A Report On Quantitative Finance

The Wall Street Club

BITS Pilani Hyderabad Campus

Table of Contents

Quantitative Finance Preview

Regular finance is about buying and selling things like stocks and bonds. Quantitative finance (quant finance for short) is like having a super powered coach for the stock market.

They use complex math and computer programs, like algorithms, which are basically fancy step-by-step instructions. These algorithms crunch tons of data – past prices, social media buzz – to find patterns and predict how much a stock might be worth. It's not a guaranteed win, but it gives quants (people who do quant finance) an edge, kind of like how a coach analyses plays and uses that info to strategize for the next game.

Algorithms act like weights. When making decisions, their output gives us a lens to view the market through, to see if the trade we want to make is reasonable or not. This is done by giving risk a numeric value, i.e., quantifying it.

While becoming a Quant (a professional in quantitative finance) is an extremely arduous task, it's easier than ever to venture into this immensely fun world equipped with basic mathematical and programming skills.

Python's Role in Quant

Python is one of the most powerful programming languages. With Python, you can effortlessly build complex programs we mentioned before. It lets you sift through mountains of data, perform intricate calculations, and translate the market's whispers into clear signals. It's like having a super-efficient lab assistant who automates all the tedious tasks, freeing you to focus on crafting the most ingenious algorithms.

To begin your journey in quantitative finance, we strongly recommend you equip yourself with the fundamentals of Python programming, like basic syntax, conditionals, loops, functions and basic data structures like lists and dictionaries. Also explore libraries like pandas for data analysis and delve into Jupyter Notebooks to bring your code and analysis together.

Some resources you can skim through to get a basic idea about what the role of python is in quant have been added below:

- Jupyter Notebook Download Tutorial
- Some basic python along with data analysis projects

We have added a plethora of online resources at the end of the report to help you get an elementary proficiency in data analytics using python.

Common Keywords Used

1. **Libraries**: Pre-built code collections in Python for tasks like data analysis
2. **DataFrame**: Powerful data structure (like a spreadsheet) for organizing financial data (rows & columns).
3. **Jupyter Notebook**: Interactive coding environment for data science, great for combining code, results, and explanations.
4. **API**: Messenger between your program and external systems (like data providers) for retrieving data.
5. **Back testing**: A method of evaluating a trading strategy using historical market data to simulate its performance.
6. **Broker**: An intermediary who executes trades on behalf of investors in financial markets, often providing additional services like research and advice. For example, Fyer, Zerodha, ICICI Securities.

Quantitative Finance Toolkit

Listed below are some of the libraries you'll be using frequently while doing data analysis using python. We'll recommend you acquaint yourself with these powerful tools, it will make your life a lot easier.

Yahoo Finance : "yfinance" is a Python library that makes it easy to grab stock data (prices, financials) from Yahoo Finance. It's like a shortcut for quant beginners to download and analyse financial information.

NumPy: NumPy is your workhorse for math. Think of it as a super-powered tool that handles massive datasets and complex calculations efficiently. It's essential for tasks like analysing prices, performing linear algebra, and generating random data – all crucial tools in your quant toolbox.

Pandas: Pandas is a powerful data analysis and manipulation library that offers data structures and tools for working with structured and time series data. It excels at handling tabular data, making it easier to load, clean, transform, and analyse datasets.

Matplotlib: Matplotlib is a plotting library for Python that allows users to create high quality figures and visualizations. It provides a wide range of plotting functions and tools for creating static, animated, and interactive visualizations across various platforms and environments.

Sessions on Quantitative Finance

The Wall Street Club organized Quantitative Finance sessions for the general body to delve into various roles within the finance industry, including Quantitative Trading. These sessions, led by Sohum Muley, Ex-Head of Quantitative Investments at the Wall Street Investment Fund, centred on practical strategies employed by professionals in live markets.

The sessions leveraged Python libraries such as pandas, yfinance, NumPy, and matplotlib. Python facilitated comprehensive analysis across the entire stock market in a streamlined manner, contrasting with the traditional approach of selecting individual stocks sequentially.

Additionally, the use of the FYERS API enabled real-time data retrieval and facilitated live web-based trading operations.

The major take-aways from each session:

- Basics of Python
- Basics of importing Pandas and NumPy and Data Cleaning.
- Basics of Momentum Investing
- Completion of Momentum Investing and Portfolio building
- Building of Overnight Carry Strategy Momentum Strategy
- Introduction to Fyers API
- Fetching Historical Data from Fyers Api

Momentum Investing Strategy

Imagine you're at the beach and see some waves building. Momentum investing is like catching those waves that are already starting to swell. You're hoping to ride them up as they get bigger (stock price goes up) before they crash (price falls).

Here's the breakdown:

Catching the Swell: You look for stocks that are already gaining momentum, their price is on an upward trend. These are your waves.

Riding the Wave: You buy the stock as the price is rising, aiming to sell it before the wave breaks (price drops).

Catching the Best Part: They want to ride the wave while it's powerful and exciting, maximizing the thrill (and maybe scoring a sweet trick).

Things to Consider:

The Wave Can Crash Early: The stock price might suddenly reverse course. You could buy right before a big drop, wiping you out (taking a loss).

Not All Waves Are Epic: Just because a wave is building doesn't mean it will become a monster. You need to understand the conditions (market trends) to pick the best waves (stocks).

Python is a powerful tool that can help you identify stocks with growing momentum. It can analyse historical price data to see which stocks have been riding an upward trend. Think of it as a wave forecasting app for the stock market.

Explainers on Momentum Investing – Momentum Portfolios, YouTube Videos

We strongly recommend you grasp the basics of python and its elementary functions to understand the topics covered ahead.

Momentum Strategies Based Solely on Price Action

Trend Following: Identify trends using technical indicators like moving averages. Trade in the direction of confirmed trends, considering factors like volume and news. Adjust position size based on trend strength and risk tolerance.

Pullback Trading: Buy assets after a temporary price decline. Use Fibonacci retracements and support/resistance levels to identify potential entry points. Consider risk management techniques like stop-loss orders to protect profits.

Chart Patterns: Recognize and trade based on specific price formations like flags, triangles, and head and shoulders. Backtest these patterns to evaluate their historical performance and identify reliable setups.

The merits of the code we have used

- **Momentum Effect:** The strategy leverages the momentum effect, which suggests that past winners tend to continue outperforming in the short term.
- **Simplicity**: The approach is relatively straightforward and easy to implement.
- **Diversification**: Selecting a portfolio of 20 momentum picks can help diversify risk.
- **Backtesting**: The function can be used to back test the strategy on historical data to assess its performance

Explanation of the logic behind the code

The logic of the momentum portfolio is that it checks the previous year's returns to rank the stocks accordingly to take a long position(buy) in the first 20 stocks for the next month and repeat the same cycle continuously.

To follow along and run the complete code use this Colab Notebook - Code for Momentum Portfolio

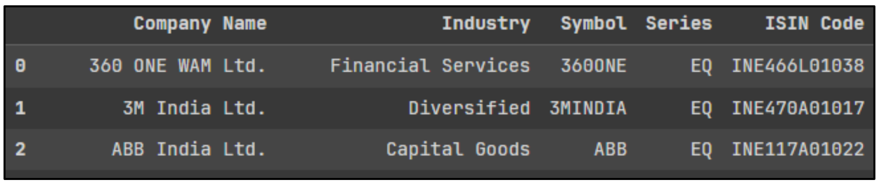
Importing the required libraries

```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
yf.pdr_override
```

Download the nifty 500 constituent list

Import data

```
stock=pd.read_csv("ind_nifty500list.csv")
stock.head()
```



```
         Company Name            Industry   Symbol  Series     ISIN Code
0     360 ONE WAM Ltd.  Financial Services  360ONE      EQ  INE466L01038
1        3M India Ltd.         Diversified  3MINDIA     EQ  INE470A01017
2       ABB India Ltd.      Capital Goods      ABB      EQ  INE117A01022
```

Use yfinance to get OHLC stock data

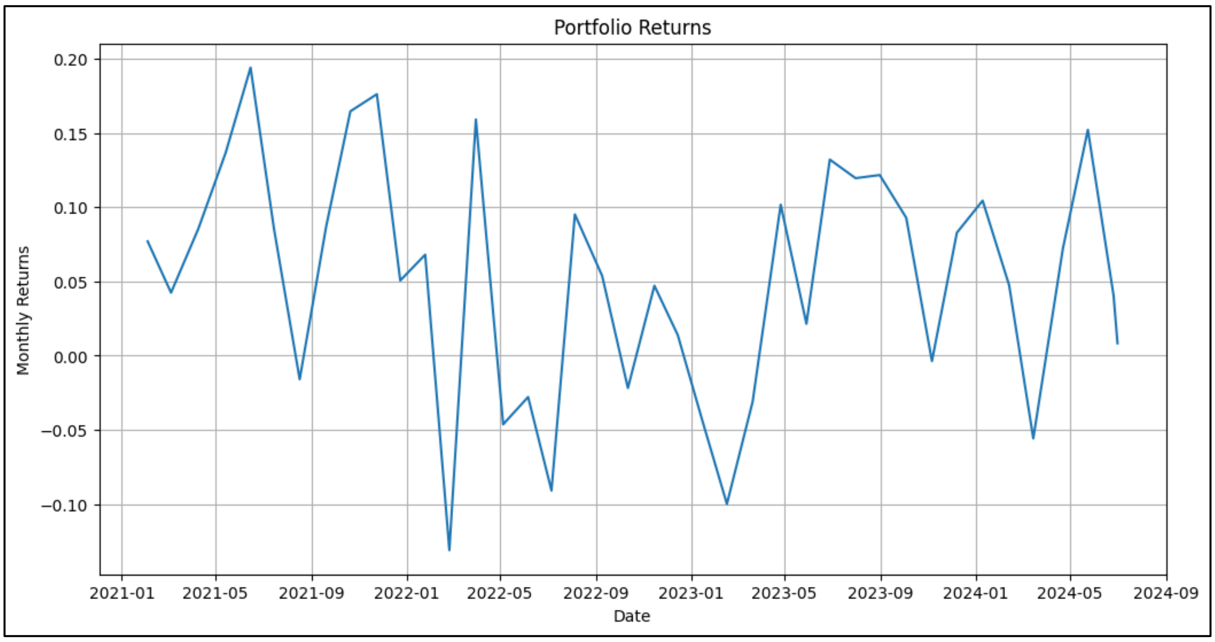
```
data  = yf.download(tickers=stock_list,start='2012-01-01')["Close"]
```

Implementing the logic: The calculate_monthly_returns(data) function below calculates monthly returns using a momentum strategy. It iterates through the dataset, analysing the previous year's data (252 trading days) to find the top 20 stocks with the highest cumulative returns (momentum picks). It then computes the portfolio's return for the following month and stores the result with the date. If there is insufficient data or no picks, the iteration is skipped. The process continues to build a list of monthly returns based on this strategy.

```python
def calculate_monthly_returns(data):
  lst = []
  for i in range(0, len(data) - 44, 22):
    previous_year_dataset = data.iloc[i+1:i+252]
    pyd_chg = previous_year_dataset.pct_change()
    ranks = np.cumprod(1 + pyd_chg).iloc[-1].T
    momentum_picks = ranks.sort_values(ascending=False).dropna().index[0:20].tolist()
    next_month = data.iloc[i+252:i+252+22]
    if not next_month.empty and momentum_picks:
      monthly_returns = np.mean(np.cumprod(1 + next_month[momentum_picks].pct_change()).iloc[-1] - 1)
      monthly_date = next_month.index[-1]
      lst.append((monthly_date, monthly_returns))
    else:
      print(f"Skipping iteration at i = {i} due to insufficient data or no momentum picks.")
  return lst
```

Backtesting Results

The graph shows the return over the time for the portfolio selected using the momentum picks.



Few key observations on the graph of back testing using the momentum investing strategy:

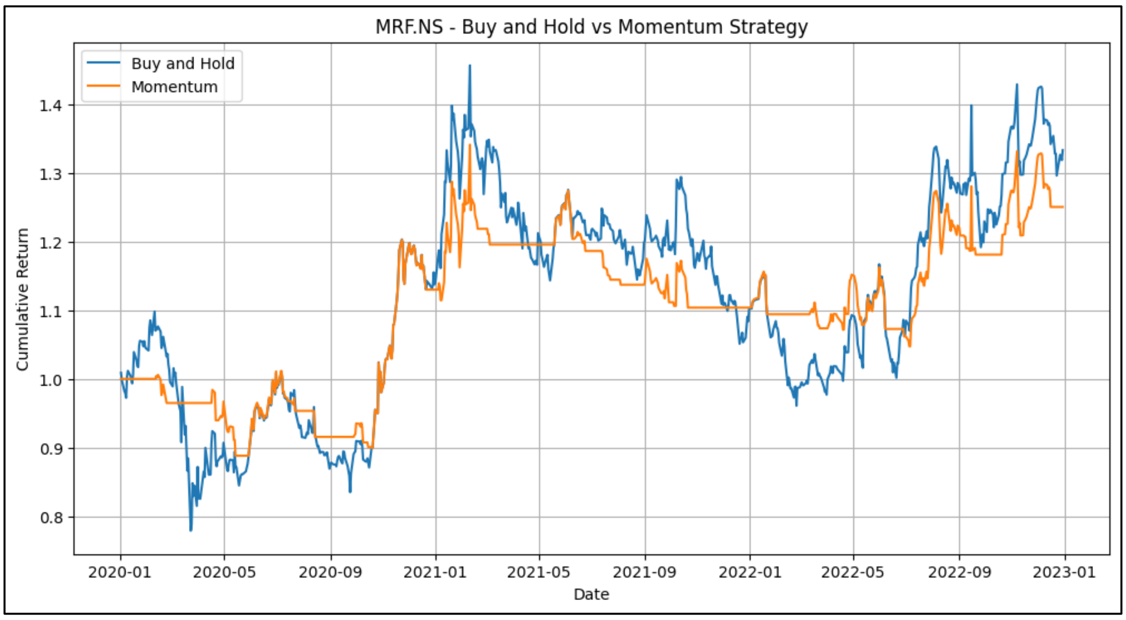
1. **Volatility**: The returns exhibit significant fluctuation, indicating a relatively high-risk investment strategy. This volatility can be attributed to factors such as the underlying asset classes, market conditions, and the portfolio's allocation to these assets.
2. **Trend**: While there's no consistent upward or downward trend, the overall trajectory suggests a slight positive bias. This could be due to a combination of factors, including the selection of assets, timing, and market conditions.
3. **Drawdown**: The graph reveals periods of significant declines, indicating potential risks associated with the portfolio. These drawdowns could be attributed to market corrections, economic downturns, or specific events affecting the underlying assets.

The following graphs illustrate a comparison between the buy-and-hold strategy and the momentum strategy for two stocks during the COVID-19 period. These charts showcase the cumulative returns—representing the total change in the value of an investment over time, factoring in the compounding effect of reinvested gains—for both strategies.

The following graphs show two different types of outcomes of momentum investing strategy are:

The Downtrend chart

In this case, we found that the Buy and Hold strategy gave better cumulative returns compared to the Momentum Strategy. This means that if you simply bought shares and held onto them for a long time, you would have ended up making more money than if you were frequently buying and selling based on recent performance trends.

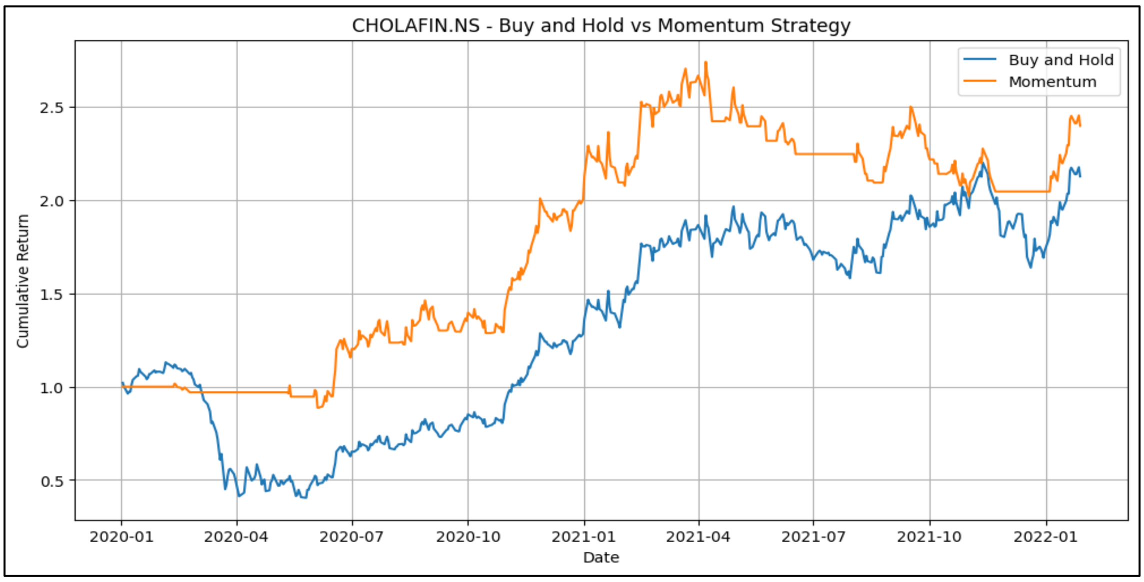


- Buy and Hold Total Return: 33.28%
- Momentum Total Return: 25.02%

- Buy and Hold Sharpe Ratio: 0.48
- Momentum Sharpe Ratio: 0.48

The Uptrend Chart

This case shows that by buying stocks that have recently been performing well and selling those that haven't, you end up making more money than if you simply held onto your stocks without selling.



- Buy and Hold Total Return: 112.73%
- Momentum Total Return: 139.86%
- Buy and Hold Sharpe Ratio: 0.92
- Momentum Sharpe Ratio: 1.33

Note -

Sharpe Ratio: The Sharpe ratio is a measure of risk-adjusted return, comparing an investment's excess return to its volatility.

Formula

$$Sharpe\ Ratio\ =\ (R_p\ -\ R_f)\ /\ \sigma_p$$

R_p — return of portfolio

R_f — risk free return

σ_p — standard deviation of the portfolio's excess return

Conclusion drawn from the Backtesting in Different Settings

The results drawn from the back testing clarify the usage of the momentum investing strategy and how does the strategy work in different environment

Downtrend Results

Momentum investing works well in trending markets, benefiting from rising prices or betting on falling ones. However, it struggles during trend reversals, where sudden market shifts cause losses. In sideways or choppy markets, the lack of clear trends leads to false signals and unprofitable trades. Volatile markets increase the risk of poorly timed trades. Momentum investing relies on consistent trends, making it less effective in unpredictable or directionless markets.

The comparison chart of MRF (MRF.NS) during the covid time period during peak market downtrend the momentum strategy does not perform better than the buy and hold strategy

Uptrend Results

During strong uptrends, momentum investing excels by capitalizing on the sustained rise in asset prices. In these periods, momentum strategies align with the prevailing market direction, leading to substantial gains as investors ride the upward trend. This environment supports momentum investing's core principle of benefiting from ongoing trends, resulting in superior returns compared to more static strategies. In clear uptrends, momentum investing harnesses the power of continued price growth, delivering strong performance.

The comparison chart of Chola Finance (CHOLAFIN.NS) during the similar time period show the benefits of using the momentum investing strategy instead of the buy and hold strategy.

Overnight Carry Strategy

In the world of investing, while chasing major market swings can be thrilling, another approach—known as the "overnight carry" strategy—offers a more nuanced opportunity. Imagine you're on a calm evening, waiting for the gentle current beneath the surface to bring in small waves. This current symbolizes the subtle price movements that occur overnight.

Catching the Current:

Focus on assets with steady, predictable price movements rather than dramatic changes. For example, buy stocks with consistent patterns or stable currencies just before the market closes to capitalize on small, but significant price shifts when the market reopens.

Holding On Through the Night:

The strategy involves buying an asset before the market closes and holding it overnight. This allows you to capitalize on price action driven by news or events that happen outside regular trading hours. For example, a company might announce earnings, or a geopolitical event might influence currency values, leading to price changes by the next trading day.

Steady Gains:

This strategy aims to exploit these small fluctuations for steady, reliable gains. Over time, these incremental profits can accumulate, providing a consistent source of income without the need for large market moves.

However, risks are inherent. Price action can be unpredictable, and unforeseen events or shifts can cause significant price gaps when the market reopens, potentially negating overnight gains. Additionally, holding positions overnight might incur costs such as financing charges or market impact.

Explanation of the logic behind the code

To follow along and run the complete code use this Colab Notebook – Code for Overnight Carry

```
def Strategy1(ticker,start_date,N):
        df=yf.download(tickers=ticker,start=start_date)
        df['Shifted Open']=df['Open'].shift(-1)
        df['profit']=((df['Shifted    Open']-df['Close'])/df['Close'])-0.0035*(df['Shifted Open']-
df['Close'])/df['Close']
        df['rets']=df['Close'].pct_change()
        df=df.dropna()
        Buy_Hold = N*(1+df['rets']).cumprod()
        Strategy = N*(1+df['profit']).cumprod()
        if(len(Buy_Hold)<1):
        return(ticker,0,0,0,0,0,0,0)
        drawdown(Buy_Hold)
        drawdown(Strategy)

        return(ticker,Buy_Hold[-1],Strategy[-
1],drawdown(Buy_Hold),drawdown(Strategy),absolute_returns(Buy_Hold),absolute_returns(Strategy))
```

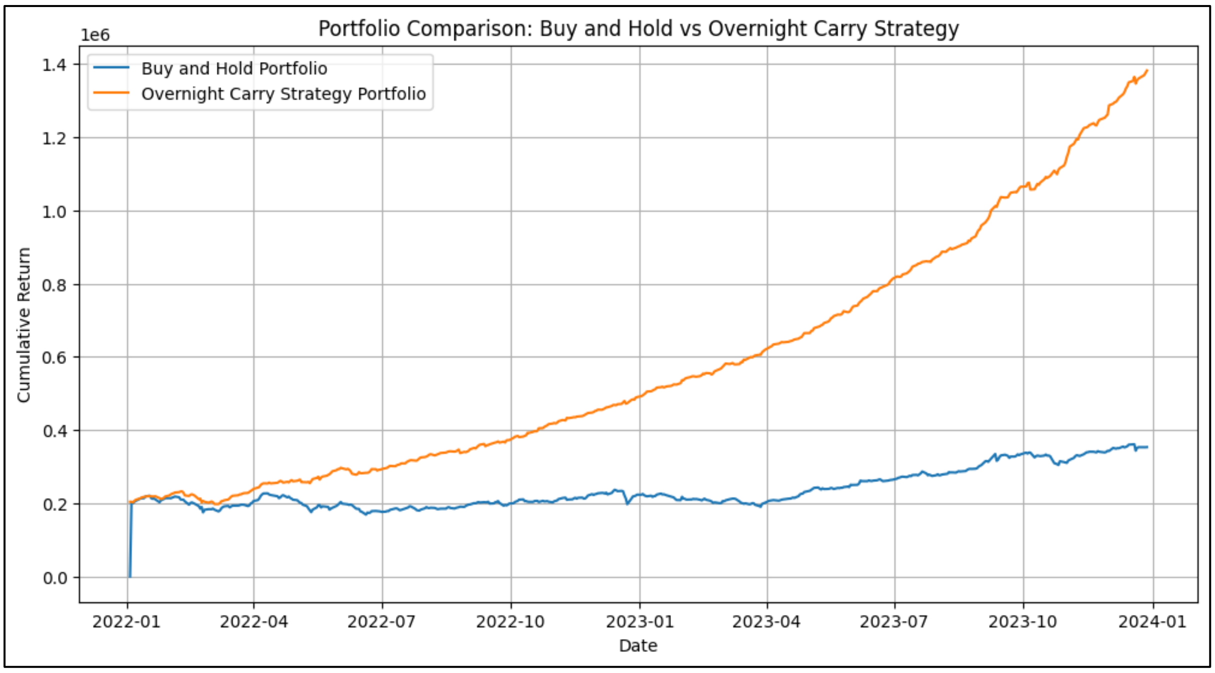
The `Strategy1` function is used to compare an overnight trading strategy with a buy-and-hold approach for a particular stock. It starts by downloading historical stock data, including daily open and close prices, using the `yfinance` library. This data is essential for evaluating the performance of both strategies over time.

For the overnight trading strategy, the function calculates potential profits by shifting the 'Open' prices to the next day and comparing them to the current day's 'Close' price. It incorporates a transaction cost of 0.35% into these calculations to reflect real trading expenses. The cumulative returns for both strategies are then computed, starting with an initial investment amount, `N`.

If the data is insufficient, the function returns zeros for all performance metrics. Otherwise, it provides a comprehensive comparison by returning the final portfolio values, maximum drawdowns, and absolute returns for each strategy. This analysis helps in understanding the effectiveness and risk associated with both trading approaches.

Backtesting Results

The strategy has been compared to the buy and hold strategy and here are the returns in the graph plotted below



The returns of the portfolio have been:

- Strategy Total Return: 577.57%
- Buy and Hold Sharpe Ratio: 0.57
- Strategy Sharpe Ratio: 4.07
- Buy and Hold Maximum Drawdown: -25.60%
- Strategy Maximum Drawdown: -14.92%

Conclusion from Backtesting Results

The overnight carry strategy offers distinct benefits by leveraging profitable gap-ups and upward market trends. Holding positions overnight allows traders to benefit from price movements outside regular trading hours, often resulting in favourable gaps at market open.

Profitability Through Gap-Ups: The strategy excels in exploiting gap-ups—instances where a security opens much higher than its previous close. These gaps, driven by after-hours news or market changes, enable consistent profits during strong upward trends.

Reduction in Drawdowns: The strategy also reduces drawdowns, or losses from peak to trough. By capitalizing on positive overnight movements and using disciplined exits, it minimizes adverse price impacts and improves capital protection.

Backtest Performance: Backtesting indicates that the overnight carry strategy outperforms the Buy and Hold approach, particularly in upward-trending markets, even after considering transaction costs. This shows its effectiveness in capturing gains more dynamically.

Adaptability and Risk Management: Successful implementation requires staying updated on economic news and market changes affecting overnight gaps. Effective risk management, such as setting stop-loss orders and adjusting trade sizes, is crucial to guard against market reversals.

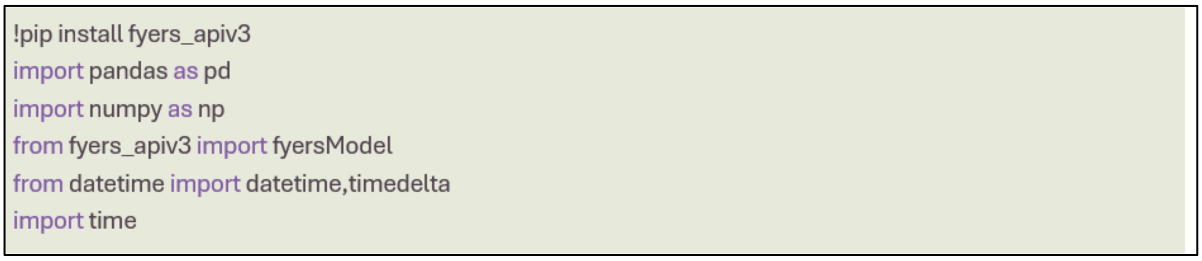
In summary, the overnight carry strategy is effective for profiting from gap-ups and upward trends. It outperforms the Buy and Hold strategy, even with transaction costs, offering significant potential for traders who use it with careful attention and strong risk management.

The Fyers API

In quantitative finance, algorithms automatically buy or sell stocks based on specific market conditions. To create these algorithms, you need market data, which often comes at a price. Luckily, the FYERS API offers a free way to access stock market data and build trading algorithms.

FYERS is a popular choice for algorithmic trading because it is user-friendly, and reliable. It lets you develop and run trading strategies without any fees.

You will learn how to access and authorize the API, setting the foundation for building your first quantitative trading strategies. **We've also attached a Python notebook** Fyers Api and we'll break down the code step by step in this document.



```
!pip install fyers_apiv3
import pandas as pd
import numpy as np
from fyers_apiv3 import fyersModel
from datetime import datetime,timedelta
import time
```

In this section, we're setting up our development environment. If you're using Google Colab, essential libraries like pandas, NumPy, datetime, and time are already included, but you'll need to install the `fyers_apiv3` library separately.

If you're working in an environment like Visual Studio Code (VSCode), you'll need to install the necessary libraries using pip commands in the terminal. For instance, just run pip install pandas to get pandas set up. We'll be importing pandas as pd, NumPy as np, and using modules like datetime, timedelta, and time for handling date and time operations.

If you're new to these libraries, resources like W3Schools can be really helpful. For details on the FYERS API, refer to the v3 documentation at Fyers Docs.

```
client_id = "XXXXXXXXXX-100"
#note that this is same as app_ID should be kept hidden(use .env file)
secret_key = "XXXXXXXXXX" #secret_ID again try to use a .env file
redirect_uri = "https://trade.fyers.in/api-login/redirect-uri/index.html" #the same link will be given to you
after you run this cell, click on it and copy the auth_code
response_type = "code" #we want it to generate an auth_code, hence response_type="code"
grant_type = "authorization_code"

session = fyersModel.SessionModel(
        client_id=client_id,
        secret_key=secret_key,
        redirect_uri=redirect_uri,
        response_type=response_type,
        grant_type=grant_type
)
response = session.generate_authcode()
print(response)
```

In this section of the code, we're setting up authentication for the FYERS API. First, we specify the `client_id`, which is basically your FYERS app ID. Then, we provide the `secret_key`, which acts as your secret ID for security purposes. The `redirect_uri` is the link where users will be directed once the process completes, and here it's set to "https://trade.fyers.in/api-login/redirect-uri/index.html". Lastly, we set the `response_type` to "code", indicating that we want to generate an authorization code.

Next, we create a session object using the `fyersModel.SessionModel` class, passing in the authentication details we just configured. This session object contains all the information required to authenticate with the FYERS API.

After that, we use the `generate_authcode()` method on the session object to retrieve the authorization code. The resulting code is stored in the `response` variable, and we print it to display the authorization code we need.

```
# Replace this value with YOUR authorization code, which you shall obtain after clicking on the redirect uri
from the top cell
auth_code = "eyJ0e...."

# Set the authorization code in the session object
session.set_token(auth_code)

# Generate the access token using the authorization code
response = session.generate_token()

# Print the response, which should contain the access token and other details
# You should get something like this: {'s': 'ok', 'code': 200, 'message': '', 'access_token': 'eyJ0e.....'}
print(response)
```

Alright, so here's how we get the access token for the FYERS API. First up, we define the `auth_code`—this is that authorization code you get after clicking on the redirect URI from the previous step.

Once we have that, we plug it into the session object using the `set_token()` method. Then, we go ahead and generate the access token by calling `generate_token()` on that session object.

Finally, we print out the response, which should include the access token along with some other details. This token is super important because it's what allows us to securely interact with the FYERS API. The response you'll see should look something like this: `{'s': 'ok', 'code': 200, 'message': '', 'access_token': 'eyJ0eXAiO....'}`.

Here's a simplified explanation of the rest of the code

1. **Imports and Constants**: The code starts by importing necessary libraries (`time`, `pandas`, `datetime`, `timedelta`, and `fyersModel`). It also sets constants for rate limiting, defining how many API calls are allowed per second and per minute (`MAX_CALLS_PER_SECOND` and `MAX_CALLS_PER_MINUTE`).
2. **FyersModel Initialization**: The `fyersModel.FyersModel` object is created using your `client_id`, `auth_code`, and an empty log path. This object is necessary for making API requests to FYERS.
3. **Rate Limiting Function:** The `rate_limit()` function ensures that the number of API calls does not exceed the defined limits. It uses the current time to check if the number of calls per second or minute is within limits and adds delays if needed.
4. **Fetching Historical Data:** The `fetch_historical_data()` function retrieves stock data within a given date range. It processes data in chunks of up to 365 days to manage large datasets. Dates are converted to epoch time for compatibility with the API. The function handles exceptions and ensures compliance with rate limits.
5. **Rate Limit Tracking:** Before fetching data, variables to track the number of API calls and timestamps are initialized. This tracking helps manage rate limits effectively.
6. **Processing Multiple Tickers:** The code iterates over a list of stock tickers, fetching historical data for each. For each ticker, it defines a date range (last five years), retrieves the data, and then stores it in a DataFrame. The DataFrame is converted to a CSV file, which is saved locally. If no data is retrieved, a message is printed.

Conclusion

Both the overnight carry and momentum investing strategies show strong promise through theory and back testing. However, their real-world success depends on market conditions, economic stability, and trader adaptability.

Overnight Carry Strategy: The overnight carry strategy benefits from interest rate differentials and overnight market movements. While back testing shows high returns, real-world success requires monitoring economic indicators and staying responsive to market changes.

Momentum Investing Strategy: Momentum investing profits from existing market trends by buying rising assets and selling falling ones. Although back tests indicate high returns, real-world success depends on timely data and effective risk management.

Real-World Considerations: Both strategies require adaptation to shifting market conditions. Effective risk management and staying updated with market data and economic news are crucial for mitigating risks and optimizing performance.

In summary, while both strategies have strong theoretical and back tested potential, their success depends on understanding market behaviour, disciplined execution, and adaptability. Practical insights and responsiveness are key to achieving long-term success.

More about Quant and the Finance Industry

Introduction to the Finance Industry

In a market economy, the financial system functions like an orchestra without a central conductor. Savers invest their capital, and producers use that capital to create goods and services, working together harmoniously. Each participant plays their part, contributing to the overall flow of the economy

Regulations serve as traffic lights, ensuring everyone follows the rules and maintains order. Beyond regulations, ethical people with strong communication skills are essential, much like a backstage crew ensuring the smooth running of a performance.

Mathematical expertise is increasingly important, akin to musicians who read sheet music flawlessly, ensuring precise coordination. When the financial system operates well, like a well-conducted orchestra, risks are managed effectively, and the benefits extend to everyone involved, with minimal disruption to the broader economy.

Key Companies in Quantitative Finance

Sell-Side Firms

Investment banks like Bank of America Merrill Lynch, Citigroup, Goldman Sachs, J.P. Morgan Chase, and Morgan Stanley create and sell financial products to institutions, retail investors, and each other, earning them the name "sell-side firms."

Sell-side firms, such as investment banks, are involved in originating, structuring, and trading these financial products. Quants play a crucial role in pricing these products and developing hedging strategies. By using complex mathematical models and quantitative techniques, quants help ensure that these products are priced accurately and managed effectively.

Buy-Side Firms

Asset management funds invest money from clients according to specific strategies, while hedge funds take calculated risks. Prominent hedge funds include AQR Capital Management, Citadel, and Appaloosa Management.

Hedge Funds: Hedge funds often take specific market positions. For example, if a hedge fund predicts that U.S. mortgage-backed bonds might default, it can purchase credit default swaps (CDS) from an investment bank. A CDS involves regular payments but provides a large payout if the bonds do default, effectively acting as insurance. This type of trade was prominent during the 2007-2010 financial crisis and was highlighted in the movie "The Big Short."

Other hedge funds use price signals to predict and profit from market movements, leveraging machine learning on electronic trading data. Quants play a crucial role in determining the financial instruments and quantities needed to execute these strategies.

Exchanges and Clearing Houses

Today, trading is largely done by computers. Market makers now set buy and sell prices with very small spreads and earn fees per trade. High-frequency traders also make quick pricing decisions.

Exchanges handle stocks, options, and futures, requiring traders to post cash margins to ensure they fulfil their contracts. Electronic trading requires advanced software and data monitoring, tasks managed by quants.

Over-the-counter contracts, like interest rate and credit default swaps, are traded directly between parties but must be recorded at clearing houses. These houses ensure contract obligations are met by requiring cash collateral and monitoring compliance, also overseen by quants.

Data Providers and Software Vendors

Financial institutions rely on data from firms like Bloomberg and FactSet, which provide both data and financial software. Software-only vendors like Numerix also play a role. These firms require experts in finance, regulation, and mathematical models, known as quants.

Insurance Firms

Insurance firms face uncertainty in matching premiums collected with future, unpredictable liabilities. They invest premiums long-term, like in life insurance. Managing this requires assets aligned with uncertain future needs. This complex task demands expertise from quants due to the role of financial markets in asset-liability matching.

Rating and Consulting

Ratings agencies such as Moody’s and Standard & Poor’s employ quants to model default risk. Consulting firms like Ernst & Young help firms without in-house quant teams to develop and assess financial market models, offering a "rented" quant team approach instead of building their own.

Future Prospects of the Club

The Wall Street Club at BPHC: A Hub for Aspiring Quants

About Us

The Wall Street Club at BPHC is a vibrant community for students passionate about finance and technology. We provide a platform for aspiring quants to explore the world of financial markets, learn practical skills, and apply them through hands-on projects.

What We Offer

- **Comprehensive Learning**: Regular workshops and seminars led by industry experts and faculty cover topics like statistical analysis, machine learning, algorithmic trading, and risk management.
- **Practical Experience**: Hands-on sessions using Python libraries like Pandas, NumPy, SciPy, Matplotlib, and Scikit-learn help students apply theoretical knowledge to real-world problems.
- **Collaborative Projects**: Work on research and development projects to gain practical experience and potentially publish your findings.

Our Mission

Our mission is to prepare students with the essential skills and knowledge to thrive in the ever-evolving finance industry. We strive to bridge the gap between theoretical academic learning and practical industry applications. By offering comprehensive training and real-world experiences, we aim to empower our members to make significant contributions and achieve success in the financial sector.

Join Us

If you’re passionate about finance, technology, and quantitative analysis, we invite you to join the Wall Street Club at BPHC. Whether you're a beginner or have advanced knowledge, our club offers a supportive, collaborative environment for skill development, hands-on projects, and connecting with like-minded peers.

Resources and References

- Sohum Muley GitHub Link to Quant Finance Sessions : GitHub link
- Link to a Roadmap for quant roles : Roadmap - to - Quant
- Link to Survivorship Bias in Stock Trading : Survivorship Bias
- Books for Quant Finance : Books
- List of Resources : Link
- Wright Research Algotrading Blogs : Articles
- Code for Momentum Investing : **Code for Momentum Portfolio**
- Code for Overnight Carry Strategy : **Code for Overnight Carry**
- Code for Fyers Api : **Fyers Api**,

Wall Street Club Previous Partners

1.Peak AI (part of the Wall Street Analytics Challenge 2023)

2.World Quant (Quantitative asset management firm)