RO1 Personalized Algo-trading Using Deep Learning & Machine Learning models

BY MUN Sun Bin
KWOK Ue Nam
KONG Kin Cheung
HO Yat Man Peter

Table of Content

01 **Introduction**
Deep Learning and Algo trading

02 **Objectives**
Goals that we aim to achieve

03 **Implementation**
Implementation of the strategy

04 **Testing**
Testing our strategy and platform

05 **Evaluation**
How well our platform performed

06 **Web Application**
Portfolio Presentation

07 **Discussion**
Challenges faced and future work

08 **Conclusion**
Achievement summary

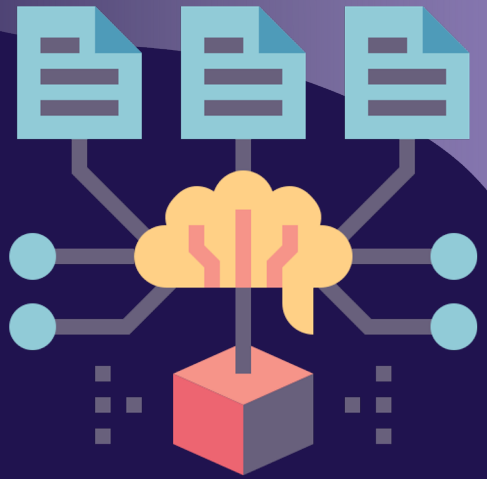
01

Introduction

Algorithmic Trading using Deep Learning

1. Introduction

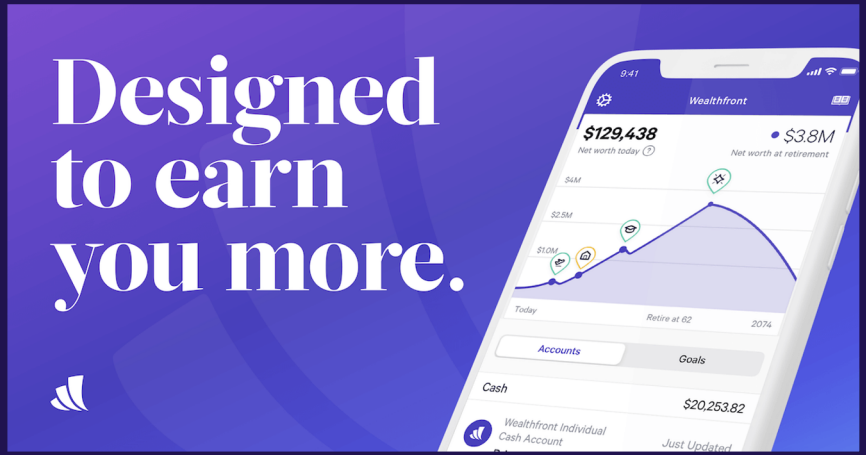
Niche our team captured

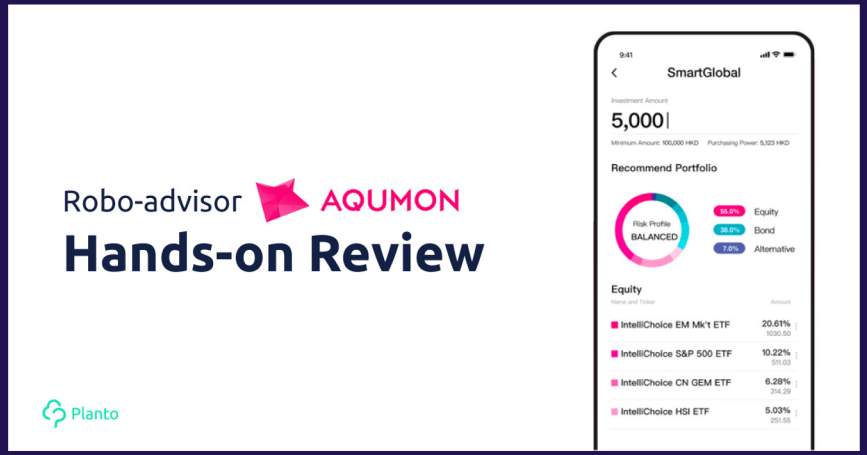


Current niche : Overflowing trading platforms & confused users

1. Introduction

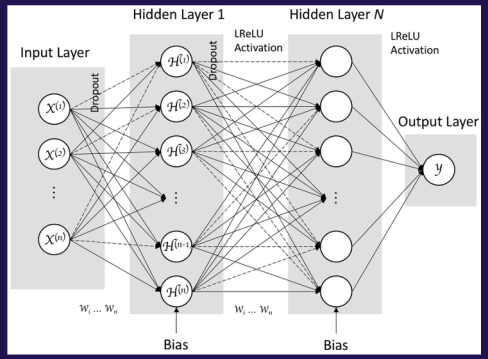
Current mobile platforms for algo trading

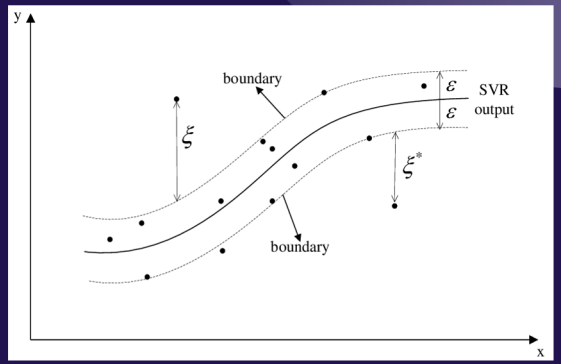


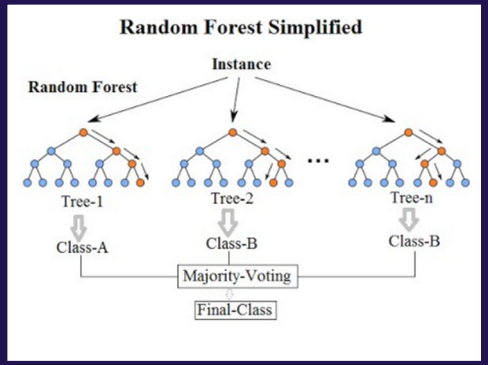


1. Introduction

Machine Learning and Deep Learning models

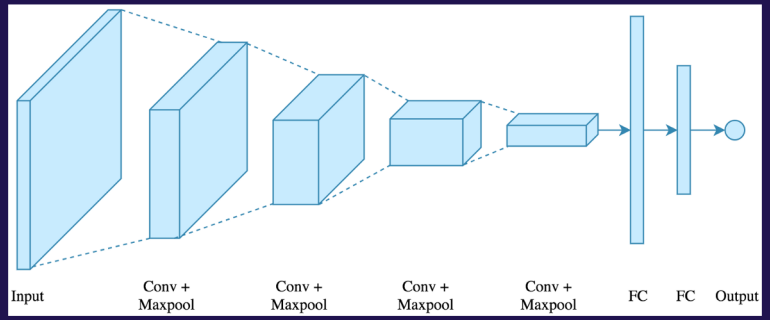


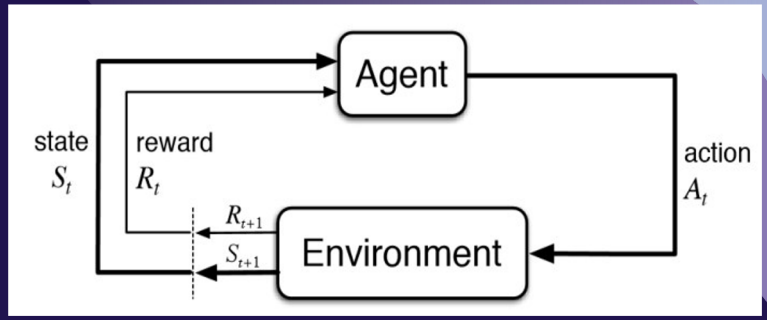


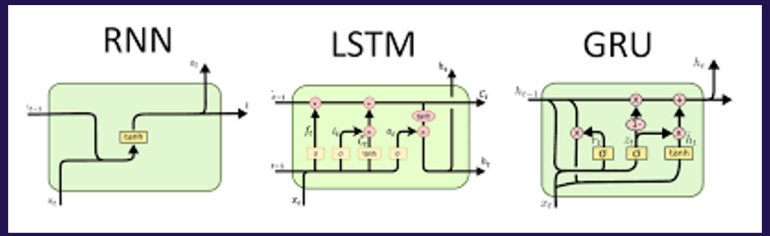


1. Introduction

Machine Learning and Deep Learning models



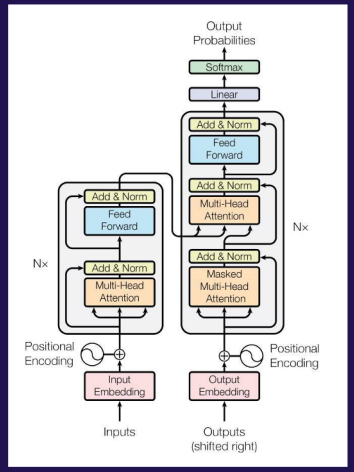


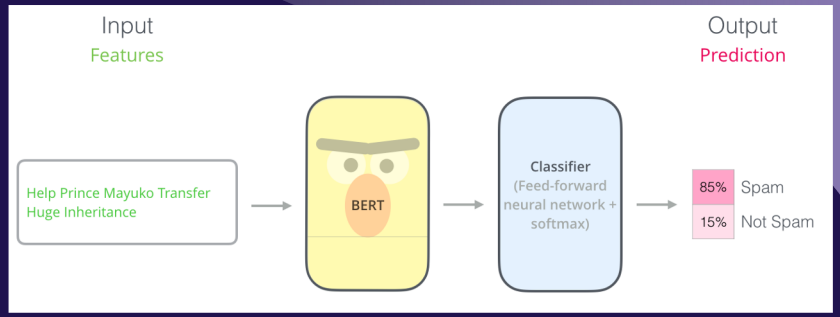


Machine Learning (ML) is a study of constructing a model operating under complex algorithms which will improve over the iterative learning process without programmed instructions.

1. Introduction

Machine Learning and Deep Learning models - further adaptations





Further implementations including:

- Mix of the existing models
- Using advanced structures for non-numerical data (BERT for sentiment analysis)

02

Objectives

What we want to achieve

Ultimate Goals

Truly automated Platform

Not requiring users to make any judgements or choices during investing

Higher Prediction Accuracy

Using multiple machine learning models for prediction

Objectives

Database Implementation

Machine Learning Models Training

Risk Quantification

Portfolio Presentation

Database Implementation

- **Historical Stock Market Data**
  - Training and testing
  - Backup purpose
- **Real-time Stock Market Data**
  - Keep our database updated
- **User Survey Responses**
  - Risk Level
- **User Portfolio**
  - Capital
  - Performances

Machine Learning Models Training

1. **Train multiple machine learning models**
2. **Combine the result to make prediction on stock price or trend**
    a. Average weighting
    b. Higher weighting on models with better performance
3. **Construct a strategy based on the model prediction results**

Risk Quantification

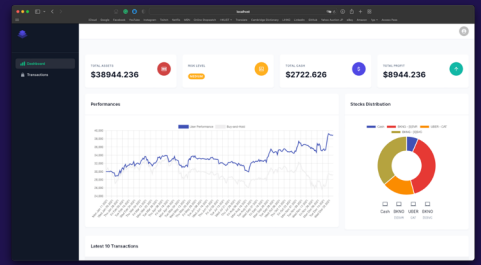
- A survey to understand users' financial situations and investment preferences
- Assign a risk-bearing level to a user based on the response from the survey
- Choose the suitable sectors and stocks for the user to invest in
- Choose the suitable models used for price and trend prediction

Portfolio Presentation

Develop a web dashboard application to display the users' portfolio

What should be displayed:

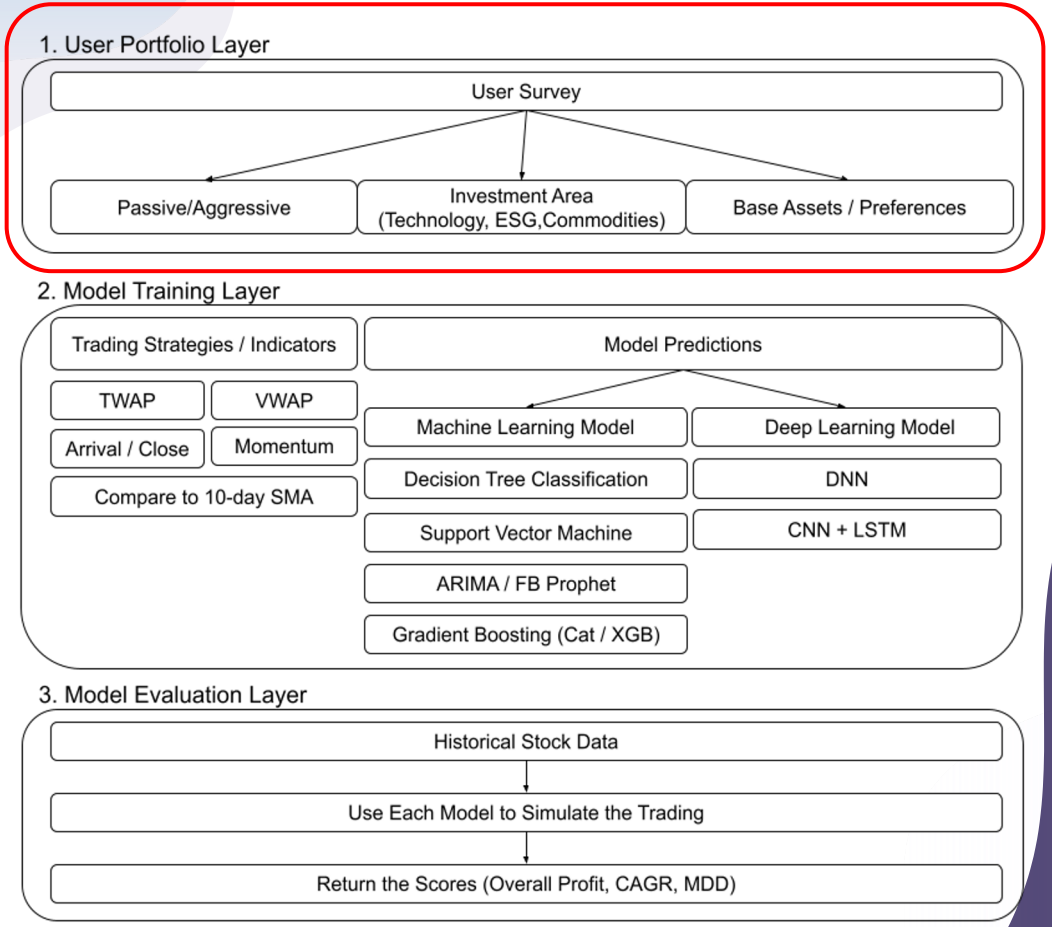
- **User Information**
- **Portfolio Distribution**
- **Portfolio Performance**
- **Baseline Performance for comparison (Buy & Hold Strategy)**



03

Methodology

How we implemented the strategy



3-1. Risk Quantification

3. Methodology

A. Risk Quantification

User Survey
Make users input their data on risk level

Quantification
Convert the user input to numeric data

1,000

Dummy users to test the module

3. Methodology

A. Risk Quantification

Demographic
Age, source of deposit, seed money

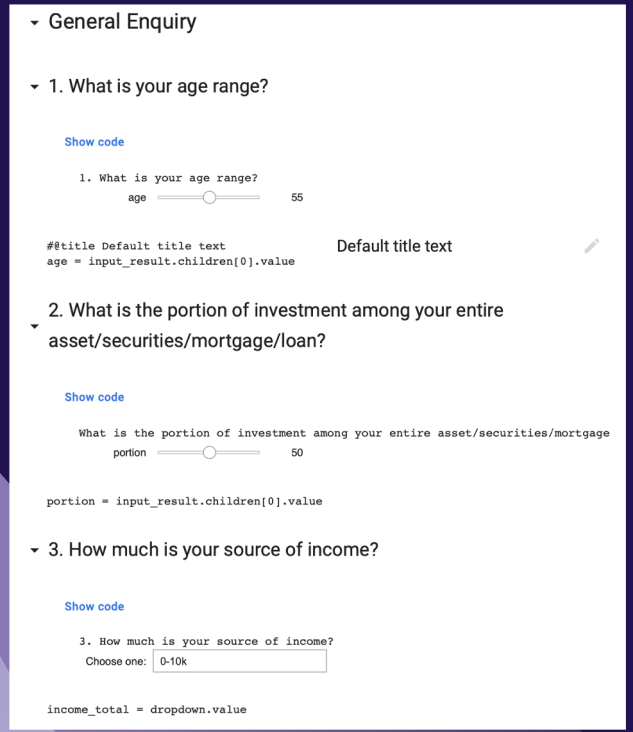
Risk
How much risk the user is willing to take

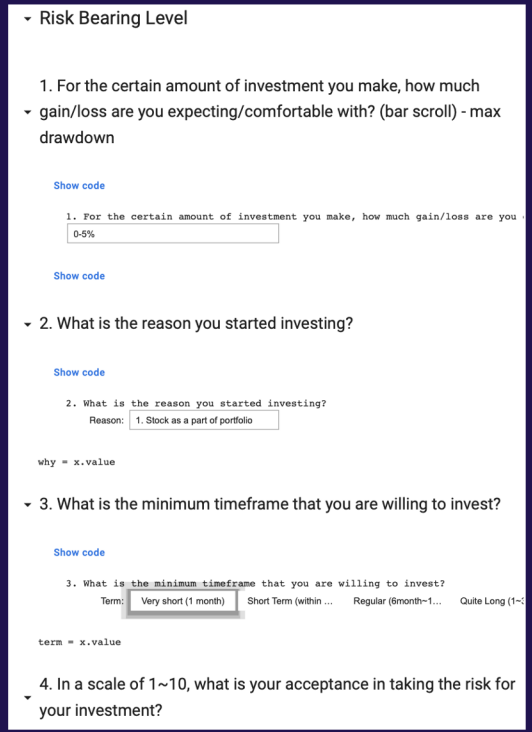
Time
Time period of the investment

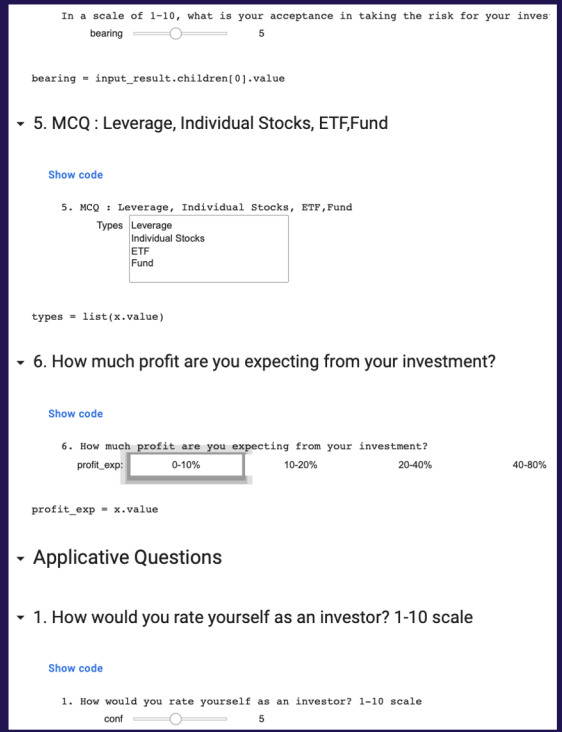
Category
Category / Sector users are interested in

3. Methodology

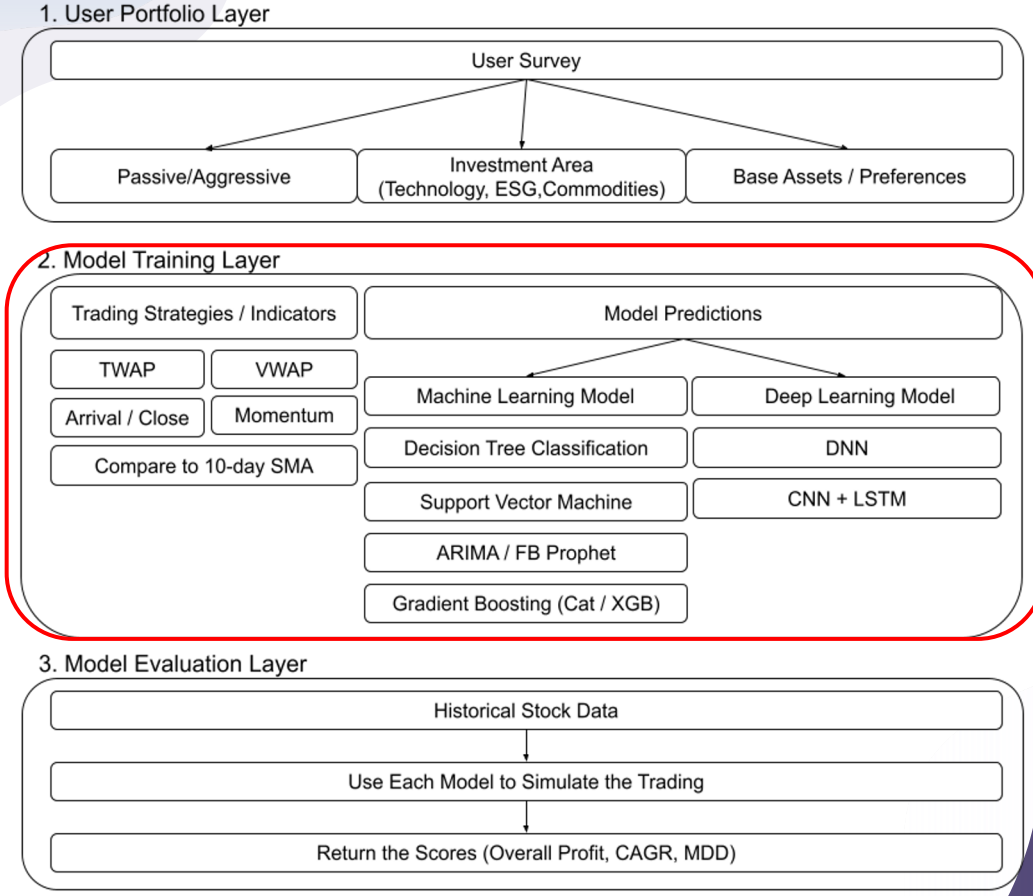
A. Risk Quantification





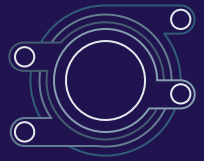


3-2. Model Selection



3. Methodology

B. Model Selection - selected



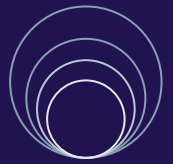
SVC
RBF kernel to perform classification

SVR
Supervised learning to predict discrete values

DNN + TWAP
Time Weighted Average Price

CNN+LSTM
sequence prediction problems with spatial inputs

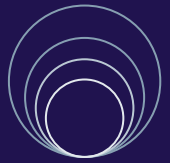
XGBoost
Extended Gradient Boosting Framework



CatBoost
Concat weak boosting models

3. Methodology

B. Model Selection - selected



CatBoost

Used XGBRegressor library
learning_rate=0.01
n_estimators=1000

Added TWAP as input for Simple DNN for decision making

```
#Performance evaluation
print('CATBoost PERFORMANCE')
print('r2 score: '+str(r2_score(new_test_y, pred_cat)))
print('RMSE : '+str(np.sqrt(mean_squared_error(new_test_y, pred_cat))))
print("Mean Absolute Error : " + str(mean_absolute_error(new_test_y,pred_cat)))
```

```
CATBoost PERFORMANCE
r2 score: 0.5254815306534676
RMSE : 1.5517341798515667
Mean Absolute Error : 1.0839976510013503
```

```
#Performance evaluation
print('XGBoost PERFORMANCE')
print('r2 score: '+str(r2_score(new_test_y, xgb_pred_y)))
print('RMSE : '+str(np.sqrt(mean_squared_error(new_test_y, xgb_pred_y))))
print("Mean Absolute Error : " + str(mean_absolute_error(new_test_y,xgb_pred_y)))
```

```
XGBoost PERFORMANCE
r2 score: 0.39090347765863553
RMSE : 1.7580605387765824
Mean Absolute Error : 1.1670200978605854
```

```
#Performance evaluation
print('TWAP PERFORMANCE')
print('r2 score: '+str(r2_score(new_test_y, pred_y)))
print('RMSE : '+str(np.sqrt(mean_squared_error(new_test_y, pred_y))))
print("Mean Absolute Error : " + str(mean_absolute_error(new_test_y,pred_y)))
```

```
TWAP PERFORMANCE
r2 score: 0.5584552566318358
RMSE : 1.4968494612872458
Mean Absolute Error : 0.94321567199208
```

3. Methodology

B. Model Selection - selected

SVC Prediction (decision)
learning_rate=0.01
n_estimators=1000

```
svc_prediction = svc.predict(new_test_X_5)
svc_confidence = svc.score(new_test_X_5, new_action_test_y)
print("svc confidence:", svc_confidence)
```

```
svc confidence: 0.875751503006012
```

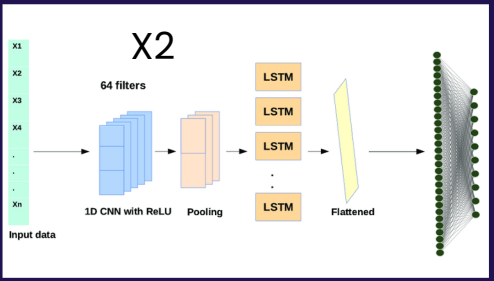
SVR Prediction (Regressor)
Selected After grid-search

```
svr = SVR(kernel=kernel, C=c, gamma=gamma)
svr.fit(new_train_X_5, new_train_y)

svr_prediction = svr.predict(new_test_X_5)
svr_confidence = svr.score(new_test_X_5, new_test_y)
print("svr confidence:", svr_confidence)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/
  y = column_or_1d(y, warn=True)
svr confidence: 0.4010474438506175
```

Used both regression and classification tasks



3. Methodology

B. Model Selection - dropped

ARIMA/GARCH
Time Series model for prediction

FB Prophet
Advanced time series model that also removes seasonality

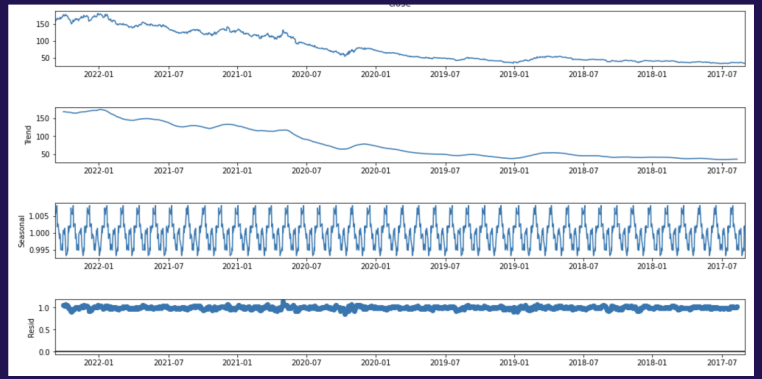
3. Methodology

B. Model Selection - dropped

ARIMA Criteria

Exists evident seasonality

ACF,PACF graph : appropriateness
AIC : Lowest
BIC : Lowest



FB Prophet

Poor r2 score, rejected from the output

```
#Performance evaluation
print('FB Prophet PERFORMANCE')
print('r2 score: '+str(r2_score(new_test_y, fb_pred_y)))
print('RMSE : '+str(np.sqrt(mean_squared_error(new_test_y, fb_pred_y))))
print("Mean Absolute Error : " + str(mean_absolute_error(new_test_y,fb_pred_y)))
```

```
FB Prophet PERFORMANCE
r2 score: 0.19141116494509347
```

3. Methodology

C. Model Volatility assignment

Step 1

Create the (stock,model) pair for all stock+models

Step 2

Calculate the S.D of the CAGR performance for each pair

Step 3

Divide all models in three levels manually

Step 4

Assign users with three highest volatility pair within sector

3. Methodology

C. Model Volatility assignment

	Stock					
Model	FB	AMZN	NFLX	IBM	LMT	PVH
CNN LSTM (Regression)	Medium	Medium	High	Low	Low	Medium
CNN LSTM (Classification)	Medium	High	High	Medium	Medium	High
TWAP	High	High	High	Medium	Medium	High
SVR	Medium	Medium	High	Medium	Medium	High
SVC	High	Medium	High	Low	Medium	High
XGBoost	Medium	Low	Low	Low	Low	Low
CatBoost	Medium	Low	Low	Low	Low	Low

04

Testing

How we tested the our strategy and platform

4. Testing

A. Backtesting on Prediction Model

- Developed a python class for backtesting procedure

- Input:
  1) Historical Price Data
  2) Prediction Action

- Output:
  1) Overall Profit with prediction
  2) Profit with Buy-and-Hold Strategy
  3) CAGR

```
class backtest:
    hpd = ""
    pred_action=pd.DataFrame()
    trade_record=pd.DataFrame(index=[],
                              columns=["Action","Price","Position","Cash","Pos_Bal","Cash_Bal","Cum_Profit","Total_Bal"],
                             )
    capital = 0
    cash_balance = 0
    profit = 0
    handle_fee = 0
    position = 0
    last_price = 0
    do_nth_profit = 0
    num_year = 0
    _tested = False

    _stock_trough = 0
    _stock_peak = 0
    _stock_all_time_low = 0
    _stock_all_time_high = 0

    _portfolio_trough = 0
    _portfolio_peak = 0
    _portfolio_all_time_low = 0
    _portfolio_all_time_high = 0

    def __init__(self,hist_price_data,pred_action,capital,handling_fee,num_year=1):
        self.hpd = hist_price_data
        self.pred_action = pred_action
        self.capital = capital
        self.cash_balance = capital
        self._portfolio_trough = capital
        self._portfolio_peak = capital
        self._portfolio_all_time_low = capital
        self._portfolio_all_time_high = capital
        self.handle_fee = handling_fee
        self.num_year = num_year

    def clear_trade_record(self,sec):
        self.trade_record=pd.DataFrame(index=[],
                                       columns=["Action","Price","Position","Cash","Pos_Bal","Cash_Bal","Cum_Profit","Total_Bal"],
                                       )
```

4. Testing

A. Backtesting on Prediction Model

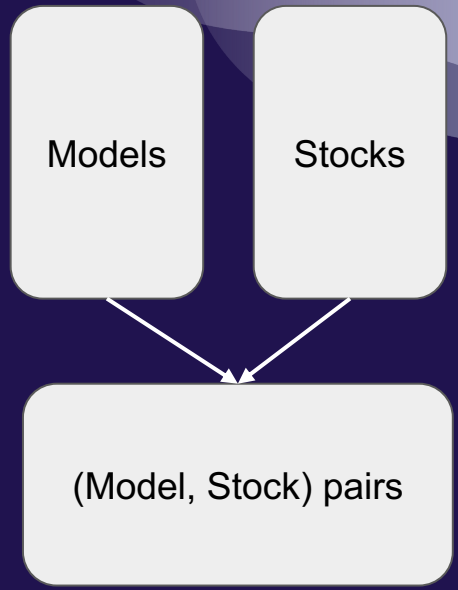
- Backtesting Setting
- Each period: 1 year long
- Number of period: 7 (2012-2018)
- Capital: $10,000
- Stock List (See Figure)

BlockChain	COIN, NVDA, FB
Airline	BA, GD, LMT
Traveling	UBER, ABNB, MAR, BKNG
Semiconductors	INTC, NVDA, QCOM, MU, AMD
Cloud Computing	IBM. AMZN, GOOG, CRM
Social Media	TWTR. SNAP. PINS. FB
Entertainment	DIS, NFLX, FB
Retail	WMT, COST, TGT, BBY, HD
Franchise	MCD, YUM, SBUX, DPZ
Real Estate	HST, EQR, AVB, PLD, SPG
Telecommunication	T, TMUS, VZ, CMCSA, CHTR
Energy & Resources	DOW, DD
Luxury goods	RACE, EL, PVH

4. Testing

B. Backtesting on User Portfolio

- For example, 8 models X 20 stocks
  ⇒ 160 pairs
- Each pairs will have 7 CAGR result
  (from year 2012-2018)
- Calculate the SD of the 7 CAGR
  ⇒ Volatility
- Sort all the pairs by volatility descendingly



Stock	Model	Volatility
AMD	TWAP	153.0592847
AMD	[5]SVR	115.5840271
AMD	CAT	107.8030016
AMD	CNN_LSTM_CLASS	106.6059767
NFLX	CAT	102.7248703
AMD	CNN_LSTM_REG	96.31024623
NFLX	CNN_LSTM_CLASS	88.87736372

4. Testing

B. Backtesting on User Portfolio

- Classify into three risk levels
  ⇒ Low : Medium : High

- Approximately in a ratio of 1:2:1

	Sector	Stock	Model	Volatility	risk_level
300	Semiconductors	AMD	TWAP	153.059285	high
75	Semiconductors	AMD	[5]SVR	115.584027	high
210	Semiconductors	AMD	CAT	107.803002	high
120	Semiconductors	AMD	CNN_LSTM_CLASS	106.605977	high
192	Entertainment	NFLX	CAT	102.724870	high
...	...	...	...	...	...
136	Airline	GD	CNN_LSTM_REG	5.337012	low
144	Energy & Resources	DD	CNN_LSTM_REG	4.644089	low
175	Telecommunication	T	CNN_LSTM_REG	4.365728	low
159	Real Estate	SPG	CNN_LSTM_REG	3.556537	low
171	Social Media	SNAP	CNN_LSTM_REG	0.947523	low

4. Testing

B. Backtesting on User Portfolio

- Each user has a user preference
- 1. Risk Level
- 2. Area of Interest
- Example: User 13

	stock_type	risk_level
9	['Telecommunication', 'Franchise', 'Airline', 'Energy & Resources', 'Cloud Com	low
10	['Retail', 'Luxury goods']	medium
11	['Retail', 'Entertainment', 'Traveling', 'Franchise', 'Telecommunication', 'Real Es	low
12	['Social Media', 'Energy & Resources', 'Semiconductors', 'Traveling', 'Franchis	medium
13	['Airline', 'Luxury goods', 'Franchise', 'Telecommunication']	low
14	['BlockChain', 'Semiconductors', 'Social Media', 'Entertainment', 'Airline', 'Ret	medium

	Sector	Stock	Model	Volatility	risk_level
0	Telecommunication	T	TWAP	12.952671	low
1	Luxury Goods	EL	CNN_LSTM_REG	12.746988	low
2	Telecommunication	T	CAT	12.556685	low
3	Telecommunication	VZ	[5]SVR	12.384909	low
4	Telecommunication	CHTR	CNN_LSTM_REG	12.348033	low
5	Luxury Goods	EL	CNN_LSTM_CLASS	12.282841	low
6	Telecommunication	CMCSA	XGB	12.140834	low
7	Airline	GD	CNN_LSTM_CLASS	11.880257	low
8	Telecommunication	VZ	CAT	11.706905	low
9	Franchise	YUM	CAT	11.386209	low
10	Airline	BA	CNN_LSTM_REG	11.335239	low
11	Airline	LMT	TWAP	11.004978	low
12	Franchise	YUM	CNN_LSTM_CLASS	10.706351	low
13	Telecommunication	VZ	[5]SVC	10.395978	low
14	Telecommunication	CMCSA	CNN_LSTM_CLASS	10.391511	low
15	Telecommunication	T	CNN_LSTM_CLASS	9.744189	low
16	Franchise	YUM	CNN_LSTM_REG	9.093985	low
17	Telecommunication	CMCSA	CNN_LSTM_REG	8.872765	low
18	Telecommunication	VZ	TWAP	8.842665	low
19	Franchise	MCD	CNN_LSTM_CLASS	8.680293	low

4. Testing

B. Backtesting on User Portfolio

- We assume each user will invest into 3 stocks, $10000 each
- Select the top 3 (model,stock) pairs for each user
- On the right, we use **2012-2018** as to measure the (model, stock) pairs volatility
- Then we perform backtesting on the selected (model, stock) pairs with **2019-2021** data

	Sector	Stock	Model	Volatility	risk_level
0	Telecommunication	T	TWAP	12.952671	low
1	Luxury Goods	EL	CNN_LSTM_REG	12.746988	low
2	Telecommunication	T	CAT	12.556685	low
3	Telecommunication	VZ	[5]SVR	12.384909	low
4	Telecommunication	CHTR	CNN_LSTM_REG	12.348033	low
5	Luxury Goods	EL	CNN_LSTM_CLASS	12.282841	low
6	Telecommunication	CMCSA	XGB	12.140834	low
7	Airline	GD	CNN_LSTM_CLASS	11.880257	low
8	Telecommunication	VZ	CAT	11.706905	low
9	Franchise	YUM	CAT	11.386209	low
10	Airline	BA	CNN_LSTM_REG	11.335239	low
11	Airline	LMT	TWAP	11.004978	low
12	Franchise	YUM	CNN_LSTM_CLASS	10.706351	low
13	Telecommunication	VZ	[5]SVC	10.395978	low
14	Telecommunication	CMCSA	CNN_LSTM_CLASS	10.391511	low
15	Telecommunication	T	CNN_LSTM_CLASS	9.744189	low
16	Franchise	YUM	CNN_LSTM_REG	9.093985	low
17	Telecommunication	CMCSA	CNN_LSTM_REG	8.872765	low
18	Telecommunication	VZ	TWAP	8.842665	low
19	Franchise	MCD	CNN_LSTM_CLASS	8.680293	low

4. Testing

B. Backtesting on User Portfolio

- Example: User 13 [(TWAP, T) + (CNN_LSTM_REG, EL) + (CAT, T)]

	Model_1	Stock_1	Value_1	Model_2	Stock_2	Value_2	Model_3	Stock_3	Value_3	Total_Value
20191209	TWAP	T	10919.766	CNN_LSTM_REG	EL	17376.683	CAT	T	14004.82	42301.269
20191210	TWAP	T	10905.026	CNN_LSTM_REG	EL	17323.003	CAT	T	13978.921	42206.950000000000
20191211	TWAP	T	10956.281	CNN_LSTM_REG	EL	17358.203	CAT	T	13971.942	42286.42600000000
20191212	TWAP	T	10958.961	CNN_LSTM_REG	EL	17361.723	CAT	T	13975.366	42296.05
20191213	TWAP	T	11005.191	CNN_LSTM_REG	EL	17564.218	CAT	T	14034.43	42603.839
20191216	TWAP	T	11071.186	CNN_LSTM_REG	EL	17564.218	CAT	T	14118.746	42754.15
20191217	TWAP	T	11071.186	CNN_LSTM_REG	EL	17555.449	CAT	T	14118.746	42745.381
20191218	TWAP	T	11125.456	CNN_LSTM_REG	EL	17605.61	CAT	T	14188.082	42919.148
20191219	TWAP	T	11245.721	CNN_LSTM_REG	EL	17596.845	CAT	T	14341.734	43184.3
20191220	TWAP	T	11256.776	CNN_LSTM_REG	EL	17586.405	CAT	T	14355.858	43199.039
20191223	TWAP	T	11214.231	CNN_LSTM_REG	EL	17832.615	CAT	T	14301.502	43348.348
20191224	TWAP	T	11183.076	CNN_LSTM_REG	EL	17696.025	CAT	T	14254.575	43133.67600000000
20191226	TWAP	T	11254.096	CNN_LSTM_REG	EL	17706.465	CAT	T	14247.456	43208.017
20191227	TWAP	T	11262.806	CNN_LSTM_REG	EL	17845.665	CAT	T	14258.506	43366.977
20191230	TWAP	T	11182.857	CNN_LSTM_REG	EL	17872.635	CAT	T	14157.079	43212.571000000000

4. Testing

B. Backtesting on User Portfolio

- Example: User 13 [(TWAP, T) + (CNN_LSTM_REG, EL) + (CAT, T)]

	Total_Value	Total_Cum_Profit	Buy&Hold_Total	Capital_Bal
20191211	42286.426000000000	12286.426	41141.227926	30000
20191212	42296.05	12296.05	41150.683926000000	30000
20191213	42603.839	12603.839	41447.179926000000	30000
20191216	42754.15	12754.150000000000	41893.363926000000	30000
20191217	42745.381	12745.381000000000	41718.643926000000	30000
20191218	42919.148	12919.148	41899.207926	30000
20191219	43184.3	13184.300000000000	42301.31592600000	30000
20191220	43199.039	13199.039	42318.09192600000	30000
20191223	43348.348	13348.348	42438.247926000000	30000
20191224	43133.67600000000	13133.676	42242.131926	30000
20191226	43208.017	13208.017	42419.395926000000	30000
20191227	43366.977	13366.977000000000	42564.787926000000	30000
20191230	43212.571000000000	13212.571	42391.97380200000	30000

05

Evaluation

How well our platform performed

5. Evaluation

A. Platform Performance Result (By year)

- Overall Performance (By year)
- Take average over 1000 users

Year	Platform Performance	Buy and Hold CAGR	With Platform CAGR
2019	9.24%	30.50%	39.74%
2020	43.81	17.15%	60.95%
2021	25.53%	18.16%	42.68%

Average of Platform Performance,
Platform CAGR and “Buy and Hold” CAGR in different years

5. Evaluation

A. Platform Performance Result (By year)

- Consider the maximum and minimum result
- When Buy-and-Hold is doing very bad, our platform will lose less
- When Buy-and-Hold is doing very good, our platform will do it much better

Year	Buy and Hold CAGR	With Platform CAGR
2019	-20.04%	-20.84%
2020	-31.80%	-16.05%
2021	-12.59%	0.2%

Minimum of Platform CAGR, “Buy and Hold” CAGR, and Platform Performance

Year	Buy and Hold CAGR	With Platform CAGR
2019	119.50%	124.75%
2020	98.76%	137.33%
2021	90.55%	116.08%

Maximum of Platform CAGR, “Buy and Hold” CAGR, and Platform Performance

5. Evaluation

A. Platform Performance Result (By year)

- **For how many users we can help to beat the “Buy-and-Hold” strategy**

Year	Outperformance
2019	80%
2020	91%
2021	89%

Percentage of users obtaining positive platform performance

5. Evaluation

A. Platform Performance Result (By year)

- **Our platform when the market is being hit severely do much better**

Year	# of Users that have Positive CAGR in “Buy and Hold” Strategy	# of Users that have Positive CAGR in our Platform	Difference
2019	981	991	+10
2020	553	981	+428
2021	956	993	+37

The number of users obtaining positive CAGR in two strategies

5. Evaluation

A. Platform Performance Result (By year)

- **Our platform can help most of the users that are getting negative CAGR in Buy-and-Hold to obtain positive return.**

Year	# of Users that have negative CAGR in “Buy and Hold” Strategy	# of Users that have (Negative CAGR in “Buy and Hold” Strategy **AND** Positive CAGR in our Platform)	%
2019	12	11	91.7%
2020	440	430	97.7%
2021	37	37	100%

Percentage of users with positive CAGR with our platform **given that** they have negative CAGR in the “Buy and Hold” strategy

5. Evaluation

B. Platform Performance Result (By risk)

- Good in the “Low” level
- Much better in the other two levels

Risk Level	# of Users	# of Users with Positive Platform Performance	%
High	148	125	84.5%
Medium	706	676	95.8%
Low	139	87	62.5%

Percentage of User Portfolio with Positive Platform Performance

06

Web Application

Portfolio Presentation

Web Application Architecture

Front-end

Showing essential information and interacting with users

Backend

Handling the requests from front-end and retrieving data from the database

Database

Storing the user information and portfolio performance

LIVE DEMO

Web Application Testing and Evaluation

Web Application Testing

Asked 8 voluntary testers to experience our web application and gathered their responses by Google forms

Result

Evaluation Metrics	Average Rating
User-friendliness	3.9 / 5.0
Informative	4.1 / 5.0
UI design	4.6 / 5.0
Interactiveness	3.8 / 5.0

07

Discussions

Challenges faced and future work

7. Discussion

A. Challenges

Risk metric

Solely depending on the average value of CAGR volatility

Model Selection

Test if Deeper models can enhance performance

Market Data Collection

- Failed to find a free-of-charge API for real time market data
- Update manually

Risk Quantification

Various risk analysis required

7. Discussion

B. Further work

Try More Ensemble Model

To capture more information from the market data

Sector-wise Model Training

Some models may behave better on some sectors

Backtest with other periods

Try on the period with different length and data range

08

Conclusion

Achievement

8. Conclusion

A. Achievements

A. Final model list:

i. SVM(Support Vector Machine) - classifier and regressor

ii. Gradient Boost models - XGBoost / CatBoost

iii. Simple DNN (Deep Neural Network) in association with TWAP

iv. Combined Deep Learning algorithms like CNN-LSTM (Convolution Neural Network with Long Short Term Memory) - classifier and regressor

B. Experimented model:

i. Random Forest / Decision Tree

ii. Time series models (ARIMA, ARCH, GARCH model, Facebook Prophet library)

iii. Simple MLP model

8. Conclusion

A. Achievements

C. Model training, testing, and evaluation using the customized backtesting class module to simulate the trading process and calculate the CAGR and MDD

D. Matched three (stock, model) pairs for 1,000 dummy users and calculated their portfolios' profit and loss

E. Built a user-friendly web dashboard application to interact with the user and visualize the portfolio by charts and graphs

8. Conclusion

A. Achievements

4.1/5

UI/UX score

25.85%

Average additional gain by our platform between 2019~2021

90%

Average % of users obtaining positive platform performance

Thanks

Do you have any questions?