Apply machine learning models on Assets allocation

Speaker: Ko Wai Mei

1. Project objective

2. Methodology

- *Workflow of the system*
- *Models*

3. Challenges

4. Solutions

5. Evaluation

- *Portfolio Statistic*
- *Crisis management*

Project objective

Are you confident in your MPF asset allocation?

Project objective

1. Want to utilize the machine learning technology to make impact on helping people with MPF funds asset allocation

2. Get positive return of the portfolio and keep the risk at a reasonable level

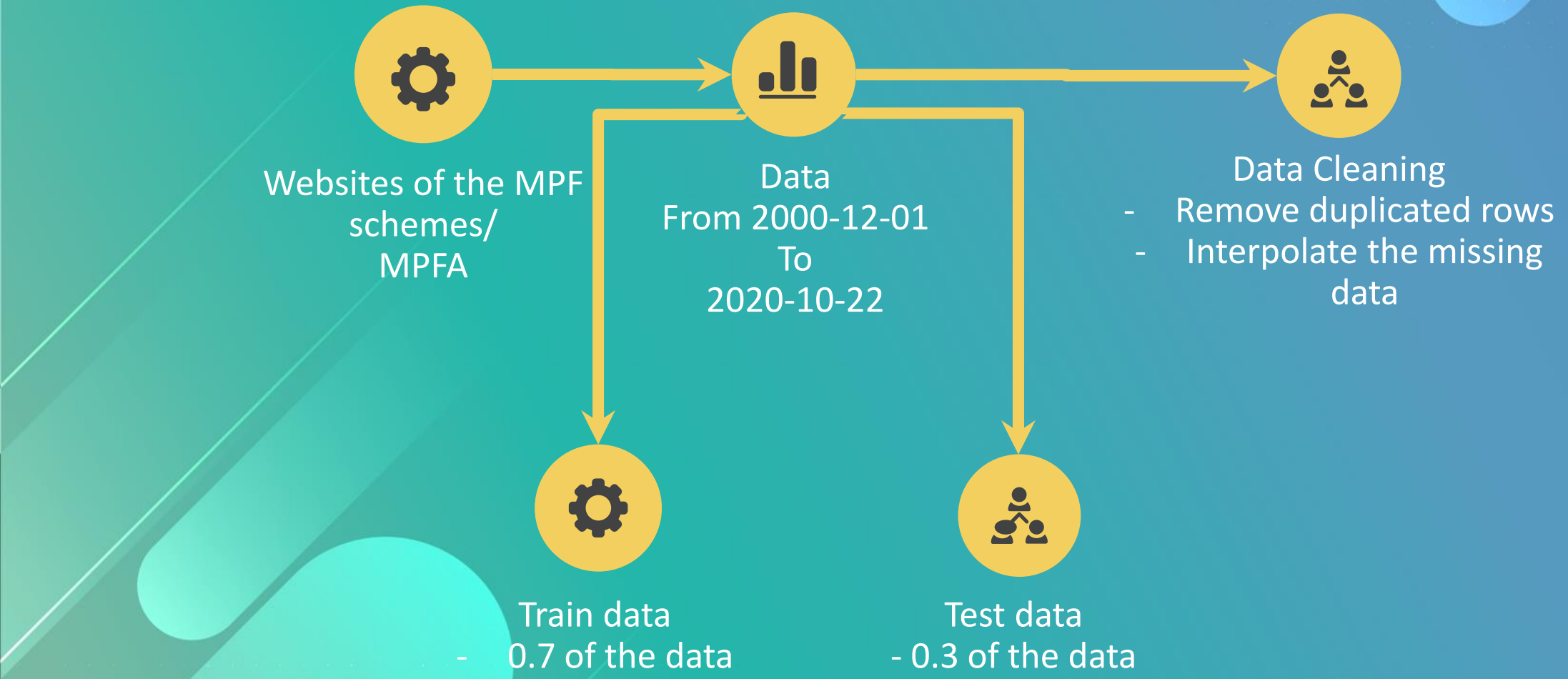
Choices of funds (Using AIA MPF funds as sample)

- Total 24 funds

65歲後基金
大中華股票基金
中港動態資產配置基金
中港基金
日本股票基金
北美股票基金
全球基金
均衡組合
亞洲股票基金
亞洲債券基金
亞歐基金
美洲基金
香港股票基金
核心累積基金
基金經理精選退休基金
強積金保守基金
富達增長基金
富達穩定資本基金
富達穩定增長基金
綠色退休基金
增長組合
歐洲股票基金
環球債券基金
穩定資本組合

Methodology

Funds allocation System



Funds allocation System

Cleaned data

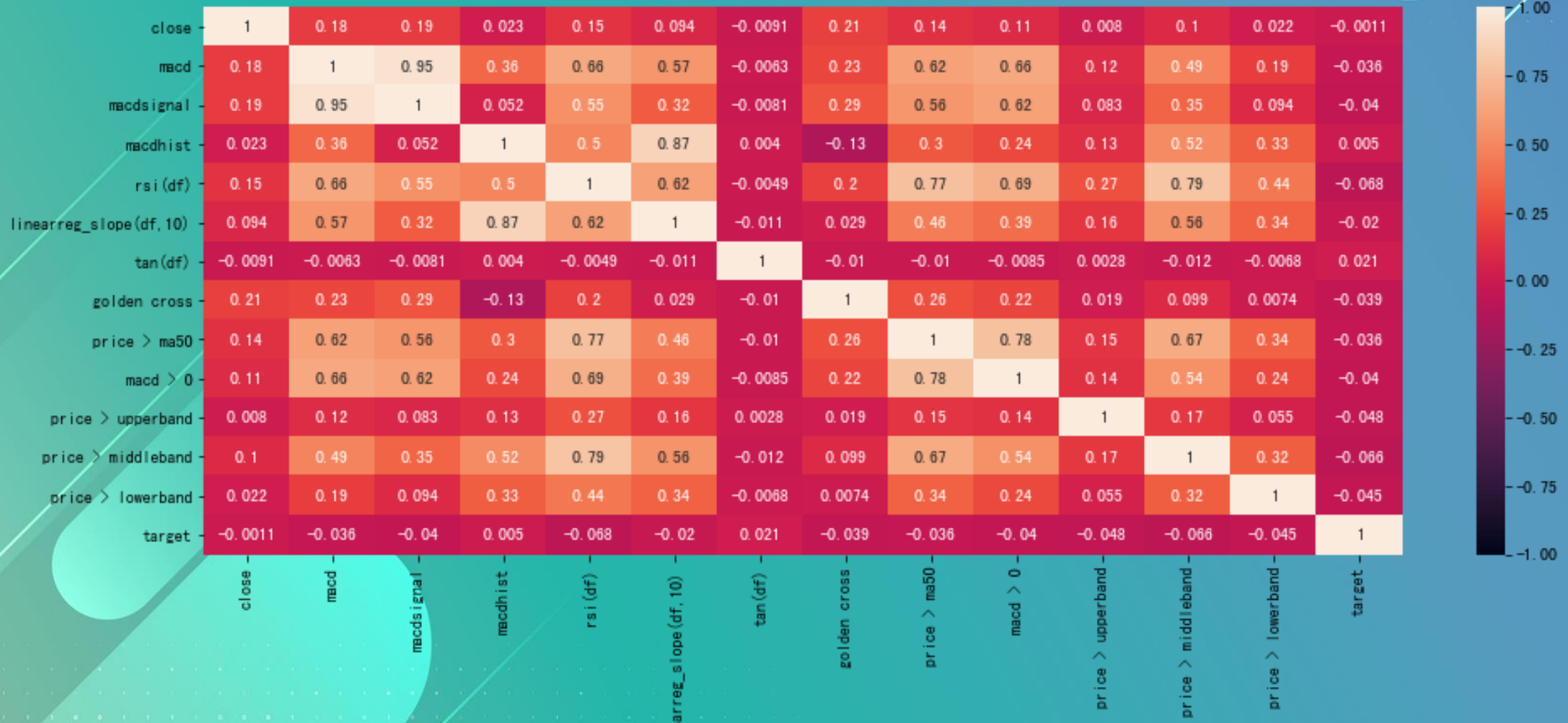
Feature engineering

Analytical Table (for each fund)

Date	close	Other featrures	Ema(50)	Golden Cross	MACD (12,26,9)	Target (Return of one month)
2001-01-31	272.80	……	275.80	208.2536	2.561320	0 (negative return)
2001-02-28	271.80	……	271.50	219.3535	2.742569	1 (positive return)

• • •

Correlation heatmap between features



Funds allocation System

Modeling data of each fund

Construct reference model and other machine learning models

- Three models are used:
  - 1. Logistic
  - regression
  - 2. SVM
  - 3. LightGBM

After analyzing the performance of the model, an ensemble model is derived

Model result:

```
In [589]:
# Emsemble prediction
pred1 = lr_model.predict(X_test)
pred2 = svc_model.predict(X_test)
pred3 = lgb_model.predict(X_test)

# Final Prediction
# finalpred=(pred1*0.2+pred2*0.5+pred3*0.3)
finalpred=(pred1+pred2+pred3)/3

#convert into binary values
for i in range(len(finalpred)):
    if finalpred[i] >= 0.5:
        finalpred[i] = 1
    else:
        finalpred[i] = 0
```

```
In [590]:
#Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, finalpred)
cm

Out[590]:
array([[101, 157],
       [ 82, 508]], dtype=int64)
```

```
In [591]:
#Accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, finalpred)
round(accuracy,2)

Out[591]:
0.72
```

Funds allocation System

Inputs:
Price of the funds

Trained ML Model
- Predict which stock will have positive return after one month
- Select the stock with label 1 for further assets allocation

Output:
Selected Stock

Selected a best weighting for the selected stocks
- Highest sharpe ratio according to its previous 3 months performance

Apply the optimal weightings to construct the portfolio

Sample of the models – Dummy Model

```python
from sklearn.dummy import DummyClassifier
from sklearn.model_selection import cross_validate
strategy = ['uniform','most_frequent','prior']
for str_item in strategy:
    # define the reference model
    model = DummyClassifier(strategy=str_item)
    pipe = Pipeline([('scaler', StandardScaler()), ('model', model)])
    # evaluate the model
    btscv = BlockingTimeSeriesSplit(n_splits=5)
    scores = cross_validate(pipe, X_train, y_train, scoring=['accuracy', 'precision'], cv=btscv, n_jobs=-1)
    pipe.fit(X_test,y_test)
    # summarize performance
    counter = 0
    for key in scores.keys():
        print('%s: %.3f (%.3f)' % (key,scores[key].mean(), scores[key].std()))
        counter += 1
        if counter == 4:
            print('\n')
```

```
fit_time: 0.016 (0.008)
score_time: 0.004 (0.001)
test_accuracy: 0.492 (0.028)
test_precision: 0.682 (0.157)


fit_time: 0.008 (0.002)
score_time: 0.004 (0.001)
test_accuracy: 0.681 (0.161)
test_precision: 0.681 (0.161)


fit_time: 0.003 (0.000)
score_time: 0.002 (0.000)
test_accuracy: 0.681 (0.161)
test_precision: 0.681 (0.161)
```

Highest Accuracy: 0.681

Highest precision: 0.681

Sample of the models – Logistic Regression

```
from sklearn.linear_model import LogisticRegression
 # define the reference model
model = LogisticRegression(solver="liblinear")
lr_model = Pipeline([('scaler', StandardScaler()), ('model', model)])
# evaluate the model
btscv = BlockingTimeSeriesSplit(n_splits=5)
scores = cross_validate(model, X_train, y_train, scoring=['accuracy', 'precision'], cv=btscv, n_jobs=-1)
lr_model.fit(X_test,y_test)
# summarize performance
counter = 0
for key in scores.keys():
    print('%s: %.3f (%.3f)' % (key,scores[key].mean(), scores[key].std()))
    counter += 1
    if counter == 4:
        print('\n')
```

```
fit_time: 0.012 (0.005)
score_time: 0.003 (0.000)
test_accuracy: 0.569 (0.157)
test_precision: 0.695 (0.130)
```

Highest Accuracy: 0.569

Highest precision: 0.695

Sample of the models – SVM

```
from sklearn.svm import SVC
strategy = ['linear','rbf','sigmoid']
for str_item in strategy:
    # define the reference model
    model = SVC(kernel=str_item)
    svc_model = Pipeline([('scaler', StandardScaler()), ('model', model)])
    # evaluate the model
    btscv = BlockingTimeSeriesSplit(n_splits=5)
    scores = cross_validate(pipe, X_train, y_train, scoring=['accuracy', 'precision'], cv=btscv, n_jobs=-1)
    svc_model.fit(X_test,y_test)
    # summarize performance
    counter = 0
    for key in scores.keys():
        print('%s: %.3f (%.3f)' % (key,scores[key].mean(), scores[key].std()))
        counter += 1
        if counter == 4:
            print('\n')
```

```
fit_time: 0.005 (0.001)
score_time: 0.003 (0.001)
test_accuracy: 0.681 (0.161)
test_precision: 0.681 (0.161)


fit_time: 0.008 (0.004)
score_time: 0.004 (0.002)
test_accuracy: 0.681 (0.161)
test_precision: 0.681 (0.161)


fit_time: 0.013 (0.005)
score_time: 0.006 (0.001)
test_accuracy: 0.681 (0.161)
test_precision: 0.681 (0.161)
```

Highest Accuracy: 0.681

Highest precision: 0.681

Sample of the models – LightGBM

```
params = {
#     'learning_rate' : [round(x,2) for x in np.random.uniform(0, 1, 10)],
    'objective':['binary'],
    'learning_rate' : [0.001,0.01,0.05,0.08,0.1],
#     'model__max_depth' : np.random.randint(5, 20,5),
    'feature_fraction' : [0.5,0.7,0.8,0.9],
#     'num_leaves': np.random.randint(1, 10,5),
    'min_data_in_leaf': np.random.randint(5, 10,5),
#     'lambda_l1': [round(x,2) for x in np.random.uniform(0, 1, 3)],
#     'model__lambda_l2': np.random.randint(0, 0.5 ,3),
    'lambda_l2': [0.005,0.05,0.1],
    'boosting_type' : ['gbdt'],
    'reg_alpha' : [0,0.05,0.5,1],
    'reg_lambda' : [0,0.05,0.5,1]
    }
# model__fit_params={'model__early_stopping_rounds':20,'model__eval_set':[(X_train,y_train)]}
# btscv = BlockingTimeSeriesSplit(n_splits=5)
gkf = KFold(n_splits=5, shuffle=True, random_state=42).split(X_train, y_train)
lgb_estimator = lgb.LGBMClassifier(objective='binary', num_boost_round=50,scale_pos_weight = 0.5)

gsearch = Pipeline([('scaler', StandardScaler()), ('model', GridSearchCV(
        estimator=lgb_estimator,
        param_grid=params,
        n_jobs=-1,
        scoring = 'precision',
        refit='precision',
        cv=gkf,  # customerised splitter subject to test
        verbose=1,
        pre_dispatch=8,
        error_score=-999,
        return_train_score=True,
        ))])

lgb_model = gsearch.fit(X_train, y_train)

# summarize performance
print(f"Best grid scores: ")
print()
means = lgb_model['model'].cv_results_['mean_test_score']
stds = lgb_model['model'].cv_results_['std_test_score']
best_index = np.where(means==max(means))[0][0]
mean = means[best_index]
std = stds[best_index]
params = lgb_model['model'].cv_results_['params'][best_index]
print("%0.3f (+/-%0.03f) for %r"
      % (mean, std, params))
```

Highest Accuracy: 0.826

Highest precision: 0.869

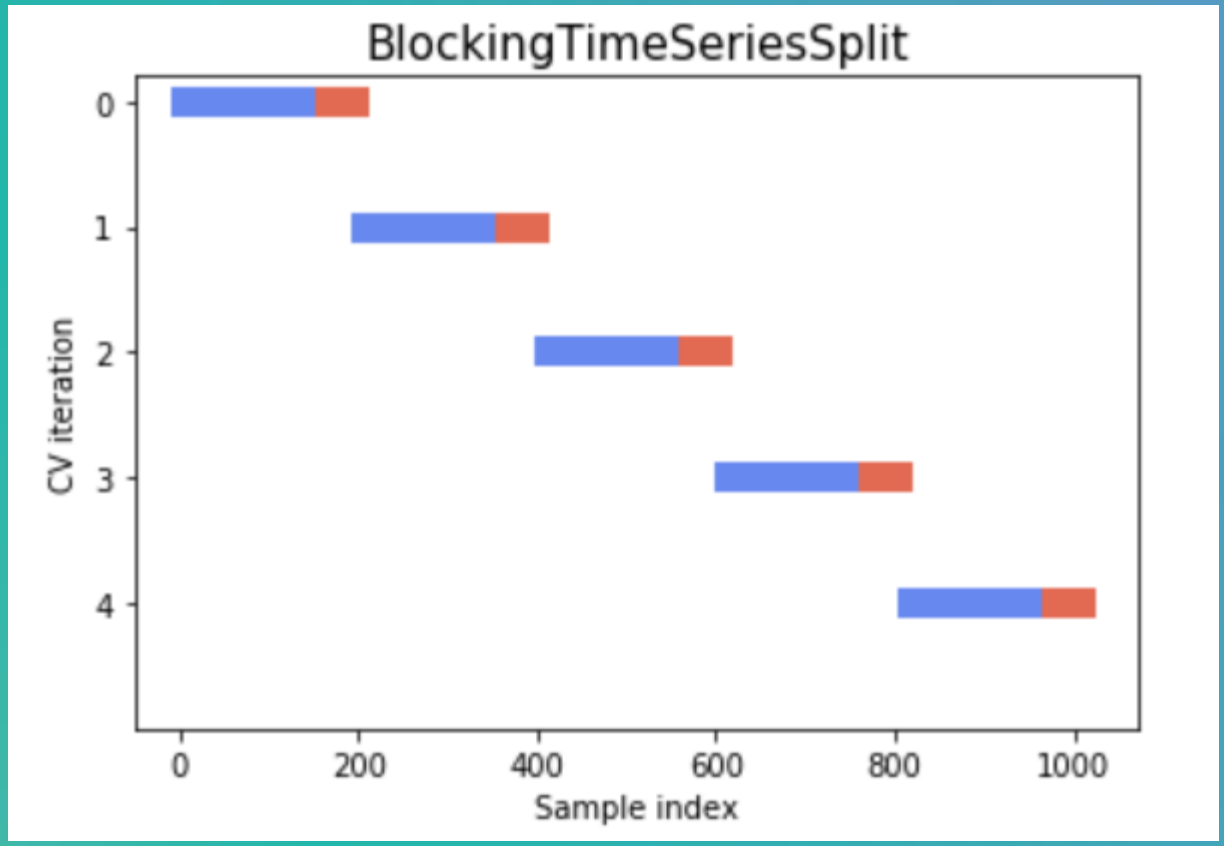
Introduction of the LightGBM model

- The most common model used by the Kaggle champions
- Fast training speed and low memory usage
- High-performance gradient boosting framework – for financial data

```
gkf = KFold(n_splits=5, shuffle=True, random_state=42).split(x_train, y_train)

param_grid = {
#     'learning_rate' : [round(x,2) for x in np.random.uniform(0, 1, 10)],
    'learning_rate' : [0.01,0.05,0.1,0.3,0.6],
    'max_depth' : np.random.randint(5, 20,5),
    'feature_fraction' : [0.5,0.7,0.8,0.9],
#     'cat_smooth' : [1,10,15],
    'num_leaves': np.random.randint(10, 50,5),
#     'min_data_in_leaf': np.random.randint(5, 20,5),
#     'lambda_l1': [round(x,2) for x in np.random.uniform(0, 1, 3)],
#     'lambda_l2': np.random.randint(0, 50 ,3),
    'boosting_type' : ['gbdt','rf','dart']
    }

lgb_estimator = lgb.LGBMClassifier(objective='binary', num_boost_round=100, metric='binary_logloss',num_class=2)
gsearch = GridSearchCV(estimator=lgb_estimator, param_grid=param_grid, cv=gkf)
lgb_model = gsearch.fit(x_train, y_train)

# Best Score
print("Best score: %0.3f" % lgb_model.best_score_)
print("Best parameters set:")

# Best params
best_parameters = lgb_model.best_estimator_.get_params()
for param_name in sorted(best_parameters.keys()):
    print("\t%s: %r" % (param_name, best_parameters[param_name]))
```

Techniques for optimising the model

- Rolling cross validation



Techniques for optimising the model

- Grid Search
- Tuning the hyper parameters of the model for better prediction
- Sample of parameters adopted by the model

```
params = {
#     'learning_rate' : [round(x,2) for x in np.random.uniform(0, 1, 10)],
    'objective':['binary'],
    'learning_rate' : [0.001,0.01,0.05,0.08,0.1],
#     'model__max_depth' : np.random.randint(5, 20,5),
    'feature_fraction' : [0.5,0.7,0.8,0.9],
#     'num_leaves': np.random.randint(1, 10,5),
    'min_data_in_leaf': np.random.randint(5, 10,5),
#     'lambda_l1': [round(x,2) for x in np.random.uniform(0, 1, 3)],
#     'model__lambda_l2': np.random.randint(0, 0.5 ,3),
    'lambda_l2': [0.005,0.05,0.1],
    'boosting_type' : ['gbdt'],
    'reg_alpha' : [0,0.05,0.5,1],
    'reg_lambda' : [0,0.05,0.5,1]
    }
```

Performance of the model (Recap)

In [589]:
```
# Emsemble prediction
pred1 = lr_model.predict(X_test)
pred2 = svc_model.predict(X_test)
pred3 = lgb_model.predict(X_test)

# Final Prediction
# finalpred=(pred1*0.2+pred2*0.5+pred3*0.3)
finalpred=(pred1+pred2+pred3)/3

#convert into binary values
for i in range(len(finalpred)):
    if finalpred[i] >= 0.5:
        finalpred[i] = 1
    else:
        finalpred[i] = 0
```

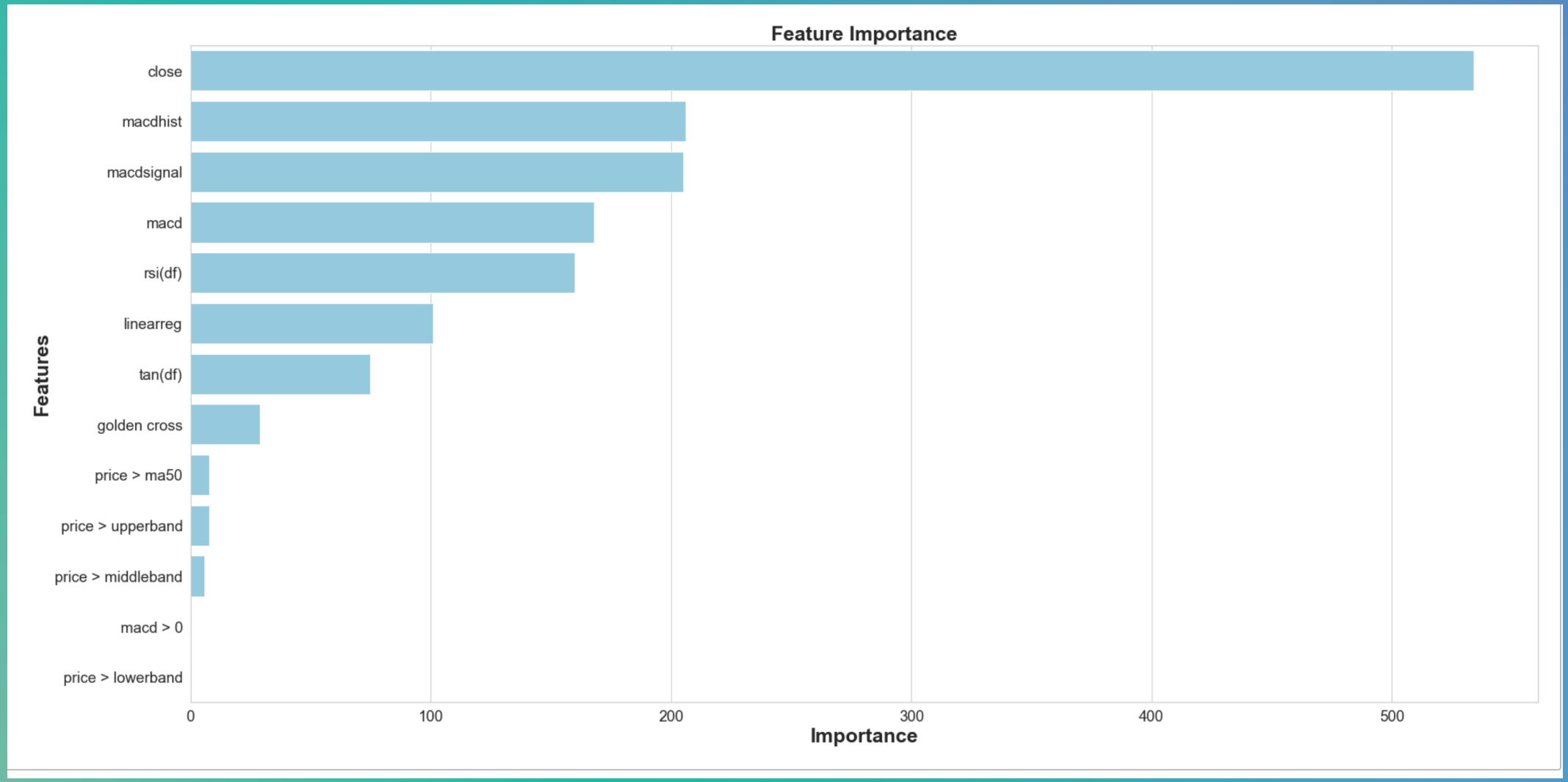
In [590]:
```
#Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, finalpred)
cm
```

Out[590]:
```
array([[101, 157],
       [ 82, 508]], dtype=int64)
```

In [591]:
```
#Accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, finalpred)
round(accuracy,2)
```

Out[591]:
```
0.72
```

Features Importance (LightGBM model)



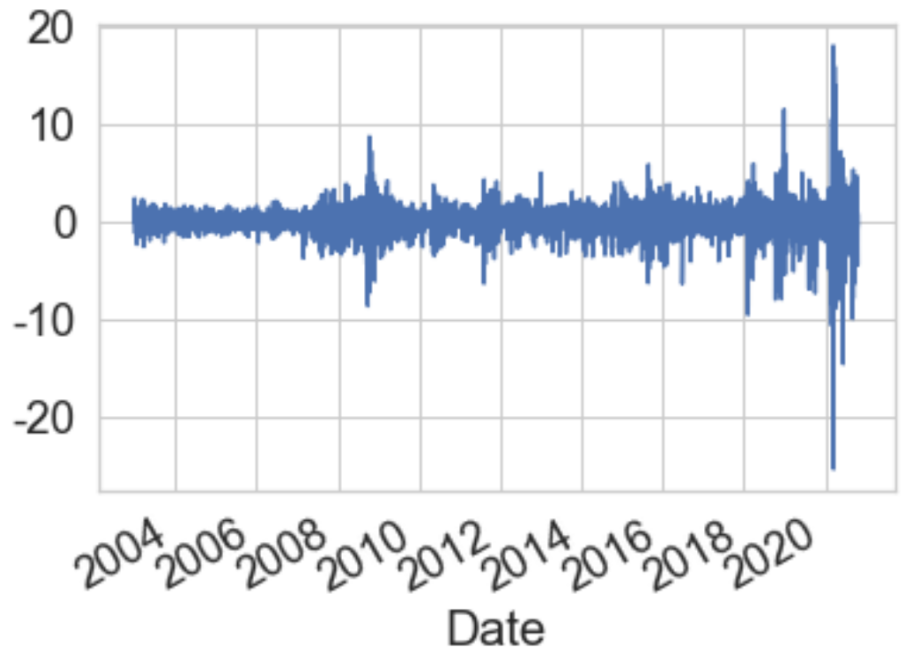
Analysis of the close price – Remove Trend

Remove trend

```
first_diff = df_w_features['close'].diff()
first_diff = first_diff.dropna()
```

```
first_diff.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x271dacc88c8>
```



Analysis of the close price – Ljung-Box test

```
acf_values = acf(first_diff,nlags=50,qstat=True,fft=False)
```

```
acf_values[2]
```

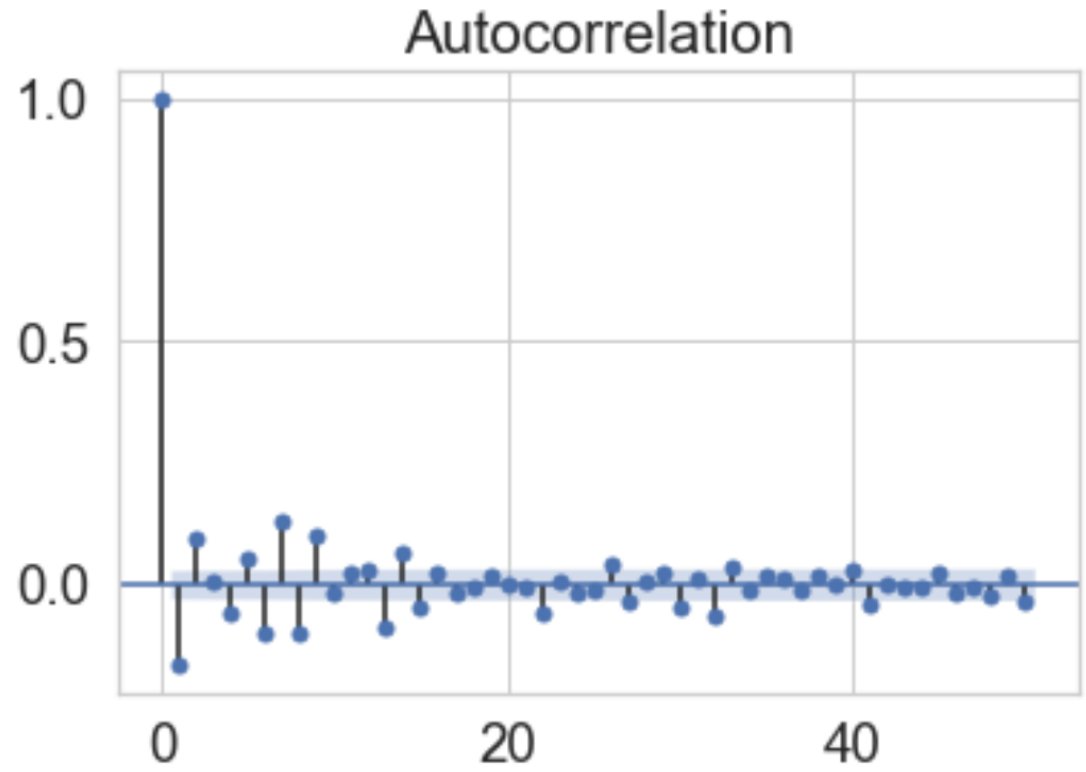
```
array([5.03148025e-29, 2.64105636e-36, 2.38517098e-35, 4.37056658e-38,
       6.33516710e-40, 2.60221426e-48, 9.76512324e-63, 4.75523098e-72,
       1.57655546e-80, 5.47279056e-80, 1.13858859e-79, 8.70666704e-80,
       1.10062938e-86, 1.54171627e-89, 4.54396670e-91, 8.98903719e-91,
       2.21837581e-90, 1.01737645e-89, 3.47579283e-89, 1.76852827e-88,
       8.44496971e-88, 3.64142310e-90, 1.52211594e-89, 4.17267149e-89,
       1.22315582e-88, 1.02339930e-89, 2.34984552e-90, 1.00785388e-89,
       1.75788535e-89, 6.56930596e-91, 1.92485717e-90, 6.11564573e-94,
       1.39123387e-94, 4.28718774e-94, 1.00747159e-93, 3.35766100e-93,
       1.06789186e-92, 2.69937968e-92, 1.00837767e-91, 5.72656073e-92,
       4.25558736e-93, 1.57999738e-92, 5.48720851e-92, 1.90762220e-91,
       2.63885404e-91, 4.37647205e-91, 1.49971360e-90, 1.25343509e-90,
       2.62391044e-90, 7.59157728e-91])
```

Result of p-values are all less than 0.5:

H0: correlations of residuals of autocorrelation = 0
-> Cannot be rejected

Analysis of the close price – Autocorrelation

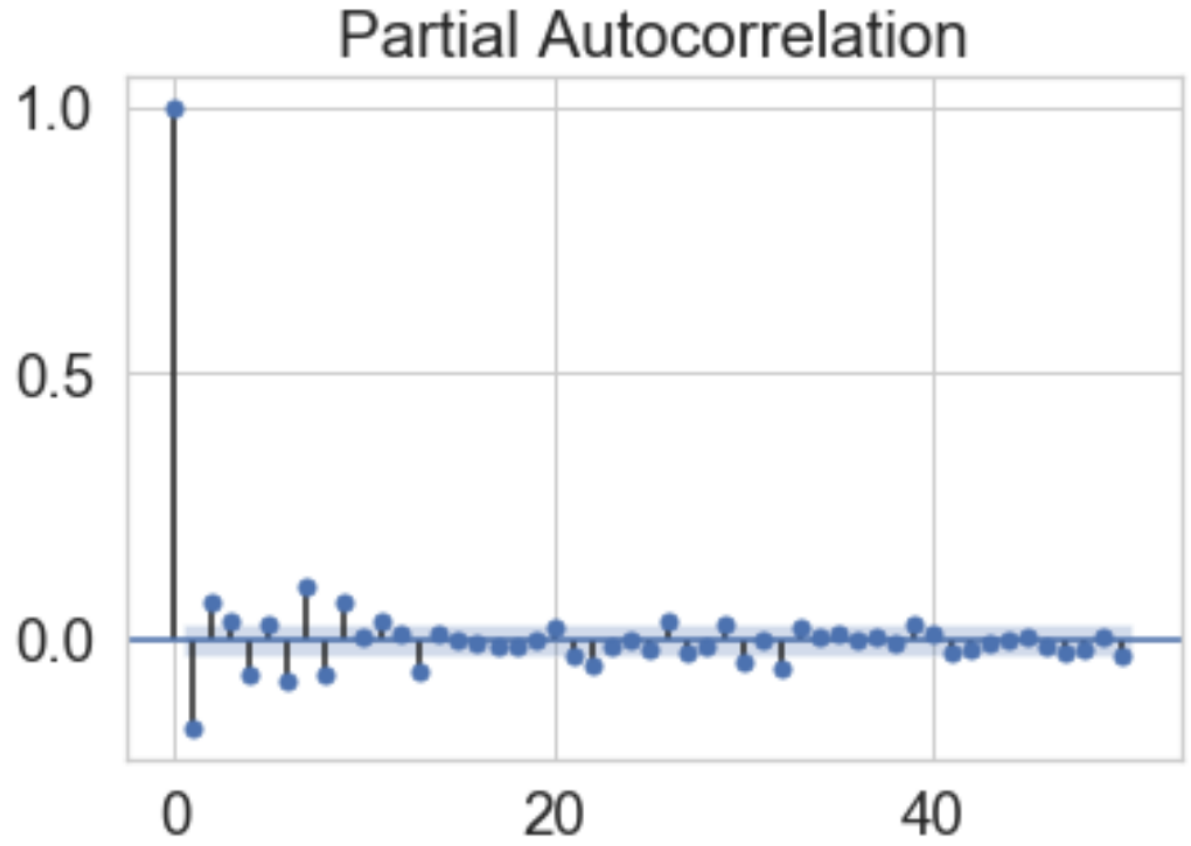
```
from statsmodels.tsa.stattools import acf,pacf
from statsmodels.graphics.tsaplots import *
plot_acf(first_diff,use_vlines=True, lags=50)
plt.show()
```



The autocorrelation after lags=30 are not significant

Analysis of the close price – P Autocorrelation

```
plot_pacf(first_diff,use_vlines=True, lags=50)
plt.show()
```



The partial autocorrelations after lags=30 are not significant

Challenges

Challenges

The separate models are not good enough before ensemble

The initial direction of the project

Initial features contain not much predictive power

Solution:

1. Make problems clear

2. Conduct research

2. Try and error

Evaluation

Portfolio Statistics

Backtesting result during 31/10/2018 - 30/9/2020

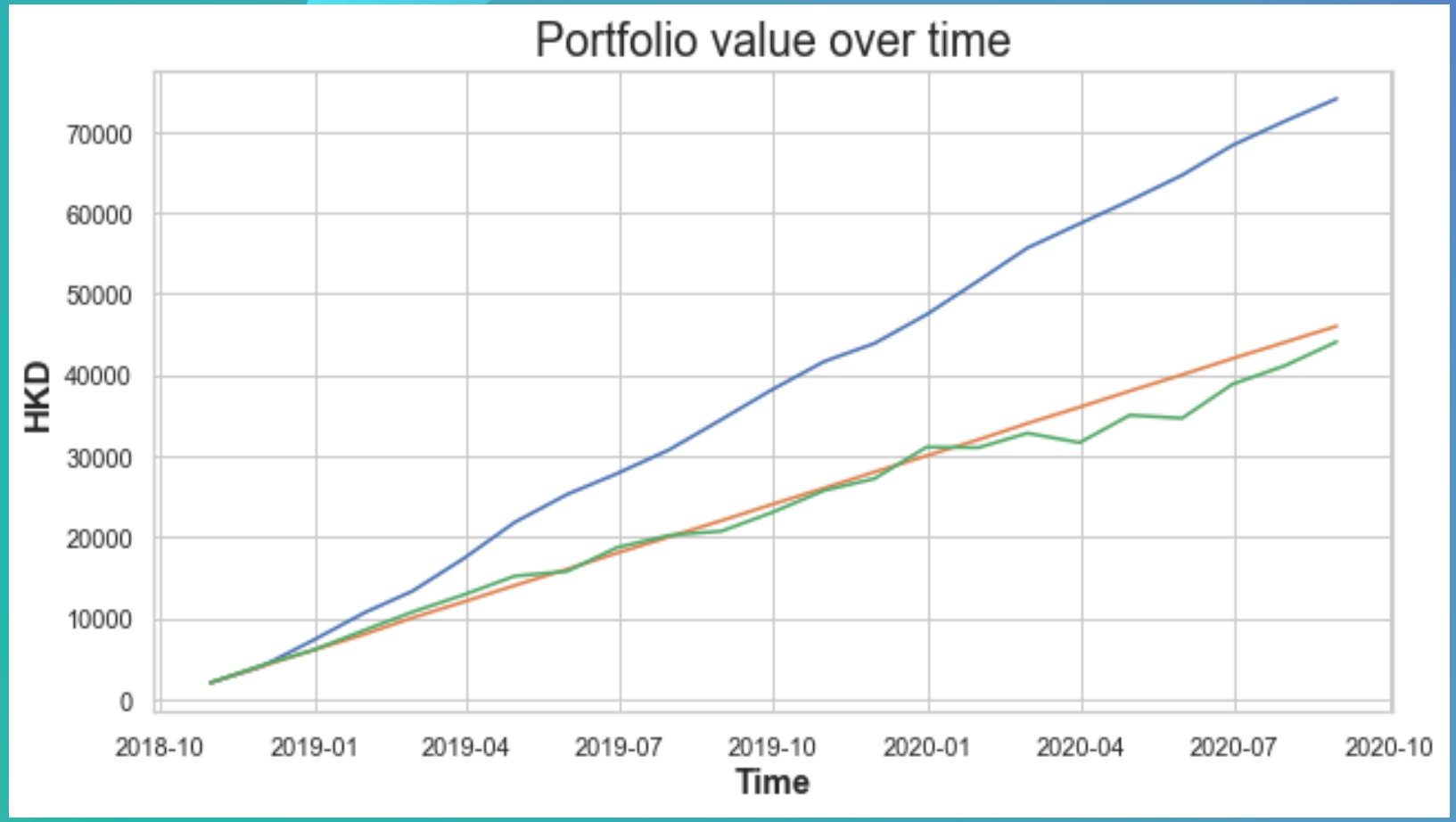
	System Selected Portfolio	HSI	Difference
Total period retuen (%)	9.69	-15.58	25.27
Total period standard deviation (%)	4.44	0.06	4.38
Annulised return (%)	4.74	-8.12	12.86
Annulised standard deviation (%)	1.13	0.79	0.34

Backtesting result during 31/10/2018 - 30/9/2020

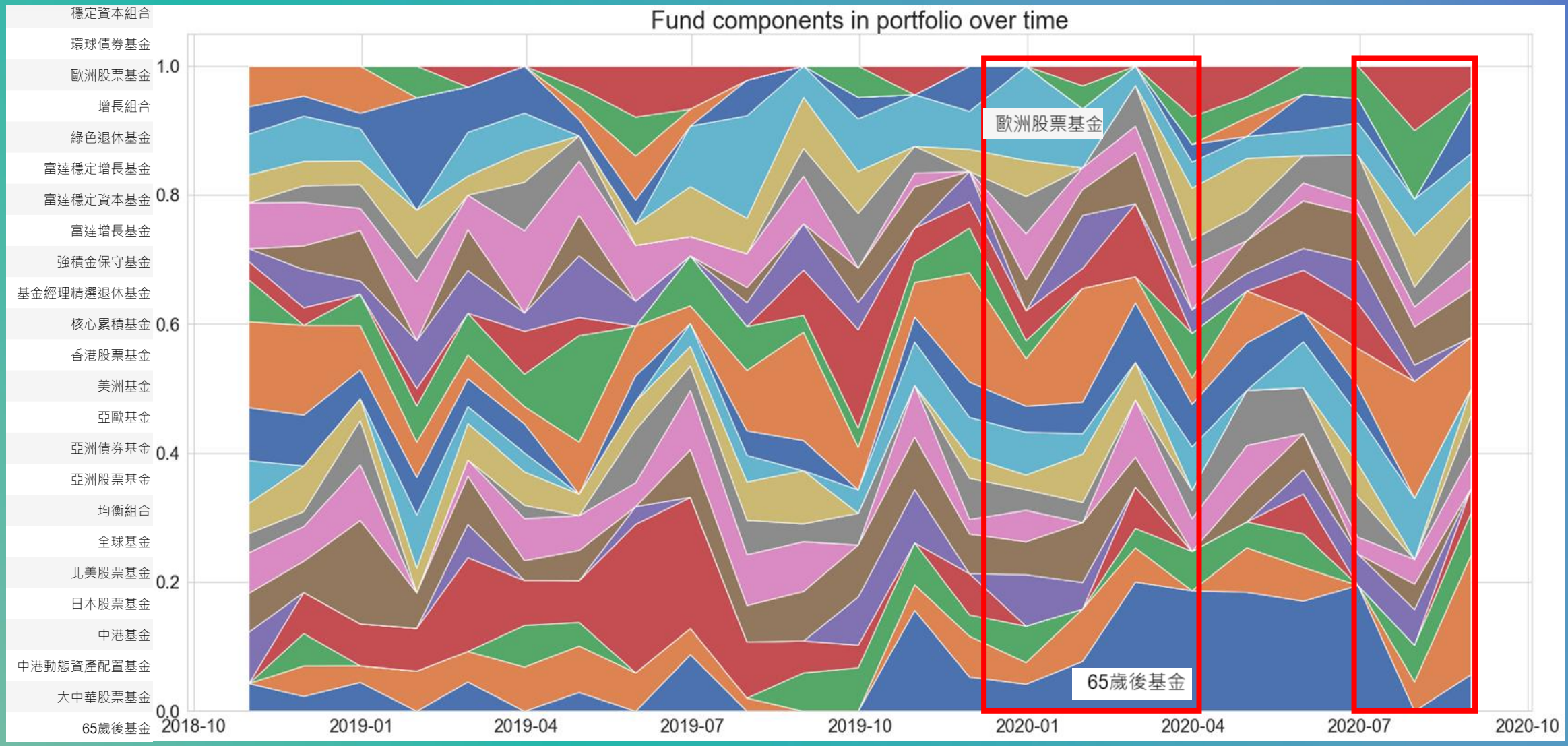
Our portfolio

VS

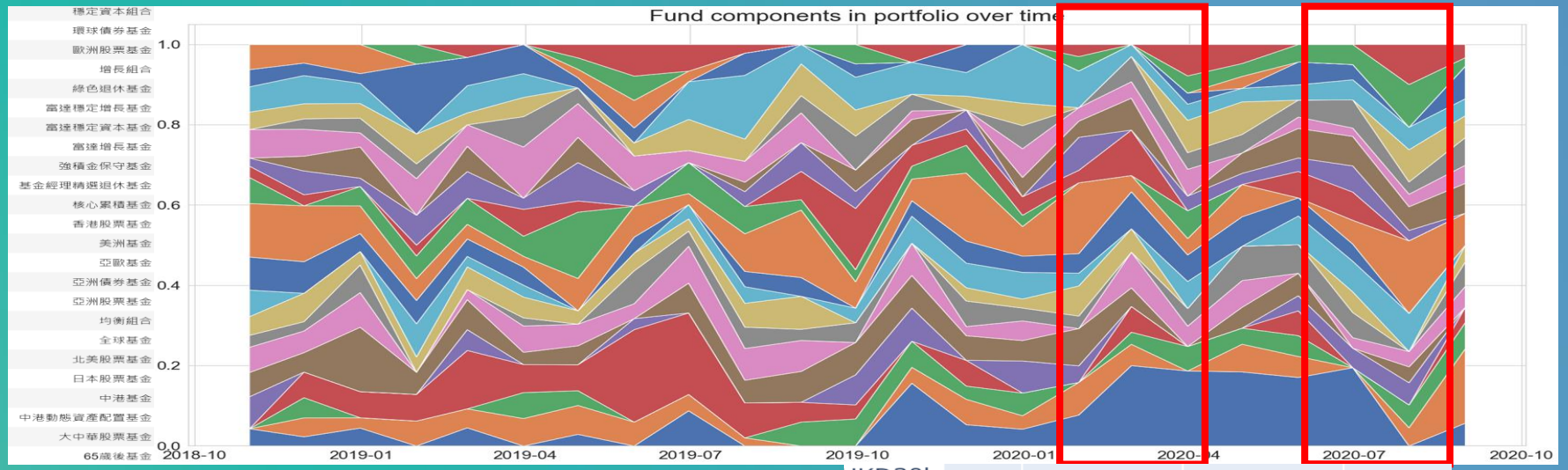
Other assets allocation



Components of fund over time



HSI in the same period




What can be next?
BUSINESS
Economy of the European Union
25 great jobs for people who love to travel
CASHBOOK

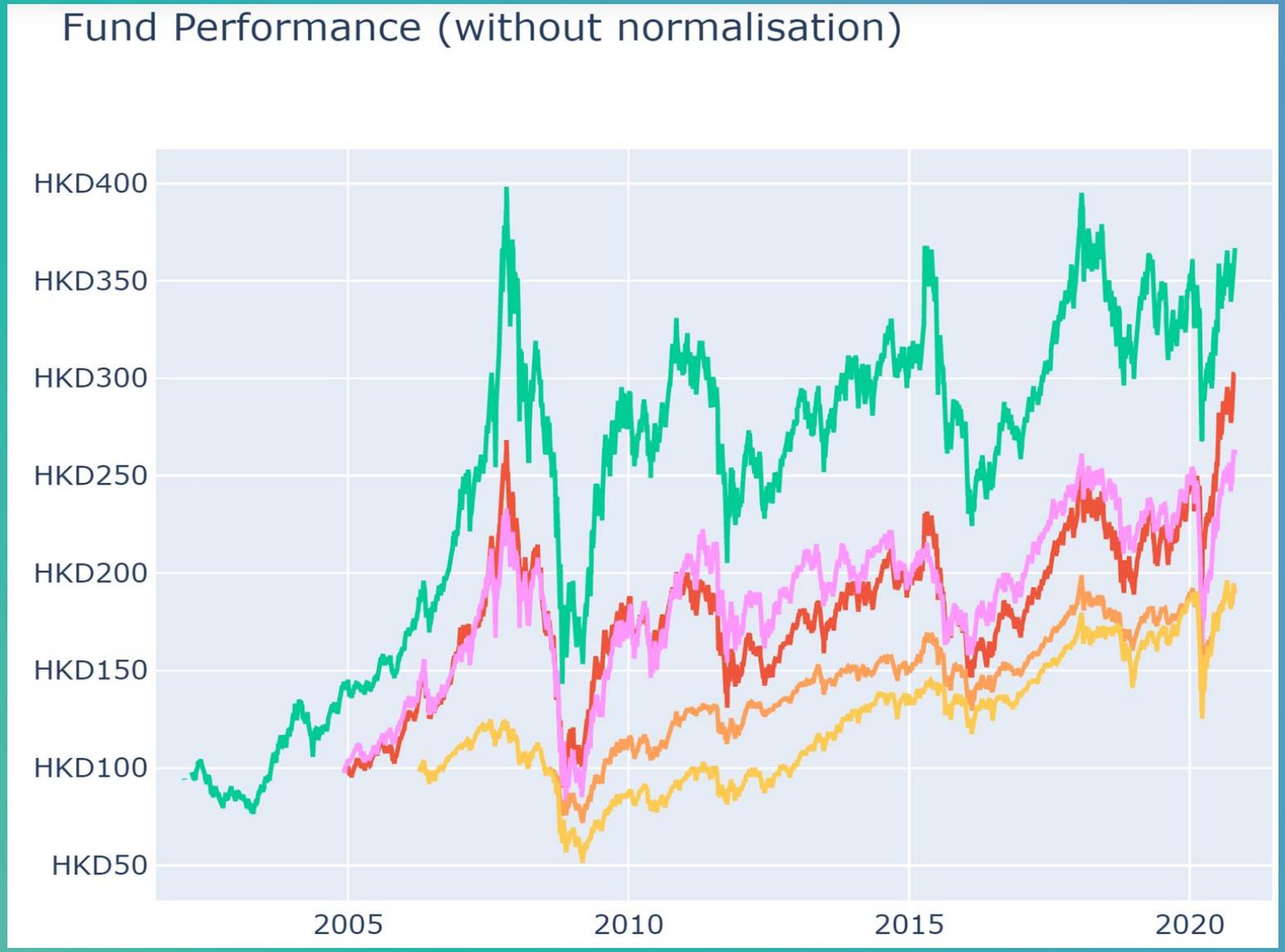
1. Gathering the info of the components of the funds over time
   -> Conduct sentiment analysis of the components

2. Push the model to live dashboard to monitor its performance

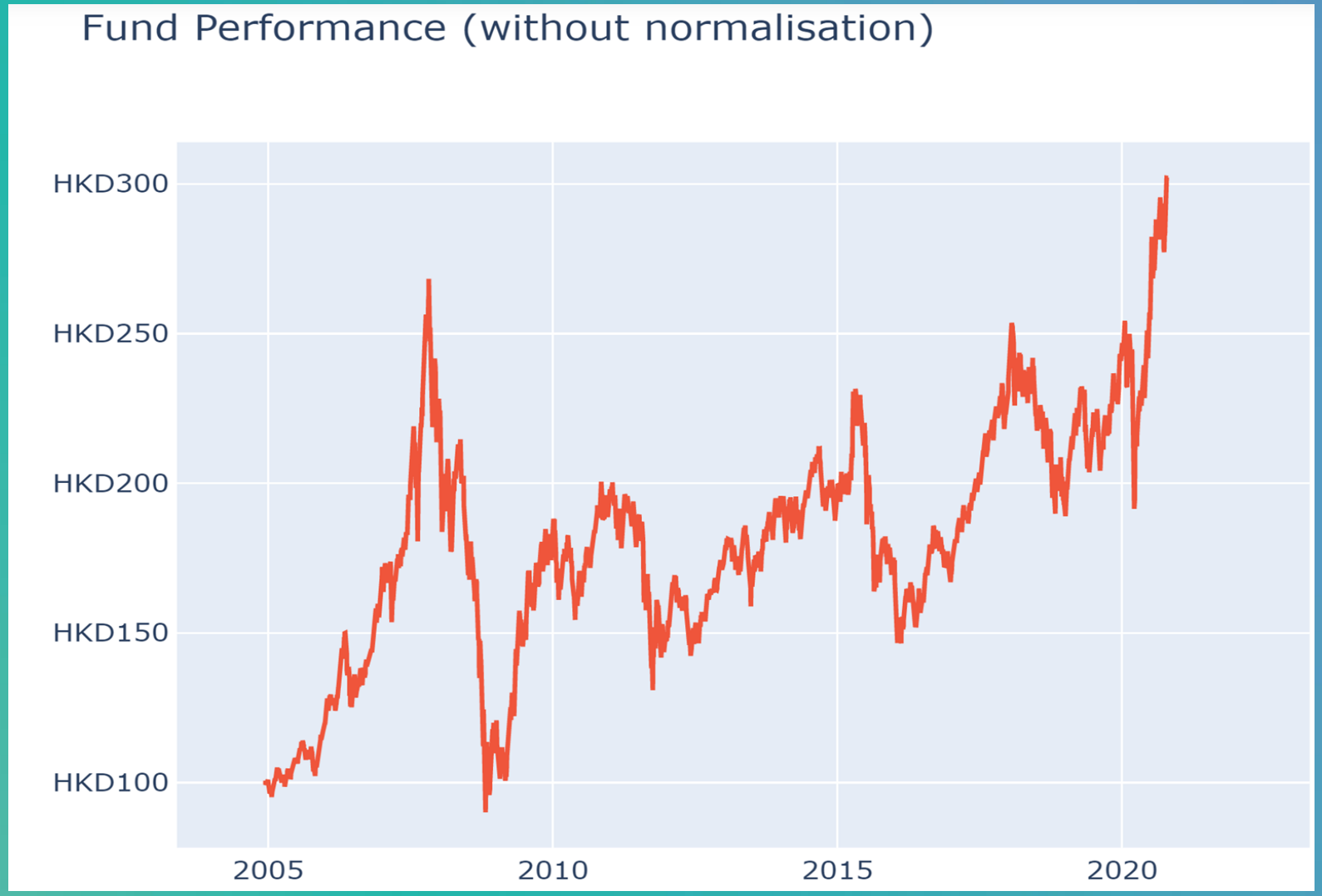
3. Come up with more ideas and make more impact

Appendix: Details of funds with high importance in model

Combined funds graph

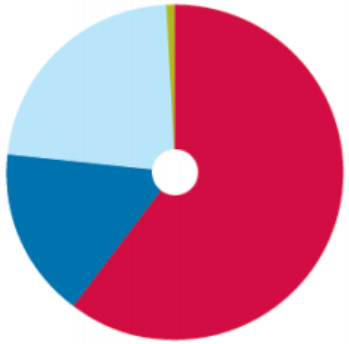


大中華股票基金


Fund Performance (without normalisation)
HKD300
HKD250
HKD200
HKD150
HKD100
2005
2010
2015
2020

大中華股票基金

資產分布 | ASSET ALLOCATION



- 60.37% 中國 China
- 16.31% 香港 Hong Kong
- 22.66% 台灣 Taiwan
- 0.66% 現金及其他 Cash and Others

十大投資項目# | TOP TEN HOLDINGS#

截至2020年7月31日 As at 31 July 2020

	佔資產淨值百分比 % of NAV
台灣積體電路製造股份有限公司 TAIWAN SEMICONDUCTOR MANUFACTURING CO LTD	9.60%
騰訊控股 TENCENT HOLDINGS LTD	9.53%
阿里巴巴集團 ALIBABA GROUP HOLDING LTD	4.87%
美團點評 MEITUAN DIANPING	3.94%
中國平安 PING AN INSURANCE (GROUP) CO OF CHINA LTD H	3.61%
友邦保險 AIA GROUP LTD	3.37%
香港交易所 HONG KONG EXCHANGES & CLEARING LTD	3.15%
金蝶國際軟件集團有限公司 KINGDEE INTERNATIONAL SOFTWARE GROUP LIMITED	2.07%
招商銀行 CHINA MERCHANTS BANK	1.98%
藥明生物 WUXI BIOLOGICS	1.89%

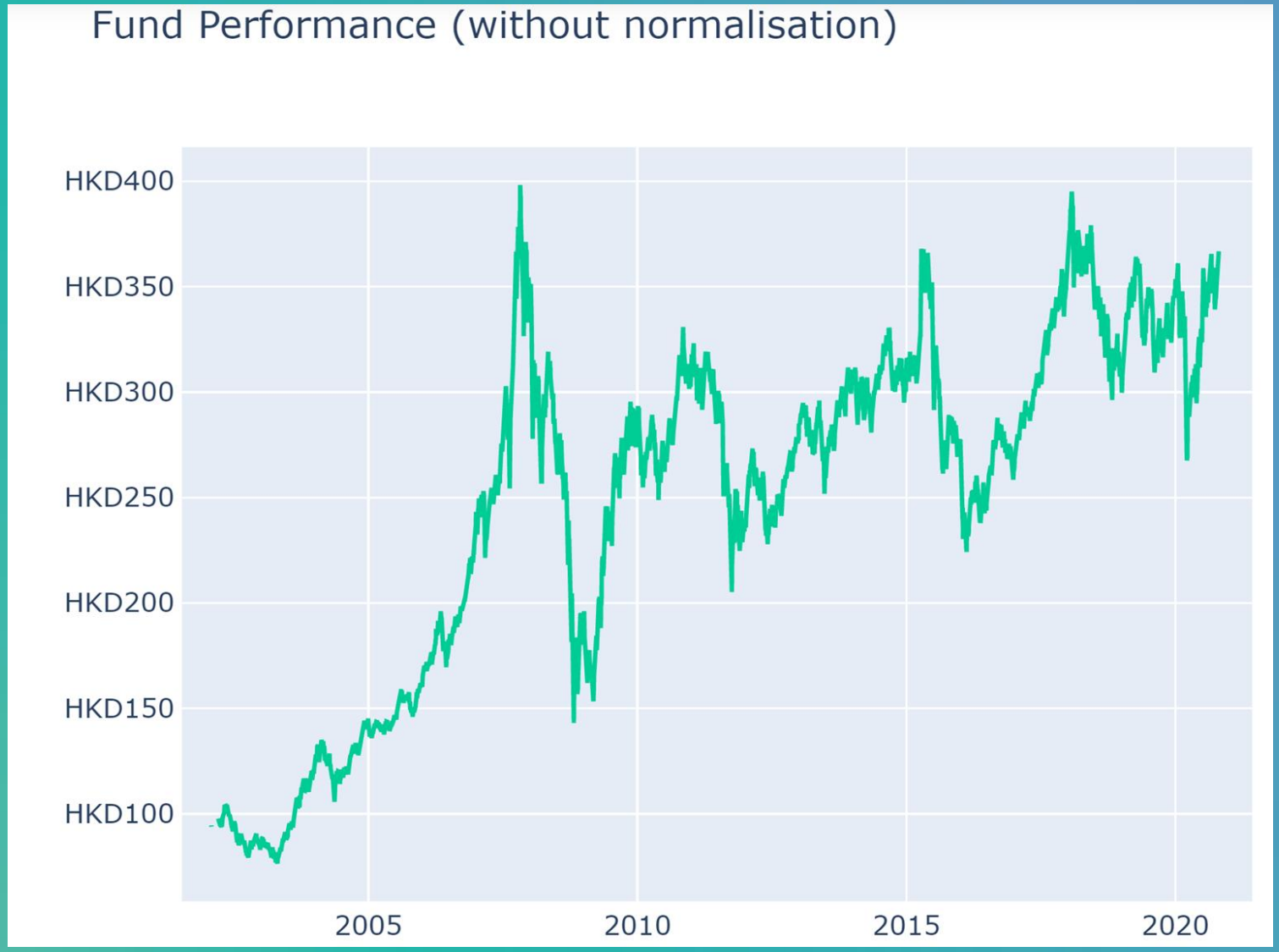
基金表現 | FUND PERFORMANCE

（資產淨值對資產淨值，以港元計算□ NAV to NAV, in HK Dollars□）

	一年 1 Year	三年 3 Years	五年 5 Years	十年 10 Years	成立至今 Since Launch	年初至今 YTD
累積回報 Cumulative Return (%)						
基金 Fund	36.64	33.61	67.77	69.01	190.39	19.20
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	21.79	28.65	42.84	52.03	69.74	19.25
年度化回報 Annualized Return (%)						
基金 Fund	36.64	10.14	10.90	5.39	7.00	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	21.79	8.76	7.39	4.28	3.42	-
曆年回報 Calendar Year Return(%)	**2019**	**2018**	**2017**	**2016**	**2015**	**-**
基金 Fund	24.57	-15.08	35.23	-2.32	-11.33	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	10.32	-11.35	12.53	2.04	-9.41	-

source:
aia mpf prime value choice fund performance review-aug-2020

香港股票基金


Fund Performance (without normalisation)
HKD400
HKD350
HKD300
HKD250
HKD200
HKD150
HKD100
2005
2010
2015
2020

香港股票基金

資產分布 | ASSET ALLOCATION

- 11.38% 消費品 Consumer Goods
- 14.60% 消費服務 Consumer Services
- 40.80% 金融 Financials
- 6.20% 健康護理 Health Care
- 3.78% 工業 Industrials
- 15.34% 科技 Technology
- 5.93% 其他行業 Other Sectors
- 1.97% 現金及其他 Cash and Others

十大投資項目# | TOP TEN HOLDINGS#

截至2020年7月31日 As at 31 July 2020

	佔資產淨值百分比 % of NAV
台灣積體電路製造股份有限公司 TAIWAN SEMICONDUCTOR MANUFACTURING CO LTD	9.60%
騰訊控股 TENCENT HOLDINGS LTD	9.53%
阿里巴巴集團 ALIBABA GROUP HOLDING LTD	4.87%
美團點評 MEITUAN DIANPING	3.94%
中國平安 PING AN INSURANCE (GROUP) CO OF CHINA LTD H	3.61%
友邦保險 AIA GROUP LTD	3.37%
香港交易所 HONG KONG EXCHANGES & CLEARING LTD	3.15%
金蝶國際軟件集團有限公司 KINGDEE INTERNATIONAL SOFTWARE GROUP LIMITED	2.07%
招商銀行 CHINA MERCHANTS BANK	1.98%
藥明生物 WUXI BIOLOGICS	1.89%

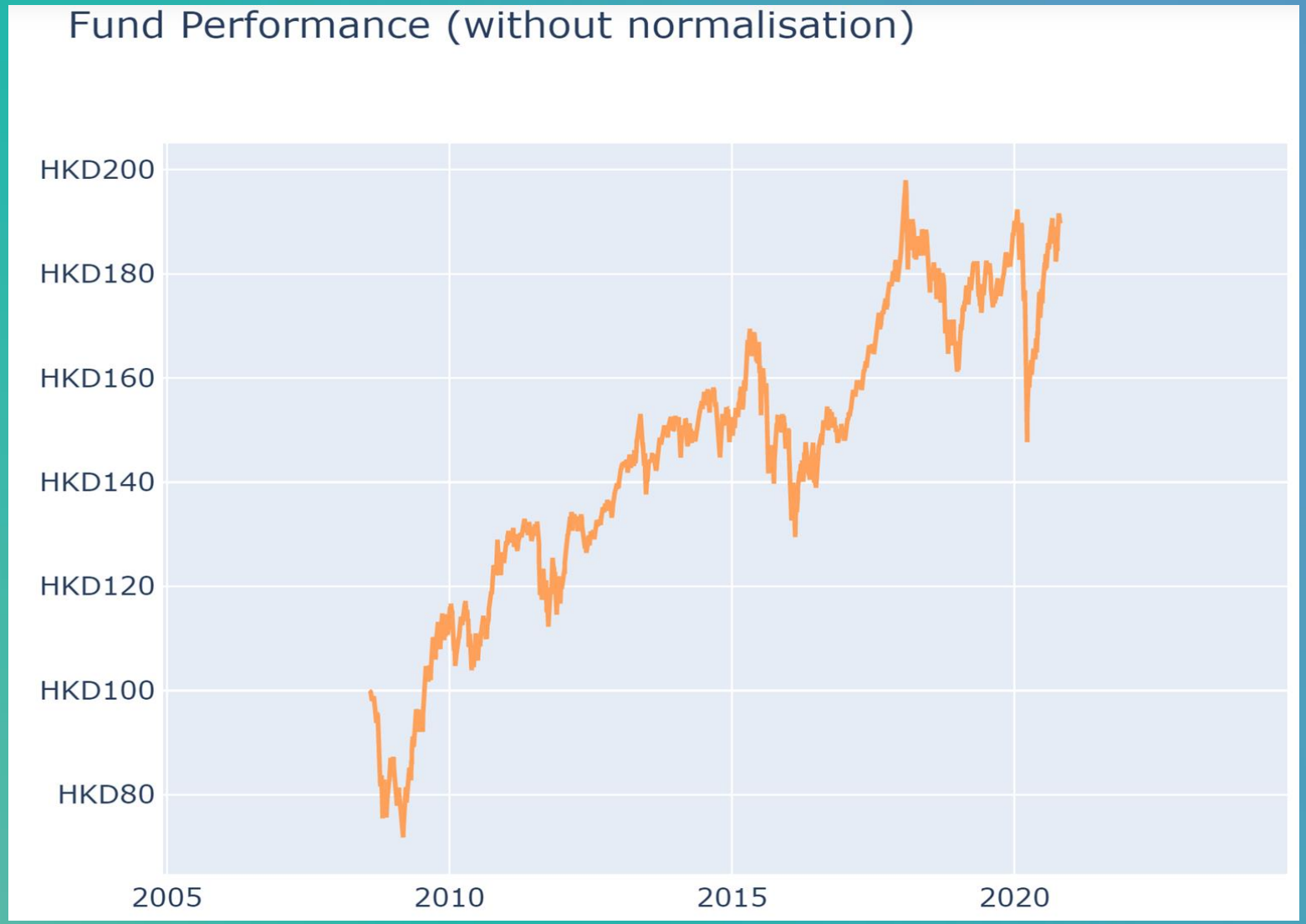
基金表現 | FUND PERFORMANCE

（資產淨值對資產淨值，以港元計算□ NAV to NAV, in HK Dollars□）

	一年 1 Year	三年 3 Years	五年 5 Years	十年 10 Years	成立至今 Since Launch	年初至今 YTD
累積回報 Cumulative Return (%)						
基金 Fund	14.04	8.55	32.20	31.11	260.67	3.49
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	11.26	7.45	17.08	21.53	73.10	12.44
年度化回報 Annualized Return (%)						
基金 Fund	14.04	2.77	5.74	2.75	7.11	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	11.26	2.42	3.20	1.97	2.98	-
曆年回報 Calendar Year Return(%)	**2019**	**2018**	**2017**	**2016**	**2015**	**-**
基金 Fund	12.89	-13.48	35.44	-4.76	-10.75	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	3.77	-10.11	13.66	1.95	-9.13	-

source:
aia mpf prime value choice fund performance review-aug-2020

基金經理精選退休基金


Fund Performance (without normalisation)
HKD200
HKD180
HKD160
HKD140
HKD120
HKD100
HKD80
2005
2010
2015
2020

基金經理精選退休基金

資產分布 | ASSET ALLOCATION

- 14.31% 歐洲股票 Europe Equities
- 22.25% 香港股票 Hong Kong Equities
- 9.20% 日本股票 Japan Equities
- 14.52% 美國股票 United States Equities
- 9.37% 其他股票 Other Equities
- 28.83% 其他債券 Other Bonds
- 1.52% 現金及其他 Cash and Others

十大投資項目# | TOP TEN HOLDINGS#

截至2020年7月31日 As at 31 July 2020

	佔資產淨值百分比 % of NAV
台灣積體電路製造股份有限公司 TAIWAN SEMICONDUCTOR MANUFACTURING CO LTD	9.60%
騰訊控股 TENCENT HOLDINGS LTD	9.53%
阿里巴巴集團 ALIBABA GROUP HOLDING LTD	4.87%
美團點評 MEITUAN DIANPING	3.94%
中國平安 PING AN INSURANCE (GROUP) CO OF CHINA LTD H	3.61%
友邦保險 AIA GROUP LTD	3.37%
香港交易所 HONG KONG EXCHANGES & CLEARING LTD	3.15%
金蝶國際軟件集團有限公司 KINGDEE INTERNATIONAL SOFTWARE GROUP LIMITED	2.07%
招商銀行 CHINA MERCHANTS BANK	1.98%
藥明生物 WUXI BIOLOGICS	1.89%

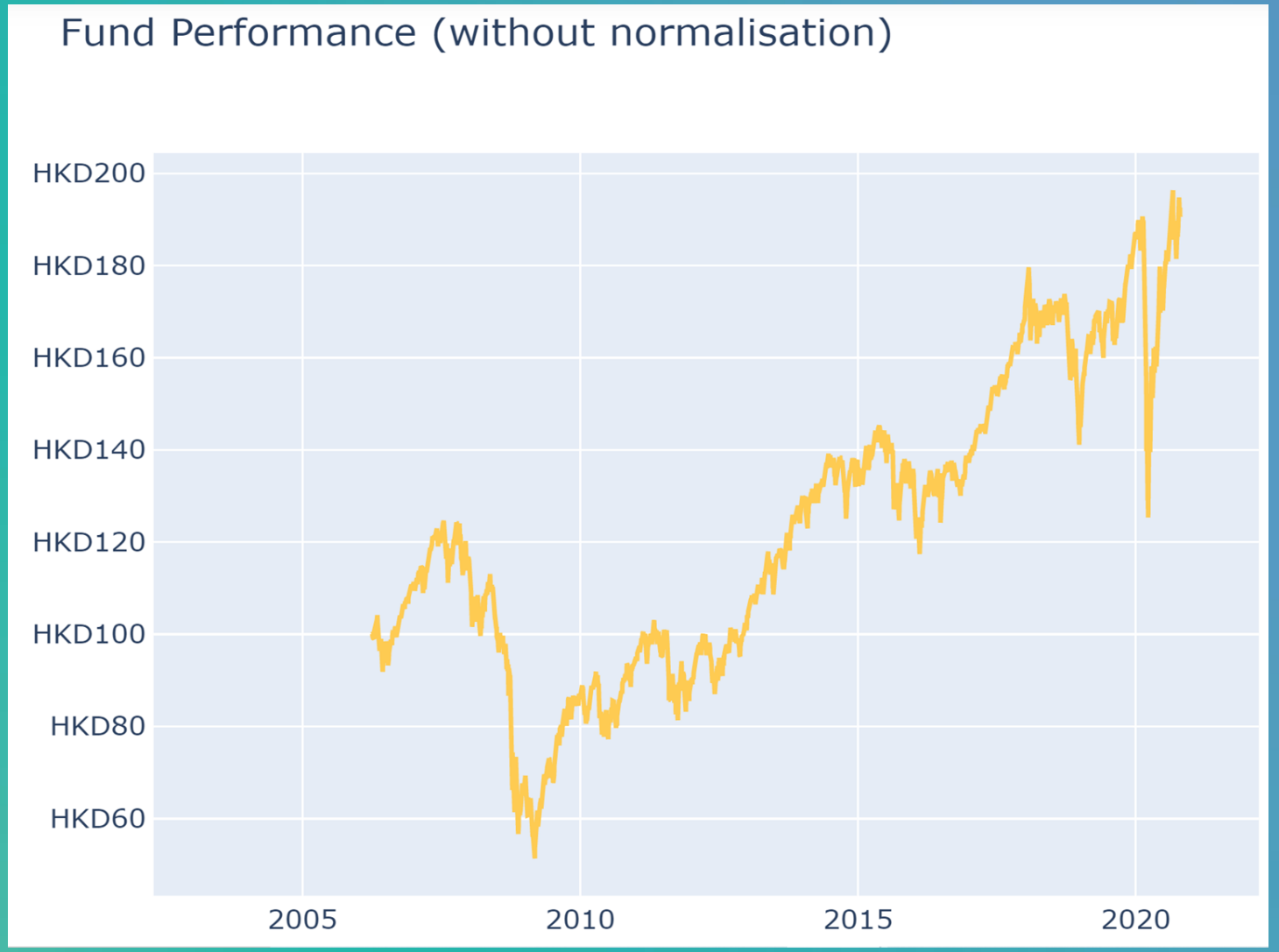
基金表現 | FUND PERFORMANCE

（資產淨值對資產淨值，以港元計算□ NAV to NAV, in HK Dollars□）

	一年 1 Year	三年 3 Years	五年 5 Years	十年 10 Years	成立至今 Since Launch	年初至今 YTD
累積回報 Cumulative Return (%)						
基金 Fund	7.63	9.31	28.76	71.76	88.99	0.11
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	6.93	6.17	13.58	24.37	36.41	8.58
年度化回報 Annualized Return (%)						
基金 Fund	7.63	3.01	5.19	5.56	5.41	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	6.93	2.01	2.58	2.21	2.60	-
曆年回報 Calendar Year Return(%)	**2019**	**2018**	**2017**	**2016**	**2015**	**-**
基金 Fund	15.41	-11.49	24.47	-1.00	-1.27	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	5.45	-8.71	9.97	1.83	-3.33	-

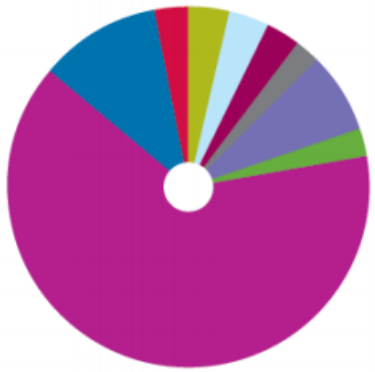
source:
aia mpf prime value choice fund performance review-aug-2020

綠色退休基金


Fund Performance (without normalisation)
HKD200
HKD180
HKD160
HKD140
HKD120
HKD100
HKD80
HKD60
2005
2010
2015
2020

綠色退休基金

資產分布 | ASSET ALLOCATION



- 3.82% 英國 Britain
- 3.62% 加拿大 Canada
- 3.09% 法國 France
- 2.08% 德國 Germany
- 7.26% 日本 Japan
- 2.57% 瑞士 Switzerland
- 63.85% 美國 United States
- 10.89% 其他國家 Other Countries
- 2.82% 現金及其他 Cash and Others

十大投資項目# | TOP TEN HOLDINGS#

截至2020年7月31日 As at 31 July 2020

	佔資產淨值百分比 % of NAV
台灣積體電路製造股份有限公司 TAIWAN SEMICONDUCTOR MANUFACTURING CO LTD	9.60%
騰訊控股 TENCENT HOLDINGS LTD	9.53%
阿里巴巴集團 ALIBABA GROUP HOLDING LTD	4.87%
美團點評 MEITUAN DIANPING	3.94%
中國平安 PING AN INSURANCE (GROUP) CO OF CHINA LTD H	3.61%
友邦保險 AIA GROUP LTD	3.37%
香港交易所 HONG KONG EXCHANGES & CLEARING LTD	3.15%
金蝶國際軟件集團有限公司 KINGDEE INTERNATIONAL SOFTWARE GROUP LIMITED	2.07%
招商銀行 CHINA MERCHANTS BANK	1.98%
藥明生物 WUXI BIOLOGICS	1.89%

基金表現 | FUND PERFORMANCE

（資產淨值對資產淨值，以港元計算□ NAV to NAV, in HK Dollars□）

	一年 1 Year	三年 3 Years	五年 5 Years	十年 10 Years	成立至今 Since Launch	年初至今 YTD
累積回報 Cumulative Return (%)						
基金 Fund	14.98	24.02	46.26	141.06	93.23	3.68
指標 Benchmark[4]	15.52	31.14	64.16	176.41	146.06	4.77
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	11.42	14.95	25.09	50.46	70.06	13.17
年度化回報 Annualized Return (%)						
基金 Fund	14.98	7.44	7.90	9.20	4.67	-
指標 Benchmark[4]	15.52	9.45	10.41	10.69	6.44	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	11.42	4.75	4.58	4.17	3.75	-
曆年回報 Calendar Year Return(%)	**2019**	**2018**	**2017**	**2016**	**2015**	**-**
基金 Fund	26.80	-12.49	22.38	2.08	-1.35	-
平均成本法回報▲ Dollar Cost Averaging Return (%)▲	9.70	-11.49	9.35	3.59	-1.71	-

source:
aia mpf prime value choice fund performance review-aug-2020