DSA5205 Data Science for Quantitative Finance Group Project Presentation

Team Confused Matrix

Outline of the presentation

1. Stock selection
    1. Select top 30 stocks
    2. Select final 10 stocks
2. Markowitz Portfolio Optimisation
3. Time series analysis
4. Feature engineering
    1. Historical Market Data
    2. Macro-economic Data
5. Machine Learning Model
    1. Fully connected neural network
    2. XGBoost Classifier
6. Final result
7. Conclusion

Investment Universe

100 Stocks

100 Stocks

20 Stocks

Fetching Data

```python
def get_ftse_symbols():
    url = "https://en.wikipedia.org/wiki/FTSE_100_Index"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', {'id': 'constituents'})
    symbols = []
    for row in table.find_all('tr')[1:]:
        ticker = row.find_all('td')[1].text.strip()
        if ticker.endswith('.'):
            symbols.append(ticker + 'L')
        elif ticker.endswith('ICG'):
            pass
        else:
            symbols.append(ticker + '.L')
    return symbols
```

30min - 60 days
60min - 370 days

Select Top 30 Stocks

First Method - Compute various metrics

```
 # log return for each period (hourly)
 df['Log_Return'] = np.log(df['Close'] / df['Close'].shift(1))
 # total return from the beginning of the period
 df['Cumulative_Return'] = (df['Close'] / df['Close'].iloc[0]) - 1
 # annualized volatility using a 5-day rolling window
 df['Volatility'] = df['Log_Return'].rolling(window=int(hours_per_day * 5)).std() * np.sqrt(hours_per_day * trading_days_per_year)
 # annualized Sharpe ratio
 df['Sharpe'] = (df['Log_Return'].mean() * (hours_per_day * trading_days_per_year)) / (df['Log_Return'].std() *
.sqrt(hours_per_day * trading_days_per_year))
 # 5-day moving average of trading volume
 df['Volume_MA'] = df['Volume'].rolling(window=int(hours_per_day * 5)).mean()
 # 5-day price momentum
 df['Momentum'] = df['Close'] / df['Close'].shift(int(hours_per_day * 5)) - 1
 return df
```

Select Top 30 Stocks

First Method - Score the metrics

```
# please play around with the scores below
if allow_short:
    total_score = (volatility_score * 0.4 +    # higher vol as it provides profit opp in both directions
                liquidity_score * 0.1 +        # liquid positions to enter and exit
                np.abs(momentum_score) * 0.1 +
                np.abs(return_score) * 0.3)     # abs(return) to see if the market moves

else:
    total_score = (return_score * 0.3 +         # more weight to higher returns
                volatility_score * 0.35 +
                liquidity_score * 0.15 +        # liquid positions to enter and exit
                momentum_score * 0.1 +
                sharpe_score * 0.1)             # risk adjusted returns
return total_score
```

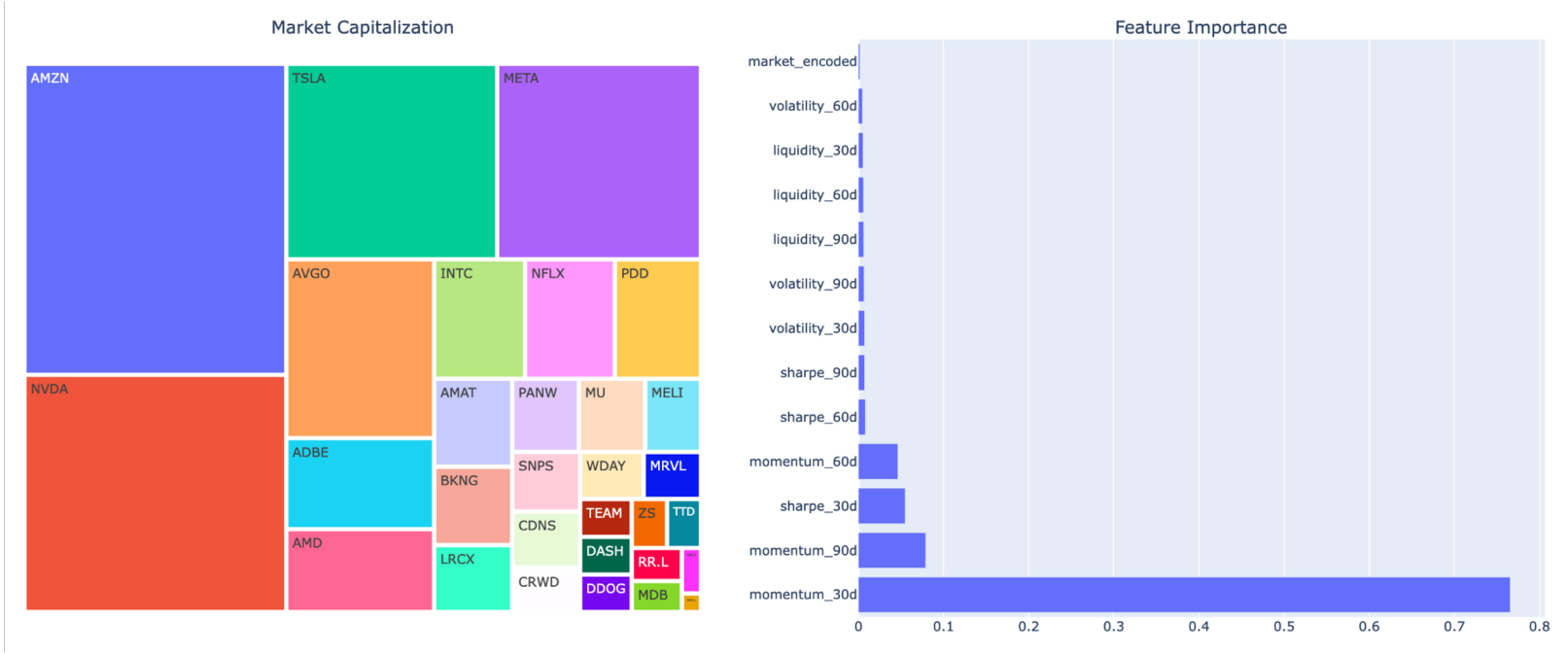
Select Top 30 Stocks

Second Method - Update Metrics

```
df['Log_Return'] = np.log(df['Close'] / df['Close'].shift(1))       # continuous compounded return between two consecutive periods
df['Cumulative_Return'] = (df['Close'] / df['Close'].iloc[0]) - 1   # total percentage return from the starting point

for window in [30, 60, 90]:
    df[f'Momentum_{window}d'] = df['Close'] / df['Close'].shift(int(hours * window)) - 1    # rate of acceleration of a stock's price
    df[f'Volume_EMA_{window}d'] = df['Volume'].ewm(span=int(hours * window)).mean()         # smoothed average of trading volume over
    df[f'Sharpe_{window}d'] = (df['Log_Return'].rolling(window=int(hours * window)).mean() * (hours * days)) / (df['Log_Return'].rolli
    df[f'Volatility_{window}d'] = df['Log_Return'].rolling(window=int(hours * window)).std() * np.sqrt(hours * days)    # stock's pric

return df
```

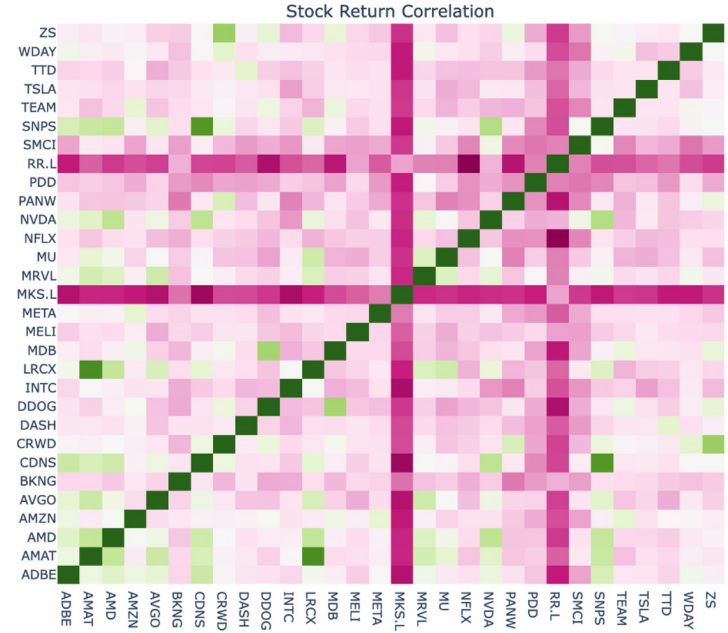
Select Top 30 Stocks

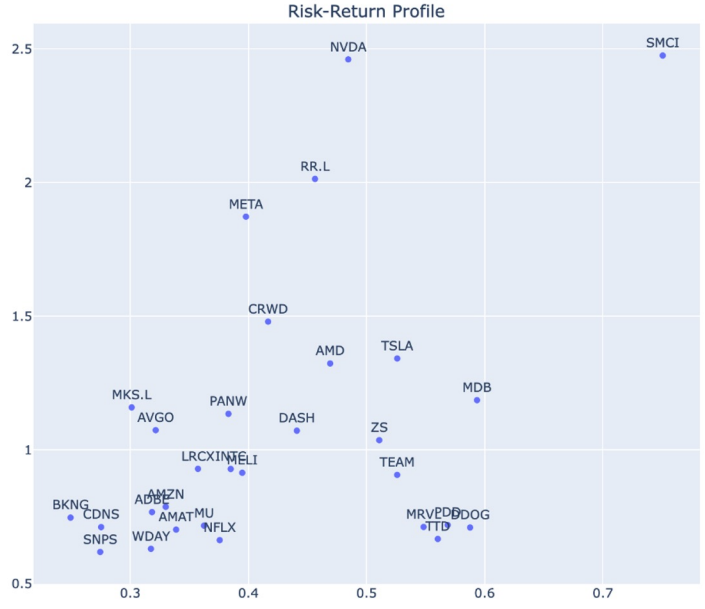
Second Method - Random Forest Regression



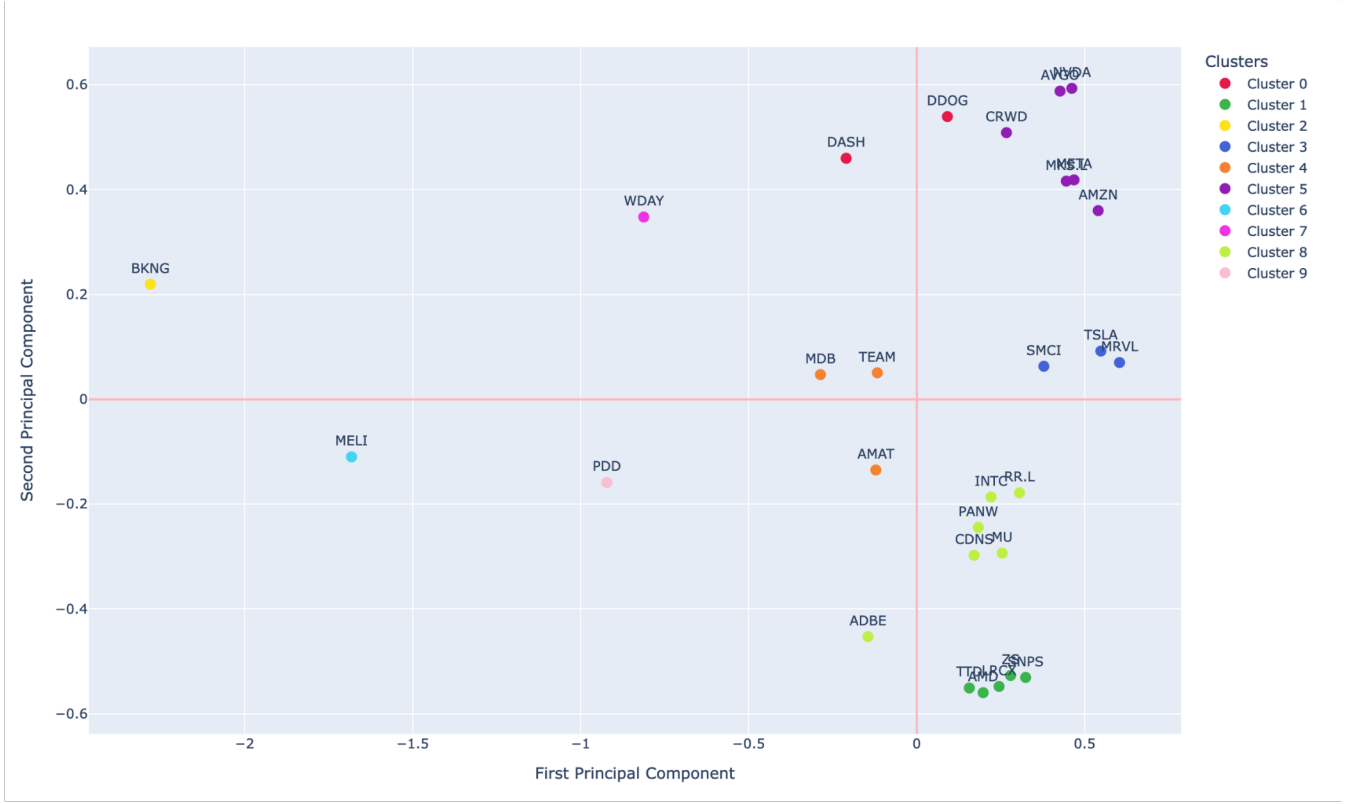
Select Top 30 Stocks

Second Method - Random Forest Regression





Select Final 10 Stocks

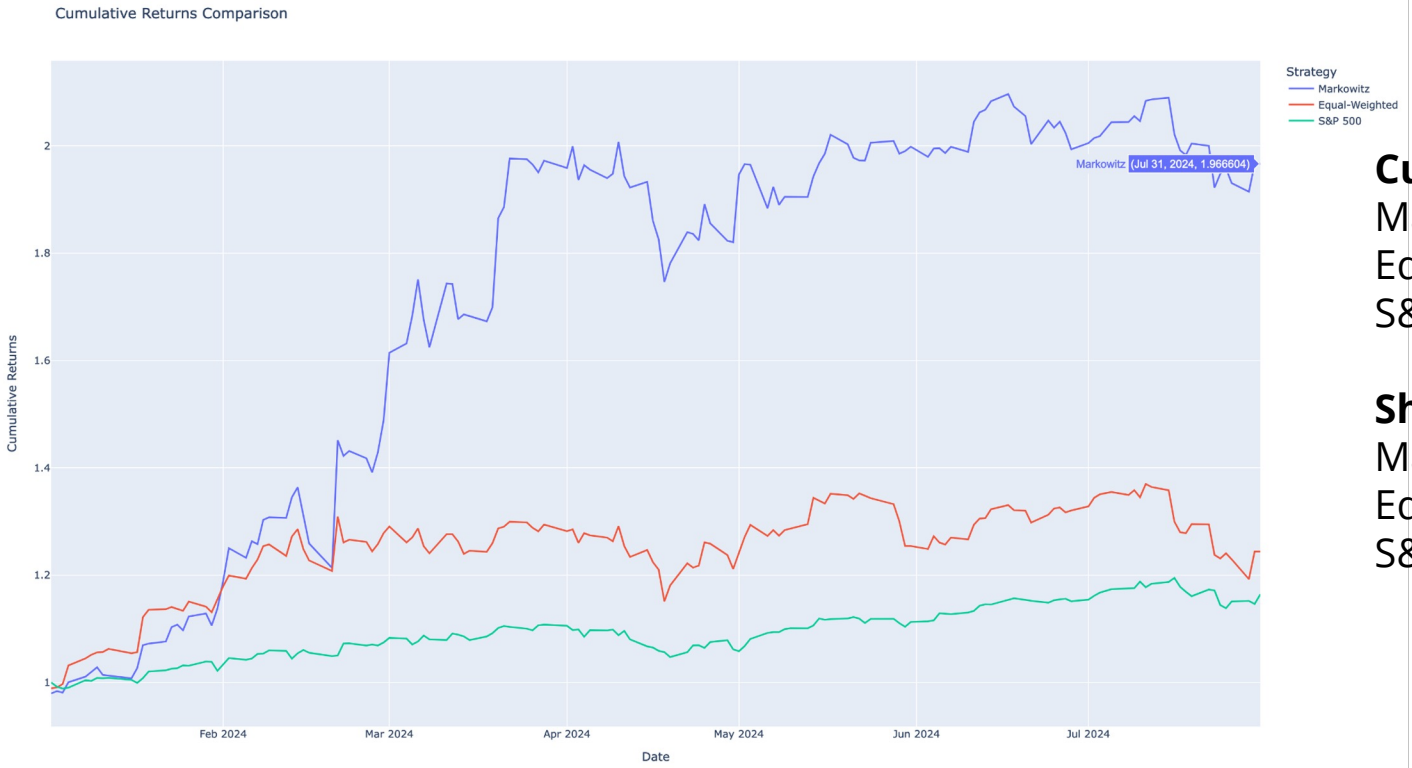


PCA

K-means Clustering

Select Final 10 Stocks

Markowitz Optimisation



Cumulative Returns:
Markowitz: 1.9666
Equal-Weighted: 1.2440
S&P 500: 1.1643

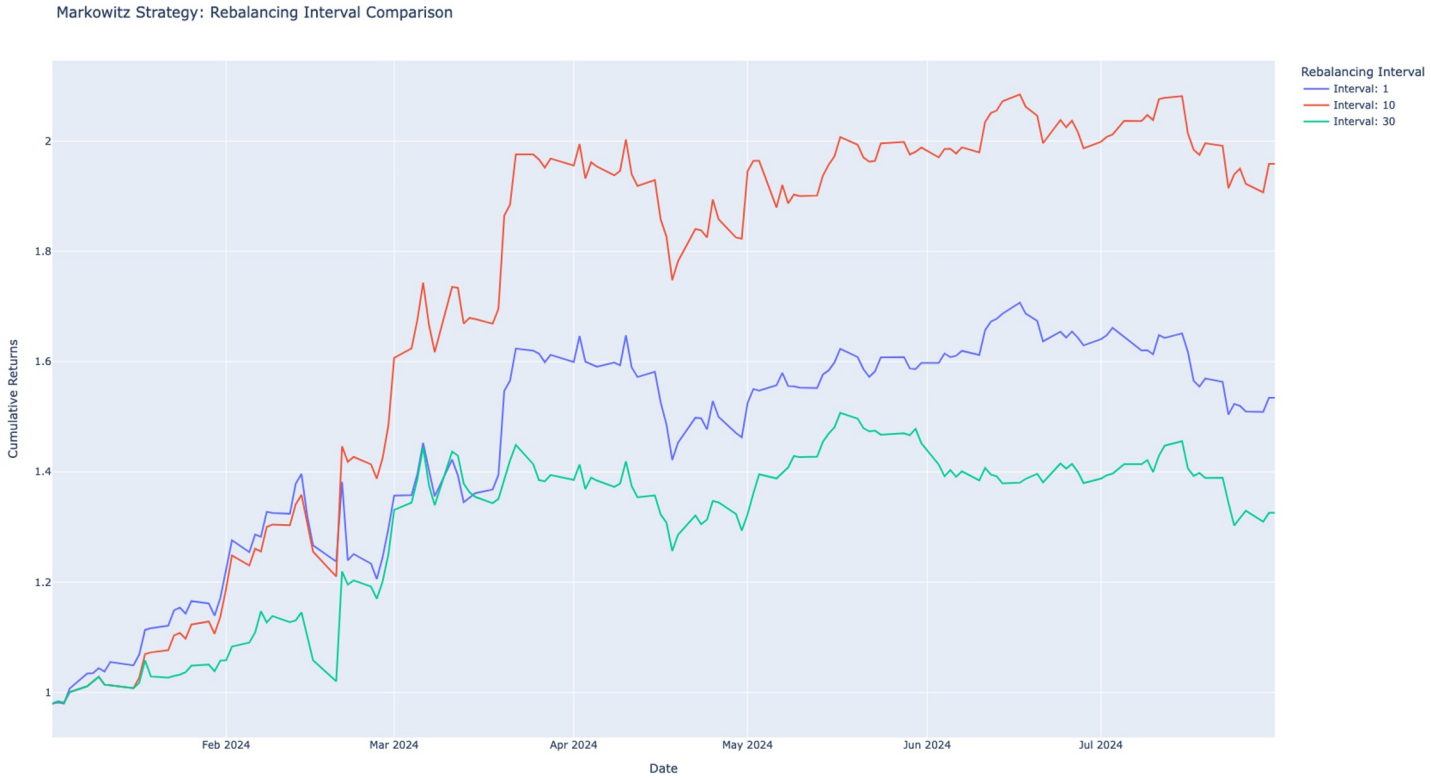
Sharpe Ratios:
Markowitz: 2.7112
Equal-Weighted: 1.2736
S&P 500: 1.9242

Markowitz Optimisation

Cumulative Returns:
Markowitz: 0.9716
Equal-Weighted: 0.9421
S&P 500: 1.0086

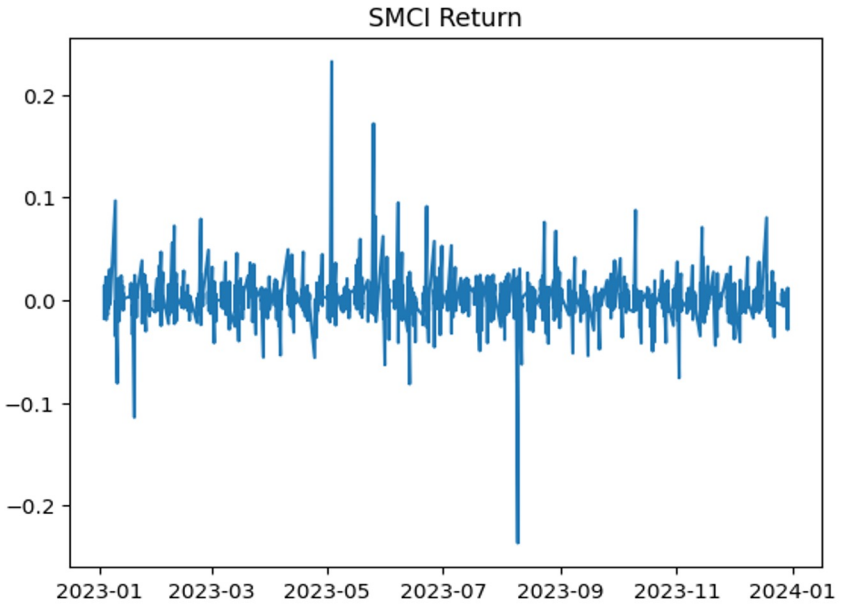
Sharpe Ratios:
Markowitz: -1.7629
Equal-Weighted: -2.3066
S&P 500: 0.3910

Markowitz Optimisation



Time Series Analysis

- Try to model returns using ARMA
- Look stationary
- Volatility clustering -> ARCH?
- ADF statistics p-value nearly 0



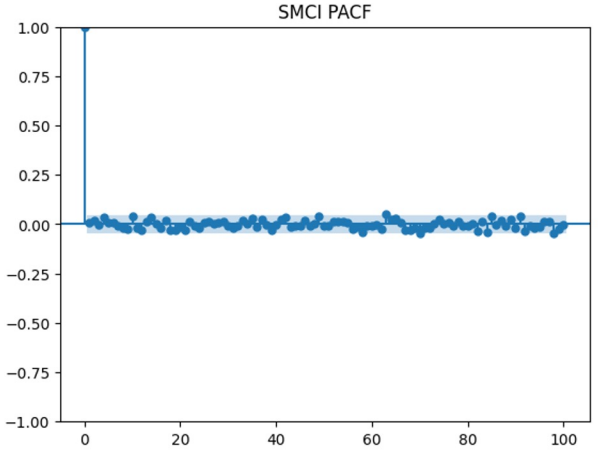
Hourly close price return for SMCI from 2023-01-01 to 2024-01-01

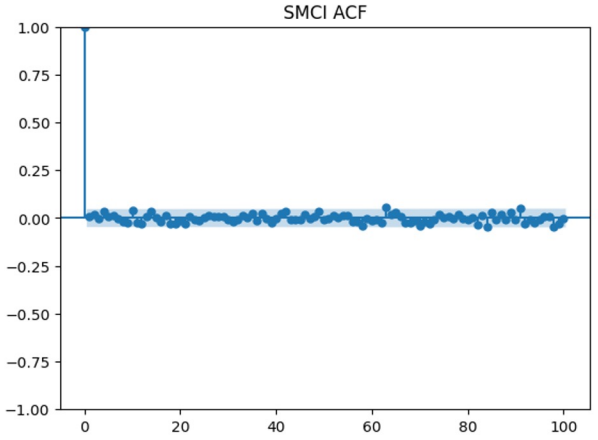
Time Series Analysis

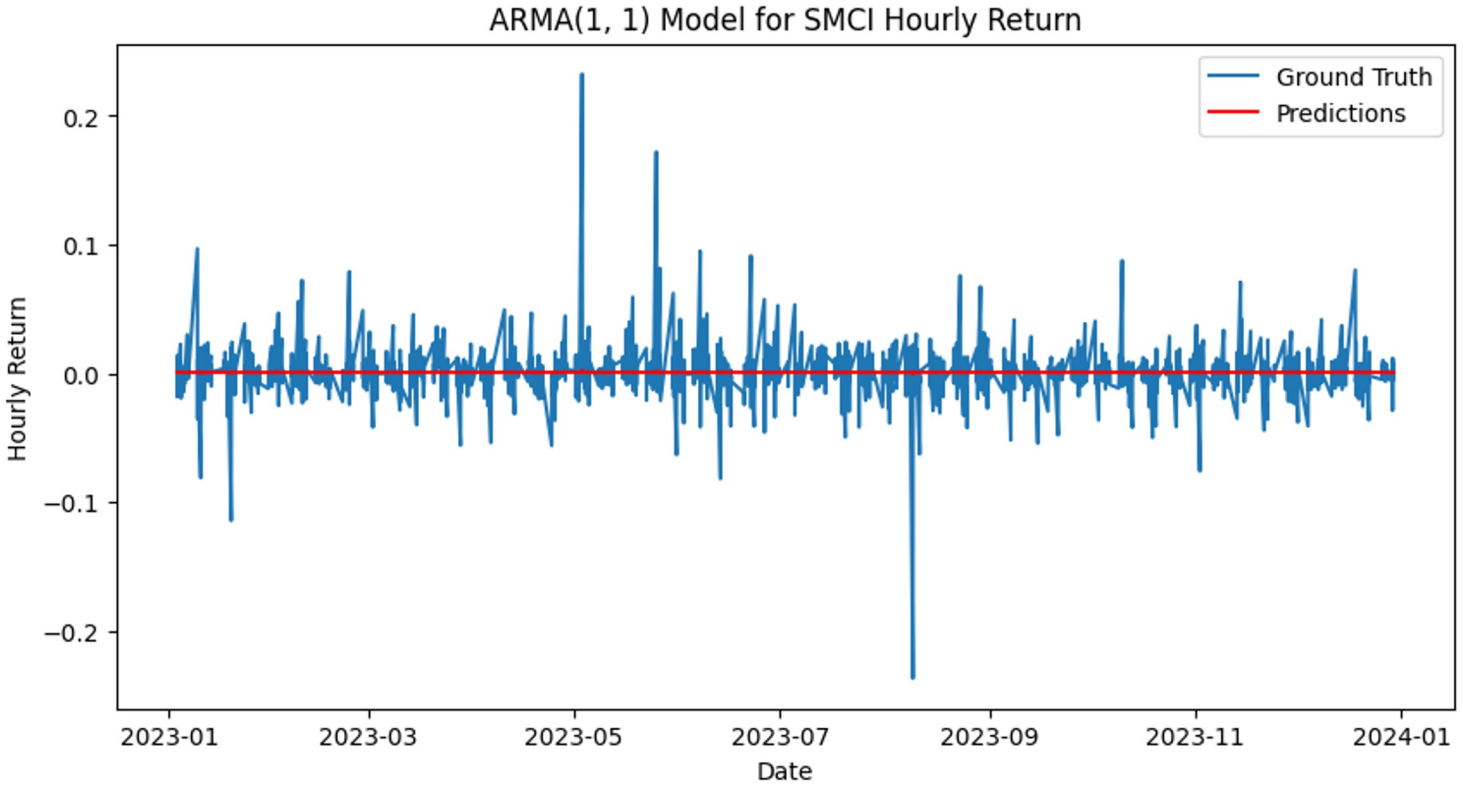
No hope using any ARMA/ARCH models

Non-linear dependence on the data?

- Feature engineering
- Non-linear models






ARMA(1, 1) Model for SMCI Hourly Return
Ground Truth
Predictions
Hourly Return
0.2
0.1
0.0
−0.1
−0.2
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date

SARIMAX Results

Dep. Variable:	Close	No. Observations:	1741
Model:	ARIMA(1, 0, 1)	Log Likelihood	4517.949
Date:	Fri, 02 Aug 2024	AIC	-9027.899
Time:	16:38:58	BIC	-9006.050
Sample:	0	HQIC	-9019.820
	- 1741		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0009	0.000	1.971	0.049	5.03e-06	0.002
ar.L1	0.0477	5.189	0.009	0.993	-10.123	10.219
ma.L1	-0.0421	5.190	-0.008	0.994	-10.215	10.130
sigma2	0.0003	2.45e-06	133.229	0.000	0.000	0.000

Trading Strategy

- Train a model to predict the position of a stock (based on expected return)
- Adjust the position every hour, equal weightage on position

Example: we have $100 to invest at the start

Signal	AMD	DASH	MSFT
9:00 am	long	neutral	short
10:00 am	long	short	long
11:00 am	neutral	long	shot

3 possible signals: long, neutral and short

% cash	AMD	DASH	MSFT
9:00 am	+50	0	-50
10:00 am	+33	-33	+33
11:00 am	0	+50	-50

Percentage of cash position in each stock

Negative means short selling, assume 0% extra margin

Feature Engineering

- Sliding windows ranging from 2 hours to 256 hours
  - SMA (Simple Moving Average)
  - EMA (Exponential Moving Average)
  - Historical Volatility
  - VWAP (Volume Weighted Average Price)
  - Momentum
- Macro-economic data
  - Market Premium
  - HML (high minus low)
  - SMB (small minus big)

Feature Engineering - SMA Price/Volume

- SMA (Simple Moving Average) of Close Price / Current Close Price
- SMA (Simple Moving Average) of Volume / Current Volume
- Added features for SMA_X, where X = [2, 4, 8, 16, 32, 64, 128, 256] trading hours as sliding window

$$\text{SMA} = \frac{A_1 + A_2 + ... + A_n}{n}$$

where:

A_n = the price of an asset at period n

n = the number of total periods

Feature Engineering - EMA / Volume

- EMA (Exponential Moving Average) of Close Price / Current Close Price
- EMA (Exponential Moving Average) of Volume / Current Volume
- Added features for EMA_X, where X = [2, 4, 8, 16, 32, 64, 128, 256] trading hours as sliding window

Formula for Exponential Moving Average (EMA)

$$EMA_{\text{Today}} = \left(\text{Value}_{\text{Today}} * \left(\frac{\text{Smoothing}}{1 + \text{Days}}\right)\right) + EMA_{\text{Yesterday}} * \left(1 - \left(\frac{\text{Smoothing}}{1 + \text{Days}}\right)\right)$$

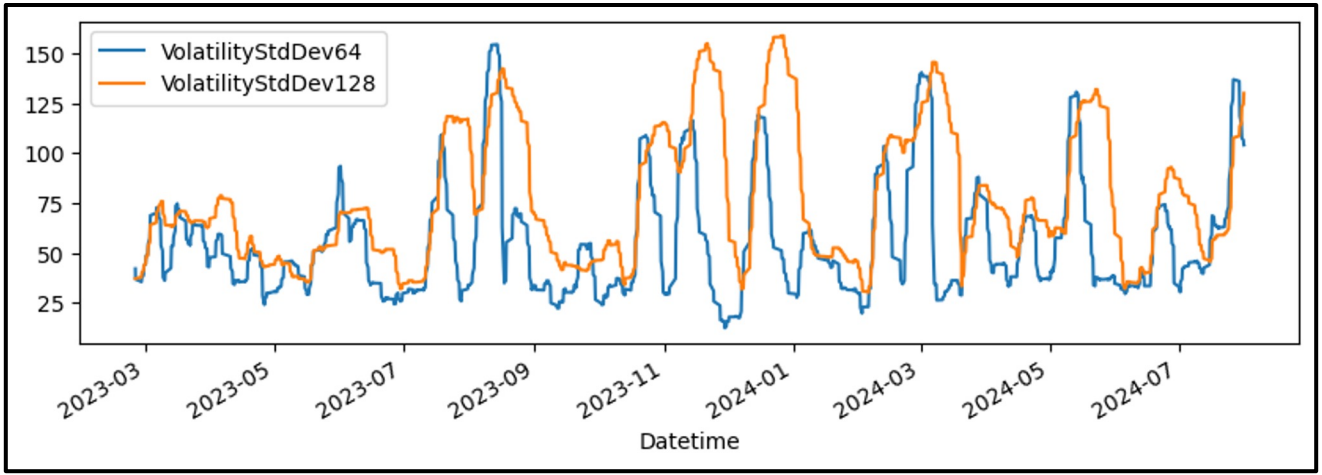
where:

$EMA = \text{Exponential moving average}$

Feature Engineering - Volatility

- Volatility is measured by rolling window standard deviation

- Added features for Volatility_X, where
  X = [2, 4, 8, 16, 32, 64, 128, 256] trading hours as sliding window



Feature Engineering - Fama French Mkt Return

- Retrieved Fama French Mkt Returns model
- Appended Ra onto feature dataset for every stock
- https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

$$R_a = R_f + \beta_1 (Mkt\text{-}R_f) + \beta_2 (SMB) + \beta_3 (HML) + \alpha$$

R_a – Expected return on asset
R_f – Risk-free rate
β_{123} – Factor coefficient
Mkt- R_f – Market risk premium
SMB (Small Minus Big) – Excess returns of small cap over large cap
HML (High Minus Low) – Excess returns of value stocks over growth stocks
α - Intercept

Feature Engineering - VWAP

- Volume Weighted Average Price / Current Close Price
- Added features for VWAP_X, where
  X = [2, 4, 8, 16, 32, 64, 128, 256] trading hours as sliding window

$$\textbf{VWAP} = \frac{\textbf{Cumulative Typical Price} \times \textbf{Volume}}{\textbf{Cumulative Volume}}$$

where:

$$\text{Typical Price} = \text{High price} + \text{low price} + \text{closing price}/3$$

$$\text{Cumulative} = \text{Total trades since the trading session opened}$$

Feature Engineering - MACD (dropped)

- Attempted MACD as a feature (moving average convergence/divergence indicator)
- Feature is meant to help traders identify momentum in the market
- Dropped this feature due to high variance and low pearson R value with target

MACD Formula

$$\text{MACD} = \text{12-Period EMA} - \text{26-Period EMA}$$

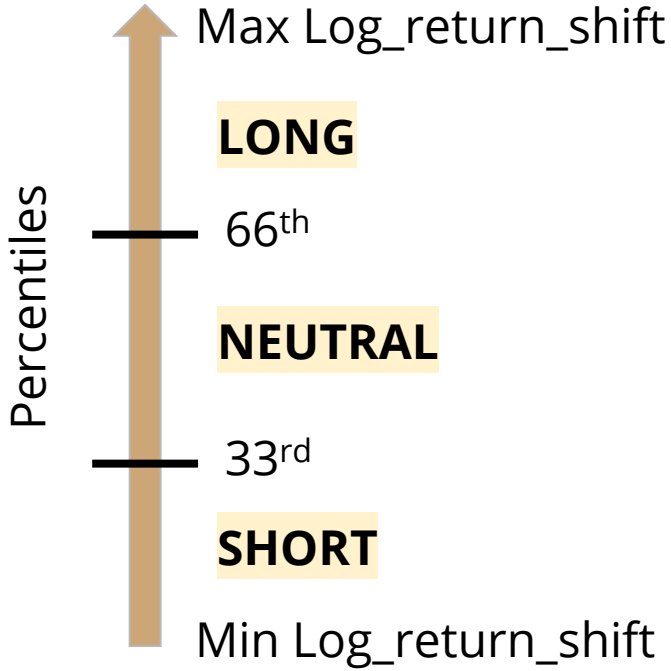
Feature Engineering - SelectKBest

- We ran SelectKBest on the final 72 features created to select the most significant 20 features

- SelectKBest runs an f_classifier, which selects the best 20 features based on the highest ANOVA F-value between label/feature for classification tasks

```
['EMAVolumeDiff2',
 'SMAVolumeDiff2',
 'EMAVolumeDiff4',
 'SMAVolumeDiff4',
 'EMAVolumeDiff8',
 'SMAVolumeDiff8',
 'Volatility8',
 'EMAVolumeDiff16',
 'SMAVolumeDiff16',
 'Volatility16',
 'EMAVolumeDiff32',
 'SMAVolumeDiff32',
 'Volatility32',
 'EMAVolumeDiff64',
 'SMAVolumeDiff64',
 'EMAVolumeDiff128',
 'SMAVolumeDiff128',
 'EMAVolumeDiff256',
 'SMAVolumeDiff256',
 'FamaFrenchMktReturns']
```

FCN



- Create targets based on Log_Returns for next timestamp
- Targets: Short, Neutral, Long
- Train-val-test splits:
  - Train: 2023-01 - 2023-12
  - Val: 2024-01 - 2024-06
  - Test: 2024-07

scikit learn

PyTorch

Sklearn FCN

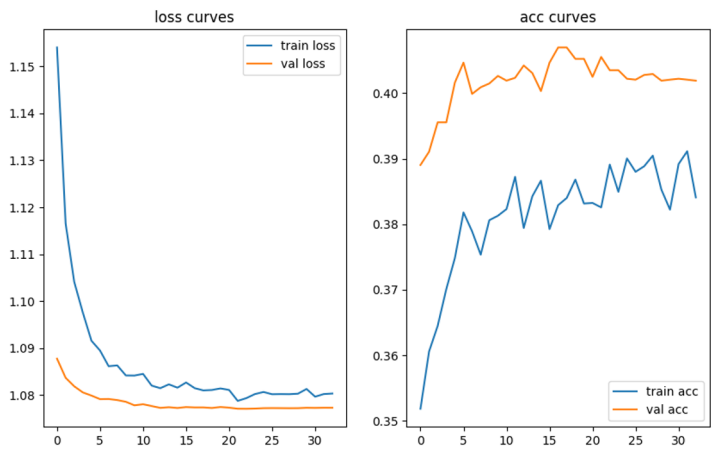
```
# Define the parameter grid
param_grid = {
    'hidden_layer_sizes': [(8,), (16,), (32,), (64,),
    'activation': ['relu', 'tanh'], # Different activ
    'solver': ['adam', 'sgd'], # Different solvers
    'alpha': [0.0001, 0.001], # Different regularizat
    'learning_rate': ['constant', 'adaptive'], # Diff
}

for params in alive_it(ParameterGrid(param_grid)):
    model = MLPRegressor(max_iter=500, **params) # in
    model.fit(X_train, y_train)
```

Limitations:

- Sklearn doesnt show real-time train-val loss curves
- Unable to debug overfitting or underfitting

Pytorch FCN

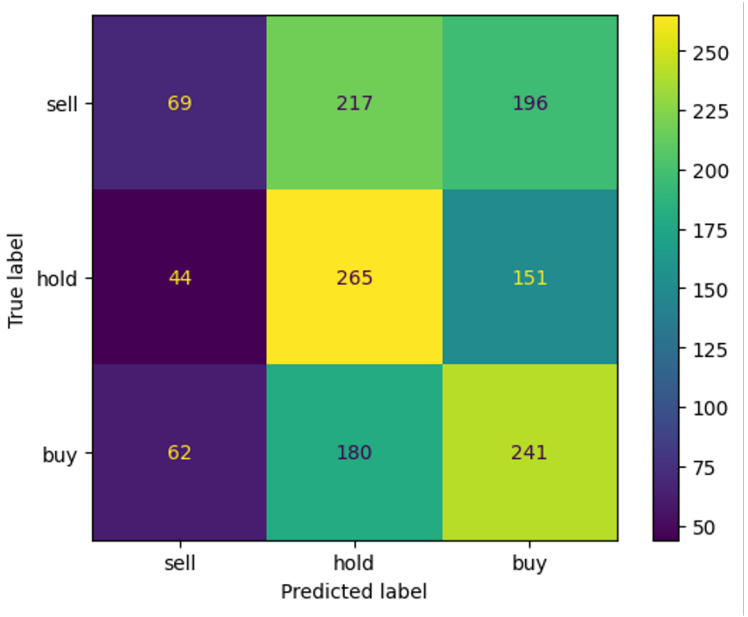
Sample Plots



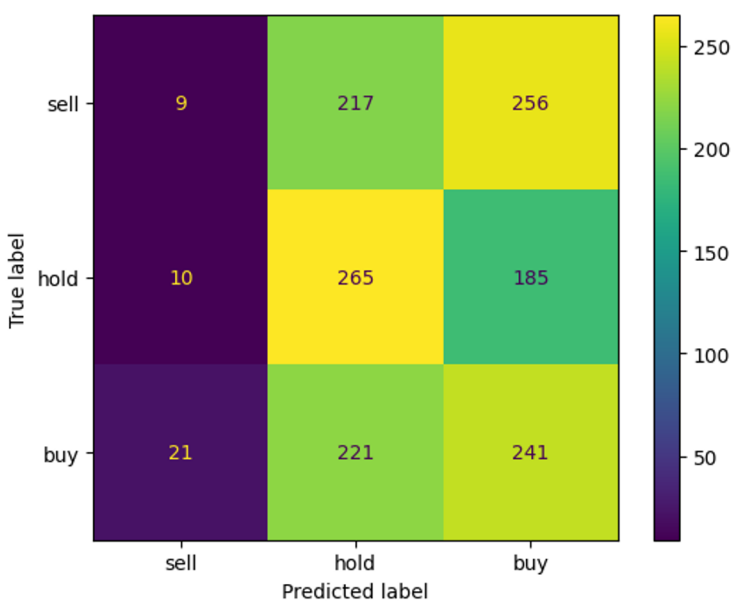
- Val loss drop very gradual
- Weird issue of val loss starting lower than train loss
- Implemented dropout and batchnorm
- Tried different model architectures

Sklearn vs Pytorch FCN

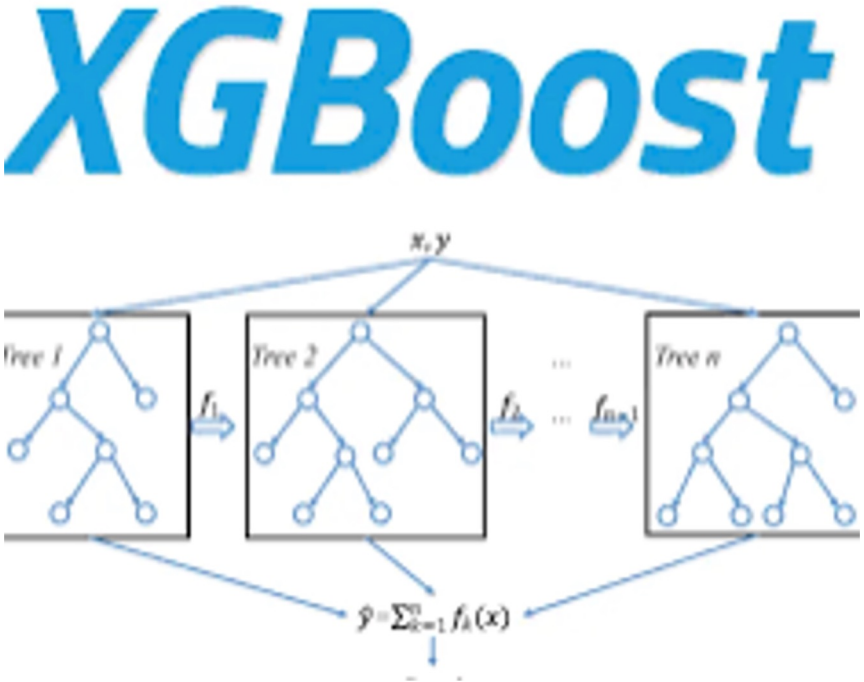
Sklearn FCN
Test-acc: 40.35 %



Pytorch FCN
Test-acc: 36.14 %



XGBoost



Result

Investment of 1 dollar, final portfolio value

	Hold S&P 500	Equal-Weighted	Markowitz	Neural Network (Sklearn)
Final Value	1.0086	0.9721	0.9716	0.9962

Limitation of the strategy

- Predicted outcome just slightly better than the baseline (always guessing the historical majority category)
- Strategy updating the position is not optimal
  - Every stock gets the same weightage
  - We should exit the position when earning money
  - Stop-loss point to minimise loss; take-profit point to lock in profit
- Only little stock fluctuation is explained by price and volume
- Unrealistic modeling assumptions
  - Each stocks have different price dynamics
  - Distribution changes over time, even as short as 1 hour
  - Does not take into account time dependence

Q and A