

In [21]:

```

import pandas as pd
import numpy as np
import datetime
#from mlxtend.feature_selection import SequentialFeatureSelector
from sklearn.feature_selection import mutual_info_classif
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import mean_squared_error
from sklearn.neighbors import KNeighborsClassifier
from skfeature.function.similarity_based import fisher_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
import pyfolio as pf
from finrl.plot import backtest_plot
from sklearn.svm import SVC

```

In [22]:

```

BIO = pd.read_csv("phase3_data/bio.csv")
BIO = BIO.loc[:, 'Report Date': 's_diff']
BIO['Report Date'] = pd.to_datetime(BIO['Report Date'])
all_date_column = BIO['Report Date']

```

In [23]:

```

BIO['Return'] = BIO['Stock price'].pct_change()
BIO['Stock price'] = BIO['Stock price'].shift(-1)
BIO['Y_boolean'] = BIO['Y_boolean'].shift(-1)
BIO['Return'] = BIO['Return'].shift(-1)
BIO = BIO.dropna()

```

In [24]:

BIO

Out[24]:

	Report Date	Stock price	Y_boolean	(Dividends + Share Buyback) / FCF	Asset Turnover	CapEx / (Depr + Amor)	Current Ratio	Debt Ratio	Dividends / FCF
0	2020-03-02	370.000000	0.0	0.039077	1.000000	0	0.066882	1.0	0
1	2020-03-03	384.929993	1.0	0.039077	1.000000	0	0.066882	1.0	0
2	2020-03-04	386.679993	1.0	0.039077	1.000000	0	0.066882	1.0	0
3	2020-03-05	374.559998	-1.0	0.039077	1.000000	0	0.066882	1.0	0
4	2020-03-06	366.819997	-1.0	0.039077	1.000000	0	0.066882	1.0	0

Report Date	Stock price	Y_boolean	(Dividends + Share Buyback) / FCF	Asset Turnover	(CapEx / (Depr + Amor))	Current Ratio	Debt Ratio	Dividends / FCF
...
778	2022-04-19	562.010010	1.0	0.002722	0.093347	0	0.388057	0.0
779	2022-04-20	545.239990	-1.0	0.002722	0.093347	0	0.388057	0.0
780	2022-04-21	534.289978	-1.0	0.002722	0.093347	0	0.388057	0.0
781	2022-04-22	532.843323	0.0	0.002722	0.093347	0	0.388057	0.0
782	2022-04-23	531.396668	0.0	0.002722	0.093347	0	0.388057	0.0

783 rows × 29 columns

In [5]:

```
# function to select data given date window
def window(start, end, df, all_date_column):
    """
    Given a start and end date, return df with data only
    from that period

    Inputs:
        start/end: start and end dates
            ex: start = '2022-01-23' ('YYYY-MM-DD')
        df: pd dataframe
        date: pd series with dates of all possible dates in data

    Returns: pd dataframe
    """

    date_column = all_date_column[all_date_column.between(start, end, inclusive=True)]
    df = pd.concat([date_column, df], axis=1, join="inner")
    df = df.iloc[:, 1:]

    return df
```

In [6]:

```
def feature_selection_FS(training_dataset, testing_dataset, bar):
    """
    find the important feature using Fishers' score
    """

    X = training_dataset.iloc[:, 3:-1]
    Y_price = training_dataset['Stock price']
    Y_dummy = training_dataset['Y_boolean']
    date_train = training_dataset['Report Date']
    date_test = testing_dataset['Report Date']

    ranks = fisher_score.fisher_score(X.to_numpy(), Y_dummy)
    feat_importances = pd.Series(ranks, training_dataset.columns[3:-1])
```

```
#     feat_importances.plot(kind='barh', color = "teal")
#     plt.show()

feat_selected = feat_importances.to_frame('importance')
# select importance > than bar
feat_selected = feat_selected[feat_selected['importance'] > bar]
select_feature_lst = ['Report Date', 'Stock price', 'Y_boolean'] + list(feat

return training_dataset[select_feature_lst], testing_dataset[select_feature_
```

In [7]:

```
def feature_selection_IG(training_dataset, testing_dataset, bar):
    """
    find the important feature using IG method
    """

    X = training_dataset.iloc[:, 3:-1]
    Y_price = training_dataset['Stock price']
    Y_dummy = training_dataset['Y_boolean']
    date_train = training_dataset['Report Date']
    date_test = testing_dataset['Report Date']
    importances = mutual_info_classif(X, Y_dummy, random_state=1241)
    feat_importances = pd.Series(importances, training_dataset.columns[3:-1])
    #     feat_importances.plot(kind='barh', color = "teal")
    #     plt.show()

    feat_selected = feat_importances.to_frame('importance')
    # select importance > than bar
    feat_selected = feat_selected[feat_selected['importance'] > bar]
    select_feature_lst = ['Report Date', 'Stock price', 'Y_boolean'] + list(feat

    return training_dataset[select_feature_lst], testing_dataset[select_feature_
```

In [8]:

```
def test_feature_select(training_dataset, validate_dataset):
    """
    check the performance of the current selected: use only training dataset acc
    """

    # run baseline logit
    logit = LogisticRegression(solver = 'lbfgs', random_state=1241)
    X = training_dataset.iloc[:, 3:-1]
    Y_dummy = training_dataset['Y_boolean']
    logit.fit(X, Y_dummy)
    Y_dummy_validate = validate_dataset['Y_boolean']
    X_validate = validate_dataset.iloc[:, 3:-1]
    Y_pred = logit.predict(X_validate)
    # pred = lr.predict(testing_dataset)
    #     score = logit.score(testing_dataset, Y_true)

    score = accuracy_score(Y_dummy_validate, Y_pred)
    return score
```

In [9]:

```
# collect all date
date_starts = []
```

```

date = datetime.datetime(2020, 3, 2)
date_starts.append(date)

for idx in range(660):#range(722):
    date += datetime.timedelta(days=1)
    date_starts.append(date)

```

In [12]:

```

# currently using BIO as example
# loop for feature selection and modeling
features_selected = []

for train_date_start in date_starts:
    print(train_date_start)
    train_date_end = train_date_start + datetime.timedelta(days=90)
    validate_date_start = train_date_end + datetime.timedelta(days=1)
    validate_date_end = validate_date_start + datetime.timedelta(days=30)
    test_date_start = validate_date_end + datetime.timedelta(days=1)
    test_date_end = test_date_start

    train_data = window(train_date_start, train_date_end, BIO, all_date_column)
    validate_data = window(validate_date_start, validate_date_end, BIO, all_date_column)
    test_data = window(test_date_start, test_date_end, BIO, all_date_column)

    bars_FS = range(0, 11)
    bars_IG = np.arange(0, 0.03, 0.005)#np.arange(0, 0.1005, 0.005)

    best_accuracy = 0
    best_bar = []
    best_features = []

    for bar_FS in bars_FS:
        for bar_IG in bars_IG:

            try:
                train_BIO_selected, validate_BIO_selected = feature_selection_FS
                train_BIO_selected, validate_BIO_selected = feature_selection_IG
                score = test_feature_select(train_BIO_selected, validate_BIO_selected)
                if score > best_accuracy:
                    best_accuracy = score
                    best_bar = [bar_FS, bar_IG]
                    best_features = train_BIO_selected.iloc[:,3:].columns
            except:
                continue
            features_selected.append(best_features)

```

```

2020-03-02 00:00:00
2020-03-03 00:00:00
2020-03-04 00:00:00
2020-03-05 00:00:00
2020-03-06 00:00:00
2020-03-07 00:00:00
2020-03-08 00:00:00
2020-03-09 00:00:00
2020-03-10 00:00:00
2020-03-11 00:00:00
2020-03-12 00:00:00
2020-03-13 00:00:00

```

2020-03-14 00:00:00
2020-03-15 00:00:00
2020-03-16 00:00:00
2020-03-17 00:00:00
2020-03-18 00:00:00
2020-03-19 00:00:00
2020-03-20 00:00:00
2020-03-21 00:00:00
2020-03-22 00:00:00
2020-03-23 00:00:00
2020-03-24 00:00:00
2020-03-25 00:00:00
2020-03-26 00:00:00
2020-03-27 00:00:00
2020-03-28 00:00:00
2020-03-29 00:00:00
2020-03-30 00:00:00
2020-03-31 00:00:00
2020-04-01 00:00:00
2020-04-02 00:00:00
2020-04-03 00:00:00
2020-04-04 00:00:00
2020-04-05 00:00:00
2020-04-06 00:00:00
2020-04-07 00:00:00
2020-04-08 00:00:00
2020-04-09 00:00:00
2020-04-10 00:00:00
2020-04-11 00:00:00
2020-04-12 00:00:00
2020-04-13 00:00:00
2020-04-14 00:00:00
2020-04-15 00:00:00
2020-04-16 00:00:00
2020-04-17 00:00:00
2020-04-18 00:00:00
2020-04-19 00:00:00
2020-04-20 00:00:00
2020-04-21 00:00:00
2020-04-22 00:00:00
2020-04-23 00:00:00
2020-04-24 00:00:00
2020-04-25 00:00:00
2020-04-26 00:00:00
2020-04-27 00:00:00
2020-04-28 00:00:00
2020-04-29 00:00:00
2020-04-30 00:00:00
2020-05-01 00:00:00
2020-05-02 00:00:00
2020-05-03 00:00:00
2020-05-04 00:00:00
2020-05-05 00:00:00
2020-05-06 00:00:00
2020-05-07 00:00:00
2020-05-08 00:00:00
2020-05-09 00:00:00
2020-05-10 00:00:00
2020-05-11 00:00:00
2020-05-12 00:00:00

2020-05-13 00:00:00
2020-05-14 00:00:00
2020-05-15 00:00:00
2020-05-16 00:00:00
2020-05-17 00:00:00
2020-05-18 00:00:00
2020-05-19 00:00:00
2020-05-20 00:00:00
2020-05-21 00:00:00
2020-05-22 00:00:00
2020-05-23 00:00:00
2020-05-24 00:00:00
2020-05-25 00:00:00
2020-05-26 00:00:00
2020-05-27 00:00:00
2020-05-28 00:00:00
2020-05-29 00:00:00
2020-05-30 00:00:00
2020-05-31 00:00:00
2020-06-01 00:00:00
2020-06-02 00:00:00
2020-06-03 00:00:00
2020-06-04 00:00:00
2020-06-05 00:00:00
2020-06-06 00:00:00
2020-06-07 00:00:00
2020-06-08 00:00:00
2020-06-09 00:00:00
2020-06-10 00:00:00
2020-06-11 00:00:00
2020-06-12 00:00:00
2020-06-13 00:00:00
2020-06-14 00:00:00
2020-06-15 00:00:00
2020-06-16 00:00:00
2020-06-17 00:00:00
2020-06-18 00:00:00
2020-06-19 00:00:00
2020-06-20 00:00:00
2020-06-21 00:00:00
2020-06-22 00:00:00
2020-06-23 00:00:00
2020-06-24 00:00:00
2020-06-25 00:00:00
2020-06-26 00:00:00
2020-06-27 00:00:00
2020-06-28 00:00:00
2020-06-29 00:00:00
2020-06-30 00:00:00
2020-07-01 00:00:00
2020-07-02 00:00:00
2020-07-03 00:00:00
2020-07-04 00:00:00
2020-07-05 00:00:00
2020-07-06 00:00:00
2020-07-07 00:00:00
2020-07-08 00:00:00
2020-07-09 00:00:00
2020-07-10 00:00:00
2020-07-11 00:00:00

2020-07-12 00:00:00
2020-07-13 00:00:00
2020-07-14 00:00:00
2020-07-15 00:00:00
2020-07-16 00:00:00
2020-07-17 00:00:00
2020-07-18 00:00:00
2020-07-19 00:00:00
2020-07-20 00:00:00
2020-07-21 00:00:00
2020-07-22 00:00:00
2020-07-23 00:00:00
2020-07-24 00:00:00
2020-07-25 00:00:00
2020-07-26 00:00:00
2020-07-27 00:00:00
2020-07-28 00:00:00
2020-07-29 00:00:00
2020-07-30 00:00:00
2020-07-31 00:00:00
2020-08-01 00:00:00
2020-08-02 00:00:00
2020-08-03 00:00:00
2020-08-04 00:00:00
2020-08-05 00:00:00
2020-08-06 00:00:00
2020-08-07 00:00:00
2020-08-08 00:00:00
2020-08-09 00:00:00
2020-08-10 00:00:00
2020-08-11 00:00:00
2020-08-12 00:00:00
2020-08-13 00:00:00
2020-08-14 00:00:00
2020-08-15 00:00:00
2020-08-16 00:00:00
2020-08-17 00:00:00
2020-08-18 00:00:00
2020-08-19 00:00:00
2020-08-20 00:00:00
2020-08-21 00:00:00
2020-08-22 00:00:00
2020-08-23 00:00:00
2020-08-24 00:00:00
2020-08-25 00:00:00
2020-08-26 00:00:00
2020-08-27 00:00:00
2020-08-28 00:00:00
2020-08-29 00:00:00
2020-08-30 00:00:00
2020-08-31 00:00:00
2020-09-01 00:00:00
2020-09-02 00:00:00
2020-09-03 00:00:00
2020-09-04 00:00:00
2020-09-05 00:00:00
2020-09-06 00:00:00
2020-09-07 00:00:00
2020-09-08 00:00:00
2020-09-09 00:00:00

2020-09-10 00:00:00
2020-09-11 00:00:00
2020-09-12 00:00:00
2020-09-13 00:00:00
2020-09-14 00:00:00
2020-09-15 00:00:00
2020-09-16 00:00:00
2020-09-17 00:00:00
2020-09-18 00:00:00
2020-09-19 00:00:00
2020-09-20 00:00:00
2020-09-21 00:00:00
2020-09-22 00:00:00
2020-09-23 00:00:00
2020-09-24 00:00:00
2020-09-25 00:00:00
2020-09-26 00:00:00
2020-09-27 00:00:00
2020-09-28 00:00:00
2020-09-29 00:00:00
2020-09-30 00:00:00
2020-10-01 00:00:00
2020-10-02 00:00:00
2020-10-03 00:00:00
2020-10-04 00:00:00
2020-10-05 00:00:00
2020-10-06 00:00:00
2020-10-07 00:00:00
2020-10-08 00:00:00
2020-10-09 00:00:00
2020-10-10 00:00:00
2020-10-11 00:00:00
2020-10-12 00:00:00
2020-10-13 00:00:00
2020-10-14 00:00:00
2020-10-15 00:00:00
2020-10-16 00:00:00
2020-10-17 00:00:00
2020-10-18 00:00:00
2020-10-19 00:00:00
2020-10-20 00:00:00
2020-10-21 00:00:00
2020-10-22 00:00:00
2020-10-23 00:00:00
2020-10-24 00:00:00
2020-10-25 00:00:00
2020-10-26 00:00:00
2020-10-27 00:00:00
2020-10-28 00:00:00
2020-10-29 00:00:00
2020-10-30 00:00:00
2020-10-31 00:00:00
2020-11-01 00:00:00
2020-11-02 00:00:00
2020-11-03 00:00:00
2020-11-04 00:00:00
2020-11-05 00:00:00
2020-11-06 00:00:00
2020-11-07 00:00:00
2020-11-08 00:00:00

2020-11-09 00:00:00
2020-11-10 00:00:00
2020-11-11 00:00:00
2020-11-12 00:00:00
2020-11-13 00:00:00
2020-11-14 00:00:00
2020-11-15 00:00:00
2020-11-16 00:00:00
2020-11-17 00:00:00
2020-11-18 00:00:00
2020-11-19 00:00:00
2020-11-20 00:00:00
2020-11-21 00:00:00
2020-11-22 00:00:00
2020-11-23 00:00:00
2020-11-24 00:00:00
2020-11-25 00:00:00
2020-11-26 00:00:00
2020-11-27 00:00:00
2020-11-28 00:00:00
2020-11-29 00:00:00
2020-11-30 00:00:00
2020-12-01 00:00:00
2020-12-02 00:00:00
2020-12-03 00:00:00
2020-12-04 00:00:00
2020-12-05 00:00:00
2020-12-06 00:00:00
2020-12-07 00:00:00
2020-12-08 00:00:00
2020-12-09 00:00:00
2020-12-10 00:00:00
2020-12-11 00:00:00
2020-12-12 00:00:00
2020-12-13 00:00:00
2020-12-14 00:00:00
2020-12-15 00:00:00
2020-12-16 00:00:00
2020-12-17 00:00:00
2020-12-18 00:00:00
2020-12-19 00:00:00
2020-12-20 00:00:00
2020-12-21 00:00:00
2020-12-22 00:00:00
2020-12-23 00:00:00
2020-12-24 00:00:00
2020-12-25 00:00:00
2020-12-26 00:00:00
2020-12-27 00:00:00
2020-12-28 00:00:00
2020-12-29 00:00:00
2020-12-30 00:00:00
2020-12-31 00:00:00
2021-01-01 00:00:00
2021-01-02 00:00:00
2021-01-03 00:00:00
2021-01-04 00:00:00
2021-01-05 00:00:00
2021-01-06 00:00:00
2021-01-07 00:00:00

2021-01-08 00:00:00
2021-01-09 00:00:00
2021-01-10 00:00:00
2021-01-11 00:00:00
2021-01-12 00:00:00
2021-01-13 00:00:00
2021-01-14 00:00:00
2021-01-15 00:00:00
2021-01-16 00:00:00
2021-01-17 00:00:00
2021-01-18 00:00:00
2021-01-19 00:00:00
2021-01-20 00:00:00
2021-01-21 00:00:00
2021-01-22 00:00:00
2021-01-23 00:00:00
2021-01-24 00:00:00
2021-01-25 00:00:00
2021-01-26 00:00:00
2021-01-27 00:00:00
2021-01-28 00:00:00
2021-01-29 00:00:00
2021-01-30 00:00:00
2021-01-31 00:00:00
2021-02-01 00:00:00
2021-02-02 00:00:00
2021-02-03 00:00:00
2021-02-04 00:00:00
2021-02-05 00:00:00
2021-02-06 00:00:00
2021-02-07 00:00:00
2021-02-08 00:00:00
2021-02-09 00:00:00
2021-02-10 00:00:00
2021-02-11 00:00:00
2021-02-12 00:00:00
2021-02-13 00:00:00
2021-02-14 00:00:00
2021-02-15 00:00:00
2021-02-16 00:00:00
2021-02-17 00:00:00
2021-02-18 00:00:00
2021-02-19 00:00:00
2021-02-20 00:00:00
2021-02-21 00:00:00
2021-02-22 00:00:00
2021-02-23 00:00:00
2021-02-24 00:00:00
2021-02-25 00:00:00
2021-02-26 00:00:00
2021-02-27 00:00:00
2021-02-28 00:00:00
2021-03-01 00:00:00
2021-03-02 00:00:00
2021-03-03 00:00:00
2021-03-04 00:00:00
2021-03-05 00:00:00
2021-03-06 00:00:00
2021-03-07 00:00:00
2021-03-08 00:00:00

2021-03-09 00:00:00
2021-03-10 00:00:00
2021-03-11 00:00:00
2021-03-12 00:00:00
2021-03-13 00:00:00
2021-03-14 00:00:00
2021-03-15 00:00:00
2021-03-16 00:00:00
2021-03-17 00:00:00
2021-03-18 00:00:00
2021-03-19 00:00:00
2021-03-20 00:00:00
2021-03-21 00:00:00
2021-03-22 00:00:00
2021-03-23 00:00:00
2021-03-24 00:00:00
2021-03-25 00:00:00
2021-03-26 00:00:00
2021-03-27 00:00:00
2021-03-28 00:00:00
2021-03-29 00:00:00
2021-03-30 00:00:00
2021-03-31 00:00:00
2021-04-01 00:00:00
2021-04-02 00:00:00
2021-04-03 00:00:00
2021-04-04 00:00:00
2021-04-05 00:00:00
2021-04-06 00:00:00
2021-04-07 00:00:00
2021-04-08 00:00:00
2021-04-09 00:00:00
2021-04-10 00:00:00
2021-04-11 00:00:00
2021-04-12 00:00:00
2021-04-13 00:00:00
2021-04-14 00:00:00
2021-04-15 00:00:00
2021-04-16 00:00:00
2021-04-17 00:00:00
2021-04-18 00:00:00
2021-04-19 00:00:00
2021-04-20 00:00:00
2021-04-21 00:00:00
2021-04-22 00:00:00
2021-04-23 00:00:00
2021-04-24 00:00:00
2021-04-25 00:00:00
2021-04-26 00:00:00
2021-04-27 00:00:00
2021-04-28 00:00:00
2021-04-29 00:00:00
2021-04-30 00:00:00
2021-05-01 00:00:00
2021-05-02 00:00:00
2021-05-03 00:00:00
2021-05-04 00:00:00
2021-05-05 00:00:00
2021-05-06 00:00:00
2021-05-07 00:00:00

2021-05-08 00:00:00
2021-05-09 00:00:00
2021-05-10 00:00:00
2021-05-11 00:00:00
2021-05-12 00:00:00
2021-05-13 00:00:00
2021-05-14 00:00:00
2021-05-15 00:00:00
2021-05-16 00:00:00
2021-05-17 00:00:00
2021-05-18 00:00:00
2021-05-19 00:00:00
2021-05-20 00:00:00
2021-05-21 00:00:00
2021-05-22 00:00:00
2021-05-23 00:00:00
2021-05-24 00:00:00
2021-05-25 00:00:00
2021-05-26 00:00:00
2021-05-27 00:00:00
2021-05-28 00:00:00
2021-05-29 00:00:00
2021-05-30 00:00:00
2021-05-31 00:00:00
2021-06-01 00:00:00
2021-06-02 00:00:00
2021-06-03 00:00:00
2021-06-04 00:00:00
2021-06-05 00:00:00
2021-06-06 00:00:00
2021-06-07 00:00:00
2021-06-08 00:00:00
2021-06-09 00:00:00
2021-06-10 00:00:00
2021-06-11 00:00:00
2021-06-12 00:00:00
2021-06-13 00:00:00
2021-06-14 00:00:00
2021-06-15 00:00:00
2021-06-16 00:00:00
2021-06-17 00:00:00
2021-06-18 00:00:00
2021-06-19 00:00:00
2021-06-20 00:00:00
2021-06-21 00:00:00
2021-06-22 00:00:00
2021-06-23 00:00:00
2021-06-24 00:00:00
2021-06-25 00:00:00
2021-06-26 00:00:00
2021-06-27 00:00:00
2021-06-28 00:00:00
2021-06-29 00:00:00
2021-06-30 00:00:00
2021-07-01 00:00:00
2021-07-02 00:00:00
2021-07-03 00:00:00
2021-07-04 00:00:00
2021-07-05 00:00:00
2021-07-06 00:00:00

2021-07-07 00:00:00
2021-07-08 00:00:00
2021-07-09 00:00:00
2021-07-10 00:00:00
2021-07-11 00:00:00
2021-07-12 00:00:00
2021-07-13 00:00:00
2021-07-14 00:00:00
2021-07-15 00:00:00
2021-07-16 00:00:00
2021-07-17 00:00:00
2021-07-18 00:00:00
2021-07-19 00:00:00
2021-07-20 00:00:00
2021-07-21 00:00:00
2021-07-22 00:00:00
2021-07-23 00:00:00
2021-07-24 00:00:00
2021-07-25 00:00:00
2021-07-26 00:00:00
2021-07-27 00:00:00
2021-07-28 00:00:00
2021-07-29 00:00:00
2021-07-30 00:00:00
2021-07-31 00:00:00
2021-08-01 00:00:00
2021-08-02 00:00:00
2021-08-03 00:00:00
2021-08-04 00:00:00
2021-08-05 00:00:00
2021-08-06 00:00:00
2021-08-07 00:00:00
2021-08-08 00:00:00
2021-08-09 00:00:00
2021-08-10 00:00:00
2021-08-11 00:00:00
2021-08-12 00:00:00
2021-08-13 00:00:00
2021-08-14 00:00:00
2021-08-15 00:00:00
2021-08-16 00:00:00
2021-08-17 00:00:00
2021-08-18 00:00:00
2021-08-19 00:00:00
2021-08-20 00:00:00
2021-08-21 00:00:00
2021-08-22 00:00:00
2021-08-23 00:00:00
2021-08-24 00:00:00
2021-08-25 00:00:00
2021-08-26 00:00:00
2021-08-27 00:00:00
2021-08-28 00:00:00
2021-08-29 00:00:00
2021-08-30 00:00:00
2021-08-31 00:00:00
2021-09-01 00:00:00
2021-09-02 00:00:00
2021-09-03 00:00:00
2021-09-04 00:00:00

2021-09-05 00:00:00
2021-09-06 00:00:00
2021-09-07 00:00:00
2021-09-08 00:00:00
2021-09-09 00:00:00
2021-09-10 00:00:00
2021-09-11 00:00:00
2021-09-12 00:00:00
2021-09-13 00:00:00
2021-09-14 00:00:00
2021-09-15 00:00:00
2021-09-16 00:00:00
2021-09-17 00:00:00
2021-09-18 00:00:00
2021-09-19 00:00:00
2021-09-20 00:00:00
2021-09-21 00:00:00
2021-09-22 00:00:00
2021-09-23 00:00:00
2021-09-24 00:00:00
2021-09-25 00:00:00
2021-09-26 00:00:00
2021-09-27 00:00:00
2021-09-28 00:00:00
2021-09-29 00:00:00
2021-09-30 00:00:00
2021-10-01 00:00:00
2021-10-02 00:00:00
2021-10-03 00:00:00
2021-10-04 00:00:00
2021-10-05 00:00:00
2021-10-06 00:00:00
2021-10-07 00:00:00
2021-10-08 00:00:00
2021-10-09 00:00:00
2021-10-10 00:00:00
2021-10-11 00:00:00
2021-10-12 00:00:00
2021-10-13 00:00:00
2021-10-14 00:00:00
2021-10-15 00:00:00
2021-10-16 00:00:00
2021-10-17 00:00:00
2021-10-18 00:00:00
2021-10-19 00:00:00
2021-10-20 00:00:00
2021-10-21 00:00:00
2021-10-22 00:00:00
2021-10-23 00:00:00
2021-10-24 00:00:00
2021-10-25 00:00:00
2021-10-26 00:00:00
2021-10-27 00:00:00
2021-10-28 00:00:00
2021-10-29 00:00:00
2021-10-30 00:00:00
2021-10-31 00:00:00
2021-11-01 00:00:00
2021-11-02 00:00:00
2021-11-03 00:00:00

```
2021-11-04 00:00:00
2021-11-05 00:00:00
2021-11-06 00:00:00
2021-11-07 00:00:00
2021-11-08 00:00:00
2021-11-09 00:00:00
2021-11-10 00:00:00
2021-11-11 00:00:00
2021-11-12 00:00:00
2021-11-13 00:00:00
2021-11-14 00:00:00
2021-11-15 00:00:00
2021-11-16 00:00:00
2021-11-17 00:00:00
2021-11-18 00:00:00
2021-11-19 00:00:00
2021-11-20 00:00:00
2021-11-21 00:00:00
2021-11-22 00:00:00
2021-11-23 00:00:00
2021-11-24 00:00:00
2021-11-25 00:00:00
2021-11-26 00:00:00
2021-11-27 00:00:00
2021-11-28 00:00:00
2021-11-29 00:00:00
2021-11-30 00:00:00
2021-12-01 00:00:00
2021-12-02 00:00:00
2021-12-03 00:00:00
2021-12-04 00:00:00
2021-12-05 00:00:00
2021-12-06 00:00:00
2021-12-07 00:00:00
2021-12-08 00:00:00
2021-12-09 00:00:00
2021-12-10 00:00:00
2021-12-11 00:00:00
2021-12-12 00:00:00
2021-12-13 00:00:00
2021-12-14 00:00:00
2021-12-15 00:00:00
2021-12-16 00:00:00
2021-12-17 00:00:00
2021-12-18 00:00:00
2021-12-19 00:00:00
2021-12-20 00:00:00
2021-12-21 00:00:00
2021-12-22 00:00:00
```

In [13]:

```
list(features_selected[0])
```

Out[13]:

```
['Current Ratio',
 'Debt Ratio',
 'Dividends / FCF',
 'Dummy_Dividends',
 'Gross Profit Margin',
 'Quick Ratio',
 'R&D / Gross Profit',
```

```
'R&D / Revenue',
'Return on Assets',
'Return on Equity',
'Share Buyback / FCF',
'tweet_polarity',
'tweet_subjectivity',
'RSI_ratio',
'MACD']
```

In [14]: `features_selected[0]`

Out[14]: `Index(['Current Ratio', 'Debt Ratio', 'Dividends / FCF', 'Dummy_Dividends',
 'Gross Profit Margin', 'Quick Ratio', 'R&D / Gross Profit',
 'R&D / Revenue', 'Return on Assets', 'Return on Equity',
 'Share Buyback / FCF', 'tweet_polarity', 'tweet_subjectivity',
 'RSI_ratio', 'MACD'],
 dtype='object')`

In [28]: `import pickle`

In [29]: `with open('BIO features', 'wb') as f:
 pickle.dump(features_selected, f)`

In [30]: `with open('BIO features', 'rb') as f:
 features_selected = pickle.load(f)`

In [31]: `models = {'random forest': RandomForestClassifier(random_state=0,
 n_jobs=-1,
 n_estimators=10,
 class_weight='balanced'),
 'logistic regression': LogisticRegression(max_iter=1000,
 penalty='l2',
 tol=0.001,
 solver='newton-cg'),
 'SVC': SVC(kernel='poly'),
 'KNN': KNeighborsClassifier(n_neighbors=5),
 'gradient boosting': GradientBoostingClassifier(max_depth= 10, random_s
 }`

In [32]: `results = {}
for model_name, model in models.items():
 if model_name not in results:
 dates = []
 predictions = []
 truth = []
 returns = []
 for train_date_start, features in zip(date_starts, features_selected):
 train_date_end = train_date_start + datetime.timedelta(days=90)
 validate_date_start = train_date_end + datetime.timedelta(days=1)
 validate_date_end = validate_date_start + datetime.timedelta(days=30)
 test_date_start = validate_date_end + datetime.timedelta(days=1)
 test_date_end = test_date_start`

```

train_data = window(train_date_start, train_date_end, BIO, all_date_
validate_data = window(validate_date_start, validate_date_end, BIO,
test_data = window(test_date_start, test_date_end, BIO, all_date_col

train = pd.concat([train_data, validate_data])

model.fit(train[features], train['Y_boolean'])
predictions.append(model.predict(test_data[features])[0])
dates.append(test_data['Report Date'].values[0])
truth.append(test_data['Y_boolean'].values[0])
returns.append(test_data['Return'].values[0])
results[model_name] = pd.DataFrame({'date':dates,
                                     'prediction': predictions,
                                     'truth': truth,
                                     'returns': returns})
results[model_name]['portfolio returns'] = results[model_name]['returns']
results[model_name]['portfolio cumulative returns'] = np.cumprod(1 + res

```

In [33]:

```

for model_name in results.keys():
    acc = sum(results[model_name]['prediction'] == results[model_name]['truth'])
    print(f'{model_name} score:{round(acc,3)}')

```

```

random forest score:0.495
logistic regression score:0.396
SVC score:0.421
KNN score:0.41
gradient boosting score:0.452

```

In [34]:

```

def backtest(df):
    df_account_value = df[['date', 'portfolio cumulative returns']]
    df_account_value.columns = ['date', 'account_value']
    backtest_plot(df_account_value,
                  baseline_ticker='SPY',
                  baseline_start = df_account_value.date.iloc[0],
                  baseline_end = df_account_value.date.iloc[-1])

```

In [35]:

```
backtest(results['gradient boosting'])
```

[*****100%*****] 1 of 1 completed

Shape of DataFrame: (456, 8)

Start date 2020-07-02

End date 2022-04-23

Total months	31
--------------	----

Backtest

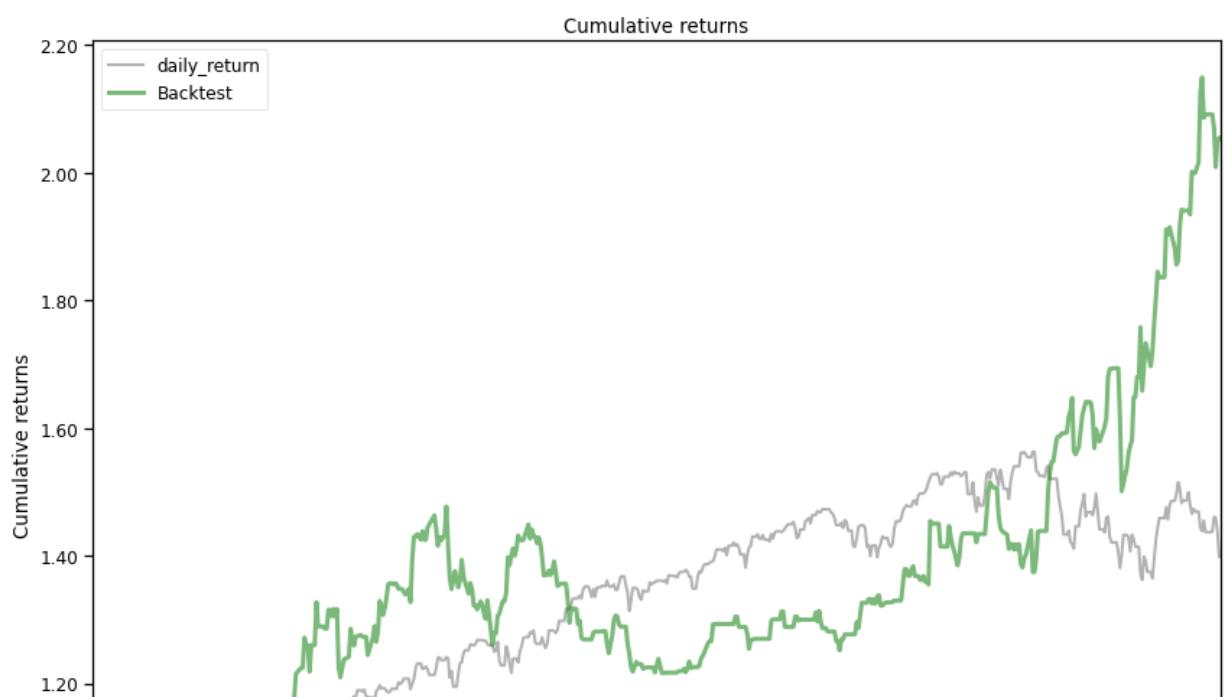
Annual return	31.592%
Cumulative returns	105.466%
Annual volatility	21.483%
Sharpe ratio	1.39
Calmar ratio	1.79
Stability	0.67

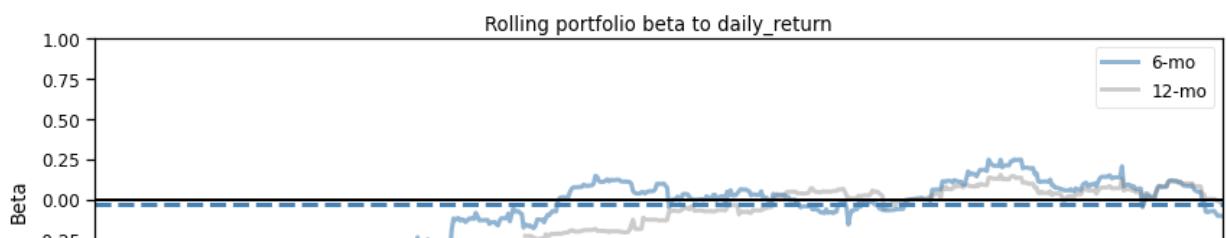
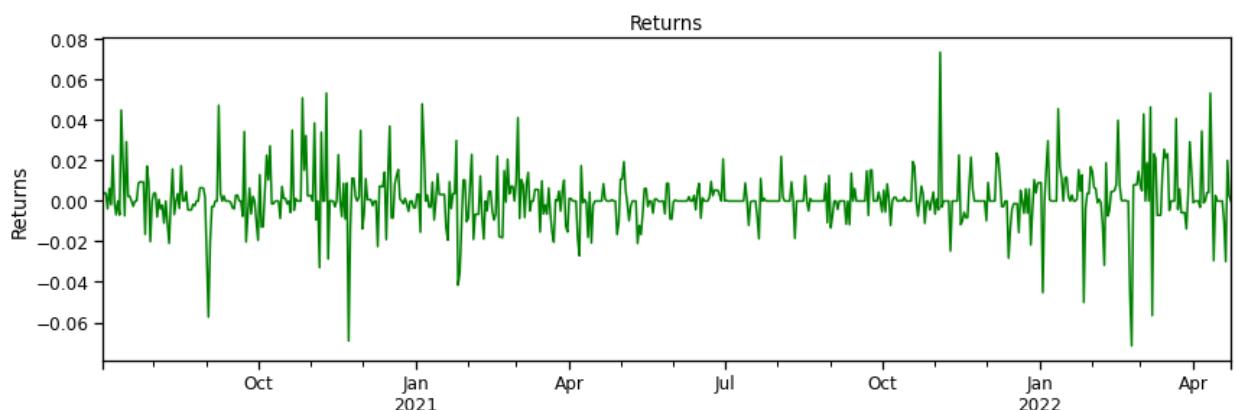
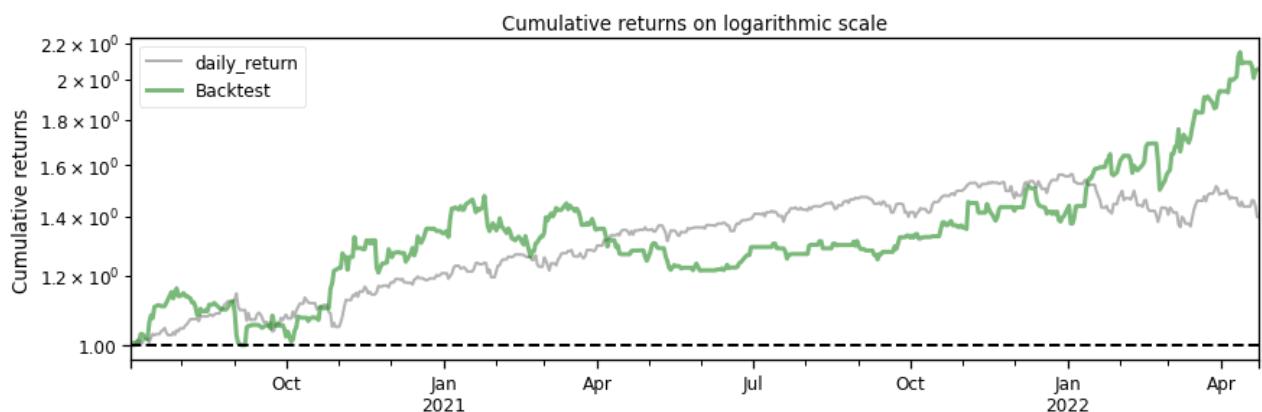
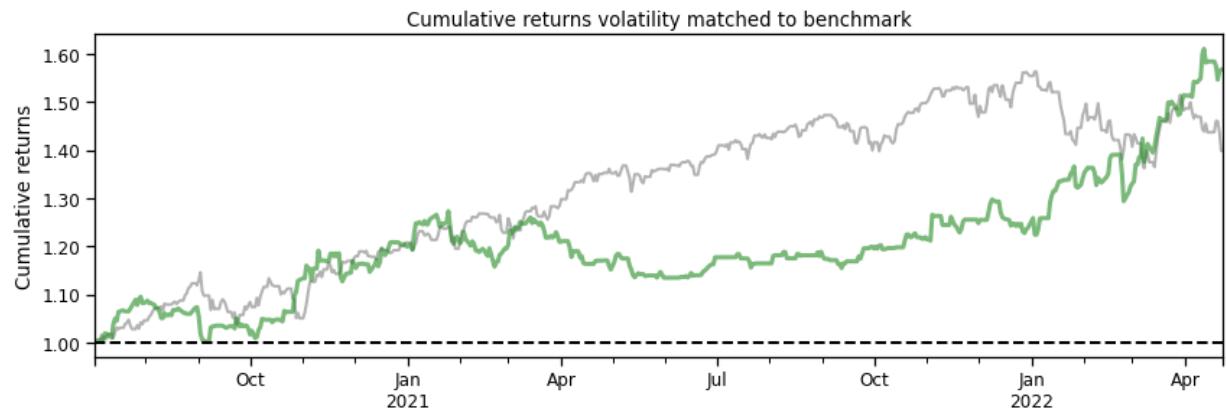
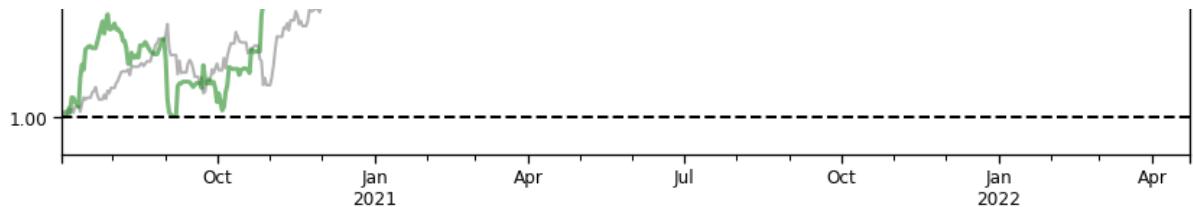
Start date 2020-07-02**End date** 2022-04-23**Total months** 31**Backtest**

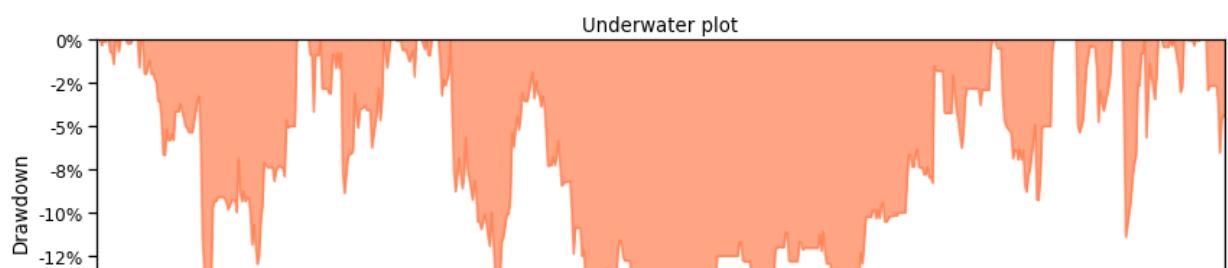
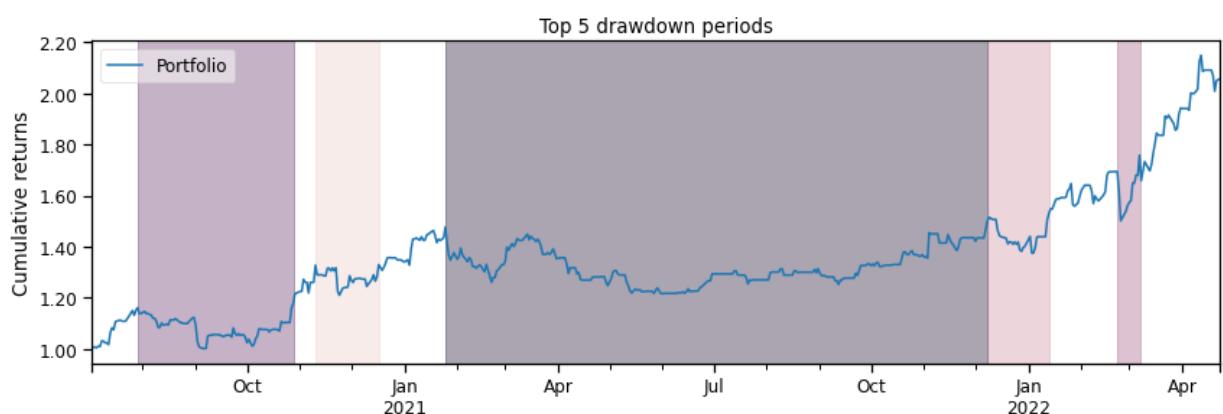
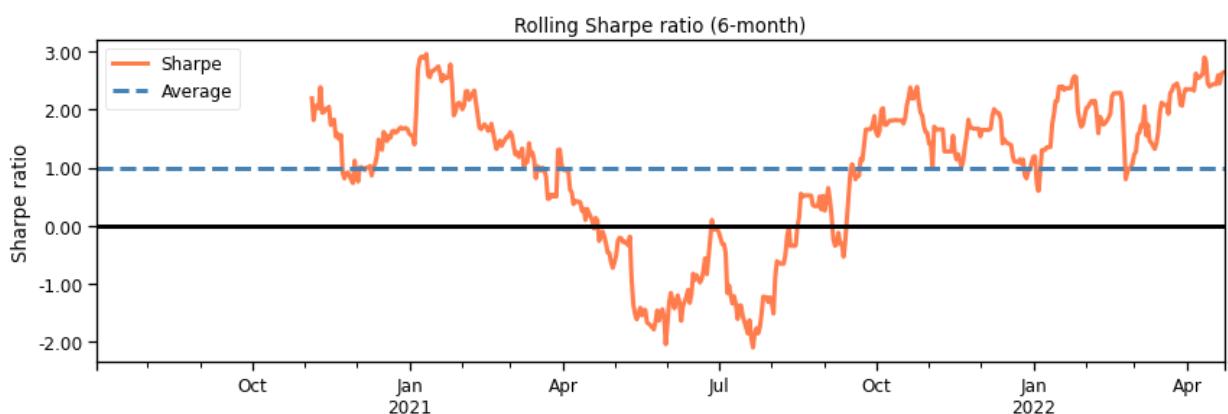
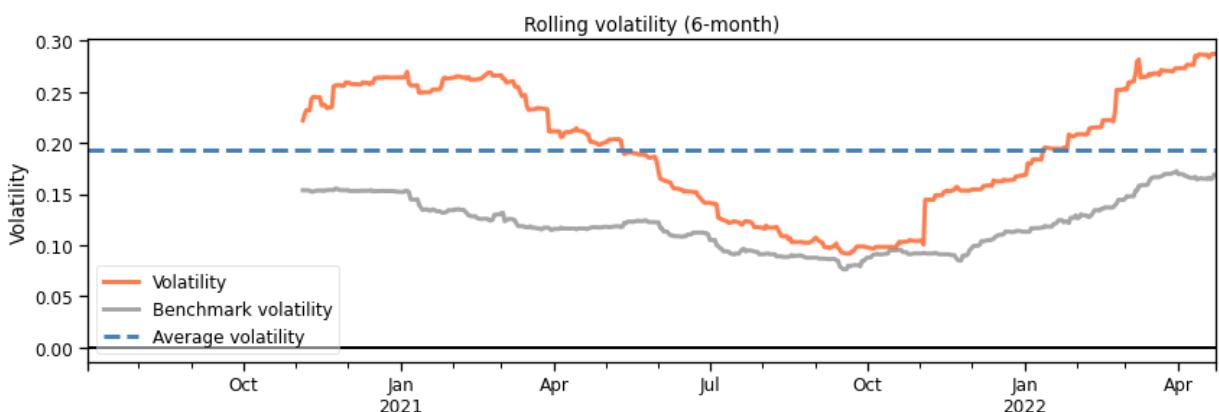
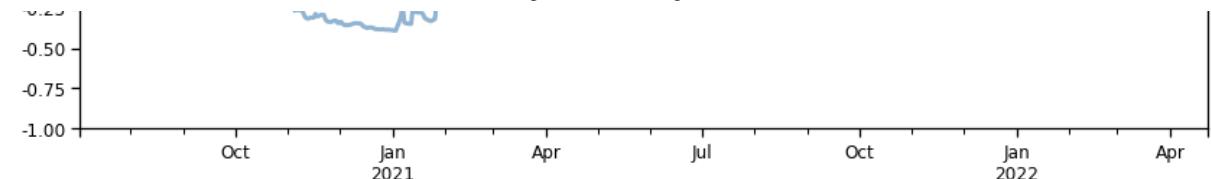
Max drawdown	-17.662%
Omega ratio	1.36
Sortino ratio	2.13
Skew	NaN
Kurtosis	NaN
Tail ratio	1.22
Daily value at risk	-2.588%
Alpha	0.37
Beta	-0.11

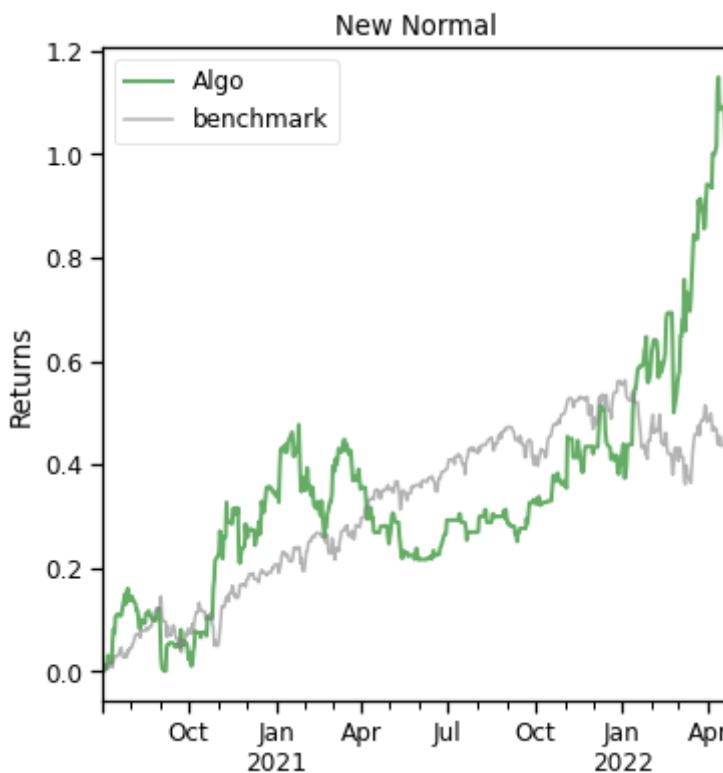
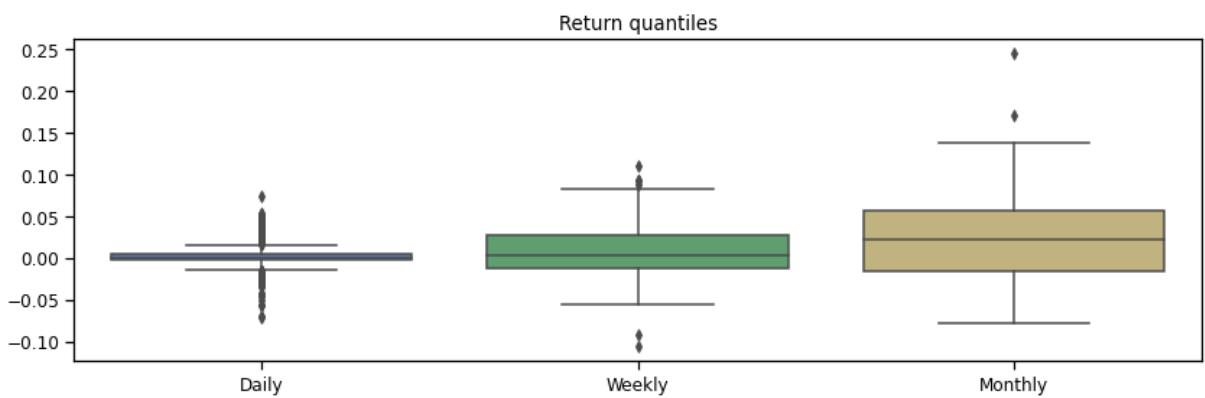
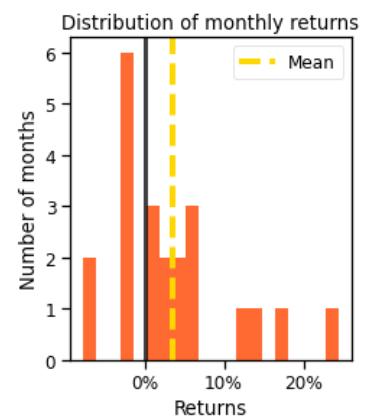
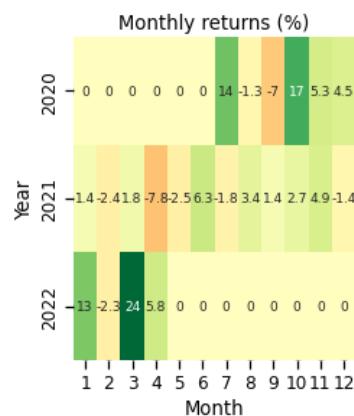
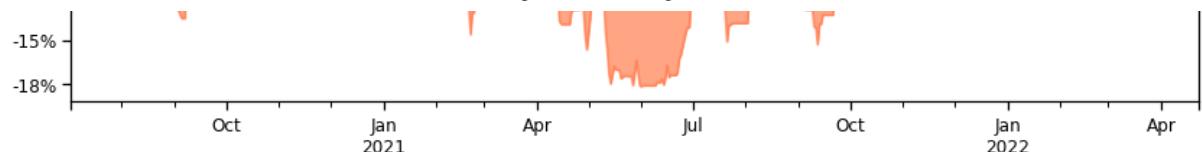
Worst drawdown periods	Net drawdown in %	Peak date	Valley date	Recovery date	Duration
0	17.66	2021-01-25	2021-06-01	2021-12-08	228
1	13.75	2020-07-29	2020-09-05	2020-10-28	66
2	11.38	2022-02-22	2022-02-24	2022-03-07	10
3	9.25	2021-12-09	2022-01-03	2022-01-13	26
4	8.86	2020-11-10	2020-11-24	2020-12-17	28

Stress Events	mean	min	max
New Normal	0.12%	-7.16%	7.36%









In [36]: `backtest(results['random forest'])`

[*****100%*****] 1 of 1 completed

Shape of DataFrame: (456, 8)

Start date 2020-07-02

End date 2022-04-23

Total months 31

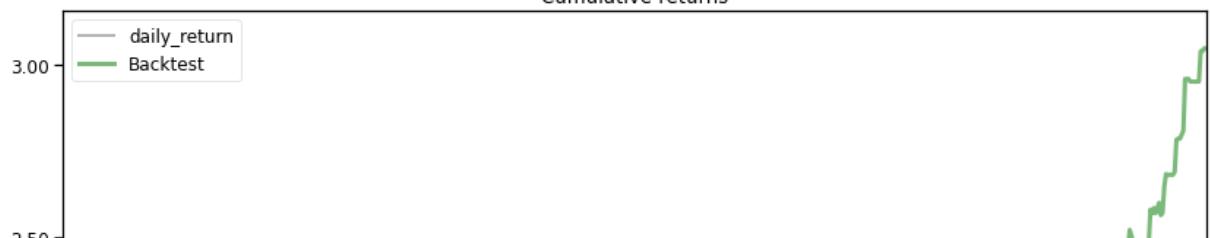
Backtest

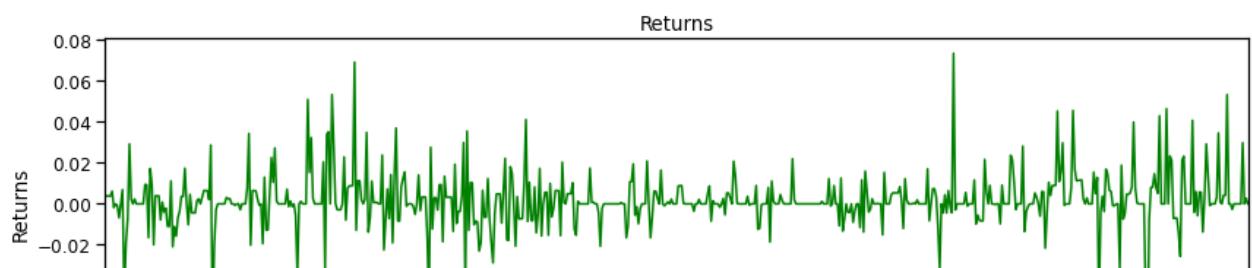
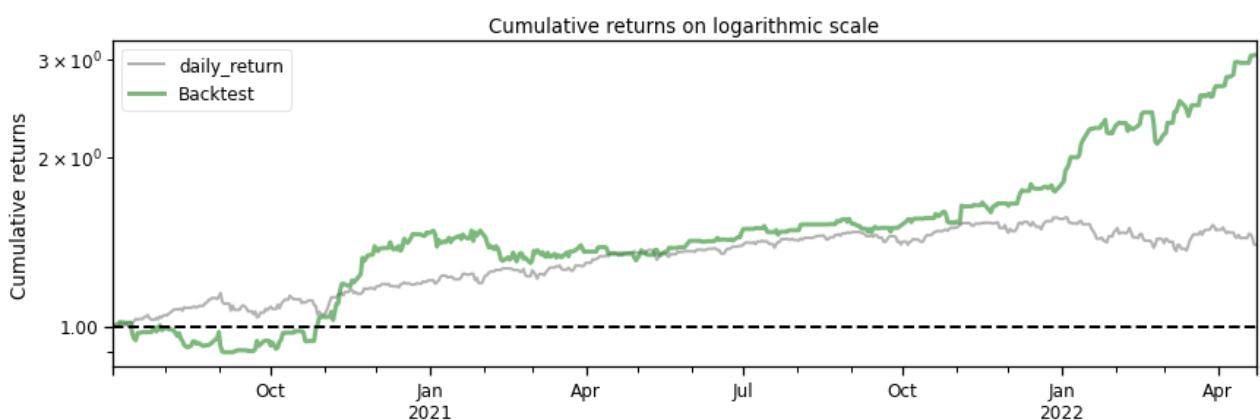
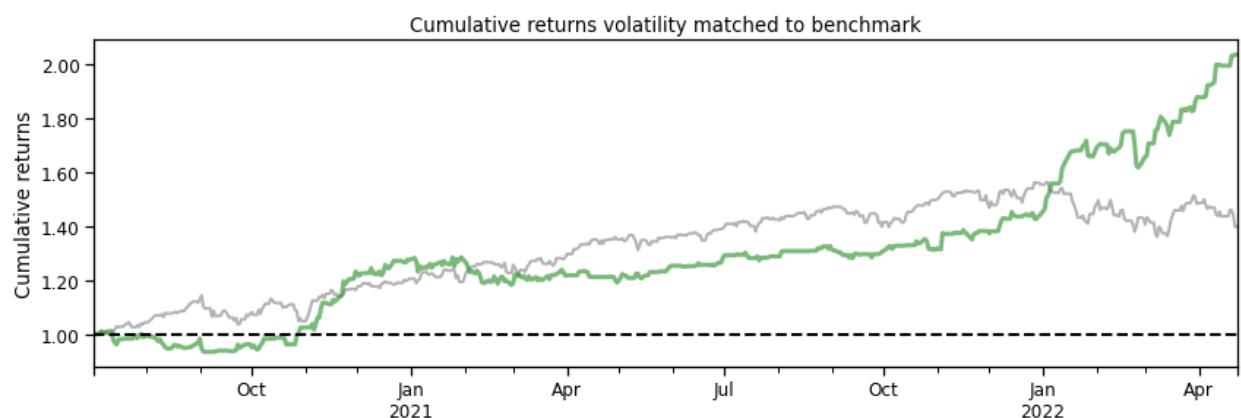
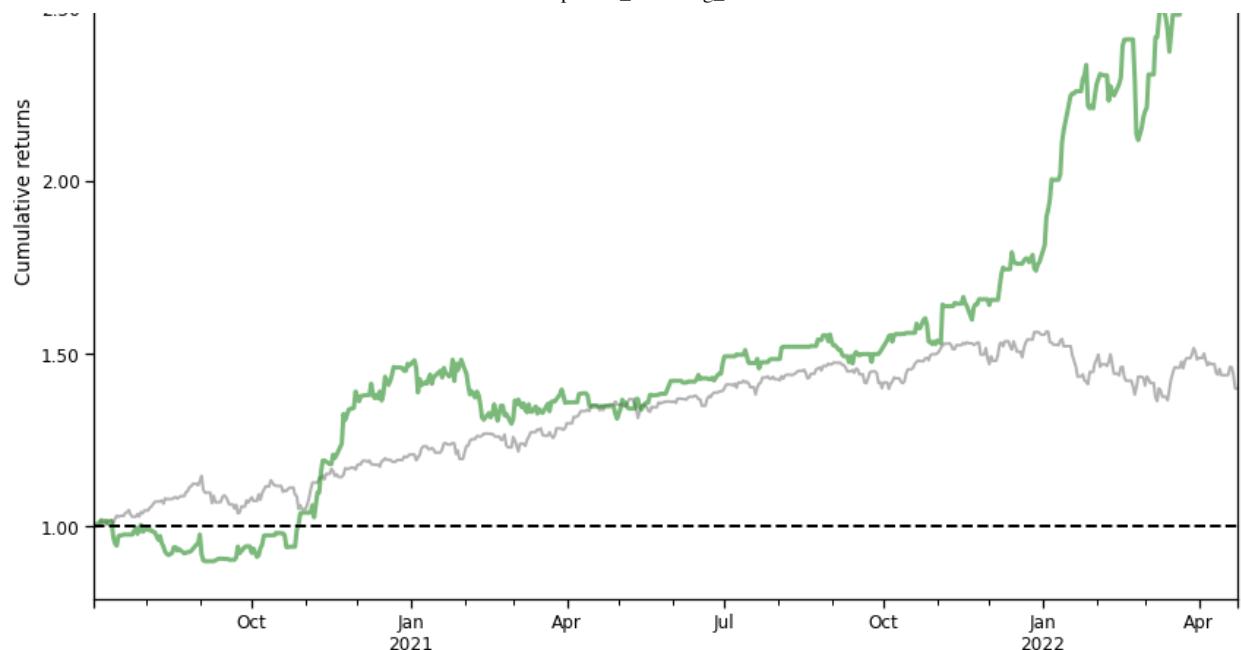
Annual return	52.916%
Cumulative returns	204.666%
Annual volatility	20.756%
Sharpe ratio	2.15
Calmar ratio	4.24
Stability	0.86
Max drawdown	-12.485%
Omega ratio	1.62
Sortino ratio	3.54
Skew	NaN
Kurtosis	NaN
Tail ratio	1.42
Daily value at risk	-2.438%
Alpha	0.59
Beta	-0.13

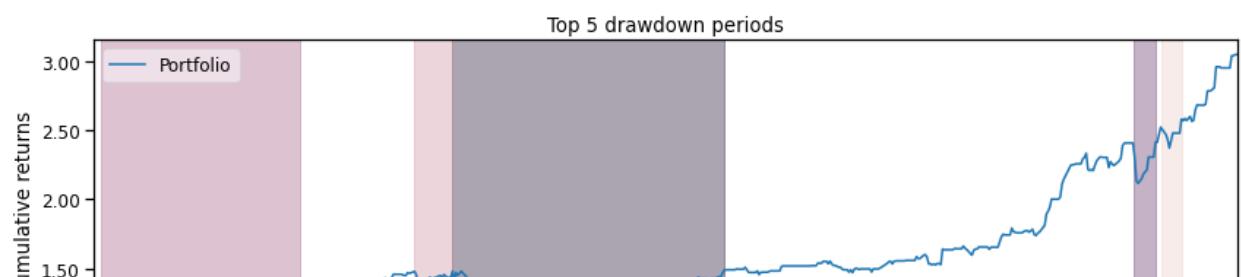
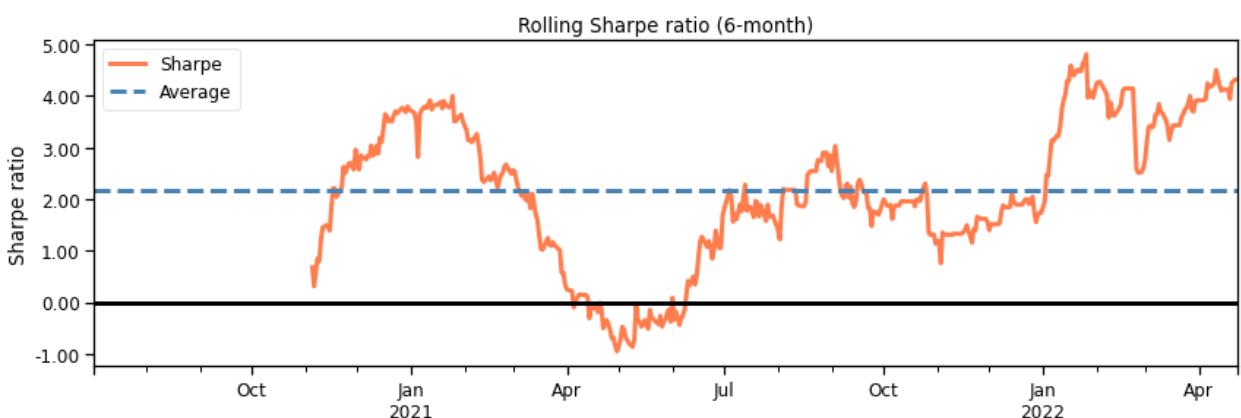
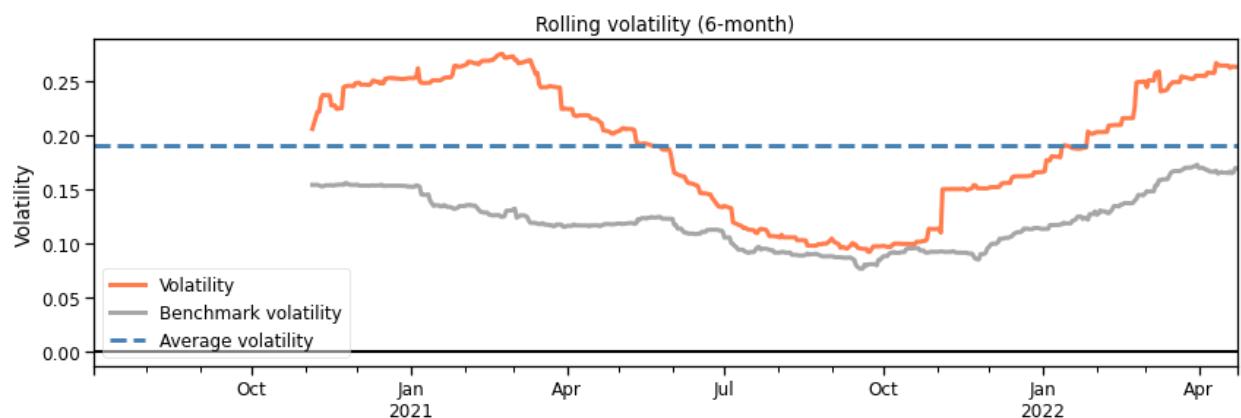
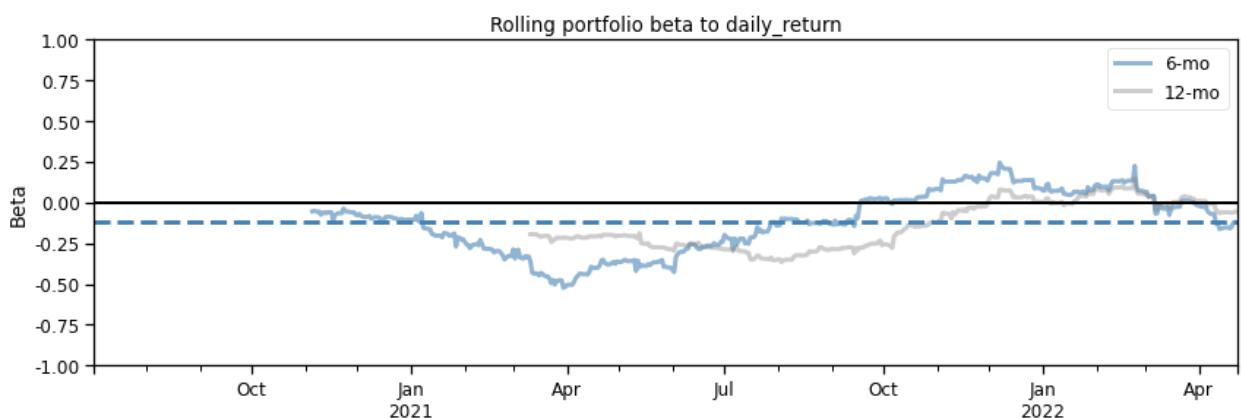
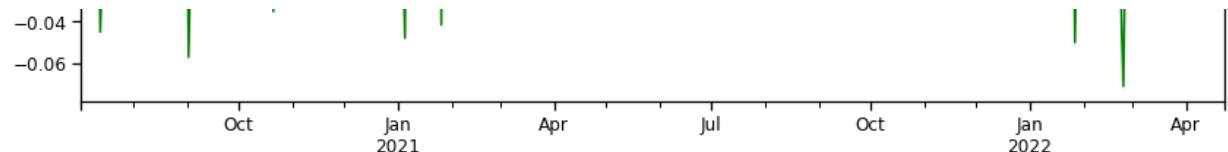
Worst drawdown periods	Net drawdown in %	Peak date	Valley date	Recovery date	Duration
0	12.48	2021-01-25	2021-02-28	2021-07-01	114
1	12.09	2022-02-22	2022-02-25	2022-03-07	10
2	11.66	2020-07-06	2020-09-04	2020-10-29	84
3	6.27	2021-01-03	2021-01-05	2021-01-25	16
4	5.94	2022-03-10	2022-03-15	2022-03-22	9

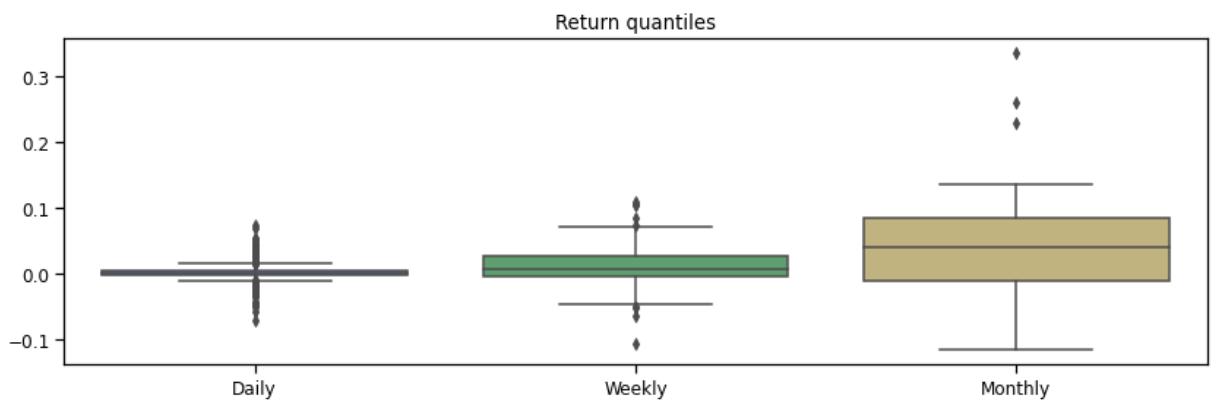
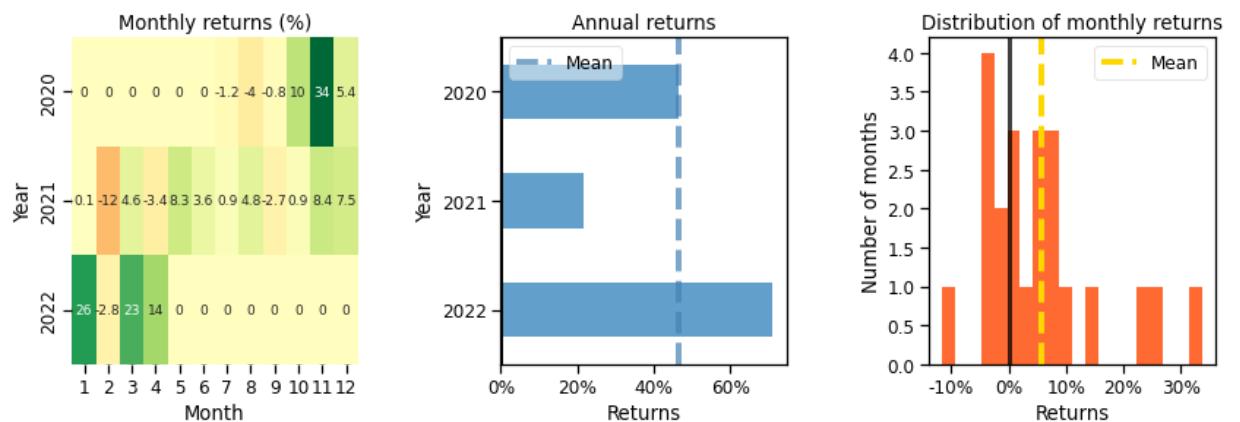
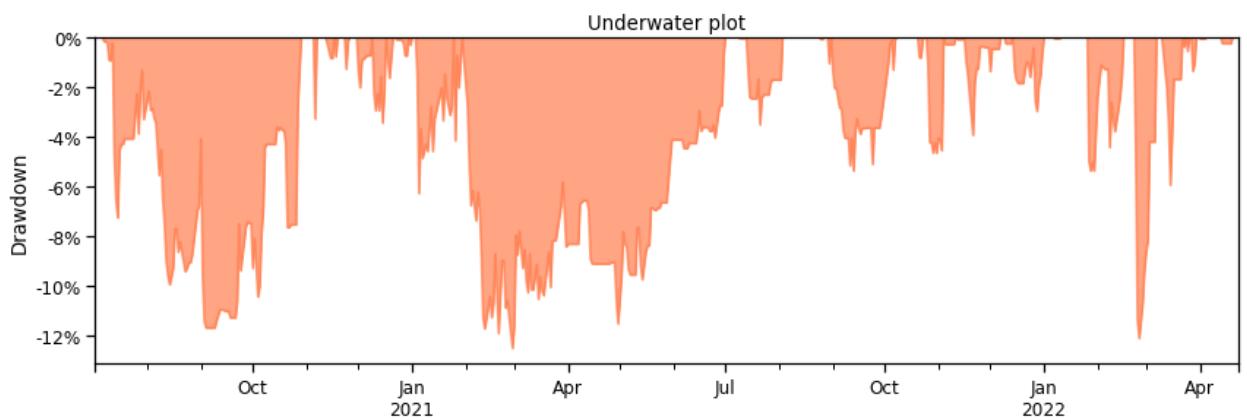
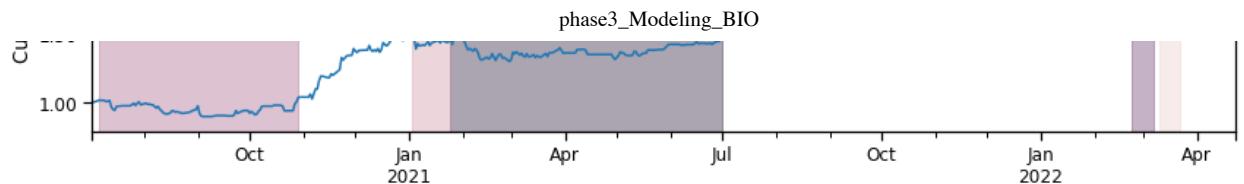
Stress Events	mean	min	max
New Normal	0.18%	-7.16%	7.36%

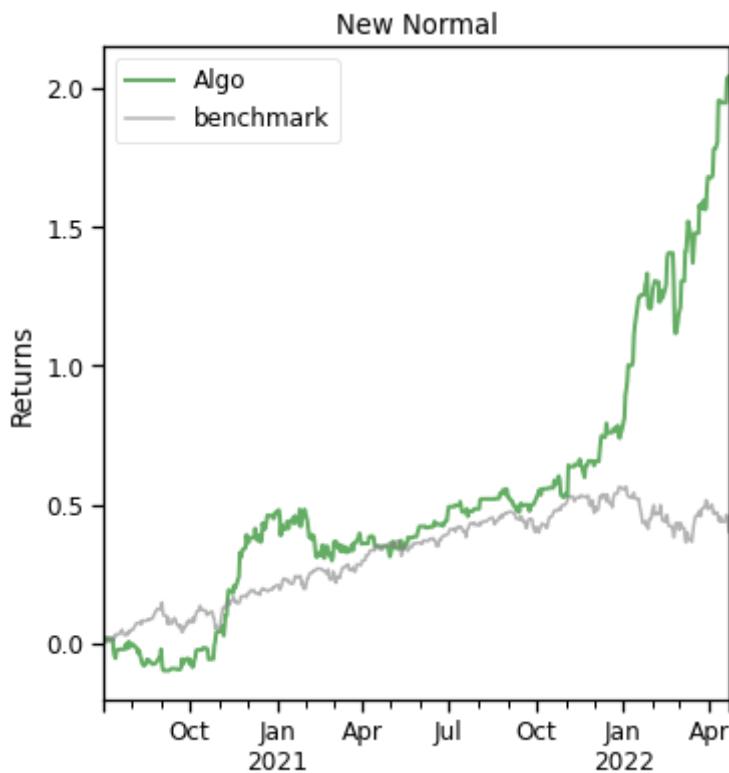
Cumulative returns











In [37]:

```
backtest(results['logistic regression'])
```

```
[*****100*****] 1 of 1 completed
Shape of DataFrame: (456, 8)
```

Start date 2020-07-02

End date 2022-04-23

Total months 31

Backtest

Annual return	10.159%
Cumulative returns	28.889%
Annual volatility	21.923%
Sharpe ratio	0.55
Calmar ratio	0.48
Stability	0.53
Max drawdown	-21.376%
Omega ratio	1.13
Sortino ratio	0.80
Skew	NaN
Kurtosis	NaN
Tail ratio	1.05
Daily value at risk	-2.714%
Alpha	0.13

Start date 2020-07-02

End date 2022-04-23

Total months 31

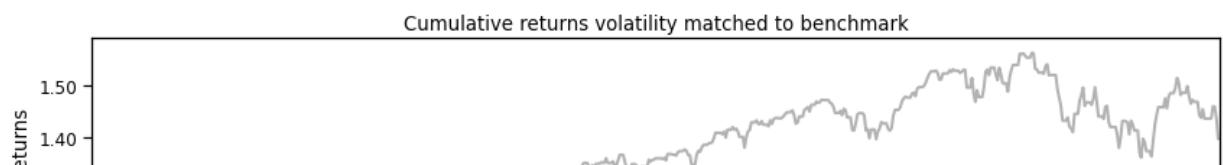
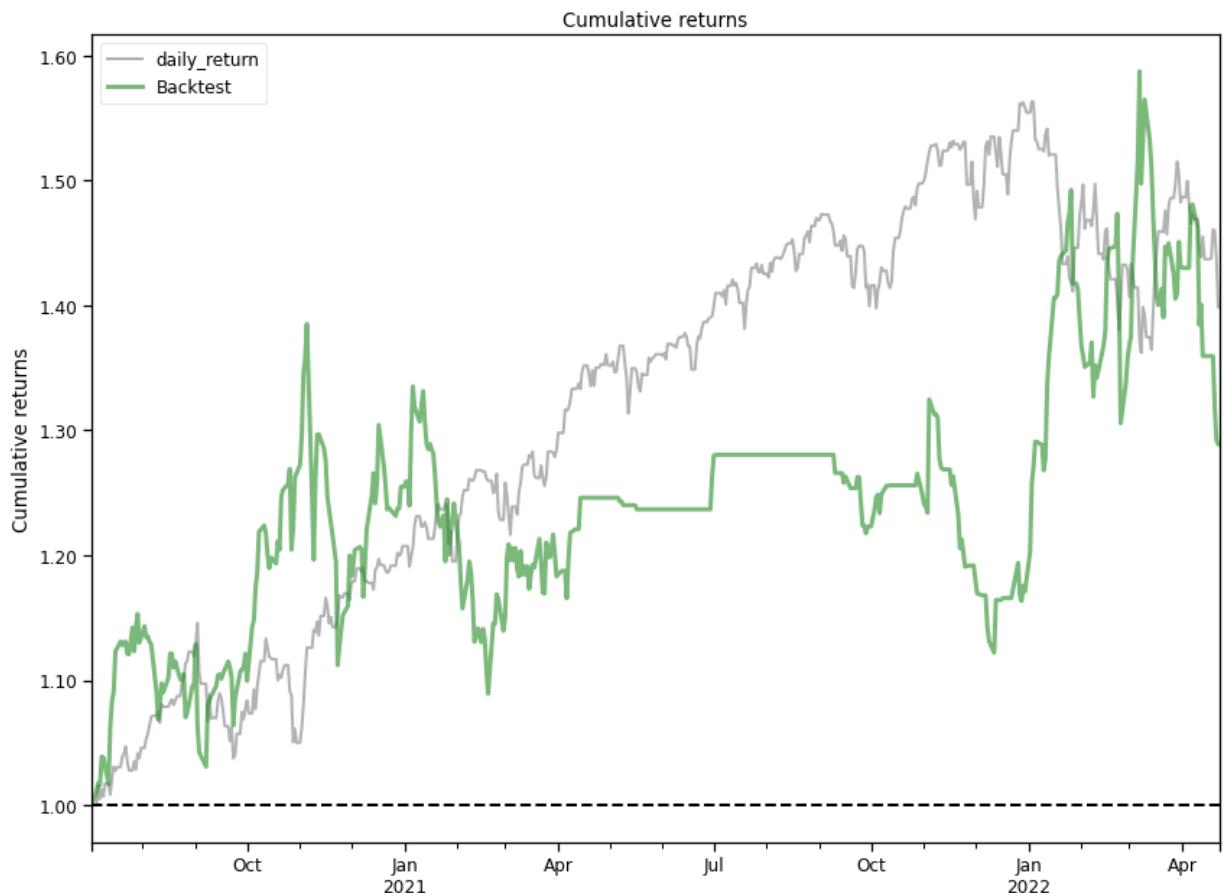
Backtest

Beta -0.02

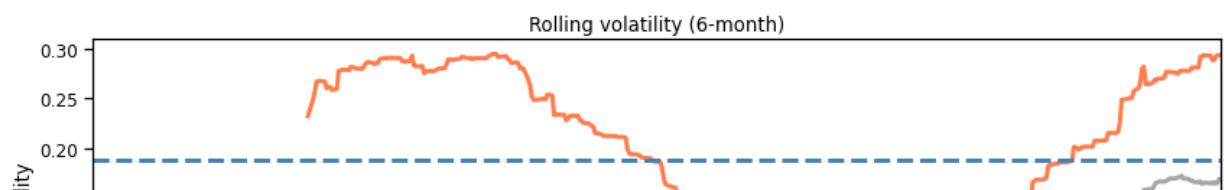
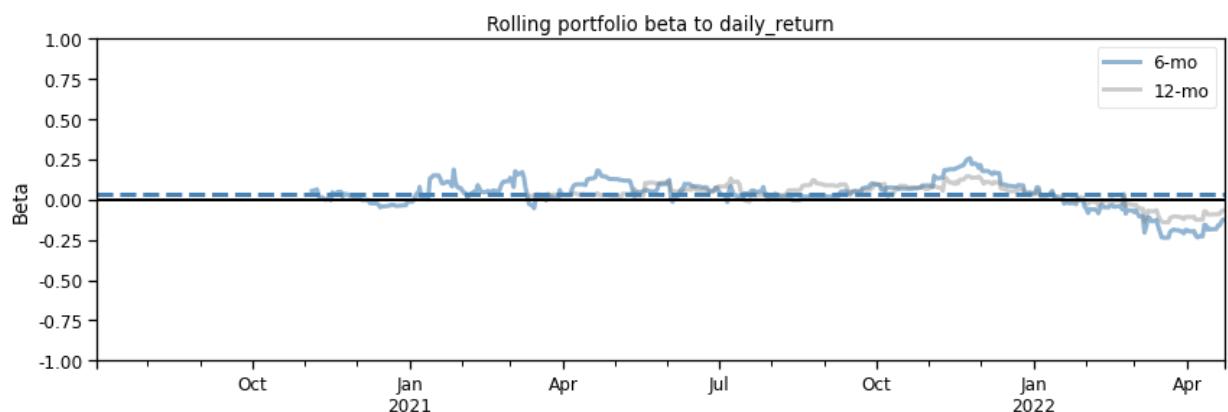
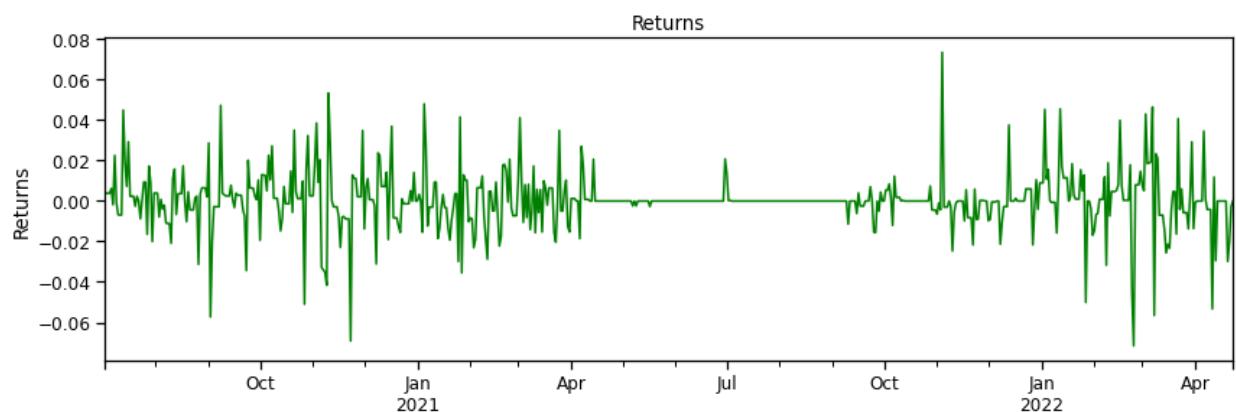
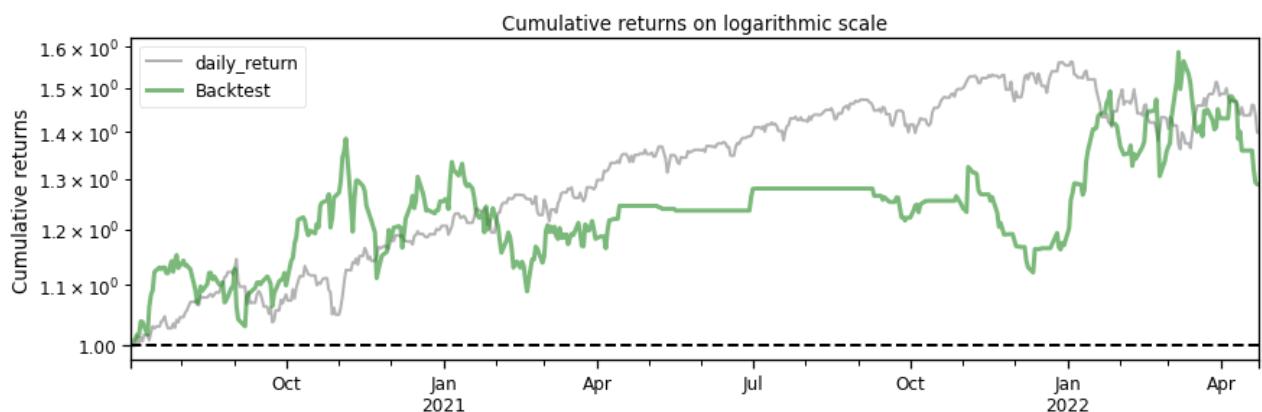
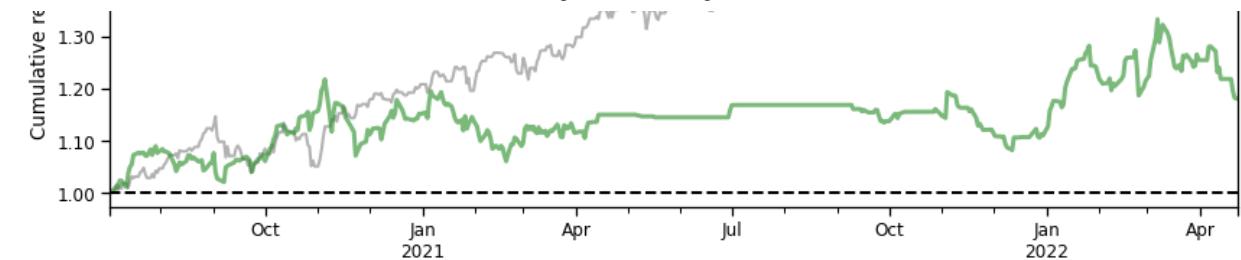
Worst drawdown periods	Net drawdown in %	Peak date	Valley date	Recovery date	Duration
0	21.38	2020-11-05	2021-02-19	2022-01-15	312
1	18.80	2022-03-07	2022-04-22	NaN	NaN
2	12.52	2022-01-26	2022-02-24	2022-03-06	28
3	10.60	2020-07-29	2020-09-07	2020-10-06	50
4	5.10	2020-10-26	2020-10-27	2020-11-01	5

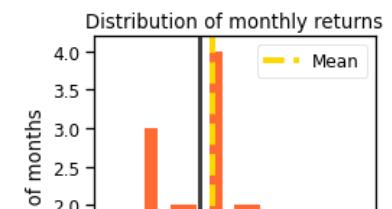
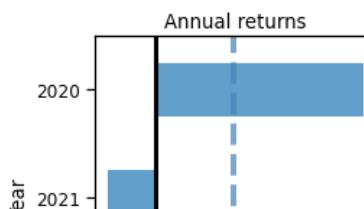
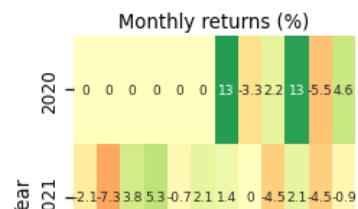
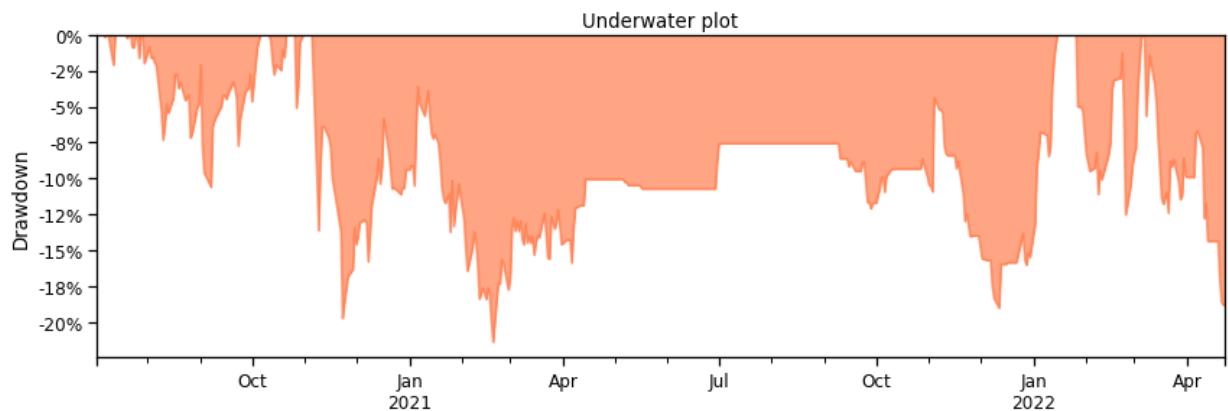
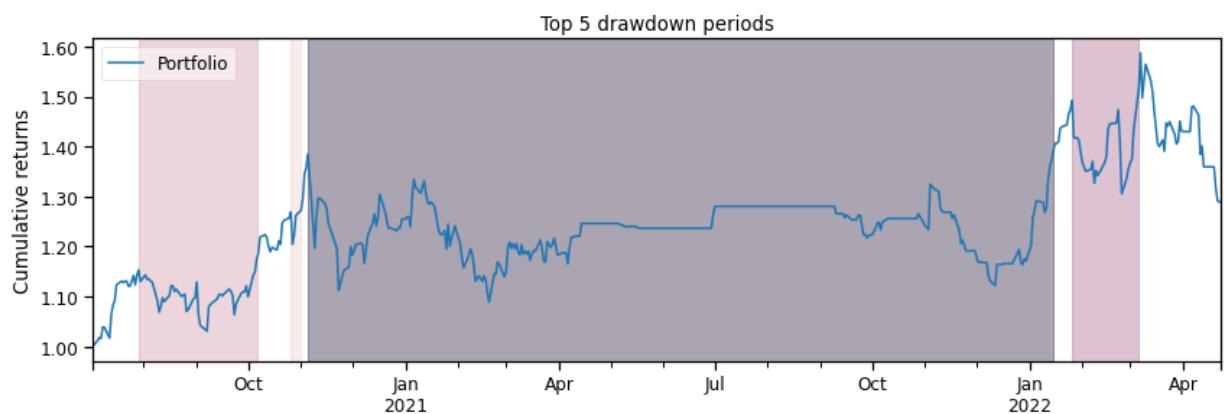
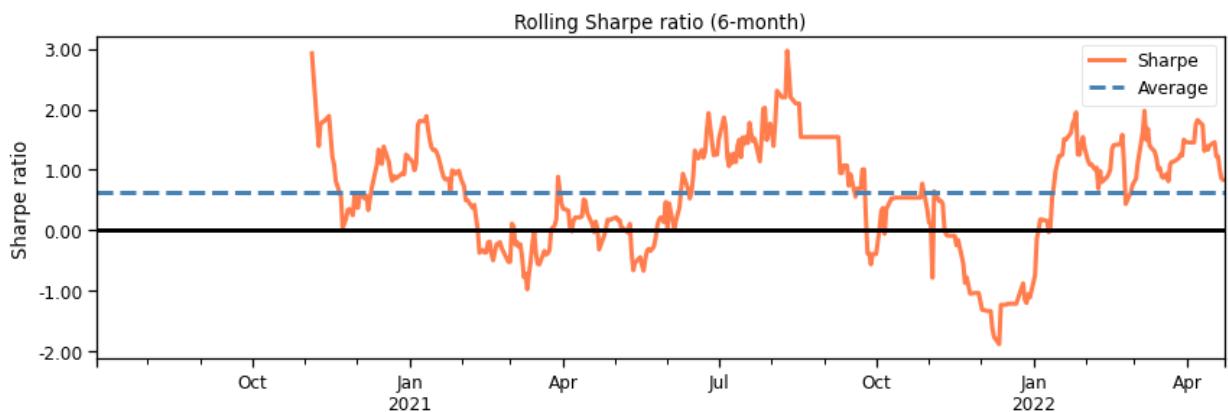
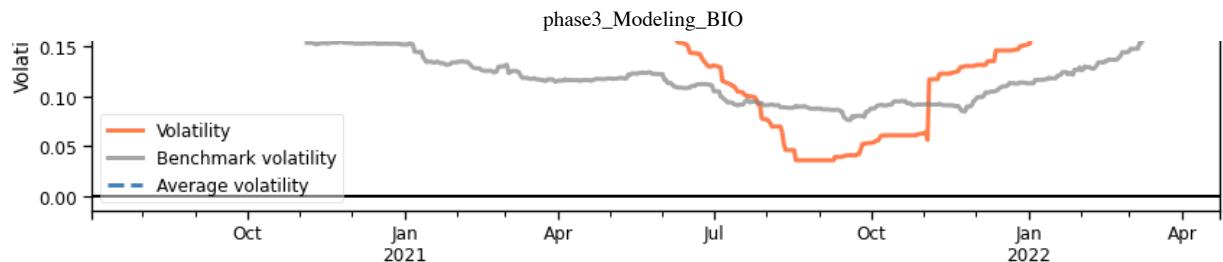
Stress Events mean min max

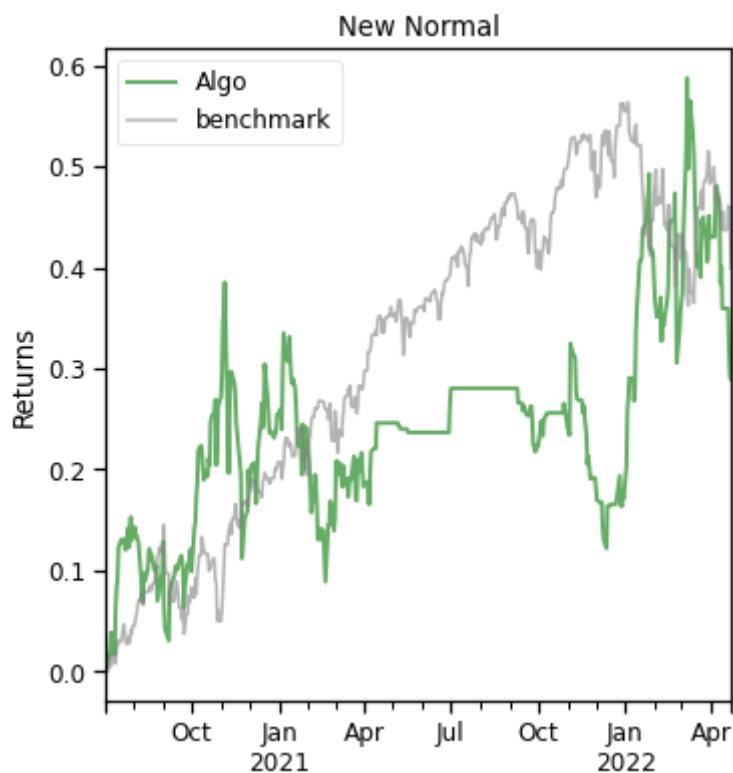
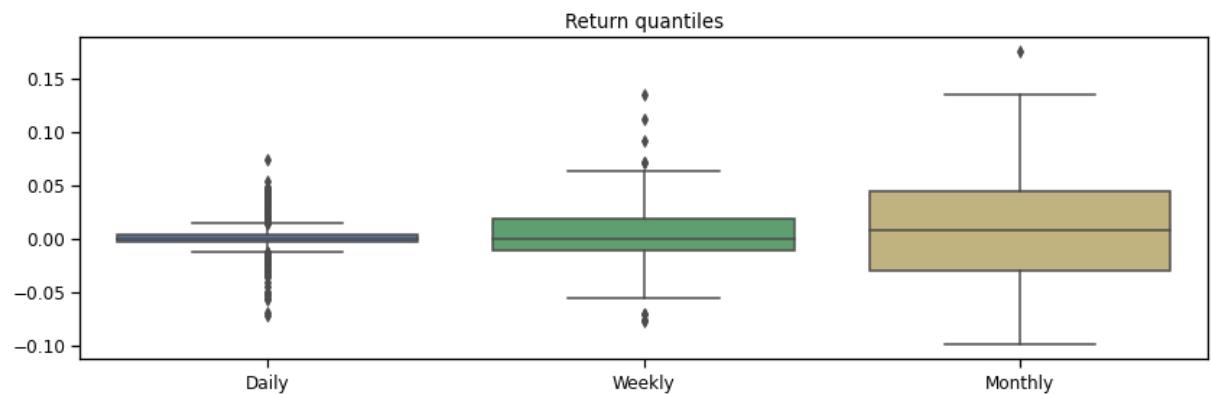
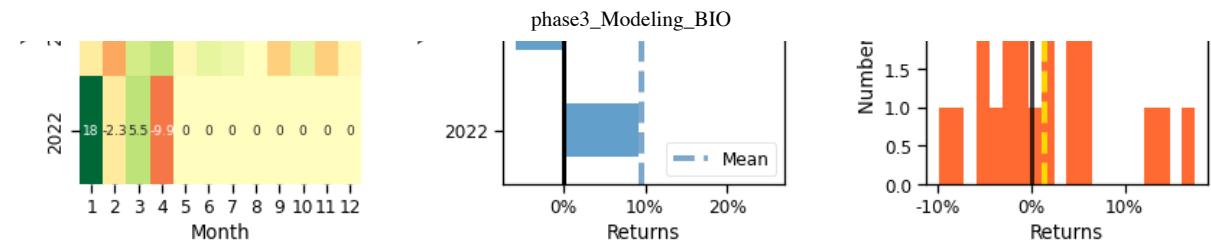
New Normal 0.05% -7.16% 7.36%



phase3_Modeling_BIO







In [38]:

```
backtest(results['SVC'])
```

```
[*****100*****] 1 of 1 completed
Shape of DataFrame: (456, 8)
```

Start date 2020-07-02

End date 2022-04-23

Total months 31

Backtest

Annual return 12.031%

Start date 2020-07-02**End date** 2022-04-23**Total months** 31**Backtest**

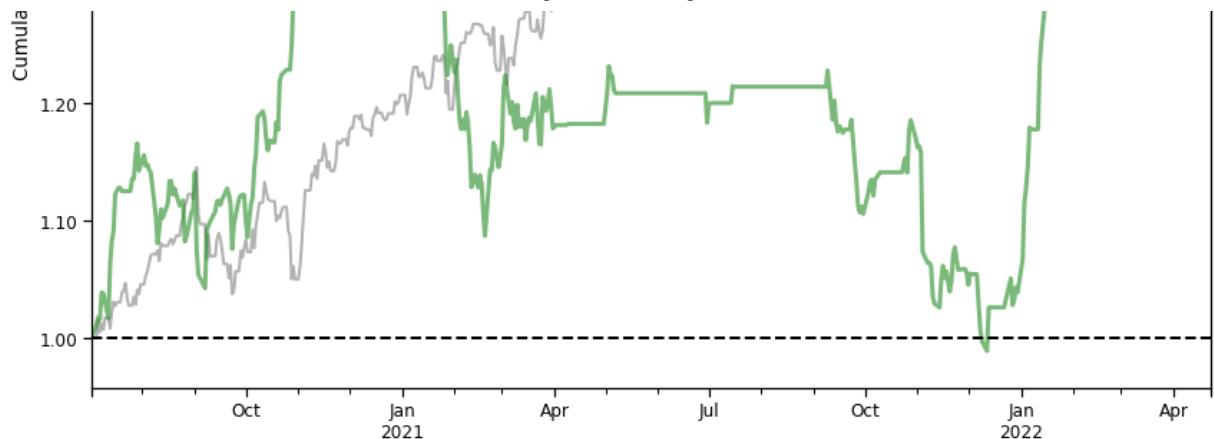
Cumulative returns	34.715%
Annual volatility	21.551%
Sharpe ratio	0.64
Calmar ratio	0.31
Stability	0.02
Max drawdown	-38.583%
Omega ratio	1.15
Sortino ratio	0.90
Skew	NaN
Kurtosis	NaN
Tail ratio	1.12
Daily value at risk	-2.661%
Alpha	0.15
Beta	-0.04

Worst drawdown periods	Net drawdown in %	Peak date	Valley date	Recovery date	Duration
0	38.58	2020-12-17	2021-12-12	NaT	NaN
1	10.78	2020-11-17	2020-11-23	2020-12-13	19
2	10.60	2020-07-29	2020-09-07	2020-10-08	52
3	6.58	2020-11-05	2020-11-07	2020-11-09	3
4	2.79	2020-10-11	2020-10-14	2020-10-21	8

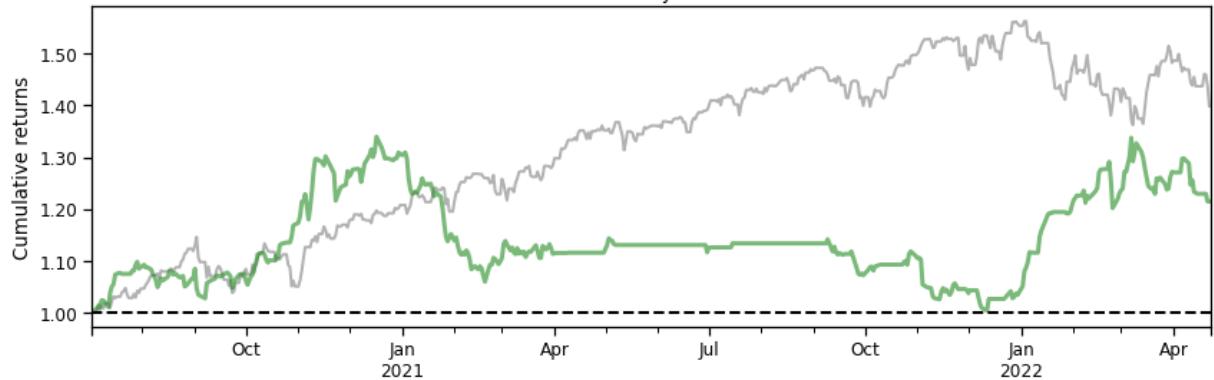
Stress Events	mean	min	max
New Normal	0.05%	-7.36%	5.34%



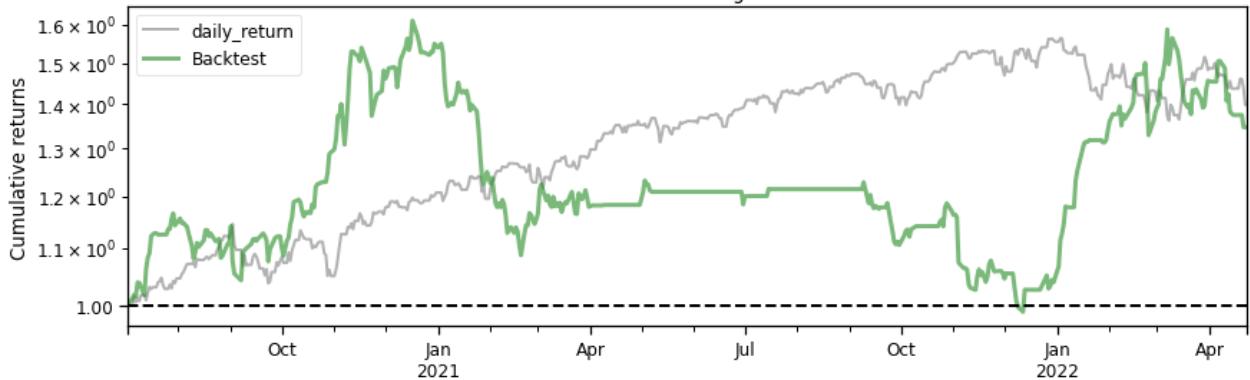
phase3_Modeling_BIO



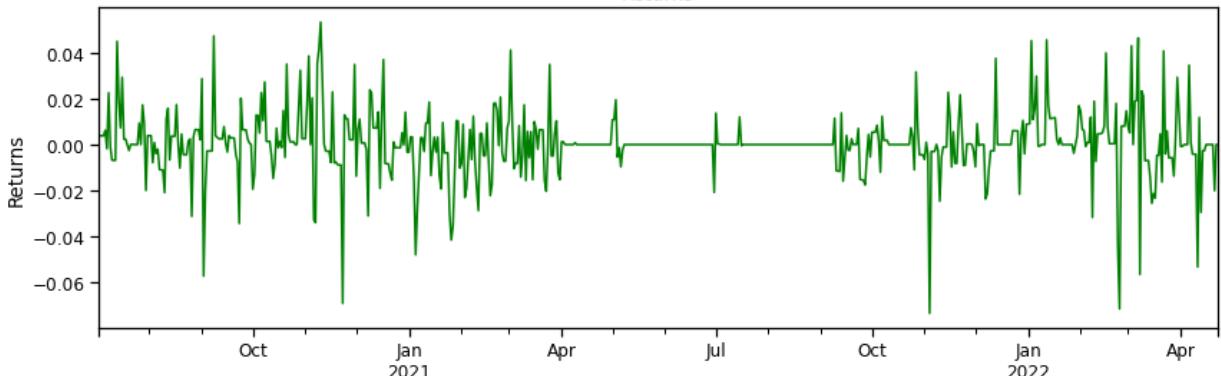
Cumulative returns volatility matched to benchmark



Cumulative returns on logarithmic scale

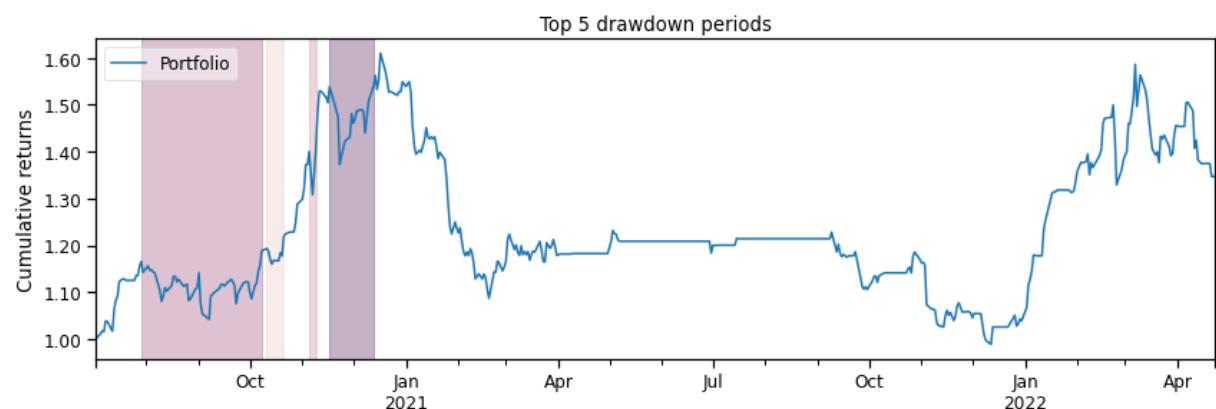
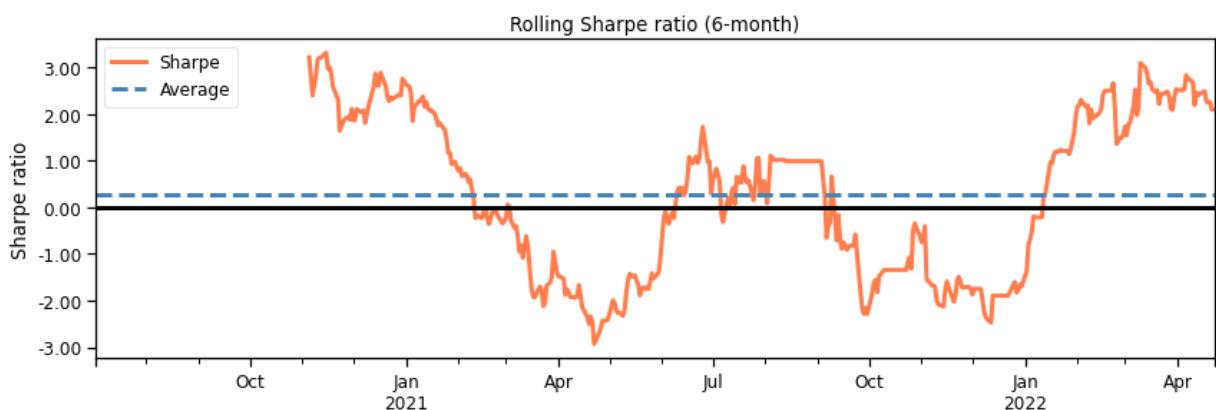
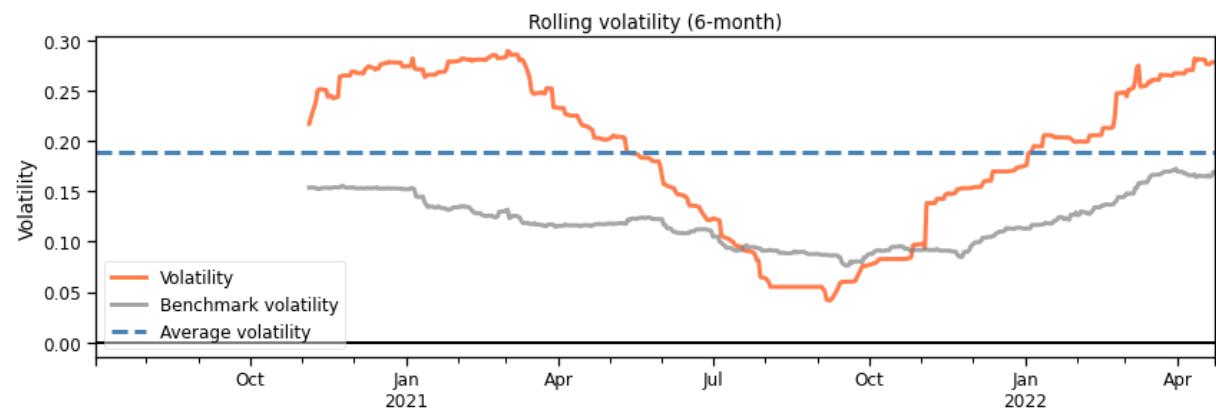
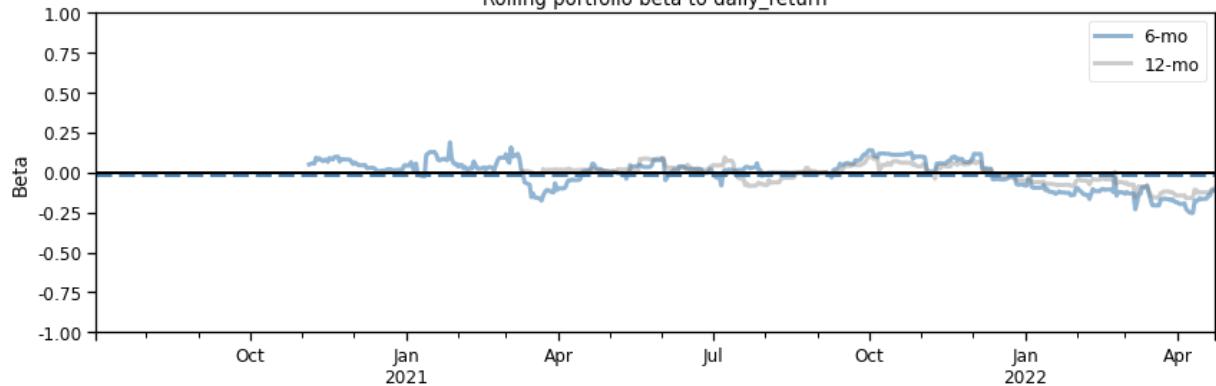


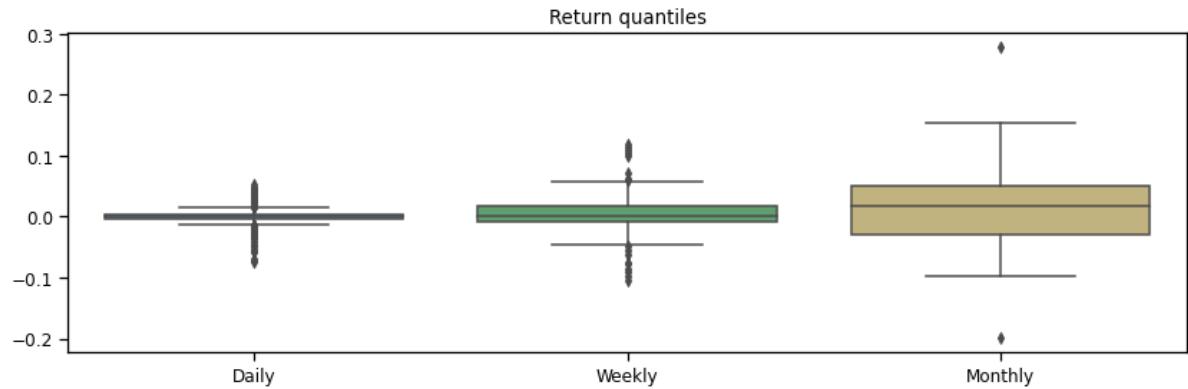
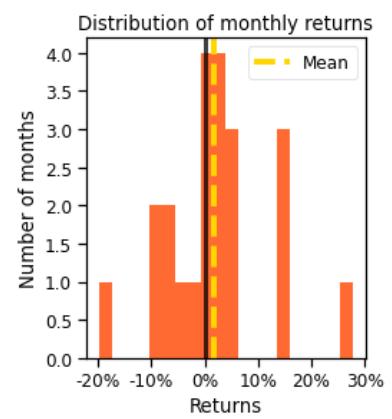
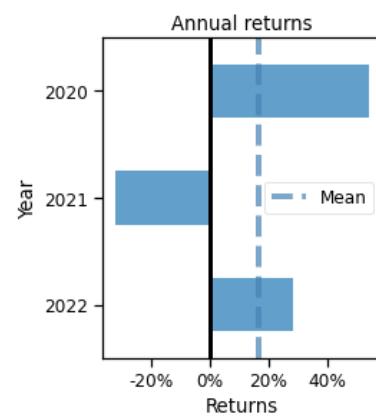
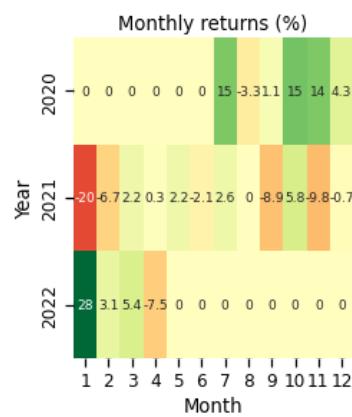
Returns

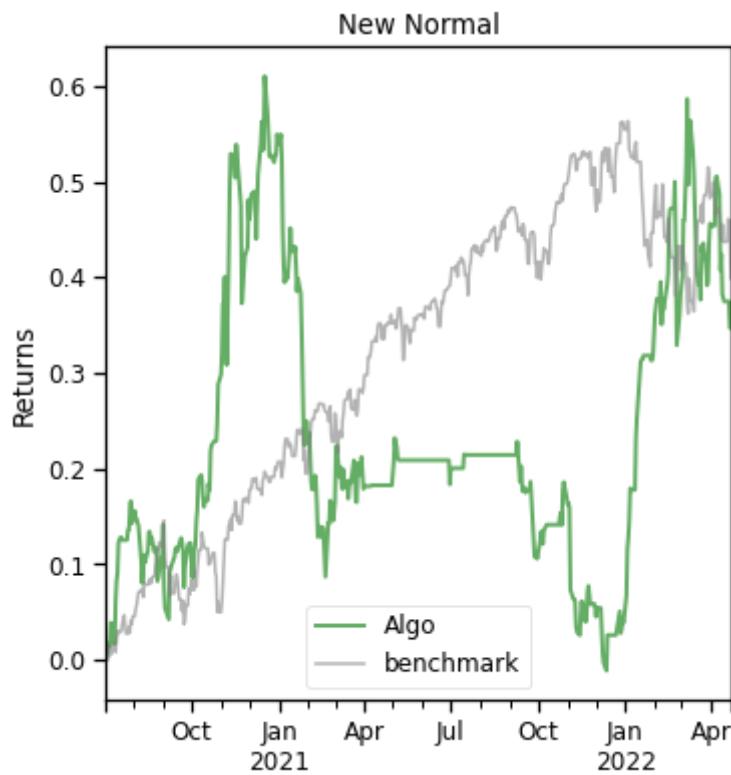


phase3_Modeling_BIO

rolling portfolio beta to daily return







In [39]:

```
backtest(results['KNN'])
```

```
[*****100%*****] 1 of 1 completed
```

```
Shape of DataFrame: (456, 8)
```

Start date 2020-07-02

End date 2022-04-23

Total months 31

Backtest

Annual return	37.242%
Cumulative returns	129.422%
Annual volatility	20.022%
Sharpe ratio	1.68
Calmar ratio	1.61
Stability	0.49
Max drawdown	-23.087%
Omega ratio	1.48
Sortino ratio	2.76
Skew	NaN
Kurtosis	NaN
Tail ratio	1.42
Daily value at risk	-2.389%
Alpha	0.41

Start date 2020-07-02

End date 2022-04-23

Total months 31

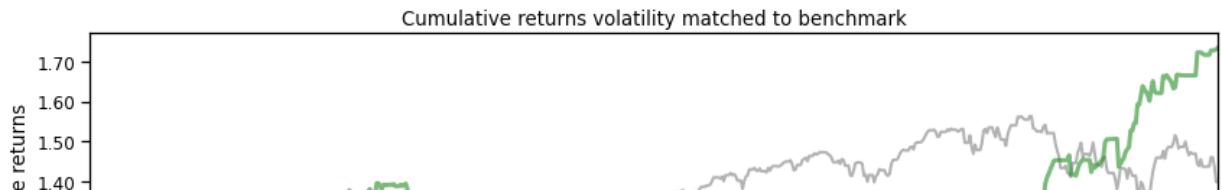
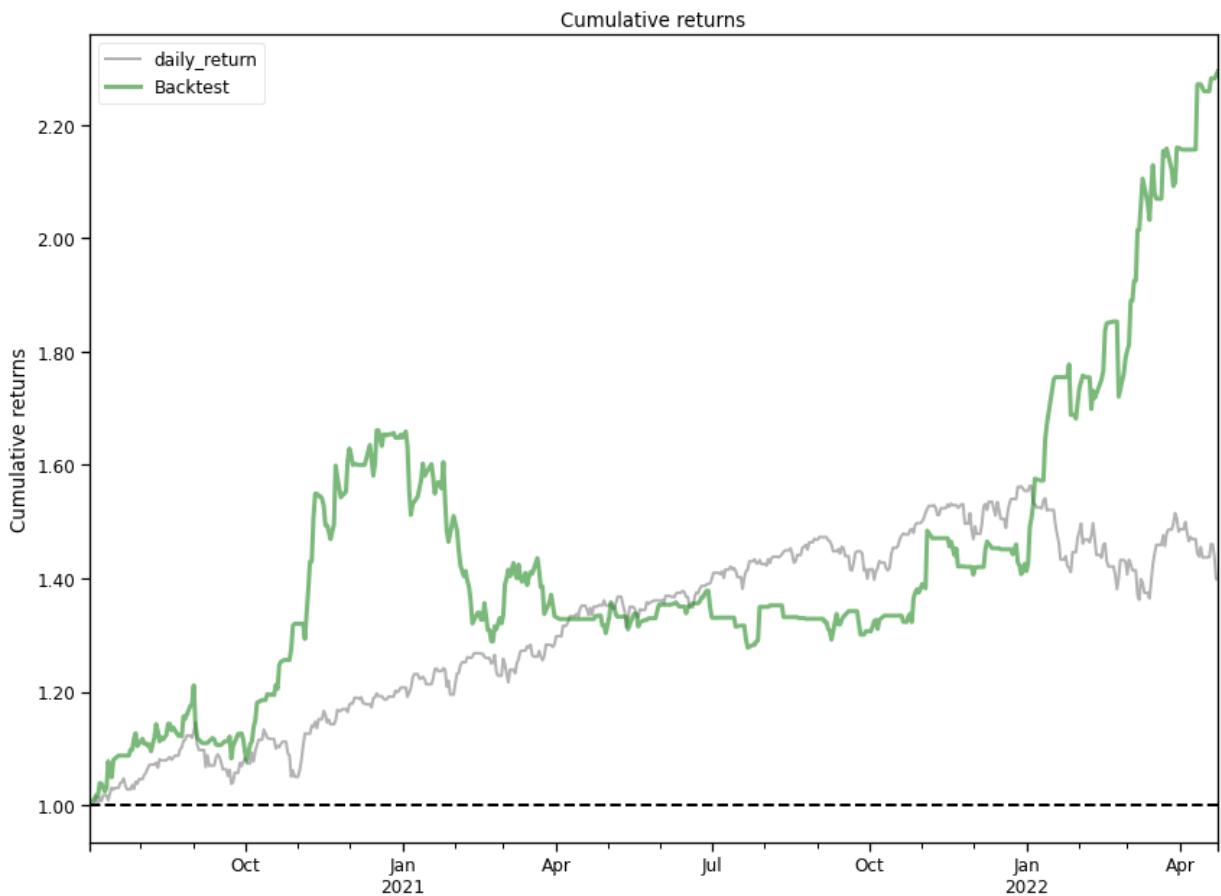
Backtest

Beta -0.04

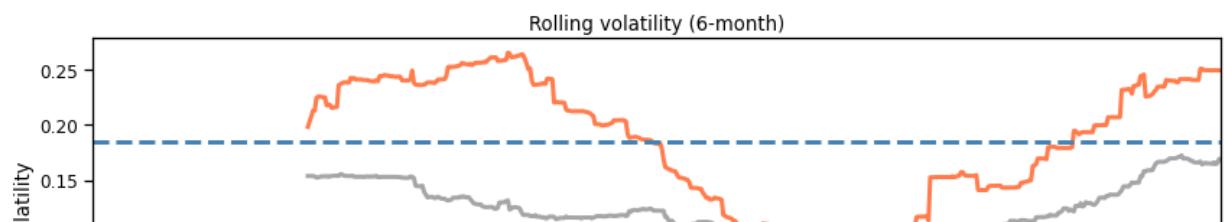
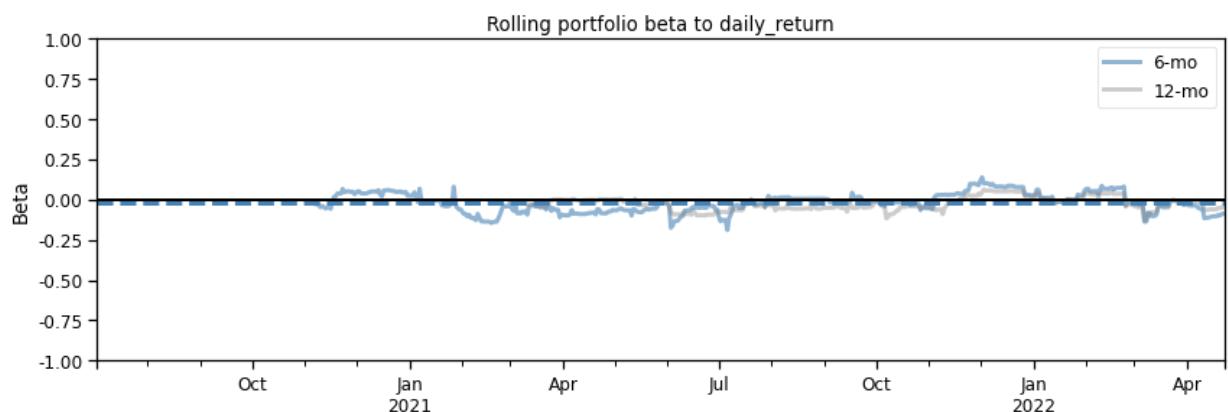
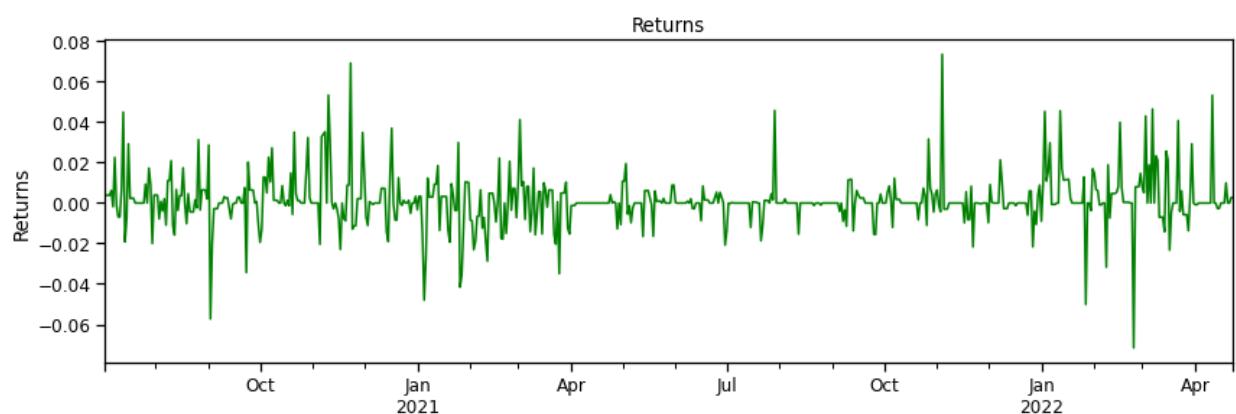
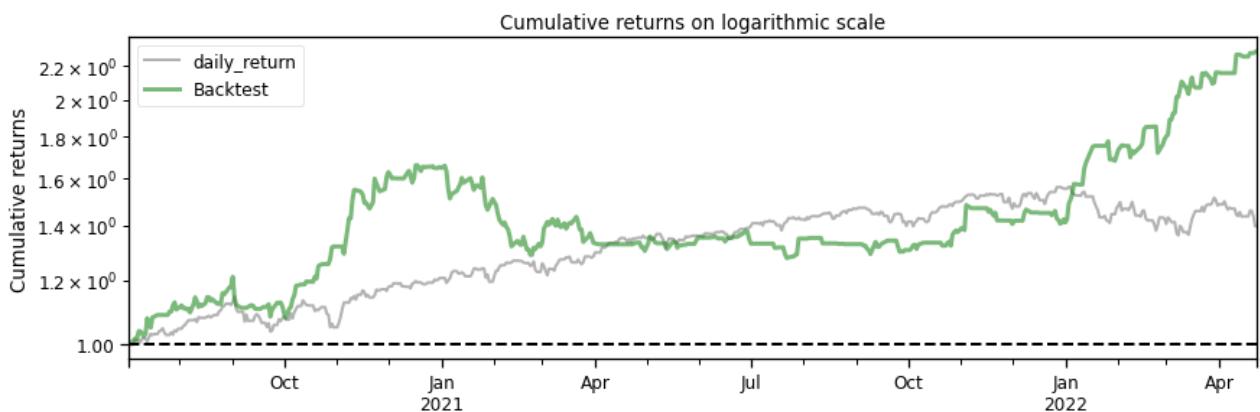
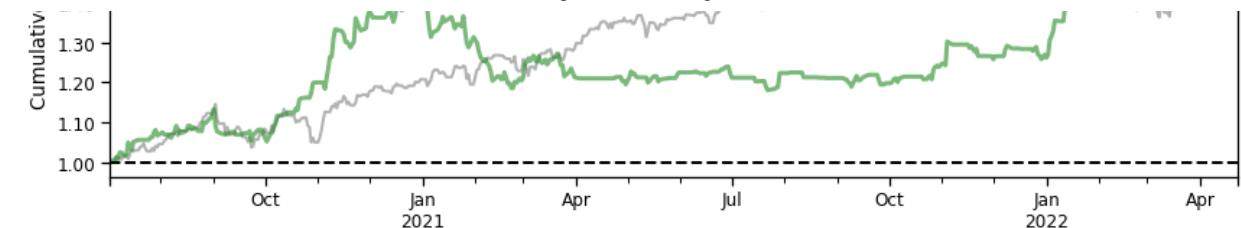
Worst drawdown periods	Net drawdown in %	Peak date	Valley date	Recovery date	Duration
0	23.09	2020-12-18	2021-07-22	2022-01-13	280
1	10.94	2020-09-01	2020-10-02	2020-10-19	35
2	7.16	2022-02-23	2022-02-24	2022-03-03	7
3	5.37	2022-01-26	2022-01-30	2022-02-16	16
4	5.18	2020-11-12	2020-11-20	2020-11-23	8

Stress Events mean min max

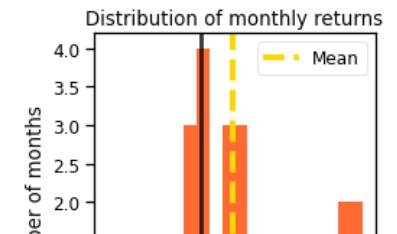
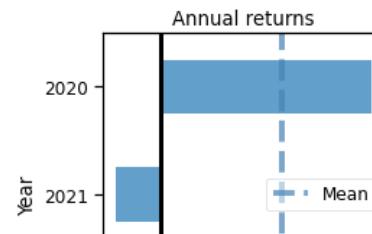
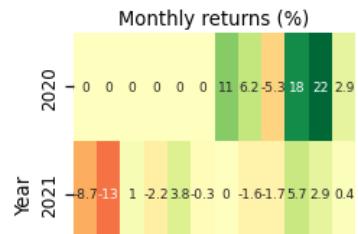
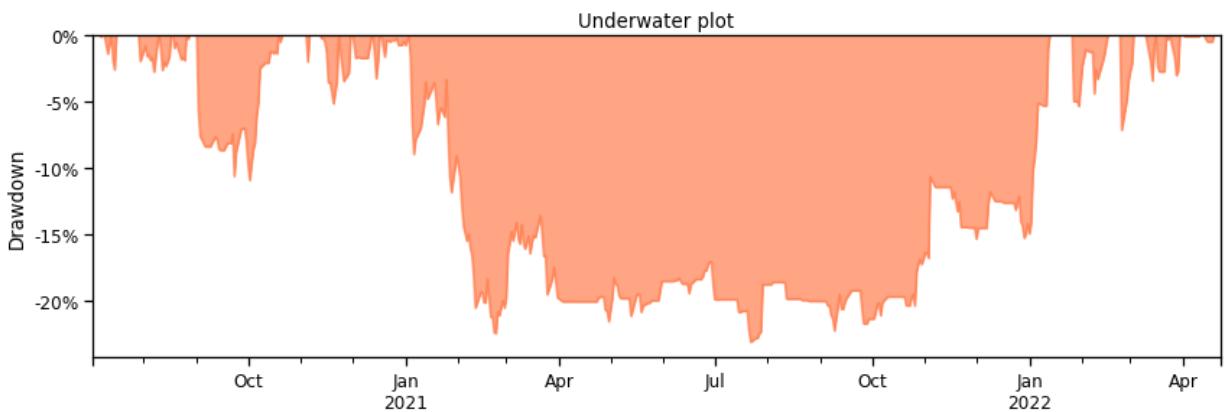
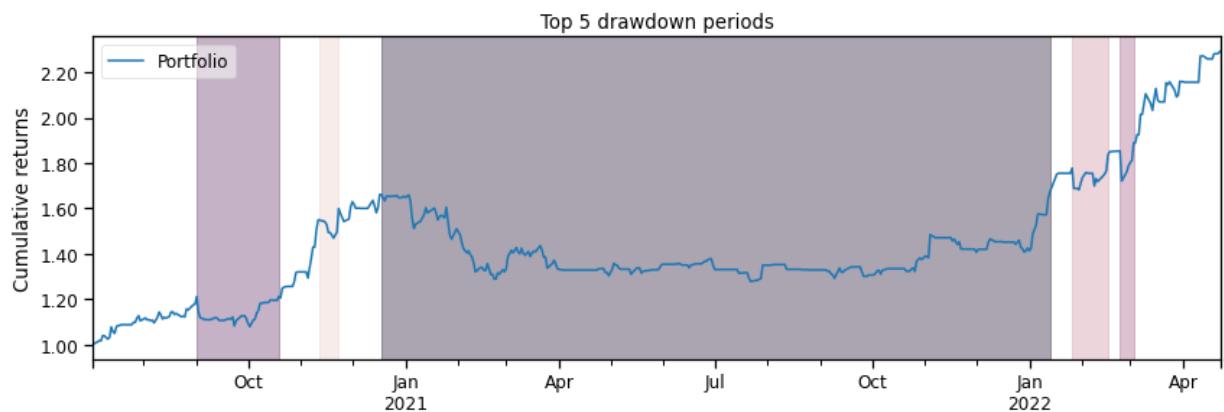
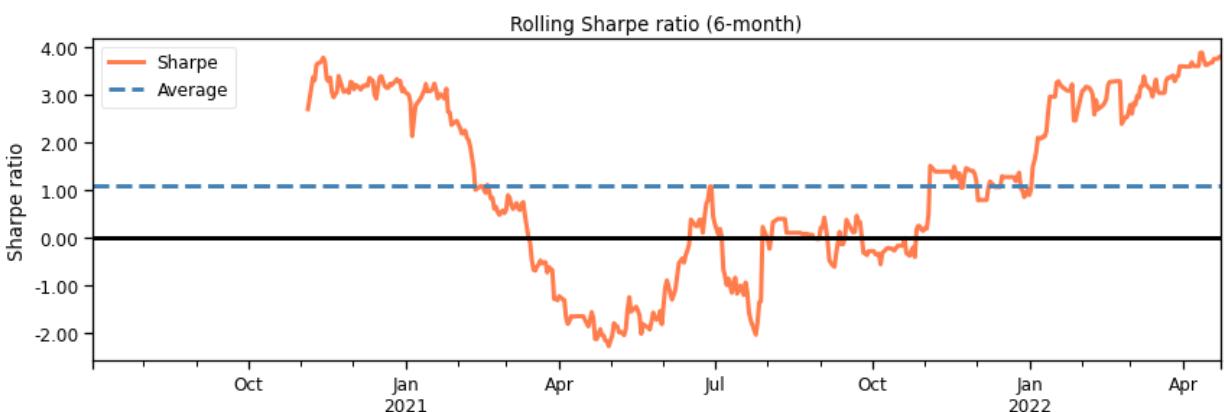
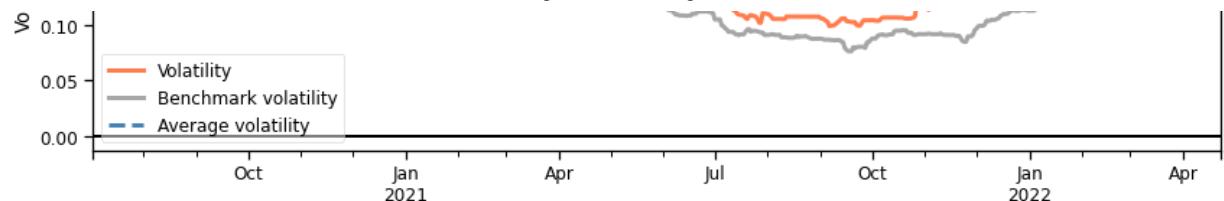
New Normal 0.13% -7.16% 7.36%

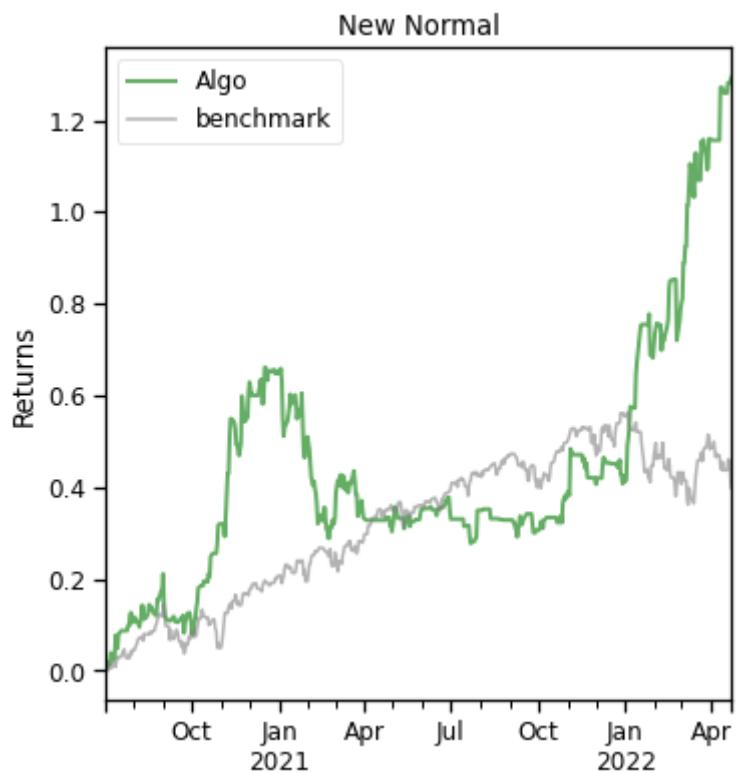
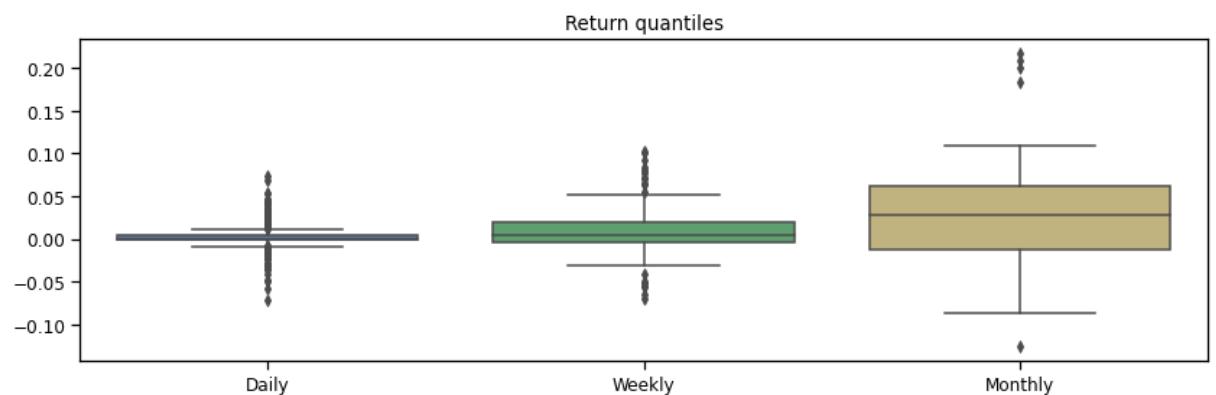
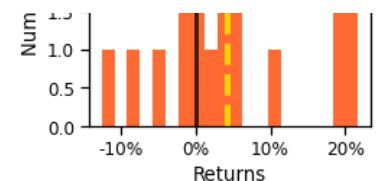
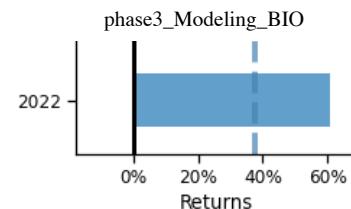
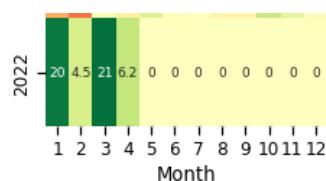


phase3_Modeling_BIO



phase3_Modeling_BIO





In []:

In []: