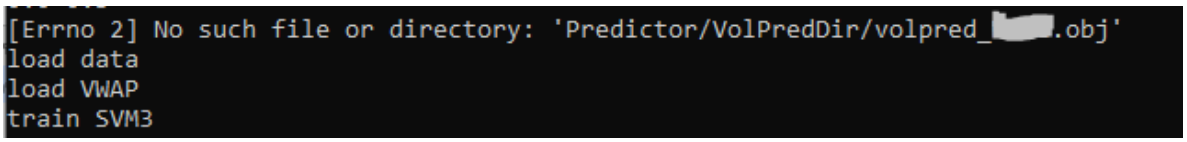
รูปที่ 21 : วิธีการแสดงผลส่วนต่าง VWAP ของระบบเมื่อเทียบกับตลาดในรูปแบบกราฟแท่ง



รูปที่ 22 : ส่วนต่าง VWAP ของระบบเมื่อเทียบกับตลาดในรูปแบบกราฟแท่ง

ข้อจำกัดของโปรแกรม หรือข้อควรระวัง

- ถ้าหากผู้ใช้งานต้องการ ซื้อ-ขาย โดยที่หุ้นตัวนั้นไม่เคยถูกสร้าง volpred หรือ SVM_df มาก่อน ขั้นตอนในการสร้างข้อมูลเหล่านี้จะใช้เวลานานพอสมควร ซึ่งทางเว็บไซต์จะไม่บอกสถานะว่ากำลังสร้าง ข้อมูลเหล่านี้อยู่ ช่องทางที่สามารถบ่งบอกได้ว่ากำลังสร้างข้อมูลอยู่มีเพียงการเข้าไปดูใน screen ของ terminal ที่สั่งใช้งาน Backend เท่านั้นดังภาพ



รูปที่ 23 : screen ของ terminal ที่สั่งใช้งาน Backend ในขณะที่โปรแกรมกำลังสร้าง volpred และ SVM_df

- ประสิทธิภาพของโปรแกรมหากนำไปใช้กับหุ้นสภาพคล่องต่ำจะแย่ลง เพราะ LO ที่โปรแกรมสั่งไว้จะถูกจับคู่กับตลาดยากขึ้น
- หุ้นที่มีราคา tick size เมื่อเทียบกับรารคาแล้วมีขนาดใหญ่อย่างเช่น หุ้นที่มีราคา 0.01 , 0.02 ความต่างระหว่าง LO และ MO จะมีค่าเท่ากับ 100 % ส่งผลให้ผลต่าง VWAP เมื่อเทียบกับตลาดมีค่าสูง
- หุ้นที่มีข้อมูลการซื้อขายน้อยจะล่งผลให้ความแม่นยำของ volume prediction ลดน้อยลง
- volume prediction จะมีความแม่นยำสูงเอใกล้เวลาปิดทำการตลาด นั่นหมายความ volume prediction ในช่วงเช้าจะมความแม่นยำน้อยกว่าในช่วงบ่าย