School	Lee Kong Chian School of Business
Programme	Master of Quantitative Finance Class 2021
Course Code	QF624 Machine Learning & Financial Applications
Instructor	Dai Bing Tian
Team Members	Ankit Rawat Chua Wen Bin ChenPeng Li Yan Yan Zhang Chuanyu Zhang Luyu
Topic	Deep Reinforcement Learning vs. ARIMA for Stock Price Prediction – **Group 3**

1. Introduction

1.1 Machine Learning Application in Stock Price Prediction

Equities market is possibly the most well covered capital market with an immense depth of data points for participants to digest and analyse when making investment decisions.

1.2 Objective

- To incorporate a set of technical indicators into a trading strategy using machine learning to determine an optimal allocation of weights within the set of technical indicators so as to maximize the P/L in each duration

1.3 Machine Learning Application

- Many of the existing ML/DL approaches try to predict price movements or trends based on supervised learning.
- In this project, we will train an agent so that it can take decisions based on the given market conditions without any supervised information from human input.

1.4 Potential Advantage in Machine Learning Application

- The use of machine learning reduces human bias and intervention and the algorithm is able to progressively take inputs from indicators and refine the model the produce an output that mini mizes the difference with realized price movements, thereby maximizing PnL.
- In contrast, without application of ML, we applied a quantitative trading model based on ARIMA models to predict the direction on the market and we used this as our benchmark to compare the PnL.

2. Dataset Definition

2.1.1 Overview

- The dataset contains daily price data of MSFT price and technical indicators.
  Technical Indicators:
  Fama Franch 5 factor, Momentum, MACD, Bollinger Bands and moving average, Dividends, Stock Split

Date	Mkt-RF	SMB	HML	RMW	CMA	RF	Return	earnings	sma_21	ma7	26ema	12ema	MACD	20sd	upper_ban	lower_ban	ema	momentun
14/5/86	0.38	-0.27	-0.4	0.41	0.09	0.023	-0.00775	0	0.070174	0.070174	0.069567	0.070123	0.000557	0.003618	0.075955	0.061481	0.070516	-0.0044
16/5/86	-0.56	0.51	0.11	0.23	-0.06	0.023	0.007812	0	0.07041	0.07041	0.069695	0.070256	0.00056	0.003411	0.075925	0.062282	0.070812	-0.00165
19/5/86	0.06	-0.41	0.08	0.05	0.08	0.023	-0.0155	0	0.07041	0.07041	0.06971	0.070193	0.000483	0.003177	0.075759	0.063053	0.070177	-0.0011
20/5/86	1.01	-0.9	-0.49	0	0.07	0.023	-0.00787	0	0.070252	0.070252	0.069675	0.070054	0.00038	0.003075	0.075721	0.063421	0.069599	-0.00055
21/5/86	-0.16	0.32	0.09	0.13	0.1	0.023	-0.01587	0	0.070017	0.070017	0.069546	0.069765	0.00022	0.003011	0.075676	0.06363	0.068673	-0.0011

- **Dataset** has 8820 rows, from 1986-03-14 to 2021-06-30, and 10 columns including date, stock price & volume, FF-5 factors and daily return.

2.1.2 Training Set Definition

- Test set on **10%** of full data sample
- Training from 1986-04-17 to 2017-09-19 ~ 7723 days (rows)

- Testing from 2017-9-20 to 2021-02-26 ~ 858 days

2.1.3 Validation

Validation applied on **5%** of the Test set

3. Building our benchmark ARIMA model

3.1 Using ARIMA model as the benchmark

Using a statistical model like ARIMA as a benchmark to measure the effectiveness of our ML model would help us give a fair assessment of our ML model performance.

3.2 Checking for Stationarity

We checked for stationarity in the stock returns to ascertain the suitability of using ARIMA model for return prediction.

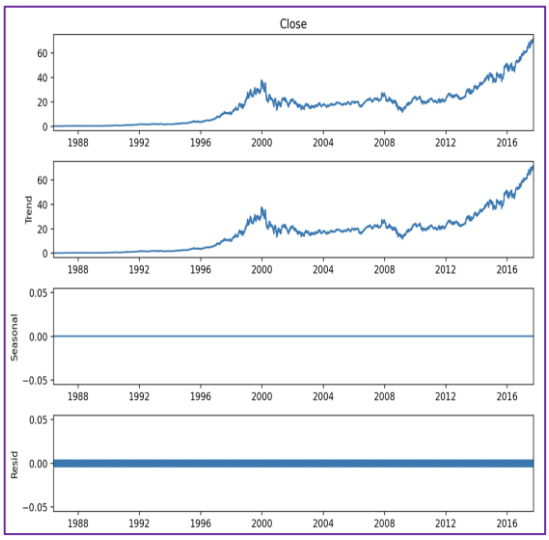
We observed MSFT stock has a significant Trend component but seasonal impact hence to make stationary we mainly need to take care of Trend component.

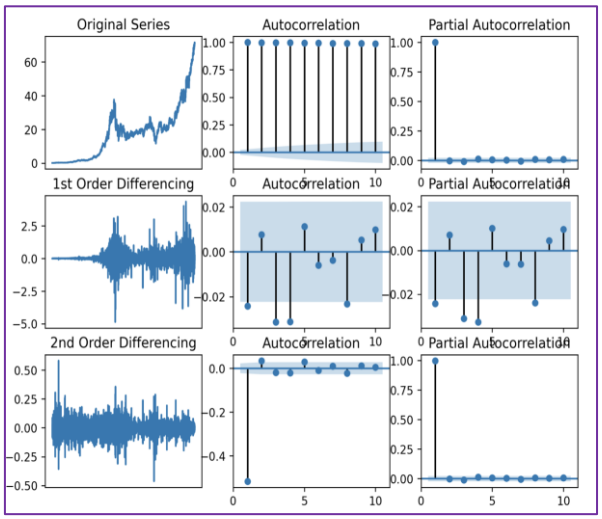
Stats below shows, numerical representation proof of stationary time series after first differencing.

```
: 1 print(test_stationarity(trainMinmax,"Close"))
Results of Dickey-Fuller Test:

Time Series is not Stationary as P value ( 0.99889 ) > 0.05
Test Statistic                    2.203087
p-value                           0.998885
# Lags Used                      36.000000
Number of Observations Used    7666.000000
Critical Value (1%)              -3.431203
Critical Value (5%)              -2.861917
Critical Value (10%)             -2.566971
dtype: float64

: 1 print(test_stationarity(trainMinmax,"Return"))
Results of Dickey-Fuller Test:

Time Series Stationary as P value ( 0.0 ) < 0.05
Test Statistic                  -46.843242
p-value                           0.000000
# Lags Used                       3.000000
Number of Observations Used    7699.000000
Critical Value (1%)              -3.431200
Critical Value (5%)              -2.861915
Critical Value (10%)             -2.566970
dtype: float64
```

3.3 Autocorrelation and Partial Auto Correlation

We checked for stationarity in the stock returns to ascertain the suitability of using ARIMA model for return prediction. In ACF and PACF graph, we observed that considering first difference convert Time-Series to stationary. Also, we observed that 2& 3 lag has significant impact post first differencing.





3.4 Using a combination of Fama-French 5 Factors and Technical Price Indicators to predict stock returns

We checked for stationarity in the stock returns to ascertain the suitability of using ARIMA model for return prediction. We started with below set of indicators and later filtered to fewer based on correlation factor.

```
'Close', 'Volume', 'Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF', 'Return',
       'earnings', 'sma_21', 'ma7', '26ema', '12ema', 'MACD', '20sd',
       'upper_band', 'lower_band', 'ema', 'momentum'
```

Below is the list of indicators post filtering based on co-relation factor. We further introduced lags on momentum and SMA and MACD to study the impact of these on predictions

- `['Mkt- RF', 'RF', 'SMB',' HML',' 26ema', 'momentumLag_1', 'upper_band','momentumLag_3','sma_21','MACDLag_2']`

Date	Mkt-RF	RF	SMB	HML	26ema	momentumLag_1	upper_band	momentumLag_3	sma_21	MACDLag_2
1986-08-12	1.08	0.022	-0.35	-0.52	0.063819	-0.005501	0.069264	-0.005501	0.062080	-0.000904
1986-08-13	0.92	0.022	-0.19	0.14	0.063951	-0.002200	0.069305	-0.007151	0.062551	-0.000802
1986-08-14	0.35	0.022	0.21	-0.24	0.064204	0.000000	0.069573	-0.005501	0.063337	-0.000590
1986-08-15	0.23	0.022	0.02	0.39	0.064456	0.003851	0.069897	-0.002200	0.064241	-0.000378
1986-08-18	0.04	0.022	-0.19	0.22	0.064622	0.004676	0.070045	0.000000	0.065106	-0.000084
...	...	...	...	...	...	...	...	...	...	...
2021-02-22	-1.12	0.000	0.68	3.14	235.182565	-0.482773	248.672340	5.228302	241.580059	5.863609
2021-02-23	-0.15	0.000	-1.05	0.90	235.001086	-7.117020	248.082531	1.345261	240.060968	5.403938
2021-02-24	1.15	0.000	1.48	1.34	234.927647	-8.622925	247.788036	-0.482773	238.653214	4.468074
2021-02-25	-2.73	0.000	-0.90	0.87	234.448745	-8.639877	248.337462	-7.117020	236.636442	3.585239
2021-02-26	-0.28	0.000	0.38	-1.56	234.255849	-13.241455	248.578748	-8.622925	234.951760	2.954576

3.5 Selection of factors by minimizing Mean Square Error

We further filtered list of variables based on impact on prediction. We considered lowest MSE as filtered criteria.

From previous PACF and ACF graph, we observed lags 1 and 2 are signification hence for test purpose we considered P =1 and Q =2 during variables filtering.

Form the test, we observed 0.0002690 is lowest mean of for below set of indictors.

['Mkt-RF', 'RF', 'SMB', 'HML', '26ema', 'momentumLag_1', 'upper_band', 'momentumLag_3', 'sma_21', 'MACDLag_2']

Function to verify change in MSE on additon of independent variables

```
from statsmodels.tsa.arima_model import ARIMA
col_list = []
d  = trainMinmax
print("Mean Square error "," | List of Columns")
for i in exog:
    col_list.append(i)
    model = ARIMA(trainMinmax["Return"], exog=trainMinmax[col_list], order=(1, 0, 2))
    model_fit = model.fit()
    fc, se, conf = model_fit.forecast(d["Return"].shape[0],exog = d[col_list], alpha=0.05)
    mse = mean_squared_error(d["Return"], fc)
    print(mse, "|", col_list)
```

```
Mean Square error   | List of Columns
0.0002972222495669858 | ['Mkt-RF']
0.0002967895176922827 5 | ['Mkt-RF', 'RF']
0.0002926543881128721 | ['Mkt-RF', 'RF', 'SMB']
0.0002780985019543996 | ['Mkt-RF', 'RF', 'SMB', 'HML']
0.0002779811832339543 | ['Mkt-RF', 'RF', 'SMB', 'HML', '26ema']
0.0002780348048034586 6 | ['Mkt-RF', 'RF', 'SMB', 'HML', '26ema', 'momentumLag_1']
0.0002779306840550364 | ['Mkt-RF', 'RF', 'SMB', 'HML', '26ema', 'momentumLag_1', 'upper_band']
0.0002778764825445505 | ['Mkt-RF', 'RF', 'SMB', 'HML', '26ema', 'momentumLag_1', 'upper_band', 'momentumLag_3']
0.0002777371143970439 9 | ['Mkt-RF', 'RF', 'SMB', 'HML', '26ema', 'momentumLag_1', 'upper_band', 'momentumLag_3', 'sma_21']
0.0002690119434921948 4 | ['Mkt-RF', 'RF', 'SMB', 'HML', '26ema', 'momentumLag_1', 'upper_band', 'momentumLag_3', 'sma_21', 'MACDLag_2']
```

3.6 Optimizing ARIMA model parameters

- From the permutation, we see that (0,0,3) and (1,0,2) are the two best P and Q values for the select model with same AIC score.
- Comparing Log Likelihood of both the models (20747.553, 20748.284) are almost the same.
- Hence for this project we will proceed with (1,0,2) so that we can have both AR and MA lags as part of our modelling. We also observed that most of the factors we considered modelling has probability = 0 hence we conclude that most of the factors have significant impact on prediction.
- Also, we constant variance and mean hence we concluded current P and Q are good to model predictions.

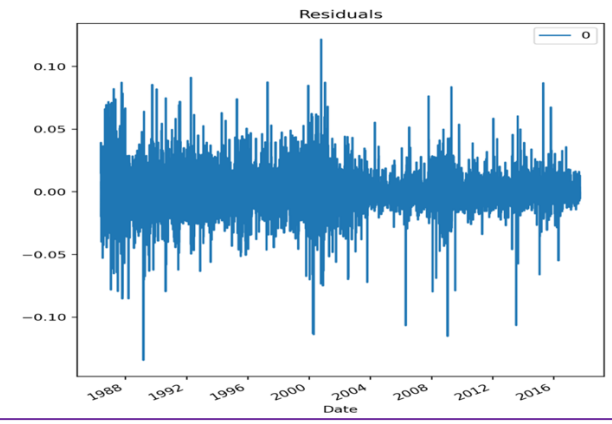
```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=-41444.740, Time=6.07 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=-41440.972, Time=7.50 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=-41446.331, Time=4.52 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=-41446.916, Time=9.33 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=-41442.367, Time=8.33 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=-41457.542, Time=12.69 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=-41464.411, Time=14.17 sec
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=-41460.501, Time=15.54 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=-41464.144, Time=10.70 sec
 ARIMA(0,0,3)(0,0,0)[0] intercept   : AIC=-41465.348, Time=17.53 sec
 ARIMA(2,0,3)(0,0,0)[0] intercept   : AIC=-41467.406, Time=9.00 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=-41459.339, Time=6.96 sec
 ARIMA(1,0,3)(0,0,0)[0]             : AIC=-41465.792, Time=8.63 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=-41466.085, Time=6.12 sec

Best model:  ARIMA(0,0,3)(0,0,0)[0]
Total fit time: 135.162 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 7703
Model:               SARIMAX(0, 0, 3)   Log Likelihood               20747.553
Date:                Tue, 06 Jul 2021   AIC                        -41467.106
Time:                        23:00:10   BIC                        -41369.815
Sample:                             0   HQIC                       -41433.742
                                 - 7703
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Mkt-RF         0.0112      0.000     92.033      0.000       0.011       0.011
RF             0.0743      0.010      7.199      0.000       0.054       0.095
SMB           -0.0030      0.000    -12.370      0.000      -0.004      -0.003
HML           -0.0062      0.000    -24.360      0.000      -0.007      -0.006
26ema         -0.0374      0.002    -19.950      0.000      -0.041      -0.034
momentumLag_1 -0.0049      0.000    -13.759      0.000      -0.006      -0.004
upper_band    -0.0004      0.000     -1.631      0.103      -0.001    7.36e-05
momentumLag_3 -0.0028      0.000     -8.493      0.000      -0.003      -0.002
sma_21         0.0377      0.002     20.860      0.000       0.034       0.041
MACDLag_2     -0.0420      0.002    -19.711      0.000      -0.046      -0.038
ma.L1          0.0293      0.008      3.454      0.001       0.013       0.046
ma.L2         -0.0418      0.009     -4.897      0.000      -0.059      -0.025
ma.L3         -0.0332      0.009     -3.688      0.000      -0.051      -0.016
sigma2         0.0003   2.41e-06    111.235      0.000       0.000       0.000
==============================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):             10658.57
Prob(Q):                              0.89   Prob(JB):                         0.00
Heteroskedasticity (H):               0.36   Skew:                             0.04
Prob(H) (two-sided):                  0.00   Kurtosis:                         8.76
==============================================================================
```

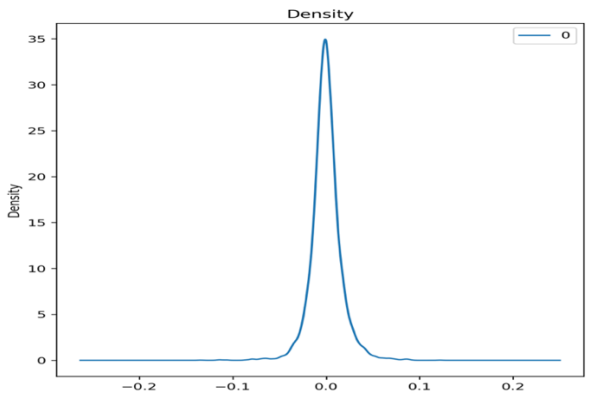
P and Q value for ARIMA modeling

-- Using beow code we will very best P and Q value for our model

```
model = pm.auto_arima(trainMinmax["Return"], exogenous=pd.DataFrame(trainMinmax[exog]),
                      start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=True,
                      start_P=0,
                      D=None,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

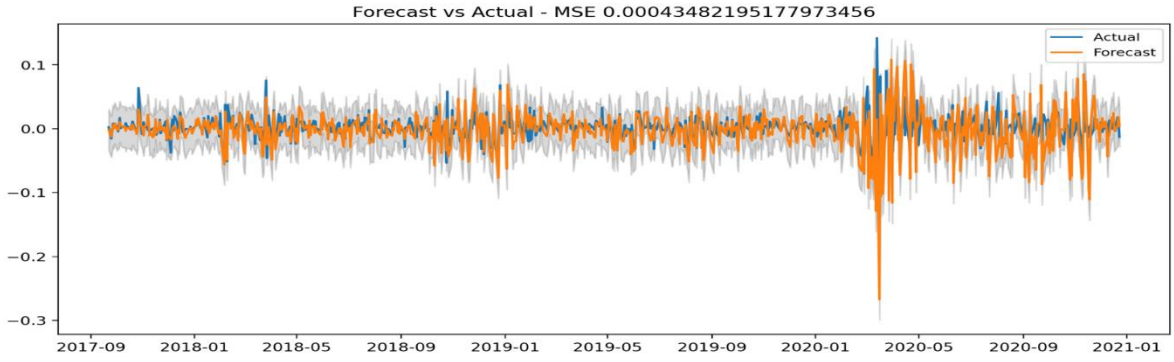
Constant mean and variance observed





3.7 ARIMA model – Forecast Vs Actual

From MSE we see the model has a low MSE hence a good fit



4. Building our Deep Reinforcement Learning (DRL) Model – A2C

4.1 Overview of DRL Application

Deep reinforcement learning is a subfield of machine learning that combines reinforcement learning (RL) and deep learning. RL considers the problem of a computational agent learning to make decisions by trial and error. Deep RL incorporates deep learning into the solution, allowing agents to make decisions from unstructured input data without manual engineering of the state space. Deep RL algorithms can take in very large inputs and decide what actions to perform to optimize an objective. In our project, we use deep reinforcement learning to make prediction and design an automated trading solution for single stock trading.

We formulate our problem as a Markov decision process in which an agent interacts with the environment at discrete time steps. At each time step t, the agent receives some representations of the environment denoted as a state S_t. Given this state, an agent chooses an action A_t, and based on this action, a numerical reward R_{t+1} is given to the agent at the next time step, and the agent finds itself in a new state S_{t+1}. The interaction between the agent and the environment produces a trajectory $\tau = [S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots]$. At any timestep t, the goal of RL is to maximize the expected return (essentially the expected discounted cumulative rewards).

4.2 Preprocess Data

- Add technical indicators: In practical trading, various information needs to be considered, for example the historical stock prices, current holding shares, technical indicators, etc. In our project, we add four trend-following technical indicators: MACD, RSI, CCI, DX and Fama-French Five Factors.
- Add turbulence index: Risk-aversion reflects whether an investor will choose to preserve the capital. It also influences one's trading strategy when facing different market volatility level. To control the risk in a worst-case scenario, such as financial crisis of 2007–2008, we employ the financial turbulence index that measures extreme asset price fluctuation.

4.3 Environment and Strategy

Our trading environments, based on OpenAI Gym framework, simulate livestock markets with real market data according to the principle of time-driven simulation.

4.3.1 State Space, Action Space and Reward Function

State space. The state space describes the observations that the agent receives from the environment. Just as a human trader needs to analyse various information before executing a trade, so our trading agent observes many different features to better learn in an interactive environment. We provide these features:

- Balance: the amount of money left in the account at the current time step t.

- Shares own: current shares for the stock.
- Closing price: one of the most used features.
- Trading volume: total quantity of traded during a trading slot.
- Technical indicators: MACD, RSI, CCI, DX
- Fama-French Five Factors.

Action space. The action space describes the allowed actions that the agent interacts with the environment. Normally, action a includes three actions: {-1, 0, 1}, where -1, 0, 1 represent selling, holding, and buying one share. Also, an action can be carried upon multiple shares. We use an action space {-k, …, -1, 0, 1, …, k}, where k denotes the number of shares to buy and -k denotes the number of shares to sell. For example, "Buy 10 shares of MSFT" or "Sell 10 shares of MSFT" are 10 or -10, respectively. The continuous action space needs to be normalized to [-1, 1], since the policy is defined on a Gaussian distribution, which needs to be normalized and symmetric.

Reward function. The change of the portfolio value when action is taken as states and arriving at new states.

```
stock_dimension = len(processed_full.tic.unique())
state_space = 1 + 2*stock_dimension + len(tech_indicator_list)*stock_dimension
print(f"Stock Dimension: {stock_dimension}, State Space: {state_space}")
```

```
Stock Dimension: 1, State Space: 7
```

```
env_kwargs = {
    "hmax": 100,
    "initial_amount": 100000,
    "buy_cost_pct": 0.001,
    "sell_cost_pct": 0.001,
    "state_space": state_space,
    "stock_dim": stock_dimension,
    "tech_indicator_list": config.TECHNICAL_INDICATORS_LIST,
    "action_space": stock_dimension,
    "reward_scaling": 1e-4

}

e_train_gym = StockTradingEnv(df = train, **env_kwargs)
```

4.3.2 Deep Reinforcement Learning Agent

In this project, we use A2C to train our agent. The A2C is proposed to solve the training problem of PG by updating the policy in real time. It consists of two models: One is an actor network that outputs the policy, and the other is a critic network that measures how good the chosen action is in the given state.

```
trained_a2c = agent.train_model(model=model_a2c,
                                tb_log_name='a2c',
                                total_timesteps=50000)
```

```
------------------------------------
| time/                 |          |
|    fps                | 283      |
|    iterations         | 10000    |
|    time_elapsed       | 176      |
|    total_timesteps    | 50000    |
| train/                |          |
|    entropy_loss       | -1.82    |
|    explained_variance | 0        |
|    learning_rate      | 0.0002   |
|    n_updates          | 9999     |
|    policy_loss        | -0.732   |
|    std                | 1.5      |
|    value_loss         | 0.238    |
------------------------------------
```

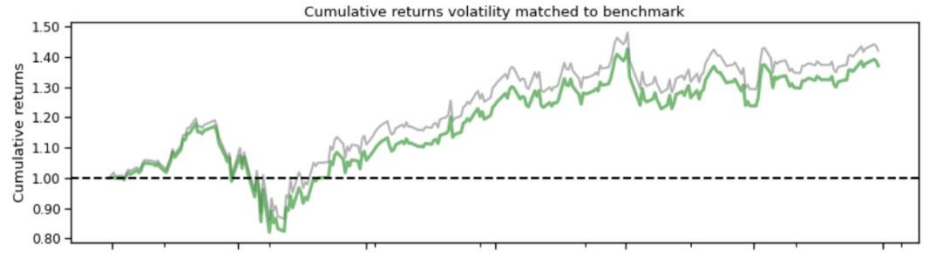
4.3.3 Back Testing Performance

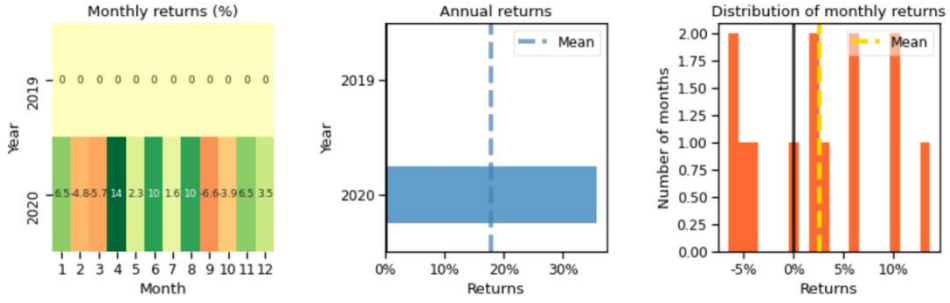
After seeing and learning from the training dataset, the agent uses the testing dataset to make prediction. The initial amount in our account is 100,000. The following is the change of our account value and the back test performance:

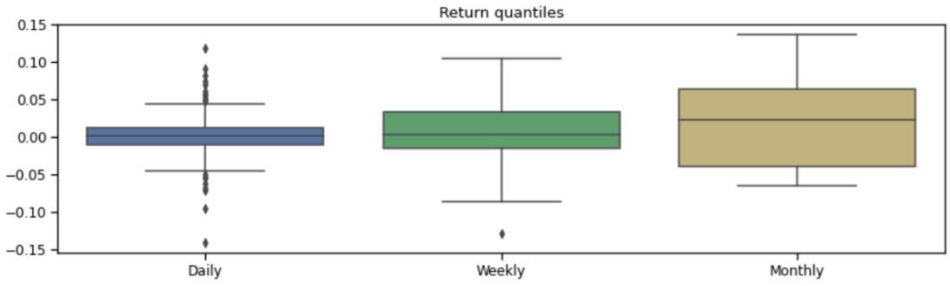
	date	account_value
0	2019-12-31	100000.000000
1	2020-01-02	100272.079956
2	2020-01-03	99862.271340
3	2020-01-06	99965.084666
4	2020-01-07	99393.836153
...	...	...
248	2020-12-23	135426.277503
249	2020-12-24	136413.488725
250	2020-12-28	137758.179753
251	2020-12-29	137266.979784
252	2020-12-30	135753.125876

	Backtest
Annual return	35.589%
Cumulative returns	35.753%
Annual volatility	42.061%
Sharpe ratio	0.94
Calmar ratio	1.22
Stability	0.70
Max drawdown	-29.19%
Omega ratio	1.19
Sortino ratio	1.35
Skew	NaN
Kurtosis	NaN
Tail ratio	1.12
Daily value at risk	-5.143%
Alpha	-0.03
Beta	0.95

Comparing DRL return with real stock return, we can see that this model fits the real trend almost perfectly.



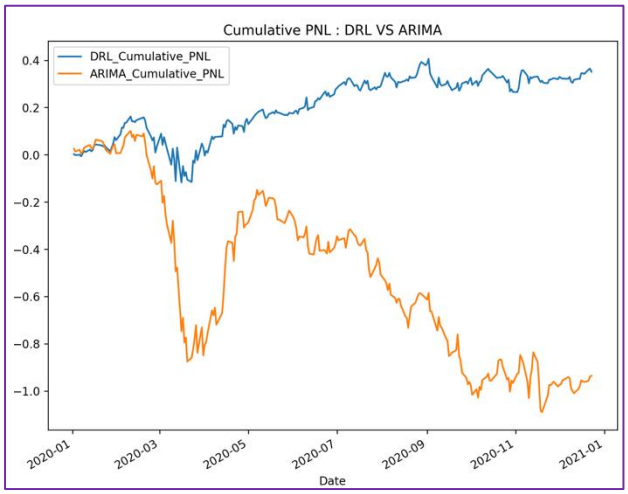




5. Results and Conclusions

5.1 Results between Classical ARIMA model vs DRL Model across a single year

We directly compare the cumulative PnL generated by the ARIMA model vs that generated by the DRL model, using the as the measure to determine if machine learning indeed can help us produce greater returns.



Date	DRL_Cumulative_PNL	ARIMA_Cumulative_PNL
2020-01-02	0.002721	0.025306
2020-01-03	-0.001366	0.013927
2020-01-06	-0.000309	0.021112
2020-01-07	-0.006181	0.004177
2020-01-08	0.005355	0.013080
2020-01-09	0.016329	0.028901
2020-01-10	0.011599	0.033916
2020-01-13	0.021568	0.041304
2020-01-14	0.014371	0.027835
2020-01-15	0.020693	0.026174
2020-01-16	0.038996	0.041387
2020-01-17	0.043554	0.063892
2020-01-21	0.039809	0.058318
2020-01-22	0.035588	0.053909
2020-01-23	0.041560	0.034349
2020-01-24	0.031487	0.020025
2020-01-27	0.014777	0.004417
2020-01-28	0.031598	0.023014
2020-01-29	0.046533	0.039018
2020-01-30	0.074689	0.043416
2020-01-31	0.062114	0.006553
2020-02-03	0.086268	0.007310
2020-02-04	0.115139	0.023074
2020-02-05	0.113803	0.038133
2020-02-06	0.134535	0.072318
2020-02-07	0.135882	0.081372
2020-02-10	0.162009	0.100929

5.2 Conclusions

- Deep Neural network has better predictive power compared to Classical models
- Machine Learning models can deliver returns without much human supervision
- Application of the machine learning approach requires less manual criteria selection processes, which reduces human biases, while still being able to deliver significant returns.

5.3 ML Performance Enhancement

- Greater training datasets could improve ML model performance