# Prediction Markets: Final Report

## Team 6

Ryan Kennedy

Andrew Clements

Nick Salem

Andrew Myers

# 1. Introduction

## 1.1 What are prediction markets?

Prediction markets are trading platforms in which participants buy and sell contracts, and the value of these contracts are derived from the outcome of specific world events. More crudely, prediction markets are just betting on real world events. But these markets are surprisingly accurate, often producing predictions that rival the accuracy of professional analysis. These markets also cover a wide variety of topics, ranging from pop culture to politics to weather. Worldwide, the two main prediction market platforms are Kalshi and Polymarket, although Polymarket is still illegal in the United States. Kalshi, on the other hand, became a legal and CFTC-regulated exchange in 2020.

These prediction markets skyrocketed in popularity during the 2024 presidential election, in which billions of dollars of contracts changed hands. Moreover, these markets proved to be more accurate indicators of Trump's eventual victory than both the media and prestigious polling organizations. One incredibly recent and intriguing prediction market was the "Pope Market", where individuals could place bets on who they believed would be elected as the next Pope of the Roman Catholic Church. Although the markets predicted Cardinal Tagle, an unexpected selection of Cardinal Prevost, now Pope Leo XIV, surprised everyone.

Prediction markets trade a new asset class, called "event contracts", which according to the Kalshi website, "give people the ability to trade based on their opinions about a specific yes-or-no question". If you are correct about this yes-or-no question, the contract is worth $1. For example, let's assume you purchase an event contract for $0.32 saying "yes" that there will be a U.S. Recession in 2025. If there is a U.S. Recession, the market will evaluate to true, and you will be filled to the dollar, receiving $0.68 of profit. These markets are a fascinating mix of retail traders and sophisticated trading entities, providing a potential goldmine of untapped gains for strategic individuals and high-frequency trading firms alike.

## 1.2 Project Overview

For this project, we planned to build a backtesting engine for prediction markets that could test trading strategies on historical prediction market data. Significant backtesting platforms exist for traditional stock markets, crypto markets, and forex markets, yet from our research, there were no backtesting platforms for prediction markets. Due to the infancy of this market type and the potential untapped profitability, we believe there is significant value in a backtesting engine.

Our project was segmented into three main phases: data aggregation, market analysis, and, finally, the construction of the backtesting tool. Each phase provided its own insights and challenges, and will be expanded upon further in the following sections.

## 2. Data Aggregation Methods

Our data aggregation process involved collecting historical market data from Polymarket and real-time orderbook data from Kalshi, creating a comprehensive dataset for backtesting prediction market strategies.

### 2.1 Polymarket Historical Data Collection

For Polymarket, we developed a data pipeline to collect historical trade fills using their public API. Since Polymarket doesn't provide complete historical order book snapshots, we implemented a custom solution to capture and store trade execution data:

1. We created scripts to systematically query an indexed Polygon node for completed trade fill events across multiple markets, focusing particularly on high-volume markets like the 2024 presidential election and other popular prediction events.
2. The collected trade data included transaction timestamps, prices, volumes, and contract identifiers, which were stored in a structured database for efficient retrieval during backtesting.
3. We implemented daily chunking in our get_all_event_timestamps function to efficiently process large volumes of historical data, allowing us to analyze specific time periods without loading the entire dataset into memory.

4. Data was processed using multiprocessing to handle the computational demands of analyzing millions of trade events, significantly reducing processing time.

### 2.2 Synthetic Orderbook Reconstruction for Polymarket

Since Polymarket doesn't provide complete order book depth data through their public API, we developed a methodology to reconstruct synthetic orderbooks:

1. We used sequential trade fills to infer market depth and liquidity at various price levels.
2. Our process_candidate_direct function analyzed trade execution patterns to estimate the likely state of the orderbook at any given time.
3. We implemented a probabilistic model that estimated bid-ask spreads based on trade frequency, volume, and price movement patterns.
4. The synthetic order books were validated by comparing predicted price impacts against actual price movements following large trades.
5. All reconstructed orderbook data was stored in Parquet format for efficient storage and query performance.

### 2.3 Kalshi Live Orderbook Streaming

For Kalshi, which offers more comprehensive market data access, we developed a direct market data feed:

1. We established a WebSocket connection to Kalshi's API to stream real-time orderbook updates for various prediction markets.
2. Our system captured both full order book snapshots and incremental deltas, allowing us to maintain an accurate representation of the market at any given moment.
3. We focused particularly on crypto-related hourly markets, which showed interesting patterns when major market makers like SIG reduced their activity after 4 PM EDT.

4. The data pipeline stored both the raw orderbook states and calculated derived metrics such as market depth, spread, and imbalance.
5. All collected data was time-stamped and synchronized to ensure accurate temporal alignment between different data sources.

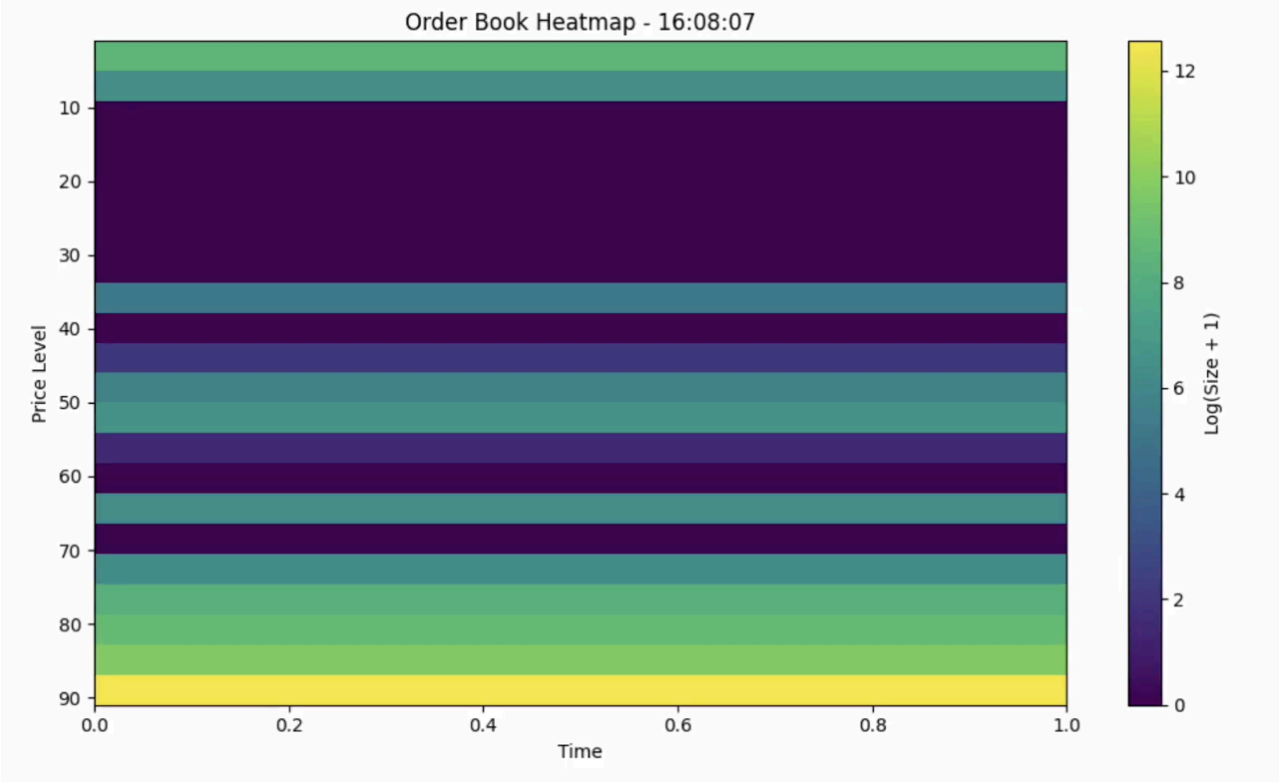
The combination of historical Polymarket data and real-time Kalshi orderbook information provided us with a robust dataset to develop and test our prediction market strategies. This dual-source approach allowed us to compare market behaviors across platforms and hopefully help us develop strategies that could potentially exploit cross-platform inefficiencies.

## 3. Market Analysis

### 3.1 Common Behaviors

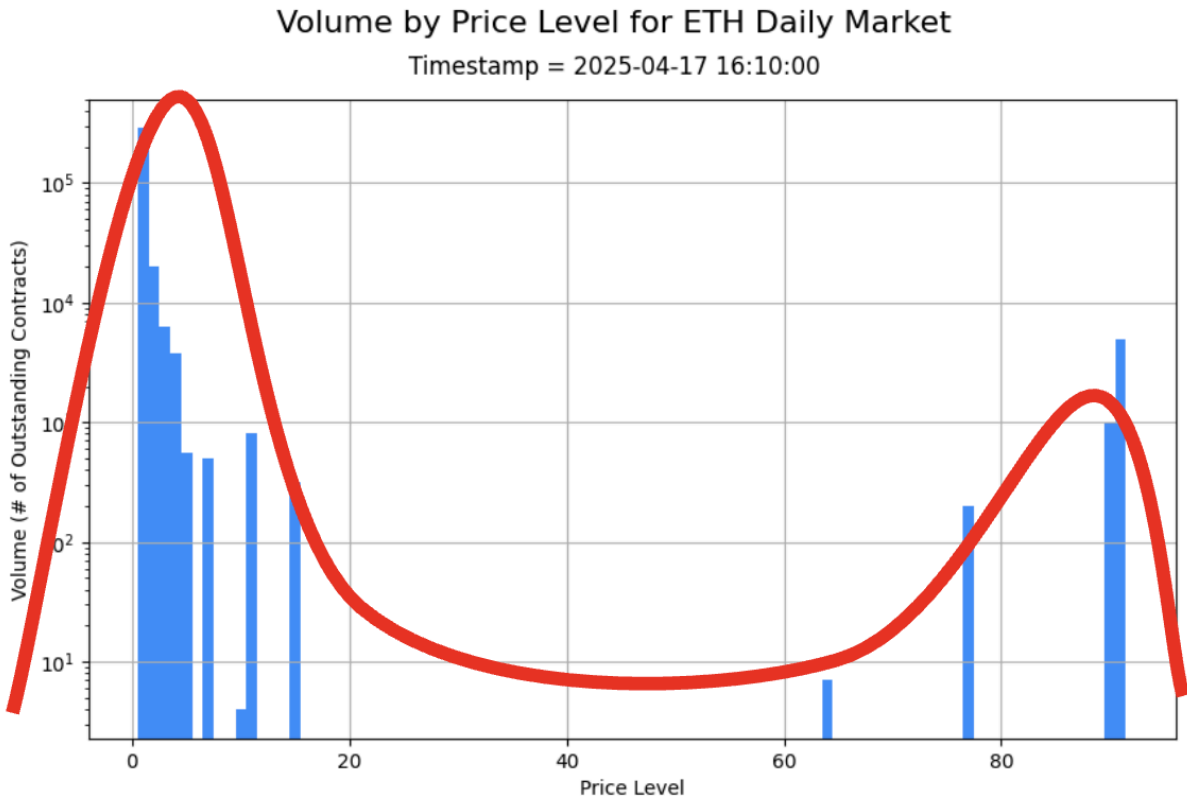
For the data analytics portion of our project, we primarily focused on hourly crypto markets listed on Kalshi. These markets run 24/7 and typically ask whether a specific crypto asset will reside within a specific price range at a specific time. To illustrate this point, a current crypto market on Kalshi asks whether ETH will be above $2,300 at 1pm EDT. Although we analyzed hourly markets during all 24 hours of the day, we zeroed in on hourly markets after 4pm EDT, as SIG often turns off their trading strategies at this time. Without a major market maker in play, there is more natural “retail flow” after 4pm, creating a more enticing, and level, playing field.

The first major trend we observed with these evening crypto markets was that they were relatively illiquid. As alluded to previously, a major cause of this could be that SIG stops trading at 4pm and leaves the market for regular retail investors. To illustrate this point, we made visualizations of a specific ETH market from April 17th, 2025, which asked “Will ETH be above $1,530 by 8pm EDT?”. This market settled to “yes”. To follow along with the written analysis of this ETH market, it would be beneficial to view the video at this link.



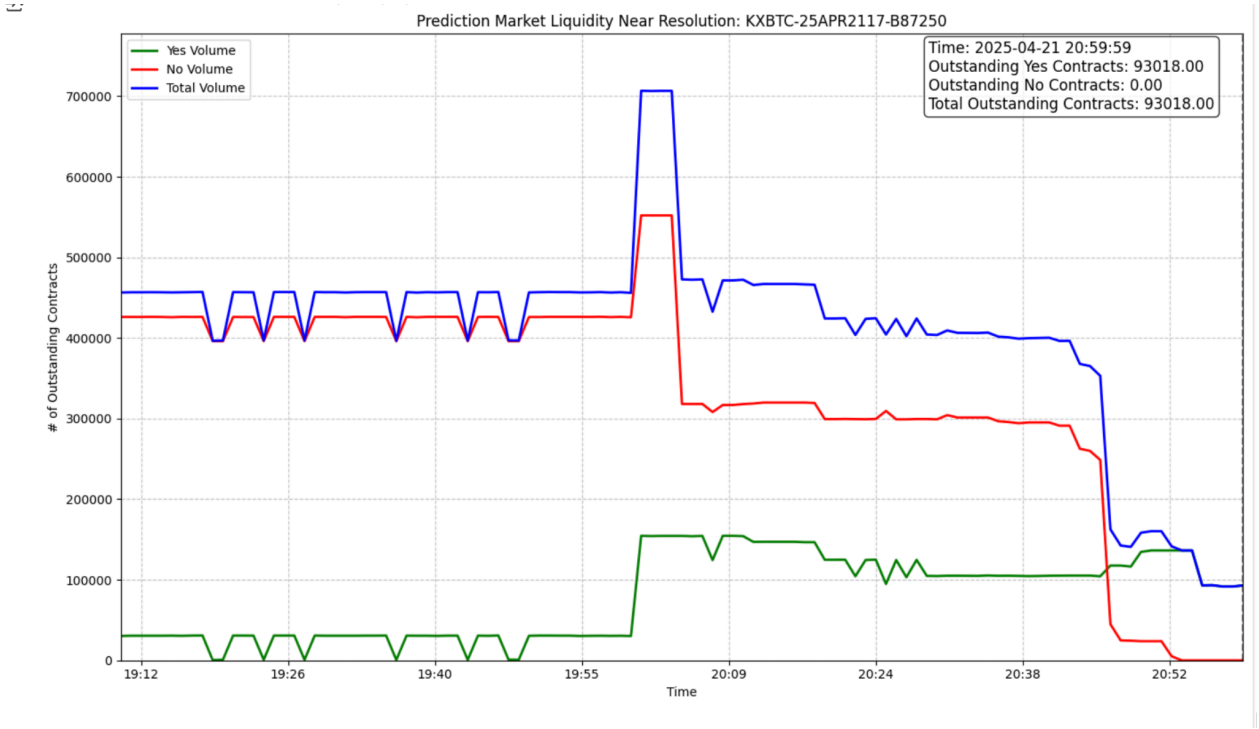

As seen in the video, the order book heatmap showed very little variation over a ten minute time period from 4:00pm to 4:10pm. On average, we found these hourly crypto markets to have low liquidity, sitting around $20M in contracts outstanding, which creates difficulties – namely a large bid-ask spread – for traders.

The second major trend we observed within these hourly crypto markets was that there existed significant volume concentrated on the “edges” of the order book, a product of the unique pricing range of event contracts. Normal assets are typically not capped at such a tight price range between $0 and $1. We discovered that many investors would place large volumes of trades at both edges of the order book. In fact, in most markets, a heavy majority of outstanding contracts were sitting on the far ends of the spectrum. The most likely theory behind this phenomenon is that people holding a contract which is likely to expire worthless are happy to trade out of their position for just a cent or two, and the market participants who placed these edge orders are happy to buy a winning contract for $0.99 if the probability of it occurring is higher than 99%. To get fills on these orders queue position is very important, hence the fact that these orders persist for a significant portion of the market’s duration.



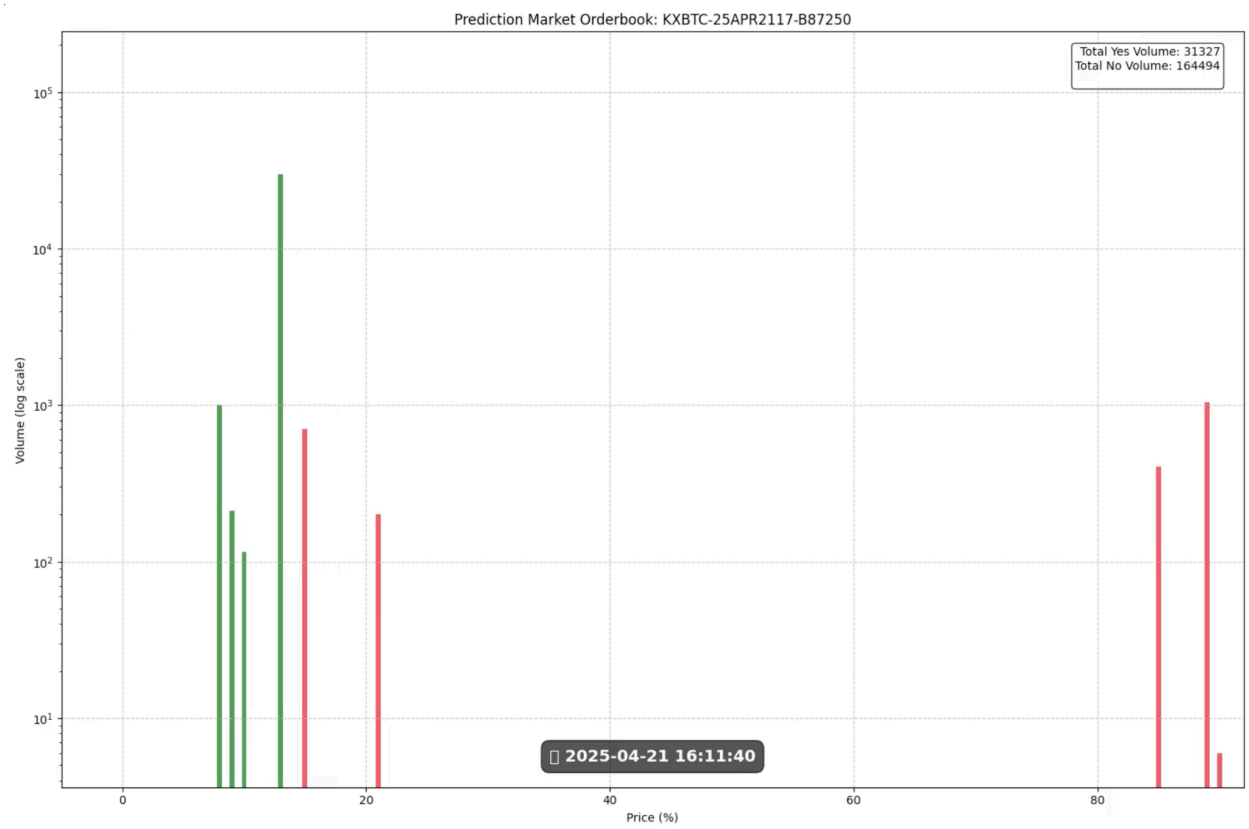

Looking at the same April 17th ETH market as before, we can see how outstanding contracts follow this general “double-bell curve”. Further note that the scale of this graph is exponential, meaning the number of outstanding contracts are even more inflated on the edges than may appear with a quick glance at the graph. The high edge concentration of outstanding contracts was consistent and strategic, as many investors left large standing orders in preparation for market resolution, which will be examined next.

The third and most interesting observation of the hourly crypto markets occurred from analyzing the market as it approached resolution. For this analysis, we analyzed a specific BTC market from April 21st, 2025, which asked “Will BTC be between $87,250 and $87,500 by 5pm EDT?”. This market settled to “yes”. To follow along with the written analysis of this ETH market, it would be beneficial to view the video at this link, although the static line graph (pictured below) displays the trend adequately:




For the sake of data visualization we did limit the x-interval size to be every 1 minute, which partially accounts for the deceptively flat # of outstanding contracts at the beginning. Nevertheless, the interesting portion of analysis comes from the right side of the graph, as the market approaches resolution at 5:00pm EDT(21:00 UTC). First, we can see that the total number of outstanding contracts decreases significantly as the market approaches the 5:00pm deadline, implying a drastic decrease in market liquidity as resolution approaches. This seems to be a very common trend among all prediction markets. Secondly, you can see a sharp decrease in the number of outstanding “no” contracts. Around 4:45 pm EDT(20:45 UTC) it became increasingly obvious that the price of BTC would settle between $87,250 and $87,500, and thus the market would resolve to “yes” at 5pm. Therefore,  the “no” side of the market reacted appropriately, by quickly canceling outstanding no contracts on the order book. Overall, we generally saw lots of movement in the order book as resolution time approached, providing a plethora of opportunities for shrewd investors to jump in, and take advantage of last minute retail panic.

We also produced a neat visualization of this same BTC market, which can be viewed at this link, and a single frame from the video is displayed below.




Throughout the hour, you can see how the orderbook "yes" price continuously shifts (although slightly) to the right, indicating that a yes resolution is becoming increasingly likely. Again, the same pattern of concentrated standing orders, as alluded to earlier, is seen with this specific BTC market. A final point of observation is the presence of flashing bids and asks. Although we cannot know for certain, it is likely that these are resulting from competing market-making algorithms responding to each others' orders.

### 3.2 Kalshi Infrastructure

Kalshi – despite being the most rigorously regulated prediction market exchange – is still very opaque about their exchange infrastructure. Even the physical location of their servers is not public information. However, it does appear that some companies are using colocation and high-frequency strategies on Kalshi. Kalshi's rulebook

describes the designation of ‘Market Maker’ for some exchange members which maintains no specific qualifications, only indicating that Kalshi can appoint or remove Market Makers at their discretion. The rulebook also describes the benefits of being an officially designated Market Maker, saying “Market Makers may receive benefits, including but not limited to financial benefits, reduced fees, differing position limits and Position Accountability Levels, and enhanced access, in accordance with any relevant Market Maker program in place at Kalshi for fulfilling the market maker obligations.” In addition to traditional liquidity-providing benefits like reduced fees, the ‘enhanced access’ item likely refers to some kind of latency advantage unavailable to other market participants. This is an extremely important advantage on an exchange where the majority of participants are click-trading or at the very least subject to the much higher latency of a wireless connection. This Market Maker distinction is a new development at Kalshi, having only gone live in April 2025. However, SIG has had a substantial presence on Kalshi in what was likely a similar role since a year earlier, first announcing the development of their Kalshi-focused trading desk in April 2024. Whether the broadening of Kalshi’s offerings will provide more market-making competition and improve the liquidity on the exchange is yet to be seen.

### 3.3 Profitability of Market Making

The primary market maker on Kalshi is SIG, and by contractual obligation they participate in virtually every major market each day. In particular, we’ve observed SIG taking very aggressive positions on certain hourly markets, most notably the Bitcoin and Ethereum price range contracts that settle at specific times. On these markets, SIG often enters quotes in lot sizes exceeding 20,000 contracts, maintaining an average inventory carry (or “hold”) of around 9%.

When we first tried to transact with SIG’s quotes profitably using a previously successful strategy on Deribit, a highly liquid crypto futures and options exchange, we found that it was rarely profitable to transact with them. Even though their spreads were relatively tight, there was little edge and the sheer volume they were willing to absorb made it difficult for us to move the market. More often than not, the edge was simply

too slim once we factored in execution costs of taking fees and the risk of adverse selection (as SIG has a whole desk focused on these markets).

However, we noticed a consistent pattern in SIG's behavior: around 4 PM Eastern each day, they abruptly pull their orders from these same daily range markets. This "switch-off" window opens up an opportunity for us to step in as liquidity providers, capturing retail flow that would otherwise have been matched against SIG's deep quotes (a note here to be made that a majority of retail on prediction markets is extremely bullish on crypto offering a lot of edge on the "No" side of many of the price ranges). By stepping in at this precise time, we can quote tighter spreads on smaller lot sizes and potentially earn the spread that SIG leaves behind.

There were two key questions we were unable to answer regarding making these markets. First, did SIG's daily strategy of holding around 9% inventory actually generate a consistent profit for them? And second, is it worthwhile for us to assume the role of market maker at 4 PM Eastern, replicating SIG's approach on a smaller scale? Data quality issues – trying to tie trade feeds from one API endpoint to market data from another, while accurately de-anonymizing both – prevented us from achieving reasonable estimates for the profitability of these strategies. Given more time, answering these questions would help us fine-tune our own strategy and determine whether we can sustainably capture that liquidity gap for the later markets.

## 4. Prediction Market Backtesting

### 4.1 Order Book Construction

The first part of creating a backtesting program is initializing an order book structure that can accurately model how a strategy might interact with a live market. This is very similar to how one might design an order book representation in a live trading program with the only difference being that, rather than waiting for messages from the exchange, all messages have already been aggregated and can be processed when ready. Additionally, rather than the traditional bids and asks used in the vast majority of financial exchanges, there are actually two assets being traded in each prediction market. Each market offers both 'Yes' and 'No' contracts, even though buying

a ‘Yes’ contract is equivalent to selling a ‘No’ contract and vice versa. By offering both ‘Yes’ and a ‘No’ contracts, the exchange ensures that traders do not have to go through the effort of locating, borrowing, and then selling a ‘Yes’ contract in order to short the market: instead they can just purchase a ‘No’ contract. Either way, the exchange aggregates offers to buy and sell both contracts into a single market, since buying one and selling the other are equivalent. Even if the only two participants in the market are one trader who wants to purchase a ‘Yes’ contract and another who wants to purchase a ‘No’ contract, that transaction can still occur because the exchange can generate both contracts, collect the amount each trader paid for their contract – which will total $1 – and later award that $1 to whoever holds the winning contract. This is slightly convoluted but thankfully the exchange’s aggregation performs most of the work, presenting all orders as limit orders to purchase either the ‘Yes’ or the ‘No’ contract. The only work that needs to be performed by the backtesting program is translating from orders to buy the ‘Yes’ and ‘No’ contracts to the traditional limit order book. The final difference this produces is that, whereas selling an asset in a limit order book generates cash, achieving a net negative in a prediction market is done by purchasing the ‘No’ contract, which costs money. In terms of PnL, the two approaches are equivalent, but the second approach does have higher capital requirements. Capital constraints are not part of the backtesting implementation, meaning that it is assumed that the user will always have enough money to express the opinion dictated by the strategy. However, if capital constraints are a concern, the limit order book representation would have to be significantly altered.

Similarly to other exchanges, the market data used in the backtesting program consists of both snapshots and ‘deltas’ or individual updates. When a backtest is instantiated, a start date and time must be provided. In order to construct the order book, the most recent snapshot preceding the start time is located and all market updates between the snapshot and the start time are applied in order to populate the order book as it should appear for the start of the backtest. At each ‘delta’ message recorded between the start and end times the order book is updated before being sent to the strategy module where any desired trading signals can be generated. This

ensures that the strategy ‘observes’ the order book the same way it would if it were trading live.

**4.2 Backtesting Details**

The second, and perhaps most nuanced part of the backtesting program is how the strategy is simulated. The ideal backtester allows for realistic execution and market impact so that performance is as similar to live trading as possible. The first feature we employed to help realistically simulate trading was built-in latency. When the strategy submits an order, the order is time stamped to the latest market data packet, however, it is not immediately executed. Instead, it is placed in a queue with all other pending orders. Each order has an attached simulated latency, which is a parameter that can be changed for each backtest. The orders are then executed only once the backtest receives, but has not yet processed, a market data packet whose timestamp is greater than the order’s timestamp plus the designated latency. This allows a user to achieve realistic execution times, ensuring that the trader cannot profitably backtest infeasible trading strategies, and allows the trader to understand how a strategy’s latency affects its performance.

The second implementation detail aimed at realistic backtesting is that, in the backtest, any fills the tester receives affect the status of the limit order book. This helps approximate market impact as the user’s trade will affect variables like weighted mid price or outstanding liquidity. Updating the order book also prevents feedback loops where the strategy continually triggers on a certain order book status which would in fact change if the user were actually trading. Changing the order book like this would alter how other market participants behave, but as we have no insight into this hypothetical counterfactual and the deviation from the historical order book is usually small, we believe this is a net improvement in test accuracy. Any market data that indicates the cancellation or filling of liquidity that the user transacted with is simply dealt with by prohibiting the offered quantity at any price from going below zero. So if there were originally 10 contracts offered at $0.50, the user’s strategy purchased 3, and then the backtest received a trade notification that another trader had purchased all 10

contracts after the user's order would have arrived, the limit order book is updated as if that purchase quantity was 7, rather than 10.

The final core tenet of the backtesting program is tracking positions, orders, and PnL. At each market update the program calculates a new PnL and records it. Additionally, all filled orders are recorded with a symbol, timestamp, side, and price. This allows for in-depth analysis of the performance of the strategy beyond just its profitability. Increased insight into when and why orders were triggered, where those orders got filled, and how that affected the profitability of the strategy is integral in the maturity of a backtesting platform.

Beyond these key functionalities, this backtesting program has many more features. Importantly, the trading strategy is almost entirely customizable: any signal generation method can be used to trade across an unlimited number of markets simultaneously. This customization allows for the use of a variety of strategies across a variety of markets, from copy trading to high-frequency strategies. This is necessary in order to build a tool that is usable across the wide array of markets available on these platforms.

**4.3 User Interaction and Frontend**

The front end of our prediction market backtesting platform serves as the user-facing interface that enables traders to visualize the efficacy of their strategy. Its objective is to configure backtests, view real-time or historical market data, simulate trades, and analyze strategy performance through an intuitive and responsive UI. Users have the ability to select between different prediction markets, namely Kalshi and Polymarket. A backtest configuration allows users to specify the market to backtest, set custom date and time windows, and adjust execution parameters like latency. The platform then produces real-time order book visualizations and displays bid and ask data, depth charts, and price heatmaps. A historical simulation tool allows users to iterate through market activity and track the performance of their strategy. A performance dashboard is dynamically generated to show P&L, order fills, latency effects, and other metrics. The front end is designed to enable users to easily toggle

between markets, analyze backtests results, and gain insights into the efficacy of their strategy. While the actual strategy still has to be written in Python following a basic template in order to interact with the backtest engine, future developments might allow for strategies to be developed and tested without the need for hands-on code.

## 5. Pivots, Challenges, and Learning Experiences

One major pivot of the project was to focus efforts on crypto hourly markets before expanding analysis to cover broader markets. This narrower scope was an effective strategy because crypto markets are some of the most active markets, and we could record multiple iterations of the same market. The high-liquidity and high-frequency nature of these markets allowed us to rigorously characterize traditional market behavior and take those lessons into account in performing other analysis and building the backtesting tool.

The biggest challenge of this project, across the board, was data aggregation. Because prediction markets are so new, there are not many APIs or products on the market that offer extensive insight into order books. As discussed above, we had to reconstruct a synthetic Polymarket order book from scratch, which was no easy feat. Additionally, the Polymarket and Kalshi APIs were helpful in re-building the order books, despite their numerous limitations. Data collection was also an issue with our Market Analysis segment, as both free and granular crypto historical data is difficult to come by. We originally wanted to compare how quickly the prediction markets reacted to changes in cryptocurrency pricing. For instance, if the price of BTC unexpectedly jumped 1%, how soon would the hourly BTC market react? But finding free high-frequency historical crypto data was prohibitive. There are a wide array of free resources that offer historical OHLCV (Open, High, Low, Close, Volume) data on cryptocurrencies, but these resources did not come close to the level of granularity needed for latency analysis on these markets. Secondly, for the services that did offer granular historical cryptocurrency prices (like CoinAPI and CoinGecko), they were, shockingly, not free. At times data

aggregation created challenges for our project team, but we were still able to collect some outstanding market data from Polymarket and Kalshi.

A major learning experience from this project was just becoming familiar with how prediction markets work. Although our team had varying levels of prior experience with prediction markets, it was cumulatively beneficial to deep-dive into the market mechanics of this new, and exciting, trading landscape. Moreover, in class we heavily discussed traditional stock markets and futures markets, so examining a third, unique market type helped compliment a more holistic understanding of the prior two.

Besides learning general prediction market principles, the most important learning experience came from constructing the backtesting program. While not incredibly difficult to program, laying out the design of the program was challenging and each step had to be done accurately in order to produce realistic results. Building a backtesting program was a great exercise in applying our programming skills, our knowledge of markets, and our ability to reason about how best to represent counterfactual events.

## 6. Conclusion

Our project can be grouped into a data aggregation phase, a market analysis phase, and the backtesting tool phase. During the data aggregation phase, we constructed an order book framework for both Kalshi and Polymarket. The Polymarket data proved to be especially challenging, although we were able to construct a synthetic order book via historical order fills. The creation of a Kalshi order book was more straightforward, as we were able to livestream real-time market data from the Kalshi API. After scraping an array of different markets from both Kalshi and Polymarket, we moved into the market analytics phase, which focused on the hourly crypto markets. During this phase, we noticed that most hourly crypto markets were relatively illiquid and that illiquidity dropped substantially near resolution time or anytime the probability of an outcome approached certainty. We observed that there is opportunity for profitability as resolution time approaches, because many investors scramble to cover their positions. Also, a “double-bell curve” was observed in many markets, revealing high

standing orders near both $0 and $1 in the order book. Last, we physically built out a backtesting engine, as well as a functioning frontend to display the backtest to potential users.

While nowhere near a fully-functional service, this project has the capability to be the start of a functional and competitive tool used by a variety of prediction market participants. Theoretically, we could begin continually scraping prediction market data, collecting a valuable and proprietary dataset that would be a competitive advantage for any backtesting service. As we continue to study these markets, scrape data, and improve the maturity of our user-facing experience, this project could grow into a full-blown viable business.

Even if that business does not materialize, this project was an excellent exercise in combining software development, data analysis, technical knowledge, and passion for exotic markets.