Student Name: LI JIAYI
Student ID: 1155188890

Technical Analysis

Review

---

1. **Common Technical Indicators**

   Technical Analysis(TA) is a methodology for determining (forecasting) the future direction of asset prices and identifying investment opportunities based on studying past market data.

2. **Backtesting**

   Backtesting can be described as a realistic simulation of our trading strategy, which assesses its performance using historical data. The underlying idea is that the backtest performance should be indicative of future performance when the strategy is actually used on the market.

   Biases

   (a) *Look-ahead bias*: when we develop a trading strategy using historical data before it was actually known/available.

   (b) *Survivorship bias*: when we backtest only using data about securities that are currently active/-tradeable. By doing so, we omit the assets that have disappeared over time (due to bankruptcy, delisting, acquisition, and so on).

   (c) *Outlier detection and treatment*: The main challenge is to discern the outliers that are not representative of the analyzed period as opposed to the ones that are an integral part of the market behavior.

   (d) *Representative sample period*: As the goal of the backtest is to provide an indication of future performance, the sample data should reflect the current, and potentially also future, market behavior.

   2 types of returns

   (a) *Arithmetic* return:

   $$r_t = \frac{S_t - S_{t-1}}{S_{t-1}}$$

   (b) *Log* return:

   $$\tilde{r}_{t+1} = ln\Big(\frac{S_t}{S_{t-1}}\Big) = ln(S_t) - ln(S_{t-1})$$

   # simply sum up the log returns from whole period to get the total return

3. **Vectorized Backtesting**

   Multiply a signal vector/matrix (containing an indicator of whether entering or closing a position) by the vector of returns

   Consider a simple example based on the SMA20

   (a) Enter a long position if the close price is above the 20-day Simple Moving Average (SMA)

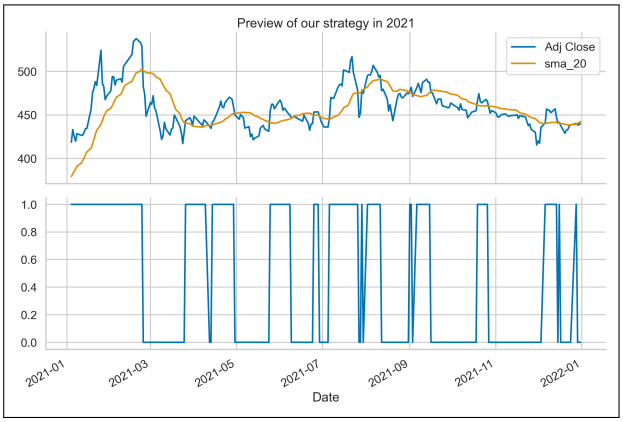
   (b) Close the position when the close price goes below the 20-day SMA

   (c) Short selling is not allowed

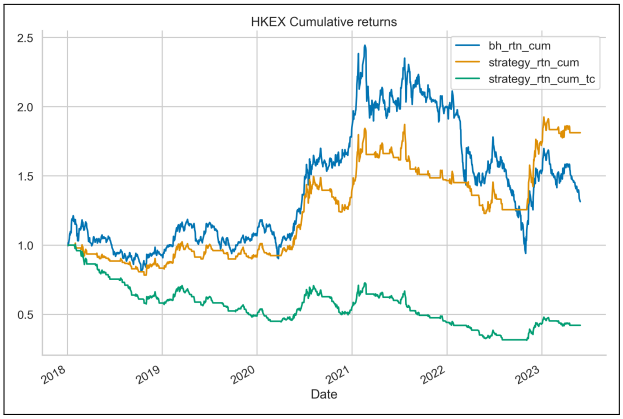
(d) The strategy is unit agnostic

Table 1: Data preview

Date	Adj Close	log_rtn	sma_20	position	strategy_rtn	strategy_rtn_cum	bh_rtn_cum	tc	strate
2018-01-02	219.85	NaN	NaN	0	NaN	NaN	NaN	NaN	
2018-01-03	218.97	-0.00	NaN	0	-0.0	1.0	1.00	0.0	
2018-01-04	223.72	0.02	NaN	0	0.0	1.0	1.02	0.0	
2018-01-05	222.49	-0.01	NaN	0	-0.0	1.0	1.01	0.0	
2018-01-08	230.22	0.03	NaN	0	0.0	1.0	1.05	0.0	



(a) Efficient frontier using optimization



(b) Cumulative returns of all strategies

Figure 1: HKEX trading strategy with SMA20 and transaction cost

4. **Event-driven Backtesting**

Event-driven backtesting aims to simulate all the actions and constraints encountered when executing a certain strategy while allowing for much more flexibility than the vectorized approach. For example, this approach allows for simulating potential delays in orders' execution, slippage costs.

Code for Figure 1

```
import pandas as pd
import yfinance as yf
import numpy as np

#Download HKEX stock prices and keep only the adjusted close price
df = yf.download("0388.HK",
                 start="2018-01-01",
                 end="2023-05-30",
                 progress=False)
df = df[["Adj Close"]]

#Calculate the log returns and the 20-day SMA of the close prices
df["log_rtn"] = df["Adj Close"].apply(np.log).diff(1)
df["sma_20"] = df["Adj Close"].rolling(window=20).mean()
```

```
#Create a position indicator and change the false and true into 0 and 1
df["position"] = (df["Adj Close"] > df["sma_20"]).astype(int)

#Count how many times we entered a long position
#shift means that the series is moving to the next period by n units
sum((df["position"] == 1) & (df["position"].shift(1) == 0))

fig, ax = plt.subplots(2, sharex=True)
df.loc["2021", ["Adj Close", "sma_20"]].plot(ax=ax[0])
df.loc["2021", "position"].plot(ax=ax[1])
ax[0].set_title("Preview of our strategy in 2021")

#The position vector is shifted by 1 to avoid the look-ahead bias.
#Flag is generated using all the information up to time t.
#We can only use that information to open a position on the next trading day at time t+1
df["strategy_rtn"] = df["position"].shift(1) * df["log_rtn"]
df["strategy_rtn_cum"] = df["strategy_rtn"].cumsum().apply(np.exp)

#Add the buy-and-hold strategy for comparison
df["bh_rtn_cum"] = df["log_rtn"].cumsum().apply(np.exp)

# assume that the transaction costs are 1%
TRANSACTION_COST = 0.01
df["tc"] = df["position"].diff(1).abs() * TRANSACTION_COST
df["strategy_rtn_cum_tc"] = (
    (df["strategy_rtn"] - df["tc"]).cumsum().apply(np.exp)
)

#Plot the cumulative returns of all the strategies
df.loc[:, ["bh_rtn_cum", "strategy_rtn_cum", "strategy_rtn_cum_tc"]]
  .plot(title="Cumulative returns")
```