

Trend-Driven Stock Trading


Leveraging hashtag data to predict fashion-stock movements

By: Audrey Ghosh, Irena Chen, Lydia Shen, Maddy Duff



Table of contents

- 01** Background and motivation
- 02** Strategy
- 03** Backend: Hashtags and stocks

- 04** Backend: Machine learning
 - 05** Results
 - 06** Limitations and Improvements
- 



01

Background and motivation



The Influence of TikTok

- Over 1 billion monthly users
- 92% of TikTok users have reported purchased something as a result of watching a video on the platform
- More than half of US Gen Z will consult TikTok Shop this holiday season, according to estimates
- Purchasing trends correlate with stock prices

Focus: Athletic-Fashion Space

- Rapidly growing industry

“For every sponsored post TikTok influencers made in 2019, they created 2.3 of such posts in 2020. This constitutes an increase of 130 percent. A different study showed that most popular TikTok influencers could count on over half a million **views per post** in 2020.” - Statista

- Specific niche within fashion



02

Strategy

**Our general approach, signals and indicators, and
determining thresholds**



Strategy Overview

Trading stocks based on patterns in groups of correlated hashtags

Why hashtags?

- Objective way of grouping videos together under categories
- Help videos circulate and content visibility
- Easily traceable for capturing trends (viral videos use same/similar hashtags)
- Content creators are often tasked by industries to promote hashtags and increase popularity of certain products
- Drive consumer purchasing (hashtags associated with specific brands – #lulu, #nike)

Driving Indicator:

Like Count/Play Count for various hashtags

- Accounts for popularity and virality
- Engagement rate

Strategy Overview

Train each ticker on hashtag data (linear regression model)

- Use a 3-day rolling average to smooth out daily volatility in video interactions without losing micro trend information and accommodate sparse data
- Pick top 3 hashtags on the training period for each ticker, combine and normalize into one signal
- Test this signal, as well as one using all combined hashtags, for comparison, on data after 2024
- Trade on this 'top 3 hashtags' combined signal for each ticker, buying, selling or holding based on specific thresholds

03

Backend, Part 1

**Coding: Exploring what works; Which hashtags do we use?
Which stocks map to those hashtags?**

Finding inter-hashtag correlations

How do we know which hashtag data to use?

Noticed there were about 10 hashtags popular in the “athleisure” space on TikTok.

These were: “#athleisure”, “#athleticwear”, “#sportswear”, “#activewear”, “#pilates”, “#gymgirl”, “#yogawear”, “#activelifestyle”, “#activewearfashion”, “#gymwear”

But, using data from all hashtags would lead to some problems:

- 1) **Dilution** of data: Some hashtags were more correlated to certain stocks than others. This makes sense! For example, “#gymwear” focuses on more athletic, functional clothing, meanwhile “#athleticwearfashion” focuses on more fashion-oriented brands, like Lululemon.
- 2) Correlation and Grouping issues may lead to **misdirected signals**: As we saw with the correlation analysis, hashtags that are more closely related to each other (like “activewear” and “gymwear”) might be grouped together as the most correlated and support each other in indicating whether we should buy or sell. But including broader and less related hashtags may confuse the analysis.

Solution!

Select the top three hashtags that correlate most with each other (correlation matrix on next slide) AND with the performance of the stock we are actively trading.

```
# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(10, 8)) # Adjust the size of the heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, fmt='.2f', cbar=True)

# Set title and labels
plt.title("Correlation Matrix of Hashtags (Without Time Differences)", fontsize=16)
plt.xlabel("Hashtags", fontsize=12)
plt.ylabel("Hashtags", fontsize=12)

# Show the plot
plt.show()

# Now, let's find the best group of 3 most intercorrelated hashtags

# List all combinations of three hashtags (using the actual hashtag names)
hashtag_combinations = list(itertools.combinations(merged_df_cleaned.columns, 3))

# Initialize a list to store the sum of correlations for each combination of three hashtags
correlation_sums = []

# Loop through each combination of three hashtags
for comb in hashtag_combinations:
    hashtag1, hashtag2, hashtag3 = comb

    # Get the pairwise correlations between the three hashtags
    corr_1_2 = correlation_matrix.loc[hashtag1, hashtag2]
    corr_1_3 = correlation_matrix.loc[hashtag1, hashtag3]
    corr_2_3 = correlation_matrix.loc[hashtag2, hashtag3]

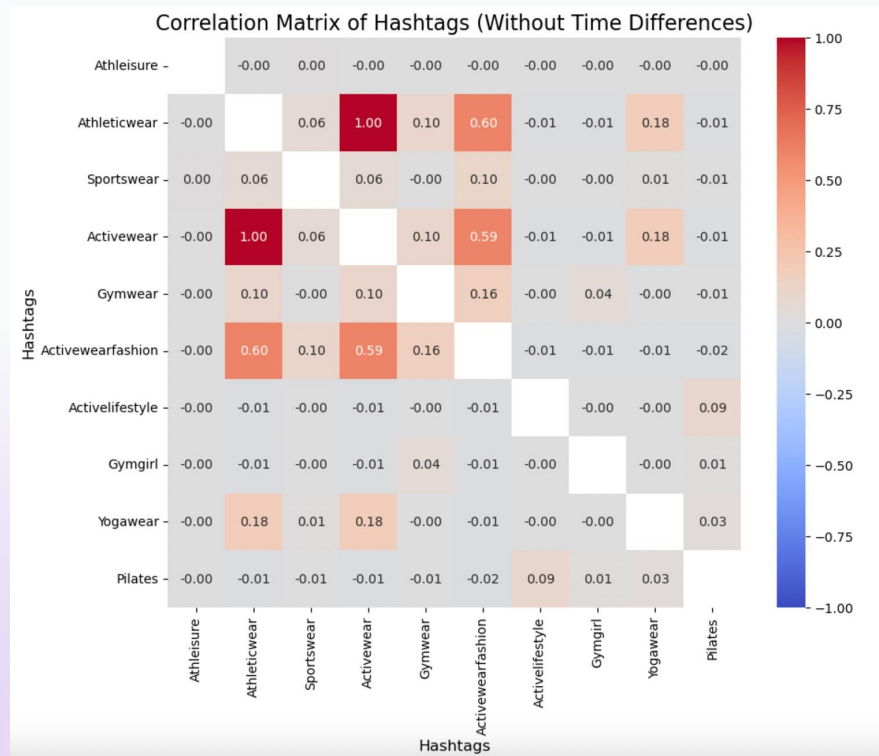
    # Compute the sum of the pairwise correlations
    correlation_sum = corr_1_2 + corr_1_3 + corr_2_3

    # Append the sum and the combination to the list
    correlation_sums.append((correlation_sum, comb))

# Sort the list by the sum of correlations in descending order to find the top group
best_group = max(correlation_sums, key=lambda x: x[0])

# Map internal column names to actual hashtag names for the best group
best_group_names = [hashtag_mapping[hashtag] for hashtag in best_group[1]]

# Print the best group of 3 hashtags and their correlation sum
print("\nBest Group of 3 Most Intercorrelated Hashtags:")
print(f"Hashtags: {best_group_names}")
print(f"Sum of Pairwise Correlations: {best_group[0]}")
```



Top 3 most correlated hashtags:
 #activewear, #athleticwear, #activewearfashion



04

Backend, Part 2

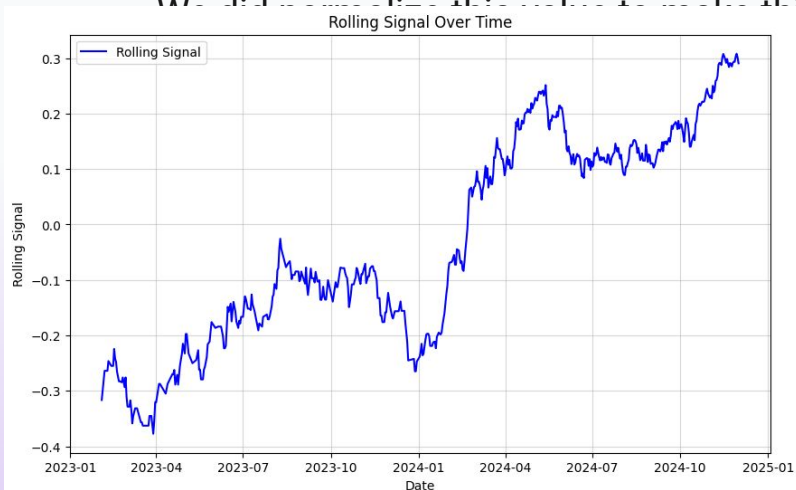
Coding: Working with sparse data and coding our strategy.



Noticing Trends – Naive Approach

Signal based on trends

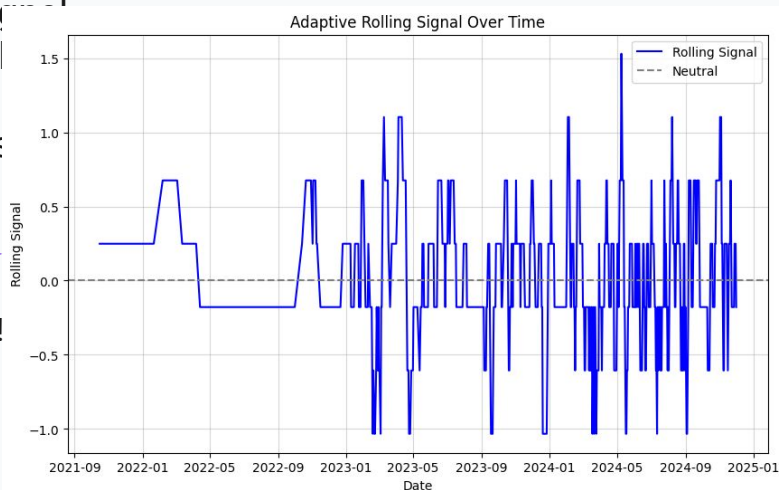
- We made every hashtag into a signal saying buy/sell if it was over a certain (variable) threshold
 - The variability accounted for longer trends (ie athleisure coming in and out of fashion cyclically) to avoid stuff like this:
- To avoid volatility (and the variability of what the TikTok algorithm promotes) we wanted to use a combination of all hashtags as the signal
- We started by trying to just combine them all!



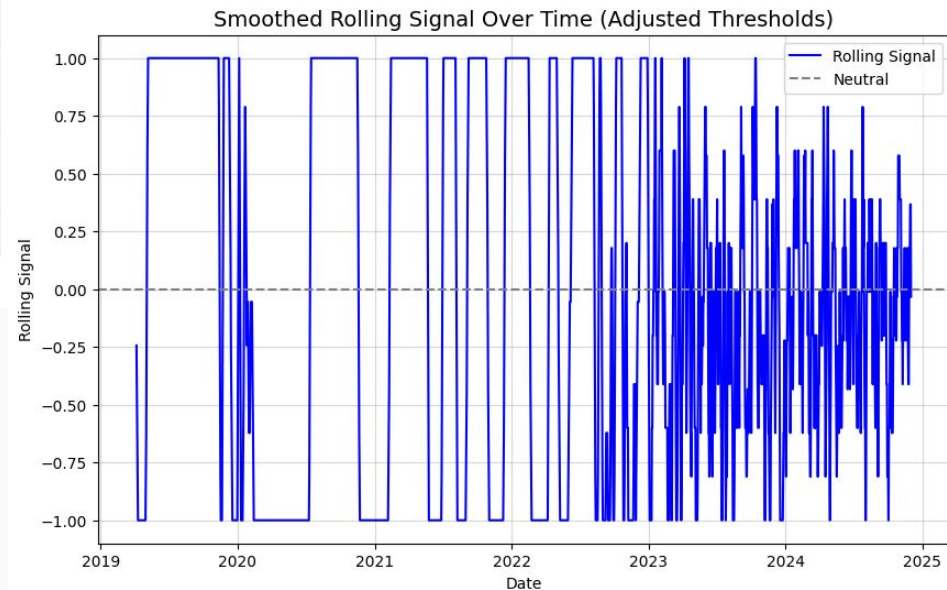
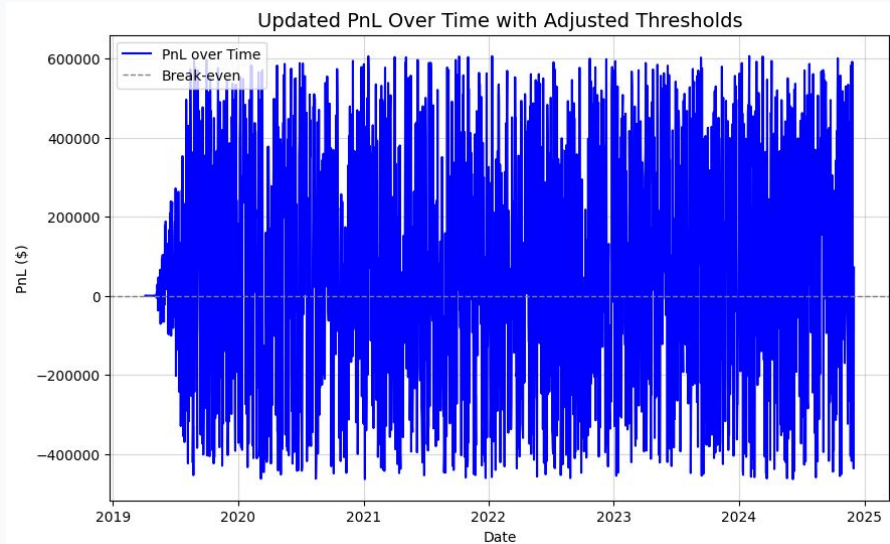
is a new signal
about a daily

between an absolute
or strong

SO MUCH
PRETTIER!!



Visualizations



A new model

- Split data into **training** and **testing** data
- Trained a basic **linear regression model** on training dataset
 - Took a **3-day rolling average** with upper and lower thresholds to create a signal for each respective hashtag
 - Smooth this signal for 5 days to get rid of daily volatility without smoothing over smaller changes.
- Take the **top 3 performing hashtags** on a given ticker, thus creating a 'combined' signal

New model (continued)

Individual forecasting:

We split the data into two components- between 2022 and 2024, and during 2024. This enabled us to look at less sparse data.

We then split up the tickers and used the model on each.

The total pnl graph you saw earlier shows the combined pnl when each ticker is traded on individually, given their unique 'top 3 hashtag' signal.

Train the models...

```
# Now select top 3 performing hashtags
top_hashtags = sorted(hashtag_pnl.items(), key=lambda x: x[1], reverse=True)[:3]
top_hashtags = [item[0] for item in top_hashtags]
print(f"Top 3 performing hashtags for {ticker}: {top_hashtags}")

# Compute 'Top3_Signal' by averaging signals of top 3 hashtags
df_combined['Top3_Signal'] = df_combined[top_hashtags].mean(axis=1)

# Now split into training and testing data
df_combined.reset_index(inplace=True)
df_combined.rename(columns={'index': 'date'}, inplace=True)
train_df = df_combined[df_combined['date'] <= train_end_date]
test_df = df_combined[df_combined['date'] >= test_start_date]
```

```
# Train models using 'Combined_Signal' and 'Top3_Signal'
# First, use 'Combined_Signal'
X_train_combined = train_df[['Combined_Signal']]
y_train = train_df['Close_Change']
model_combined = LinearRegression()
model_combined.fit(X_train_combined, y_train)

X_full_combined = df_combined[['Combined_Signal']]
df_combined['Predicted_Change_Combined'] = model_combined.predict(X_full_combined)
df_combined['Predicted_Change_Combined'] = df_combined['Predicted_Change_Combined'].clip(-0.05, 0.05)
```



05

Results

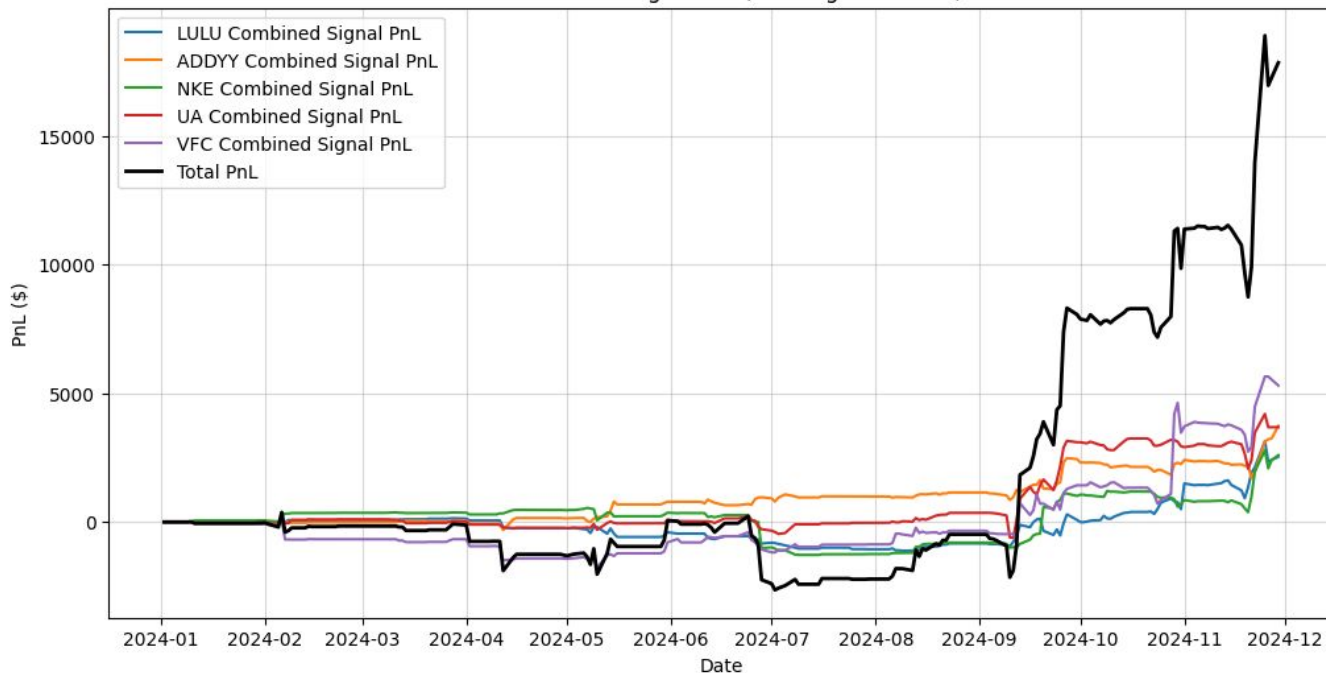


Results

Black line:

Total PnL Sharpe Ratio: 1.9001

PnL Over Testing Period (Starting from Zero)

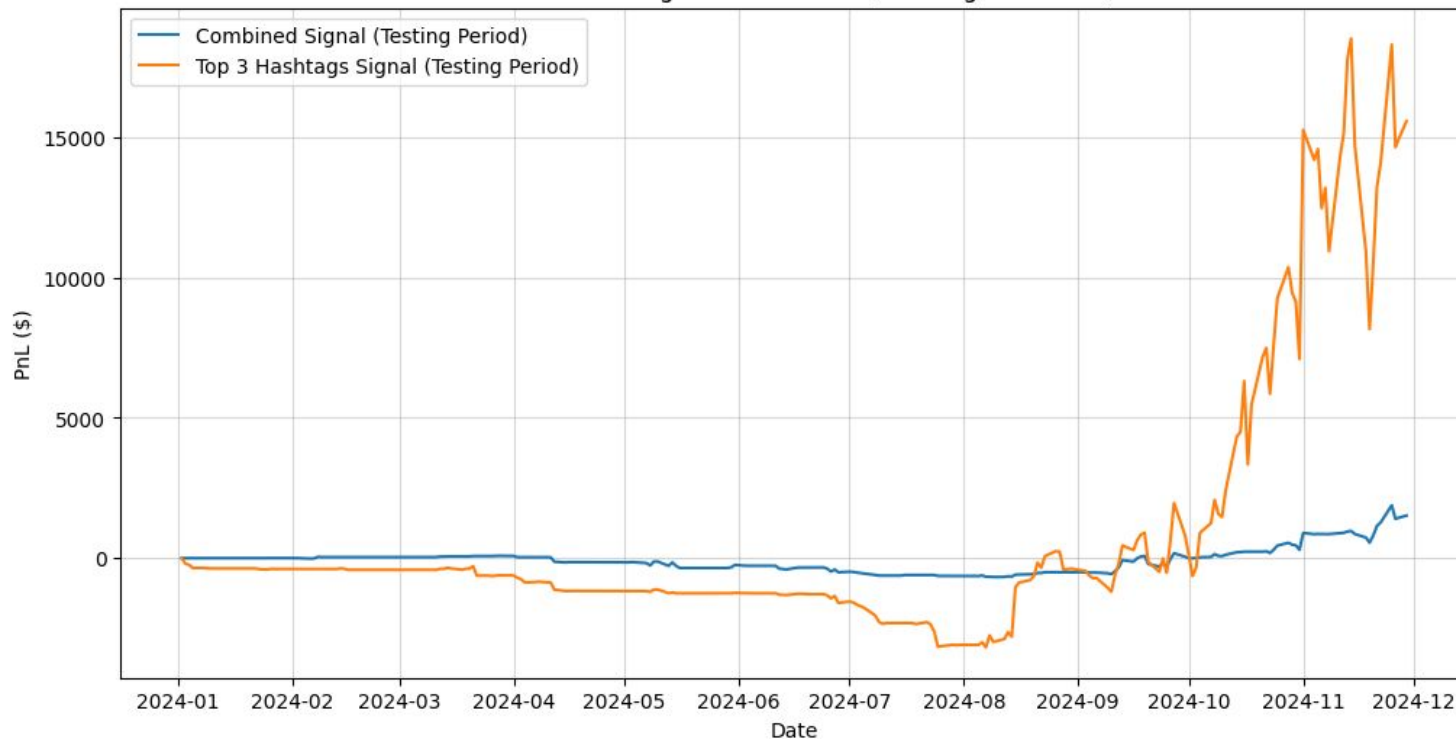


Total PnL during Testing Period: \$17833.27
Maximum Drawdown during Testing Period: \$3015.41
Total Invested Amount during Testing Period: \$637558.30
Return on Investment during Testing Period: 2.7971%

LULU

Top 3 performing hashtags for LULU: ['Activelifestyle', 'Activewearfashion', 'Pilates']

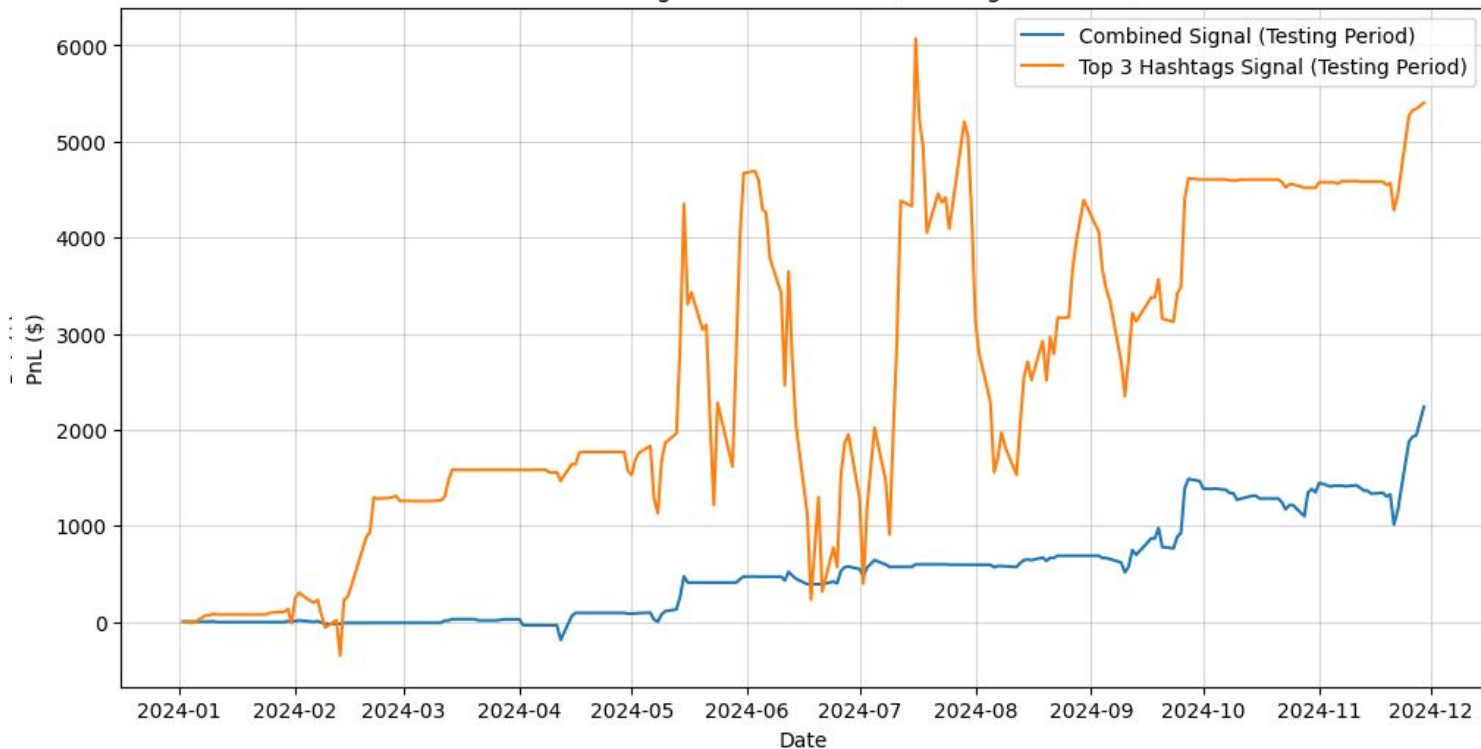
PnL Over Testing Period for LULU (Starting from Zero)



ADDYY

Top 3 performing hashtags for ADDYY: ['Athleisure', 'Gymwear', 'Sportswear']

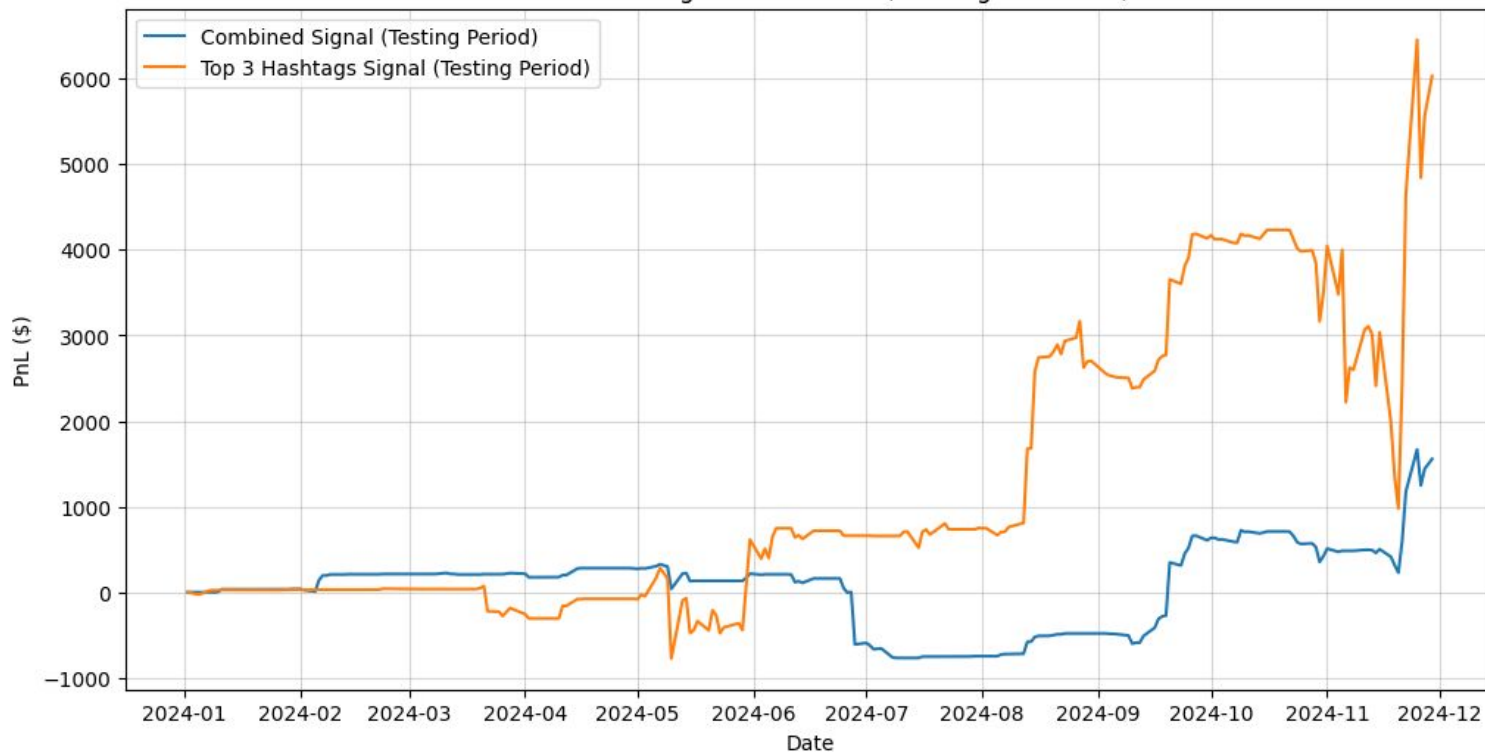
PnL Over Testing Period for ADDYY (Starting from Zero)



NKE

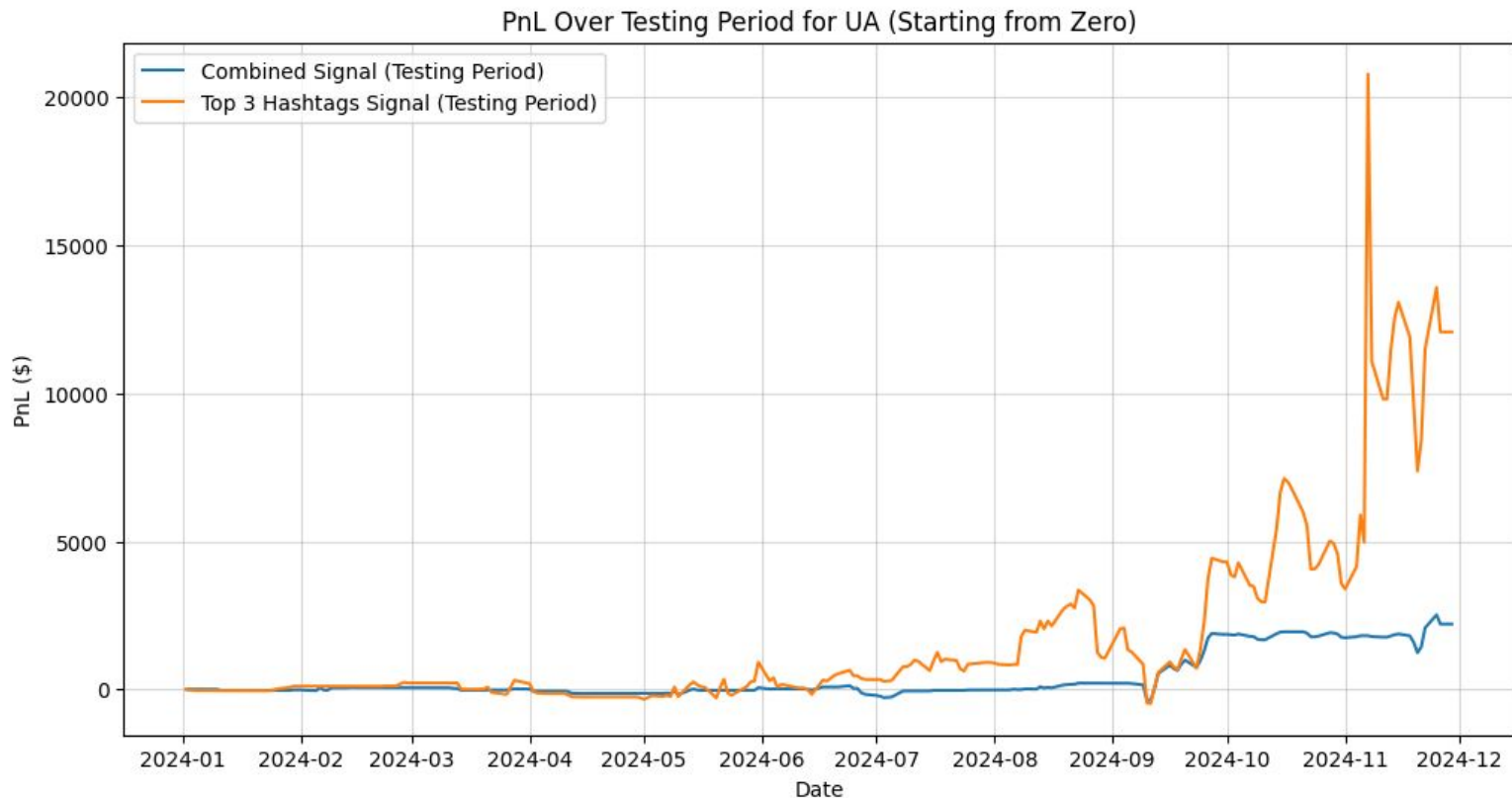
Top 3 performing hashtags for NKE: ['Sportswear', 'Activewearfashion', 'Gymgirl']

PnL Over Testing Period for NKE (Starting from Zero)



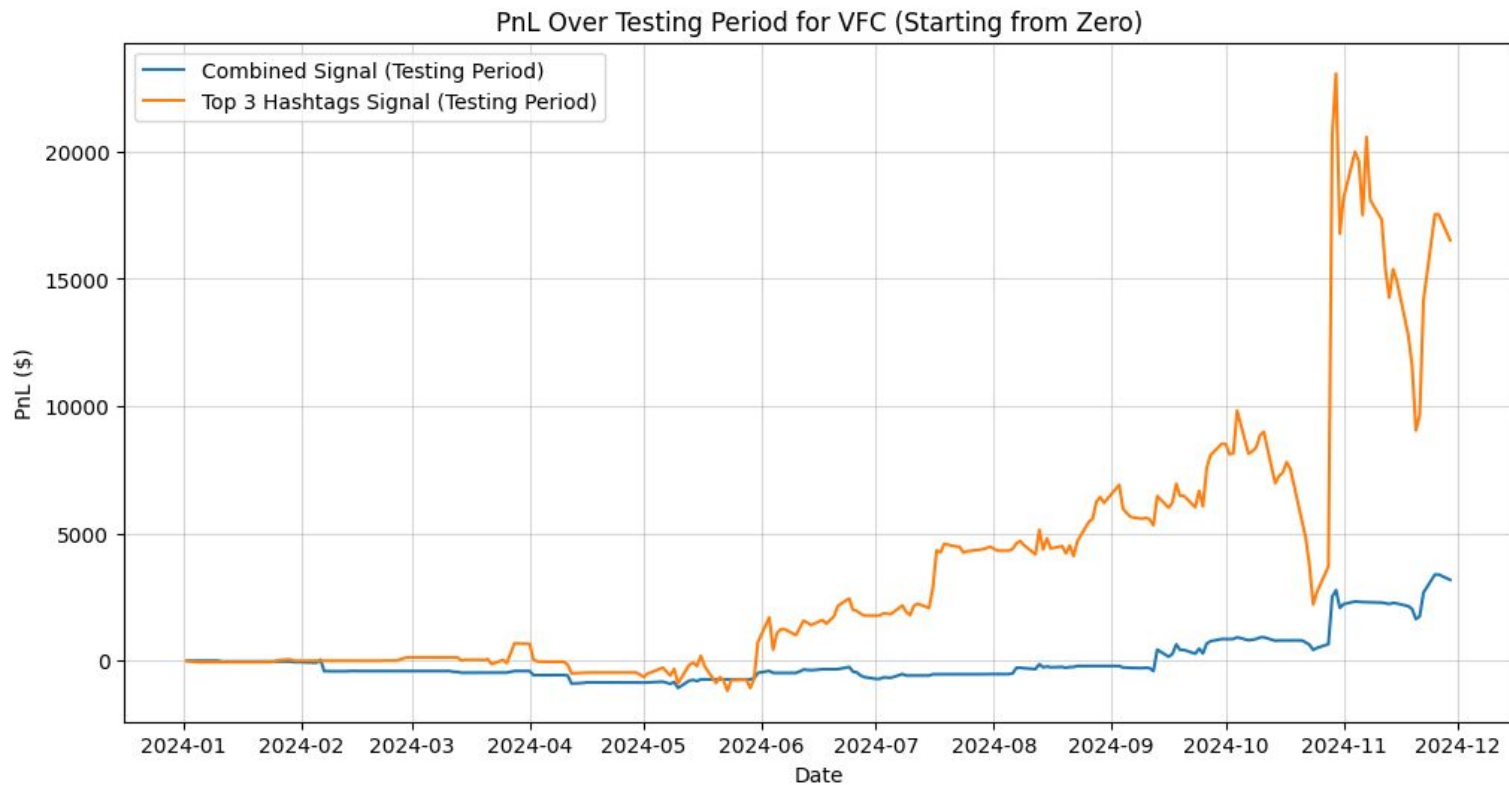
UA

Top 3 performing hashtags for UA: ['Sportswear', 'Activelifestyle', 'Gymgirl']



VFC

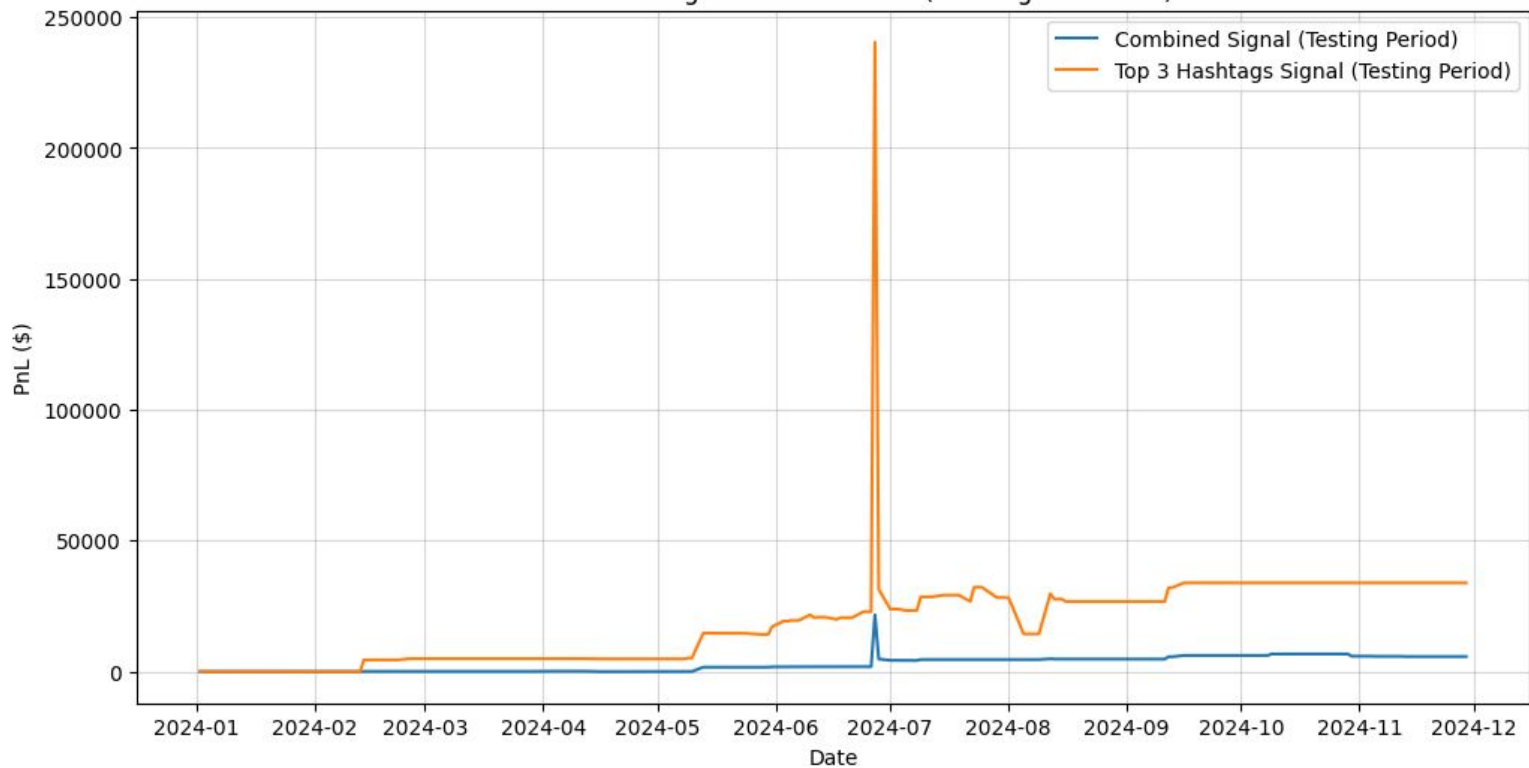
Top 3 performing hashtags for VFC: ['Gymgirl', 'Sportswear', 'Activelifestyle']



ASCCF

Top 3 performing hashtags for ASCCF: ['Athleisure', 'Gymwear', 'Sportswear']

PnL Over Testing Period for ASCCF (Starting from Zero)



Confirmation

```
for aggregated_df in hashtags_list:
    for prices in prices_list:

        # Print the names of the dataframes (optional for debugging)
        #print_dataframe_name(aggregated_df)
        #print_dataframe_name(prices)
        count += 1

        # Merge 'aggregated_df' and 'prices' on 'date' (use inner join to keep only common dates)
        merged_df = pd.merge(
            aggregated_df,
            prices[['Date', 'Open', 'Close']],
            left_on='date',
            right_on='Date',
            how='inner'
        )

        # Calculate daily change in ticker price
        merged_df['ChangePriceDay'] = merged_df['Open'] - merged_df['Close']

        # Only keep relevant columns (for this example, let's use 'diggCount', 'playCount', 'ChangePriceDay', and 'diggCount/playCount')
        merged_df = merged_df[['date', 'diggCount', 'playCount', 'diggCount/playCount', 'ChangePriceDay']]

        # Calculate percentage change in 'diggCount/playCount'
        merged_df['change in d/p'] = merged_df['diggCount/playCount'].pct_change()

        # Calculate percentage change per time
        merged_df['deltaTime'] = merged_df['date'].diff().dt.days

        # Calculate the change in 'diggCount/playCount' per time
        merged_df['change'] = merged_df['change in d/p'] / merged_df['deltaTime']

        # Keep only the necessary columns for correlation analysis
        merged_df = merged_df[['date', 'change', 'ChangePriceDay', 'diggCount/playCount']]

        # Calculate the correlation between 'diggCount/playCount' and 'ChangePriceDay'
        merged_df_clean = merged_df.dropna(subset=['diggCount/playCount', 'ChangePriceDay']) # Ensure no NaNs for correlation
        corr, p_value = pearson(merged_df_clean['diggCount/playCount'], merged_df_clean['ChangePriceDay'])
        #print(count, "Correlation:", corr)

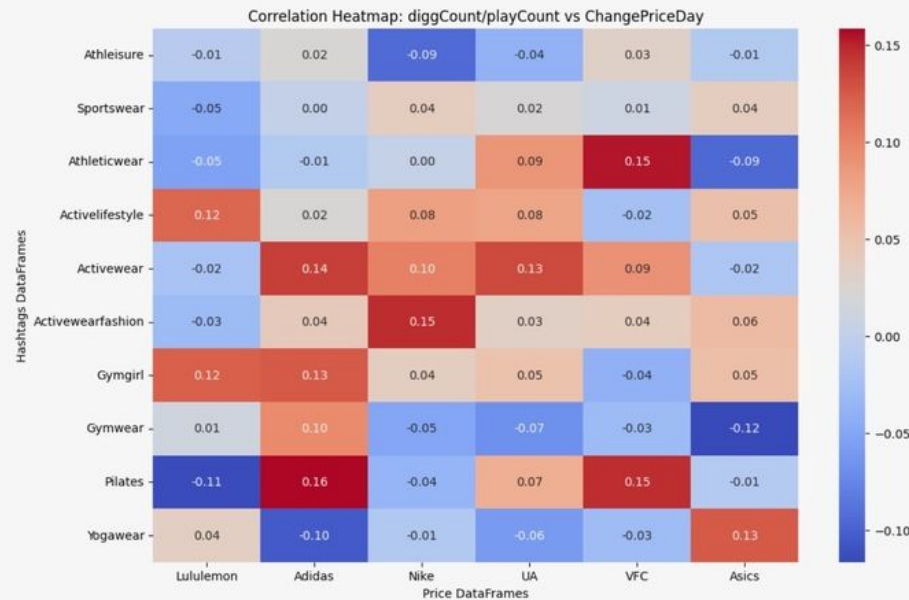
        # Append correlation result for each combination
        correlation_data.append({
            'hashtags_df': str(aggregated_df),
            'prices_df': str(prices),
            'Correlation_diggCount/playCount_vs_ChangePriceDay': corr
        })

# Convert the correlation data into a DataFrame for easier handling
correlation_df = pd.DataFrame(correlation_data)

# Pivot the correlation DataFrame to get a matrix format
pivot_corr_matrix = correlation_df.pivot(index='hashtags_df', columns='prices_df', values='Correlation_diggCount/playCount_vs_ChangePriceDay')

# Plot the heatmap using Seaborn
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', cbar=True, xticklabels=['Lululemon', 'Adidas', 'Nike', 'UA', 'VFC', 'Asics'],
            # Adding title and labels for clarity
            plt.title('Correlation Heatmap: diggCount+playCount vs ChangePriceDay')
            plt.xlabel('Price DataFrames')
            plt.ylabel('Hashtags DataFrames')

# Show the plot
plt.show()
```



Improvements on Model

- The linear regression model may not accurately capture highly correlated hashtags
 - Though we don't expect extremely strong correlations as large corporations won't move majorly based on daily swings in trends, we hoped the model would capture high correlations as good predictors. This was not *always* the case (though sometimes was)
- In the future:
 - Use a more sophisticated model (could use random forest to nonlinearly combine signals as predictors)
 - Use a wider bucket of tickers
 - Use more data!

Results

Sharpe Ratios during Testing Period:

LULU Combined Signal Sharpe Ratio: 1.1559

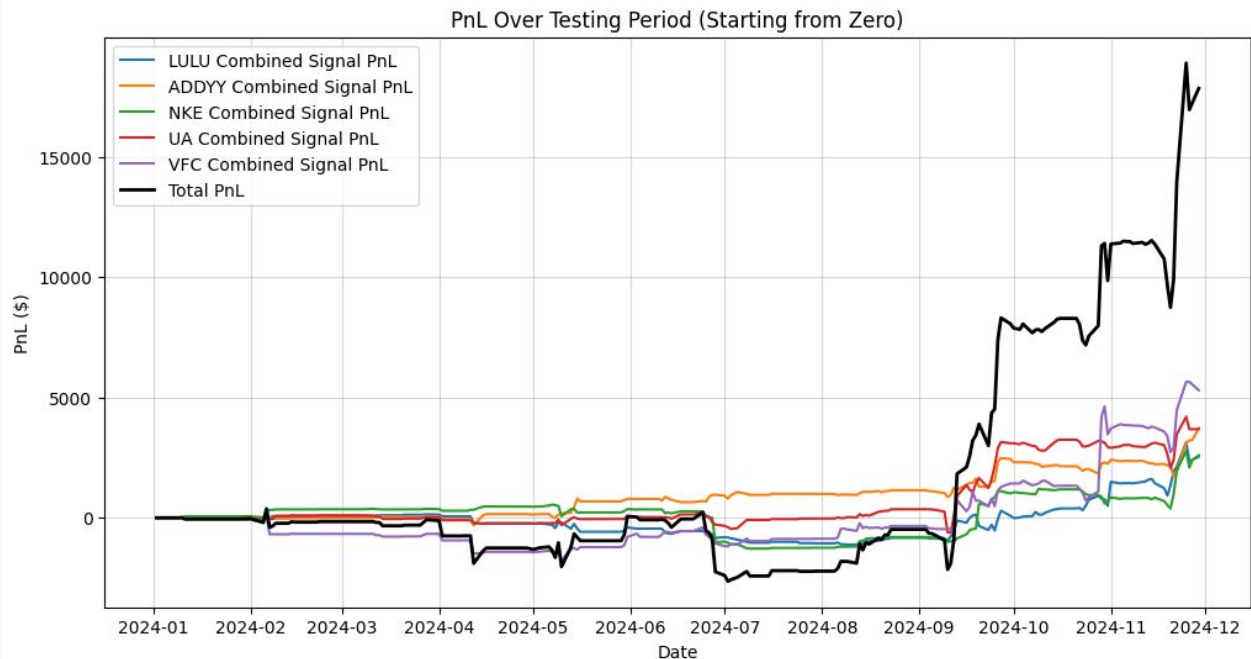
ADDYY Combined Signal Sharpe Ratio: 1.9787

NKE Combined Signal Sharpe Ratio: 1.1314

UA Combined Signal Sharpe Ratio: 1.4681

VFC Combined Signal Sharpe Ratio: 1.1996

Total PnL Sharpe Ratio: 1.9001



Other stats:

Total PnL during Testing Period:

\$17833.27

Maximum Drawdown during Testing Period: \$3015.41

Total Invested Amount during Testing Period: \$637558.30

Return on Investment during Testing Period: 2.7971%



06

Limitations and Future Improvements



Limitations

Sample Size

- Tiktok scraper provided limited data (~200 videos for each hashtag)
- Very hard to train a well-rounded model on so little data, especially with fashion trends being so cyclical over long periods

Company Size

- Companies had to be publicly traded (so are often very large)
- These are big corporations that may not be as easily influenced by Tiktok trends



Improvements

Better Scraper

- A scraper that provided more data would have been ideal

Wider Range

- Our strategy is based on individual companies' performances – an index with far less volatility and susceptibility to microtrends would almost certainly perform better

Other Trends

- Our project focused on athleisure, but the same ideas could be applied to other fashion trends

Other Indicators

- We used likeCount / playCount, but we could also explore:
 - likeCount
 - likeCount * playCount



Thanks!

Do you have any questions?

Feel free to ask us now.