Intraday SPY Next Move Data Analysis & Potential Applications

Christopher Lam
Portfolio Website
Github Repo Link

Objective of Study

The objective of this study is to analyze historical S&P 500 (SPY) data on an hourly granularity to identify potential ranges and price behaviors. To achieve this, SPY data is cleaned, processed, and ultimately visualized into various charts, each containing information of if the closing price of SPY was above, below, or within a plus or minus percent change threshold for every hour in the dataset. This process was also repeated for various block-sized hours, (e.g. a 2 hour rolling window, where the above analysis applied as applicable). From this analysis and visualization, insights on potential future behavior may be inferred and acted upon to derive profit through derivative, options, and/or other transactions.

Dataset Information

The dataset used is sourced by Kaggle user Geowtt, who compiled hour-by-hour data for the year 2023. This dataset incorporates several tickers, OHLCV (Open, High, Low, Close, Volume), and other technical indicators; Many of the provided entries are filtered out due to the scope of this study. For more information about the dataset, see this link.

Method of Analysis

Data conversion, Preprocessing, and Validation

The raw data is first converted from a CSV file into a parquet file. This design choice was made in an attempt to make the code scalable for future use. As a columnar storage system, parquet gives several advantages compared to CSV:

- Advanced compression for smaller file size
- Faster access to specific columns than CSV
- Built in schema metadata for data integrity

Although the utilized dataset is relatively small, consisting of approximately 500 entries, this file conversion gives the code the opportunity to perform well under larger amounts of historical data

The data was then preprocessed to remove any non-applicable rows and columns. Specifically, this study revolved around the data of the S&P 500 (SPY), though the original dataset held additional stock information. Thus, any entry with a symbol that was not “SPY” was removed. Further, technical metrics were removed (e.g. macd), as was the given “target_up_next” column; Ultimately, the following columns left after preprocessing are:

- Timestamp (UTC & rounded to whole hourly blocks)
- Symbol (SPY)
- Open
- Close
- High
- Low

- Volume

The next step in preprocessing was to convert the “timestamp” column (given in UTC) into EST. Further, this timestamp is then filtered to only incorporate rounded open market hours (9:00 AM EST - 4:00 PM EST). These columns are then validated for correctness (e.g. timestamps are in fact within market hours, columns exist, etc.).

The processed data is then resaved into a new parquet file, where analysis is then conducted.

Analysis of data

For each row in the cleaned data set's applicable open market hours (9:00 AM EST - 4:00 PM EST) the OHLCV is analyzed and given a trinary variable, **target_next_move** (1, 0,-1). This variable represents the price being above, in between, or below a **threshold** percent for each row.

For example:

- Let **threshold** equal 0.0025 (equivalent to a plus or minus move of 0.25% for SPY)
- Let SPY's open equal 100.00 for the hour range 9:00 AM EST to 9:59 AM EST
- Let SPY's close equal 100.50 for the hour range 9:00 AM EST to 9:59 AM EST

By the specified **threshold** of 0.0025, the requirements for **target_next_move** to equal 1 must result in a SPY closing price above 100.25. Inversely, **target_next_move** will result in a value equal to -1 if the closing price of SPY is below 99.75. Finally, the default value of 0 will be assigned where the closing price to be in between these two values.

Thus, by this example, the row will be marked with a value of 1, since the closing price exceeded the established **threshold** (closing price of 100.50 is greater than the **threshold** of 102.5); This process is repeated for each hour for each day available across the applicable timeframe (i.e. each row in the dataset).

Further, the above process is then repeated across several block hours. The previous implementation described is incorporated on the most granular level (i.e. each row is one hour, so **target_next_move** is reported as such), though by using block hours, analysis can be conducted across broader time frames.

For example:

- Let **threshold** equal 0.0025 (equivalent to a plus or minus move of 0.25% for SPY)
- Let **block_size** equal 2 (i.e. spans 2 hours instead of the original 1).
- Let SPY's open equal 100.00 for the hour range 9:00 AM EST to 9:59 AM EST
- Let SPY's close equal 101.00 for the hour range 9:00 AM EST to 9:59 AM EST
- Let SPY's open equal 101.00 for the hour range 10:00 AM EST to 10:59 AM EST
- Let SPY's close equal 100.10 for the hour range 10:00 AM EST to 10:59 AM EST

Like before, the **threshold** for **target_next_move** to be 0 (in between) is the range of [99.75, 100.25) inclusive, with values of 1 (above) and -1 (below) being outside of the ranges as applicable. Despite having a large move upward in the period of 9:00 AM EST to 9:59 AM EST, the classification for this section is in between (i.e. **target_next_move** equals 0), since the closing price utilized is based on the **block_size** of 2. This process is then repeated for every applicable 2 hour interval for each day in the data set; That is, a **block_size** of 2 does NOT include analysis starting at 3:00 PM EST (since it would bleed into 5:00 PM EST, which is not within the utilized dataset).

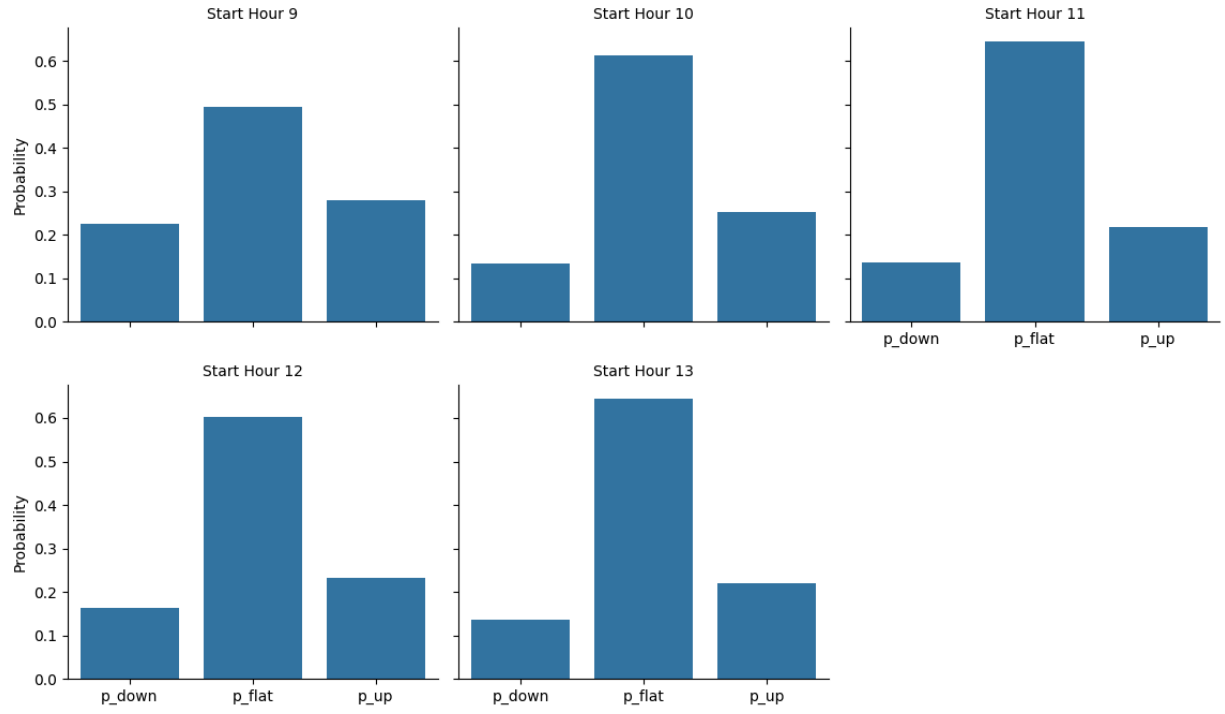
As a whole, this repo analyzes **thresholds** 0.001 to 0.0045 in 0.0005 increments (0.001, 0.0015, ..., 0.0045) for all **block_size** 1 to 6 inclusive.

Visualization of Data

For each **threshold** and **block_size** combination, a bar chart is created that showcases the starting hour and the distribution of p_down, p_flat_ and p_up for the accumulated data.

For example, the chart below displays the visualization for CLEANED_FIG_2_block_size_0.0025_threshold_target_next_move.png. The nomenclature for each chart is: CLEANED_FIG{**block_size**}_block_size_{**threshold**}_threshold_target_next_move.png

That is, this figure has a **block_size** of 2 and a **threshold** of 0.0025 (0.25%).



This data was visualized using Seaborn and MatPlotLib.

Results, Figures, & Observations

To see all the generated charts, check the GitHub link and proceed to data->charts->target_next_move.

When analyzing the data, a few key patterns can be observed. The first is that there is a direct relationship between threshold and p_flat. That is, as the threshold increases, so does the likelihood that the price stays within the threshold bounds. Intuitively this makes sense and does not give much valuable insight; A wider tolerance band increases the likelihood that the price remains within said tolerance, and does not provide much opportunity to realize profit. This can be visualized with the two charts below:

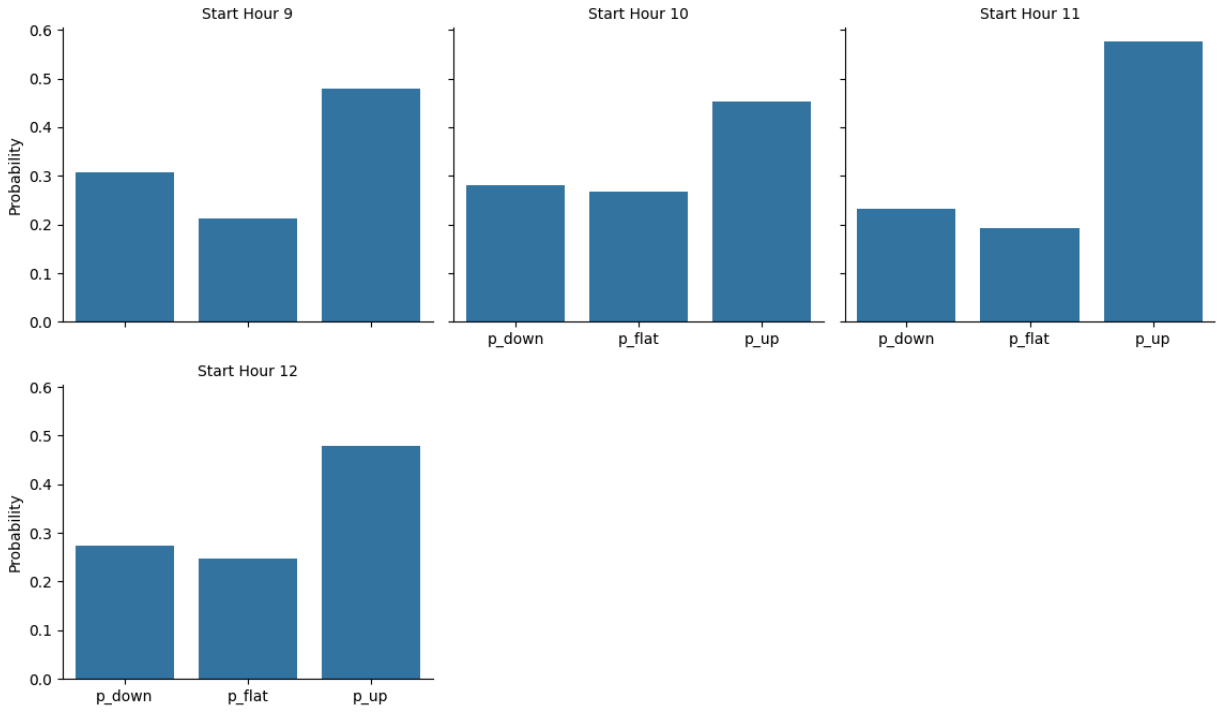


Fig A: CLEANED_FIG_3_block_size_0.001_threshold_target_next_move.png

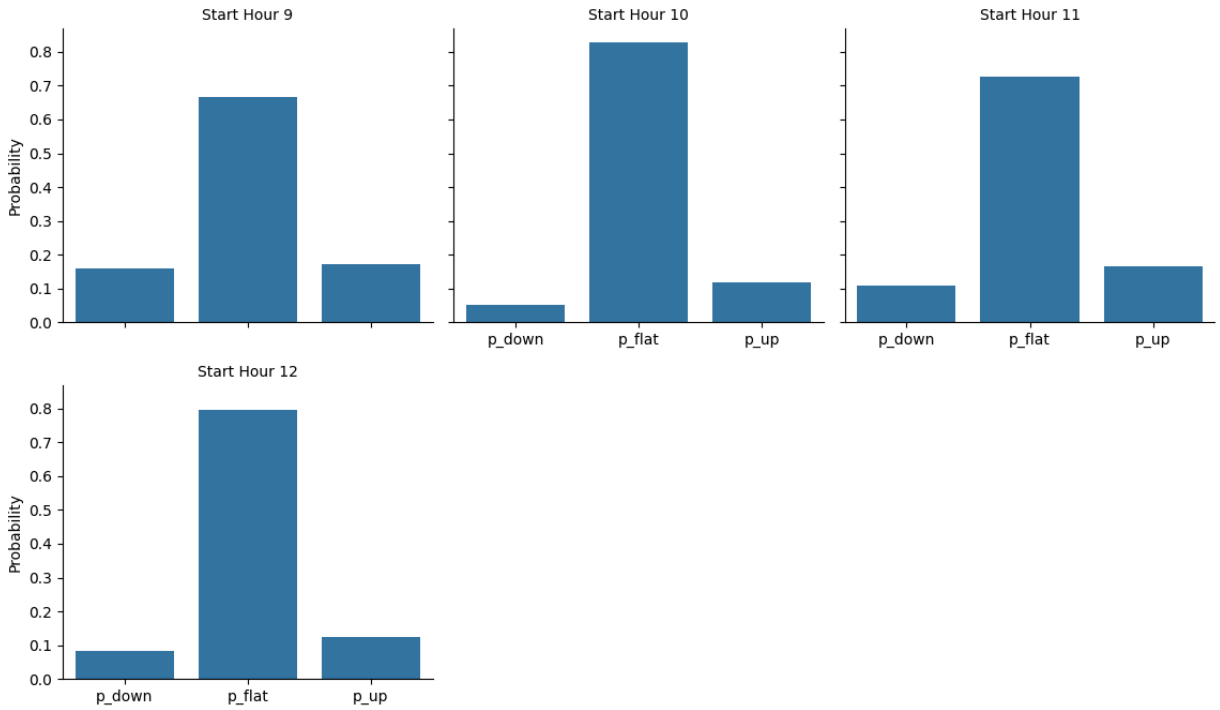


Fig B: CLEANED_FIG_3_block_size_0.0045_threshold_target_next_move.png

Here the difference is in the threshold, with **Fig A** having a threshold of 0.001 and **Fig B** having a threshold of 0.0045. As mentioned, a larger threshold increases the likelihood of p_flat drastically.

The second observation is the relationship between **block_size** and **p_flat**. Specifically, as **block_size** increases, the likelihood of **p_flat** begins to decrease with all else being equal. Intuitively, this also makes sense, since a larger **block_size** allows for more intraday fluctuations. For example, a **block_size** of 4 is a 4 hour window in which SPY can progressively move up or down, thus existing outside of the **threshold**. This can be visualized with the charts below:

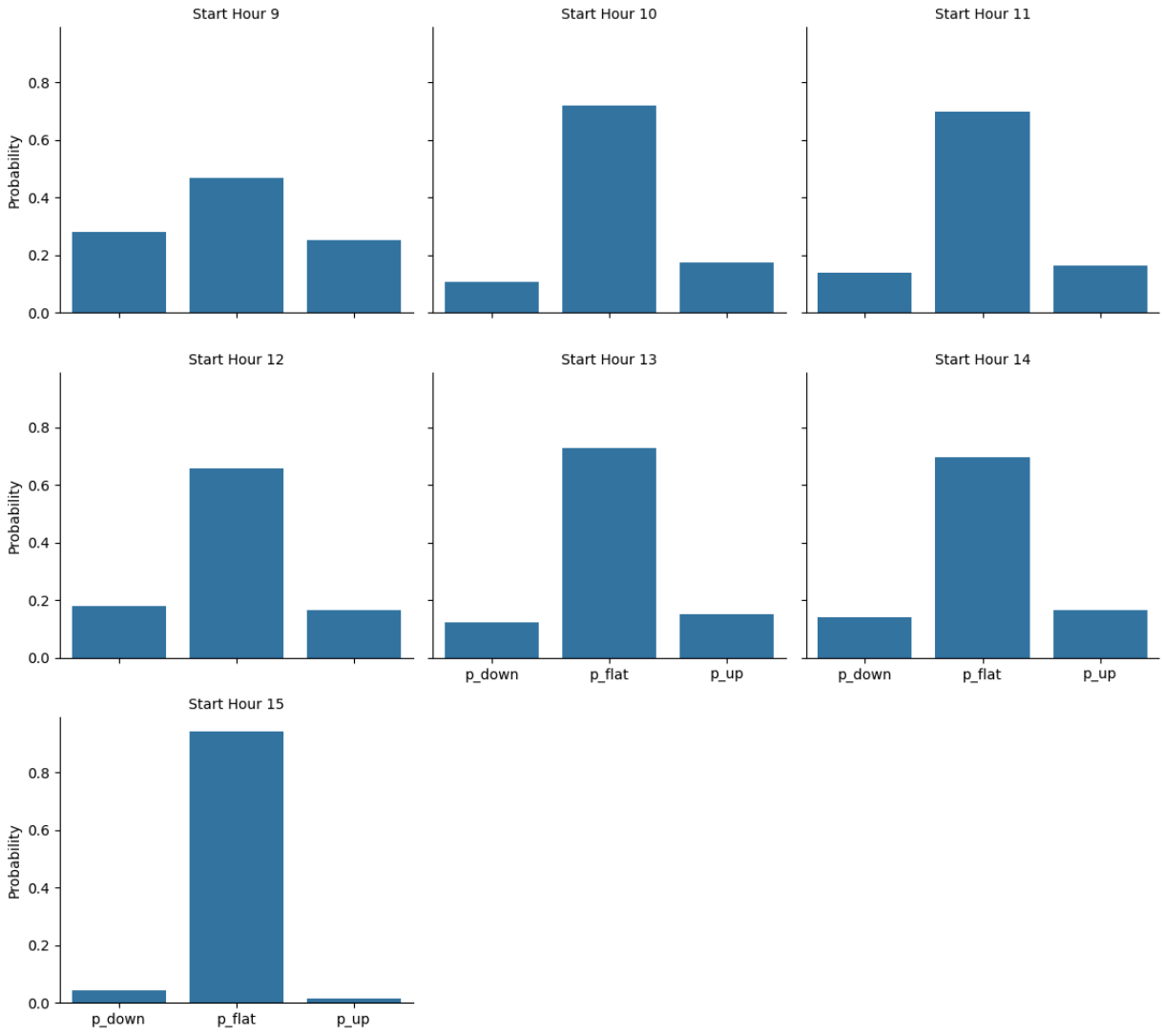


Fig C: CLEANED_FIG_1_block_size_0.002_threshold_target_next_move.png

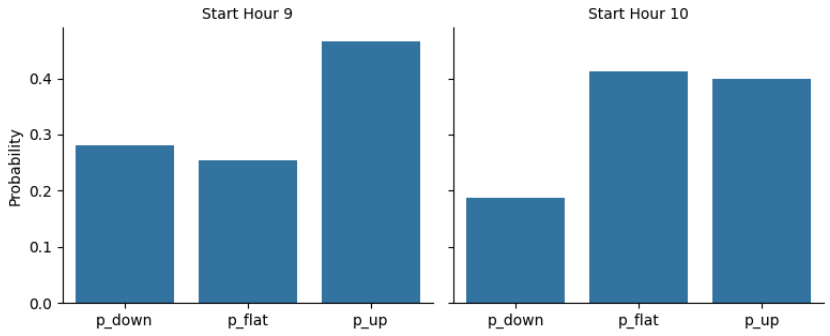


Fig D: CLEANED_FIG_5_block_size_0.002_threshold_target_next_move.png

Here, the difference in the block size gives **Fig D** more time to fluctuate in price. As a consequence of this, the opportunity for the final price to exist outside of p_flat appears to increase, explaining why p_flat is not significantly higher in a larger **block_size** compared to a smaller **block_size.**

A third observation is that p_flat appears to “lag” at a start hour of 9:00 AM EST when compared to other blocks for lower thresholds, though only significantly for low **block_sizes**. Intuitively, this also makes sense, due to the nature of the “morning power hour”. Here, overnight orders and institutional buying can lead to large swings of movement, resulting in a decreased chance of price stability reflected through p_flat. Thus, for small **block_sizes** a larger threshold is needed to capture a comparable p_flat percent. This can be visualized with the charts below:

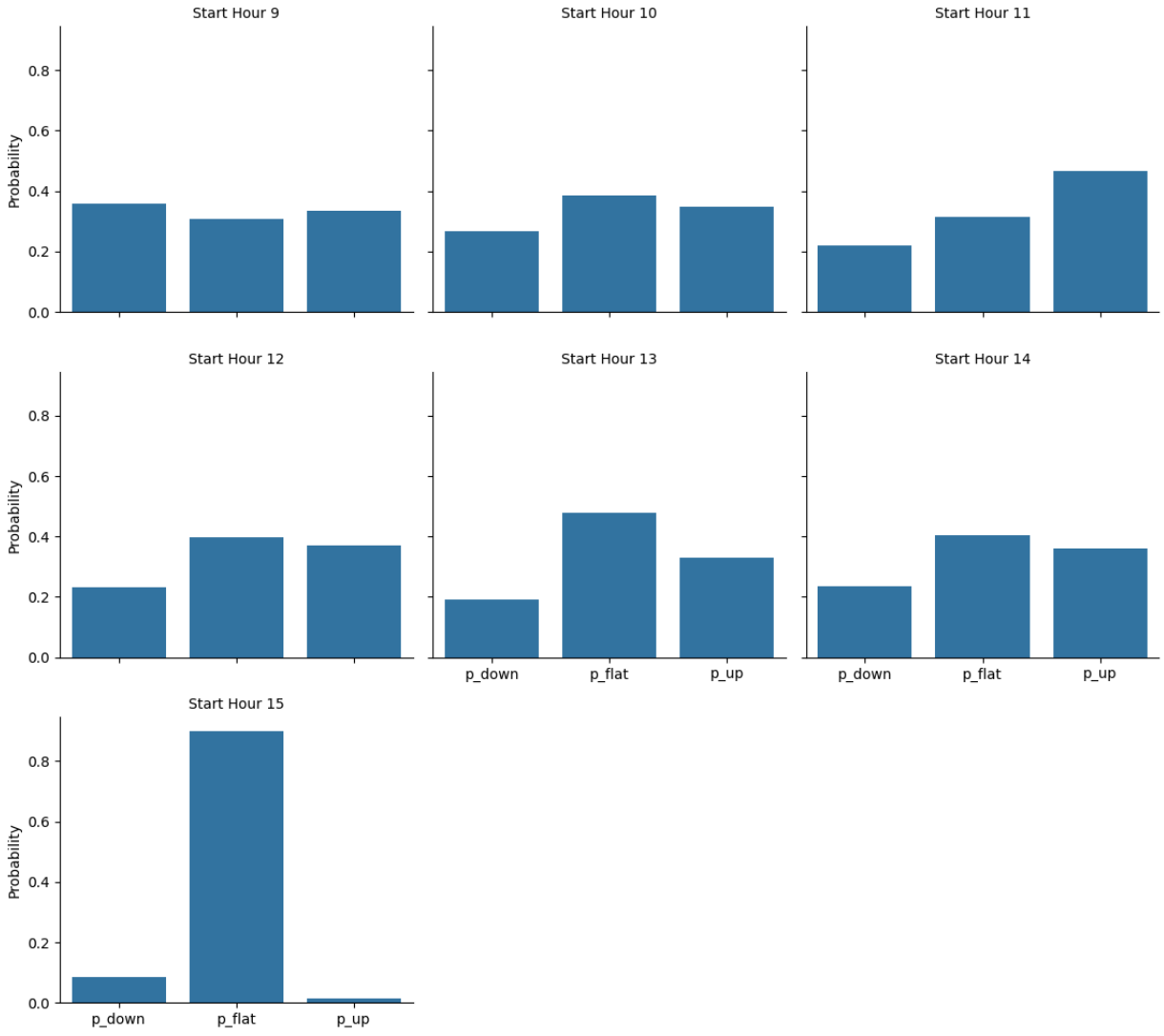


Fig E: CLEANED_FIG_1_block_size_0.001_threshold_target_next_move.png

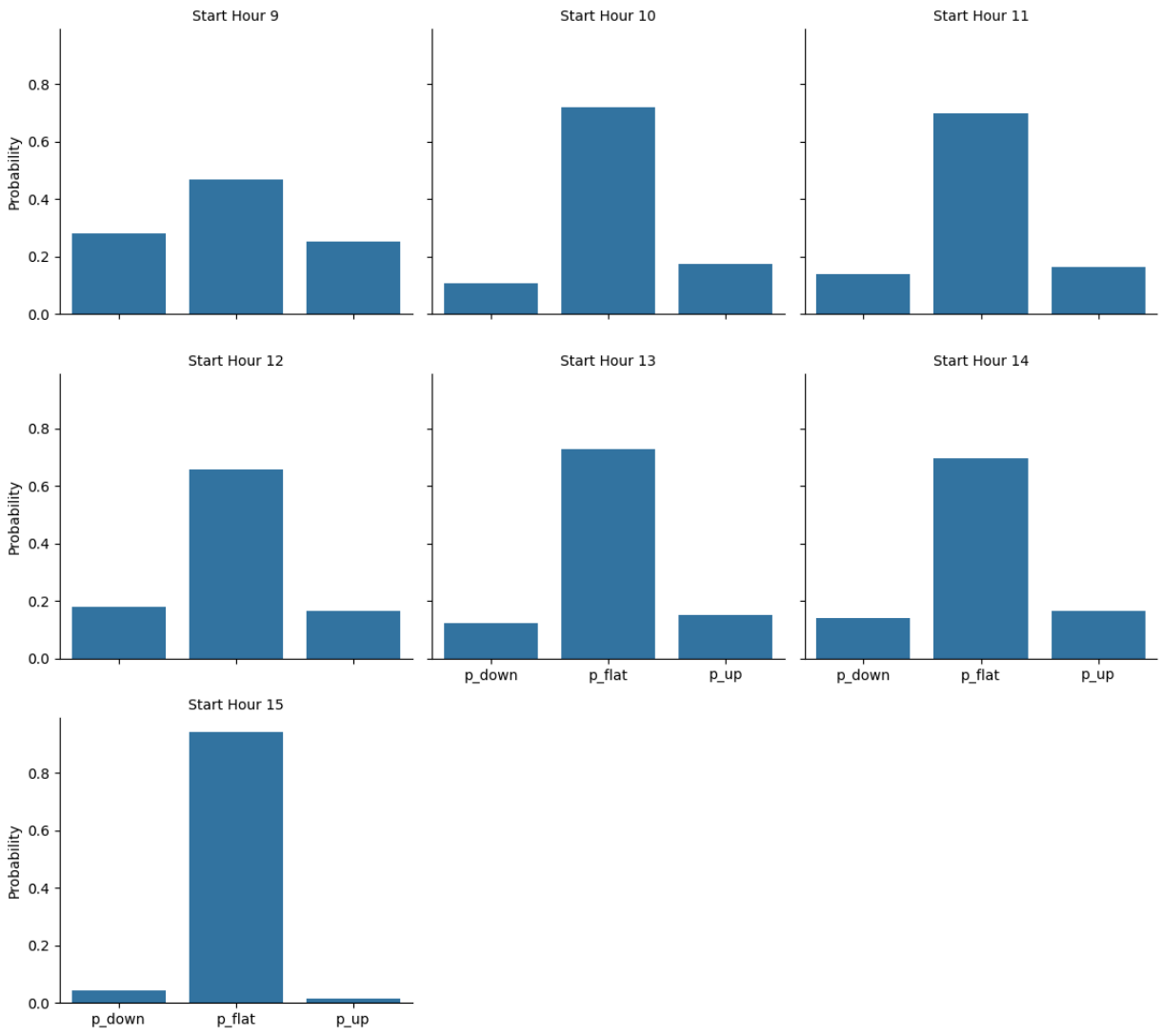


Fig F: CLEANED_FIG_1_block_size_0.002_threshold_target_next_move.png

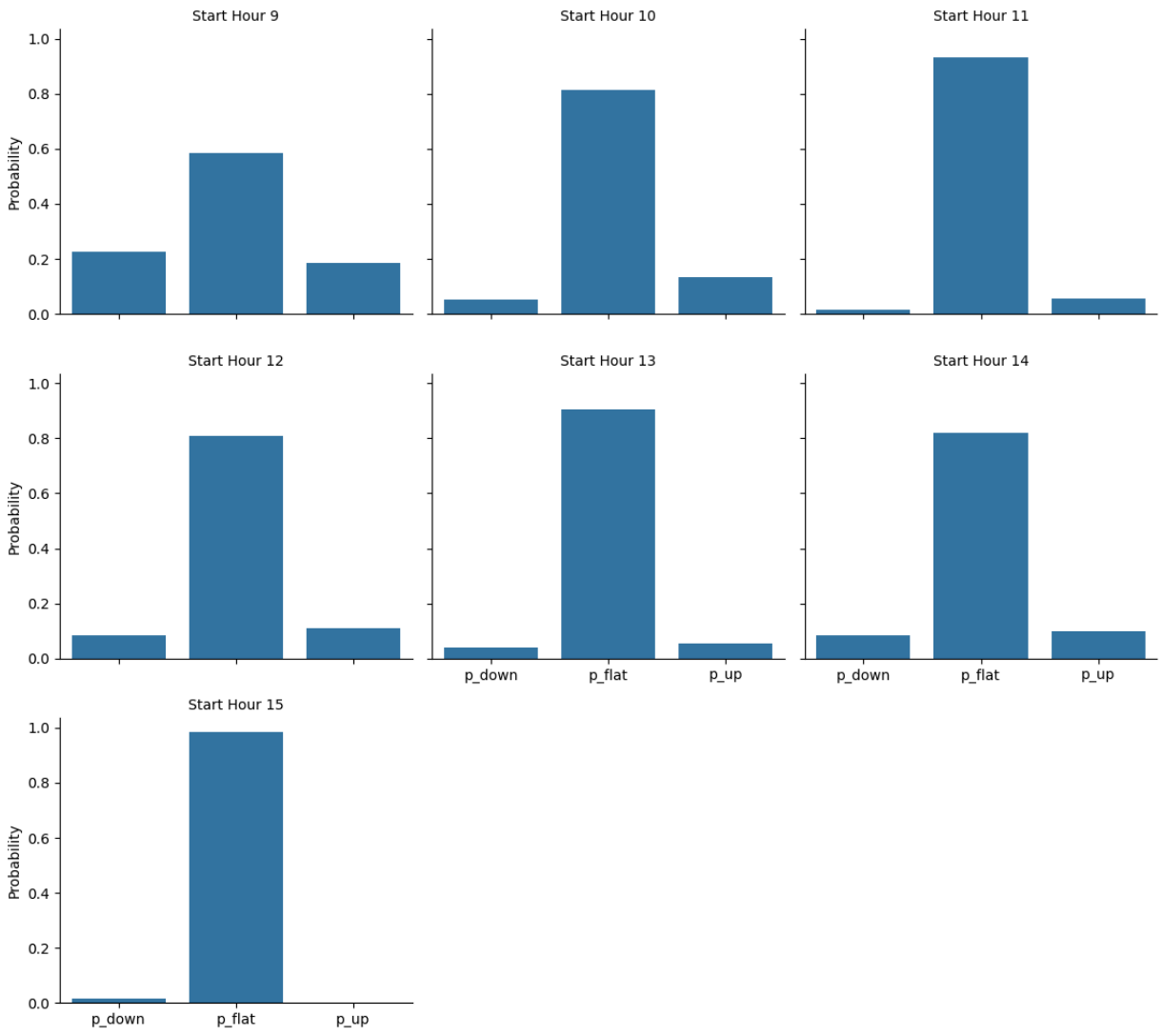


Fig G: CLEANED_FIG_1_block_size_0.003_threshold_target_next_move.png

As shown in Fig **E, Fig F,** and **Fig G**, p_flat for “Start Hour 9” appears to “lag” the other starting hours. As mentioned, this is likely due to institutional buying/selling and overnight orders, which create a more volatile window during this morning session. It is worth noting that increasing the threshold and the block size allow p_flat for “Start Hour 9” to “catch-up” with other hours. This also makes sense due to the nature of increasing the **threshold** and **block_size** as previously discussed.

Critiques

Perhaps the biggest critique of this study is the lack of significant data. In total, there are approximately 500 relevant entries after preprocessing. Further, larger **block_sizes** by design have a smaller subset of the overall preprocessed data, since the size of the hours in the block cannot roll into the next day. As a consequence of this, the analysis lacks significant reliability and all inferences should be taken only as a frame of reference. Were there to be an extension of data, the patterns currently observed in the dataset may continue to exist, and thus have more merit when it comes to application.

Extention

Extending this work and analysis can take on several different stances.

From a programming and software perspective, a UI can be incorporated to better analyze the data. This would create an opportunity for a user to configure what kind of analysis they wish to study and make inferences on, rather than surveying the premade charts. Overall, this extension would grant a more complete software that is more ergonomic for use.

From a data analysis perspective, more refined analysis using the pre and post processed data may be performed, providing the ability to extract new insights. Volume analysis, use of technical indicators, and other calculatable metrics can be incorporated and visualized to attempt to gain an edge using this data.

From an investment and trading standpoint, various vehicles can be explored and backtested based on this analysis. For instance, utilizing the Black-Scholes model and neutral options strategies (e.g. Iron Condors) can provide opportunities for increasing one's edge in deriving profit. A Backtested performance would be able to gain insights on valuable metrics such as drawdown, CAGR, and overall percent return.

Concluding Thoughts

In conclusion, the data analysis and visualization of the 2023 SPY hourly data provides a foundation for the intraday behaviors of the broader S&P 500. By modeling the distribution and probability of price movement on an intraday hourly level, one can theoretically take actions to derive profit through different vehicles. While the data used for this analysis may not be large enough to create reliable estimates, the process and methodology gives opportunity for extension using more data. Further, this study leaves room for extension in a multitude of dimensions, spanning from programming, data analysis, and financial investment.