Applications of Probability: Prediction Markets

Edgar Yepez

Math 482, Texas A&M, Spring 2026

Abstract

With the rising popularity of prediction markets and their remarkable predictive accuracy, they present an interesting case study for the application of probability. We will explore how prediction markets operate, the role of implied probabilities, and the mathematical principles that underpin their functionality. By examining real-world examples and theoretical models, we will illustrate how probability theory is applied in these markets to forecast outcomes and inform decision-making. Finally, we will examine the application of mathematical principles through a Python simulation of a prediction market.

1 Introduction

You walk into a grocery store, grab a carton of eggs, and head to the checkout just as you do every week. But then you notice the price has changed, it's higher, maybe a lot higher. You pay, and you think nothing of it.

But looking at the big picture, eggs are more than just breakfast; they are a direct physical representation of how the global economy is impacting the everyday person.

You don't know what the price of eggs will be next month, you try asking some friends, or spend some time looking online, and you get a general idea.

However, you can potentially save yourself some time by just looking at the wisdom of the crowds. A prediction market is a weighted collection of beliefs about a future event, aggregated into a single price. For example, a market maker will offer two securities: YES and NO. The YES security

will pay 1 dollar if the price of eggs is over 2.50 dollars next month, and 0 otherwise. Its complement is the NO security, which will pay 1 dollar if the event does not happen, and 0 if it does. The current price of the contract could take any value between 0 and 1, to make it directly proportional to a probability, having YES $= p$ and NO $= 1 - p$. A trader who is confident the price will be over 2.50 with probability p should be willing to pay up to p cents for the security. Similarly, he would be willing to buy a NO security up to p cents if he believes the probability/price of the YES security is too high. Because the price is determined through a supply and demand, buying and selling will cause the price to converge to a price reflecting all available infromation. Thus, the price of the security can be interpreted as the market's collective belief about the probability of an event occurring.

2 Preliminaries

2.1 Discrete Probability Model

To model the behavior of prediction markets, we introduce the framework of discrete probability theory. A probability space provides the mathematical structure for modeling random phenomena—in our case, the movement of security prices over time. Formally, a probability space is defined as a triple $(\Omega, \mathcal{F}, P)$ where:

- Ω (Sample Space) is the sample space, representing all possible outcomes or states of the world;
- $\mathcal{F}$ (sigma-algebra) is the collection of events, representing the information available at each point in time;
- P is the probability measure, assigning probabilities to each event in $\mathcal{F}$.

In the context of prediction markets, this framework allows us to rigorously describe how prices evolve as new information arrives.

2.1.1 Sample Space

Definition 2.1. *Let Ω represent all possible states of the world. Each element $\omega \in \Omega$ can be viewed as a sequence of price movements:*

$$\Omega = \{\omega : \omega = (a_1, a_2, \ldots, a_T)\}, \quad a_t \in \{u, d\}$$

By listing all possible future prices, we set all the possible states of the world. As time passes, more information is revealed about the true state of the world, which is a subset of Ω, $A \subset \Omega$. Having information about A gives us the values that are not in A, $\Omega \setminus A = \bar{A}$.

2.1.2 Fields of Events

Let $\mathcal{F}$ be the information available to investors up to time t.

Definition 2.2. *F_t is a σ-algebra of sets if*

1. $\emptyset, \Omega \in \mathcal{F}$
2. $A \in \mathcal{F} \Rightarrow A^c \in \mathcal{F}$
3. $A_i \in \mathcal{F}, i \in \mathbb{N} \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$

2.1.3 Filtration

Definition 2.3. *A filtration $\mathbb{F} = (\mathcal{F}_t)_{t \geq 0}$ is a collection of σ-algebras*

$$\mathbb{F} = \{\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_t \subseteq \mathcal{F}_{t+1} \subseteq \cdots \mathcal{F}_T\}$$

$\mathbb{F}$ is used to show the flow of information; as time passes, more and more information is revealed to the observer. As t increases, $\mathbb{F}$ becomes a finer partition of Ω.

2.1.4 Stochastic Process

Definition 2.4. *A stochastic process is a collection of random variables $\{X(t)\}$. For any fixed t, $t = 0, 1, \ldots, T$, $X(t)$ is a random variable on $(\Omega, \mathcal{F}_T)$.*

Definition 2.5. *A stochastic process is called* adapted *to filtration $\mathbb{F} = \{\mathcal{F}_t\}$ if for all $t = 0, 1, \ldots, T$, $X(t)$ is a random variable on $\mathcal{F}_t$, that is, if $X(t)$ is $\mathcal{F}_t$-measurable.*

Definition 2.6 ($\mathcal{F}_t$-Measurability)**.** *A random variable is a function $X : \Omega \to \mathbb{R}$. We say that X is $\mathcal{F}_t$-measurable if for every Borel set $B \subseteq \mathbb{R}$, the inverse image $X^{-1}(B)$ is an element of the σ-algebra $\mathcal{F}_t$:*

$$X^{-1}(B) \in \mathcal{F}_t \quad \textit{for all Borel sets } B \subseteq \mathbb{R}$$

2.1.5 Probability

At each node of a tree the price could move either up or down. Let p denote the probability of upwards movement and $1-p$ denote the probability of a downward movement. Then for a single step

$$p + (1 - p) = 1$$

2.1.6 Expectation

The expected value of a random variable is the theoretical mean of the random variable. To calculate the expected value for a discrete random variable X:

$$E(X) = \sum_{i=1}^{k} x_i P(X = x_i)$$

2.2 Martingales

To demonstrate that the future price is equivalent to the current probability of an event occurring, we need to introduce the concept of a martingale. In simple terms, a martingale is a model for a fair game, where given everything you know has happened up to now, X_n, then your best guess of the future value X_{n+1} is just the current value X_n. In other words, the expected value of the next outcome, given all the information you have, is equal to the current value.

Definition 2.7. *Let $X = X_1, X_2, \ldots, X_t$ be a sequence of random variables adapted to a filtration $F = F_1, F_2, \ldots, F_t$. We say that X is a martingale with respect to F if for all t, we have:*

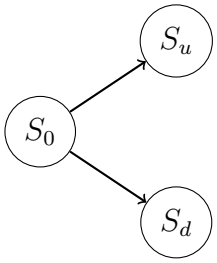
1. *$E[|X_t|] < \infty$ (the expected value of X_t is finite)*
2. *$E[X_{t+1}|F_t] = X_t$ (the conditional expectation of the next value given the past is equal to the current value)*

3 Binomial Trees

3.1 1-Level Trees

Definition 3.1 (II.1.1)**.** *Let the underlying price at time $t = 0$ be denoted by S_0. The 1-level tree model is the assumption that at time $t = T$ the underlying*

price can take one of two values, $S_u > S_0$ ("market up") or $S_d < S_0$ ("market down").



3.1.1 Risk-Neutral Probability Measure

Proposition 3.1. *The fair price of a security in a 1-level binomial tree is given by the risk-neutral expectation $E_q[S_T] = S_0$.*

Proof. Consider the binomial tree above: if S_0 represents the value of the "YES" security, then a fair price S_0 must satisfy:

$$E[S_T] = S_u \cdot p + S_d(1-p) = S_0$$

However, considering the time value of money with a risk-free interest rate r, the present value is:

$$E[S_T]e^{-rT} = S_0$$

Since we are considering short time periods, T is very small, so we approximate $e^{-rT} \approx 1$. To determine the probability, we define the risk-neutral measure q based on the inequality $S_d < S_0 < S_u$:

$$0 < \frac{(S_0 - S_d)}{(S_u - S_d)} < 1$$

Let $q = \frac{S_0 - S_d}{S_u - S_d}$. We now verify that this choice of q satisfies the martingale property by substituting back into the expectation:

$$E_q[S_T] = S_u \cdot q + S_d(1-q)$$

Substituting $q = \frac{S_0 - S_d}{S_u - S_d}$ and $1 - q = \frac{S_u - S_0}{S_u - S_d}$:

$$E_q[S_T] = S_u \cdot \frac{S_0 - S_d}{S_u - S_d} + S_d \cdot \frac{S_u - S_0}{S_u - S_d}$$

Combining the fractions:

$$E_q[S_T] = \frac{S_u(S_0 - S_d) + S_d(S_u - S_0)}{S_u - S_d} = \frac{S_u S_0 - S_u S_d + S_d S_u - S_d S_0}{S_u - S_d}$$

Simplifying the numerator:

$$E_q[S_T] = \frac{S_u S_0 - S_d S_0}{S_u - S_d} = \frac{S_0(S_u - S_d)}{S_u - S_d} = S_0$$

This confirms that the discounted price process is a martingale under measure q, establishing that the fair price of a security is equal to the risk-neutral expectation of its terminal value. □

4 Multi level Binomial Trees

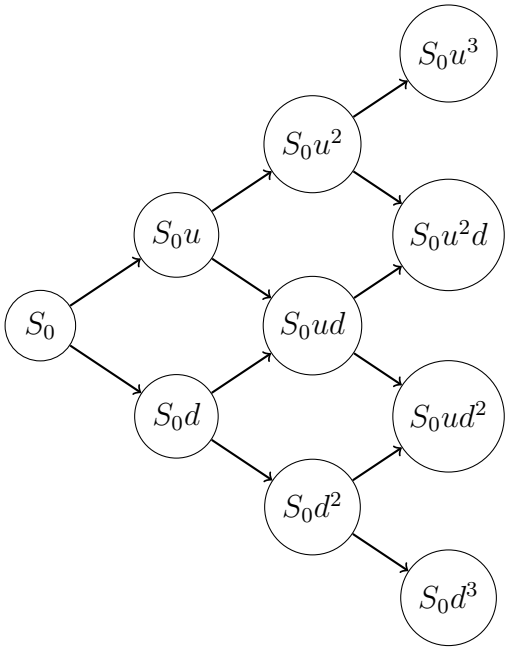
We can expand the previous model to multiple levels, where the price can move up or down at each step. For a termination point T and a given L number of levels, we set $\Delta t = T/L$. The up and down factors are:

$$u = \frac{S_u}{S_0}, \qquad d = \frac{S_d}{S_0}$$

At each node, the price branches as:

$$S^{\text{next}} = \begin{cases} S^{\text{prev}} \cdot u & \text{(up)} \\ S^{\text{prev}} \cdot d & \text{(down)} \end{cases}$$

This approach allows us to model prediction markets similarly to how we model stock and option prices. The price of the security at time t is the expected value of the future price at time T, discounted by the risk-free rate, and calculated using the risk-neutral probability measure. The following tree illustrates the price movements for a 3-level binomial tree, where the price can move up or down at each step.



With the previously established properties of the binomial tree, we can rewrite the risk-neutral probability measure as follows:

$$q = \frac{1-d}{u-d}$$

However, due to the multiplicative nature of the tree that is used to model price movements with a possibly unbounded gain and limited loss, we can see that the tree becomes asymmetrical as we expand the number of levels. Intuitively, this is not how the price of a prediction market should behave, as the price of a security should represent the probability of an event occurring.

Instead, to model the price of a prediction market, we can use an arithmetic binomial tree, where the price movements are characterized by a random variable, $X = \mu + \sigma \cdot Z$, where Z is the standard normal random variable $Z_i \sim N(0,1)$. We impose the risk-neutrality assumption $\mu = r$, where r is the risk-free rate. As stated previously, we will set $r = 0$ due to the short time periods we are considering. Finally, to maintain time consistency, we will scale X by the square root of the time increments $\sqrt{\Delta t}$. This results in a binomial tree where each node branches to:

$$S^{Next} \begin{cases} S^{Prev} + \sigma \cdot \sqrt{\Delta t} \\ S^{Prev} - \sigma \cdot \sqrt{\Delta t} \end{cases}$$

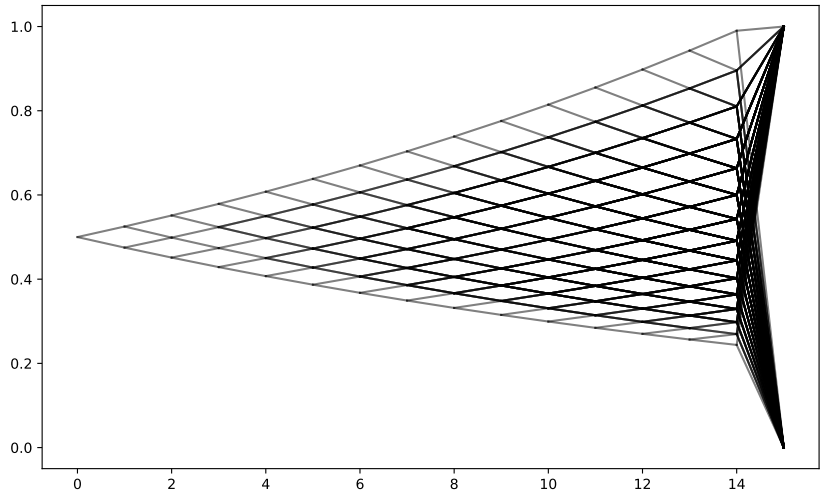


Figure 1: Geometric Binomial Tree Possible Paths

Now to account that in a prediction market once the price reaches 0 or 1, the event is consider to be resolved,w e will force the price to be 0 or 1 once it reaches those values. Additonaly to ensure that the price converges at the termination point, we will force the price to be 0 or 1 at the last time step, depending on weather the price is above or below 0.5. This results in the following set of equations for the price at time t_i:

$$S(t_i) = \begin{cases} 0 & \text{if } S(t_{i-1}) \leq 0 \\ 1 & \text{if } S(t_{i-1}) \geq 1 \\ \mathbb{I}(S(t_{i-1}) > 0.5) & \text{if } i = L \\ S(t_{i-1}) + \sigma \cdot \sqrt{\Delta t} Z_i & \text{otherwise} \end{cases}$$

It is important to note that the reason we use $\sqrt{\Delta t}$ for scaling is due to how variance scales up over time. The variance of one step in the model is $\text{VAR}(\sigma \cdot \sqrt{\Delta t} Z_i) = \sigma^2 \cdot \Delta t \cdot \text{VAR}(Z_i)$. Over L timesteps, we have $\sigma^2 \cdot \Delta t \cdot \text{VAR}(\sum_{i=1}^{L} Z_i) = \sigma^2 \cdot \Delta t \cdot L = \sigma^2 \cdot T$. The Δt cancels out perfectly, having the total amount of variance only dependent on the total time T and volatility σ.

Theorem 4.1. *The risk-neutral probability measure for the arithmetic binomial tree is* $q = 1/2$.

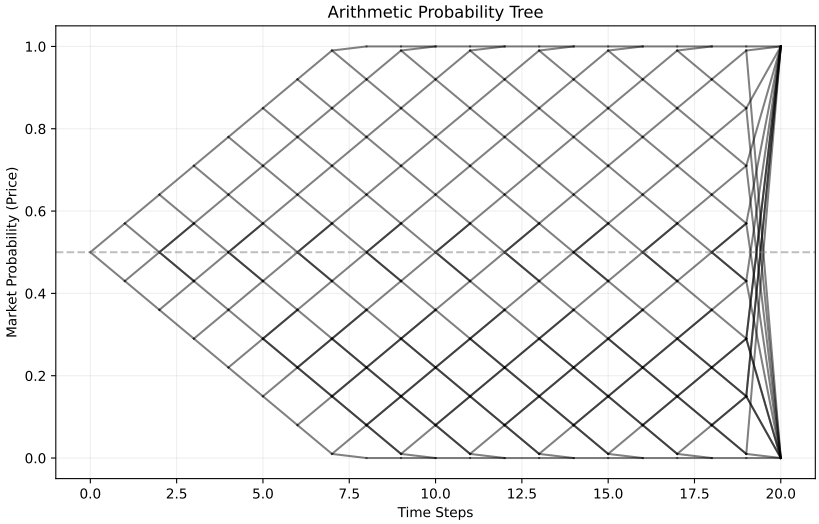


Figure 2: Arithmetic Binomial Tree Possible Paths

Proof. Using the martingale property of the tree, we can find the risk-neutral probability measure for the arithmetic binomial tree as follows:

$$\begin{aligned}
E_q[S_T] &= (S_u) \cdot q + (S_d) \cdot (1-q) = S_0 \\
E_q[S_T] &= (S_0 + \sigma \cdot \sqrt{\Delta t}) \cdot q + (S_0 - \sigma \cdot \sqrt{\Delta t}) \cdot (1-q) = S_0 \\
S_0 \cdot q + \sigma \cdot \sqrt{\Delta t} \cdot q + S_0 - S_0 q - \sigma \cdot \sqrt{\Delta t} + \sigma \cdot \sqrt{\Delta t} \cdot q &= S_0 \\
2\sigma \cdot \sqrt{\Delta t} \cdot q + S_0 - \sigma \cdot \sqrt{\Delta t} &= S_0 \\
2\sigma \cdot \sqrt{\Delta t} \cdot q &= \sigma \cdot \sqrt{\Delta t} \\
q &= \frac{1}{2}
\end{aligned}$$

□

We can see that the risk-neutral probability measure for the arithmetic binomial tree is 0.5, which is what we would expect due to the Efficient Market Hypothesis. The Efficient Market Hypothesis states that the price of a security should reflect all the information available, and thus the price should be an unbiased estimator of the true probability of an event occurring. Having a probability other than 0.5 would imply that there is an arbitrage opportunity, and thus the price would not be an unbiased estimator of the true probability.

5 Pascal's Triangle and the Binomial Distribution

The binomial tree model inherently links discrete probability to combinatorial structures. As we expand the tree, the number of distinct paths to reach a specific node follows the patterns found in Pascal's Triangle.

$$
\begin{array}{ccccccccccc}
 & & & & 1 & & & & & & \\
 & & & 1 & & 1 & & & & & \\
 & & 1 & & 2 & & 1 & & & & \\
 & 1 & & 3 & & 3 & & 1 & & & \\
1 & & 4 & & 6 & & 4 & & 1 & &
\end{array}
$$

Lemma 5.1. *In a binomial tree with n steps, the probability of the price finishing at a node corresponding to k upward movements and $n-k$ downward movements is given by the binomial distribution:*

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Proof. Each unique path from S_0 to a terminal node at time n consists of a sequence of n independent movements. If a node represents k up-steps (each with probability p) and $n-k$ down-steps (each with probability $1-p$), the probability of following any *single* such path is $p^k(1-p)^{n-k}$ due to the multiplicative nature of independent probabilities.

The number of ways to choose k up-steps out of n total steps is given by the binomial coefficient $\binom{n}{k}$, which corresponds exactly to the k-th entry of the n-th row in Pascal's Triangle. Summing the probabilities of all disjoint paths to this node yields the result. □

6 Brownian Motion

Definition 6.1. *Given our probability space, Brownian motion is defined as a stochastic process $W(t)$ with the following properties:*

- $W(0) = 0$;
- $W(t)$ *has independent increments;*

- *$W(t)$ has normally distributed increments with mean 0 and variance t, that is, $W(t) - W(s) \sim N(0, t-s)$ for $0 \leq s < t$;*
- *$W(t)$ has continuous paths.*

How will we simulate a Brownian process? Let N be a positive integer, and we divide the interval $[0, T]$ into N equal subintervals of length $\Delta t = T/N$. We can define the Brownian motion at time t_i as follows:

$$W_t = (W_1 - W_0) + (W_2 - W_1) + \cdots + (W_i - W_{i-1}) = \sum_{i=1}^{N} \Delta(W_i) = \sum_{i=1}^{N} \sqrt{\Delta t} \cdot Z_i$$

where Z_i is a standard normal random variable $Z_i \sim N(0, 1)$.

Notice that we can plug Brownian motion directly into our arithmetic binomial tree model—both have increments drawn from a normal distribution with mean 0 and variance $\sigma^2 \Delta t$, which accumulates to $\sigma^2 T$ over time. To ensure that the price of the security in our prediction market model behaves like a Brownian motion, we can define the price at time t_i as follows:

$$S(t_i) = \begin{cases} 0 & \text{if } S(t_{i-1}) \leq 0 \\ 1 & \text{if } S(t_{i-1}) \geq 1 \\ \mathbb{I}(S(t_{i-1}) > 0.5) & \text{if } i = L \\ S(t_{i-1}) + \sigma \cdot W_t & \text{otherwise} \end{cases}$$

Because of the conditions that we have set, the distribtion of the endition points will behavive differently from a standard brownian motion, we will model it using python code to see how the deistibtion behaves as time passes.

From the simulation, we can see that the distribution is normal at the start. As time moves forward, variance increases; by half-time, the distribution looks arguably more uniform, and as we get closer to the end of the time period, the distribution becomes more bimodal, with peaks at 0 and 1. This is because as time passes, more and more paths of the Brownian motion will hit either 0 or 1, which represent the event not occurring or occurring, respectively.

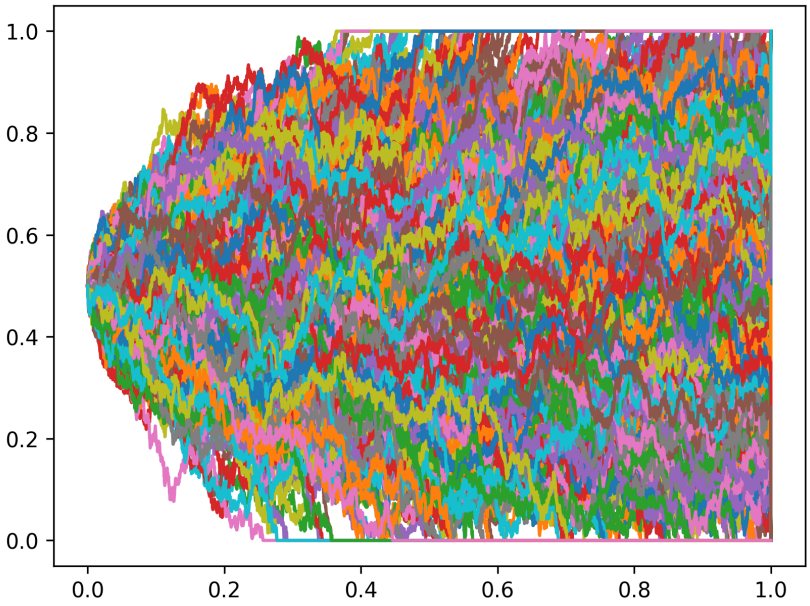


Figure 3: Brownian Prediction Markets

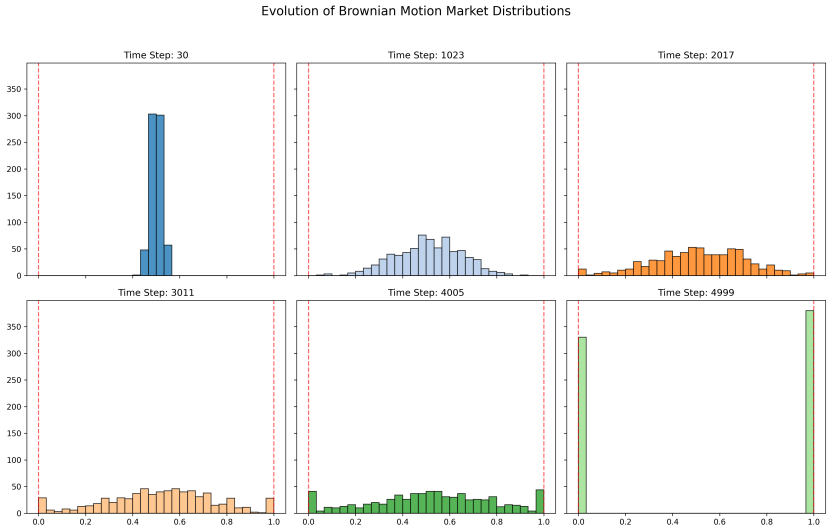


Figure 4: Brownian Prediction Markets Distribution Evolution

7 Real-World Observations

To draw similarities between our model for prediction markets and real-world examples, we will utilize data from platforms such as Kalshi and Polymarket.

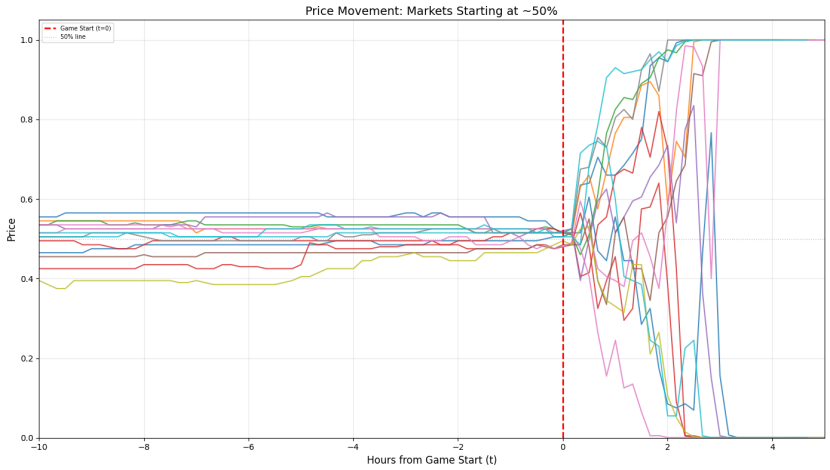


Figure 5: Sports Markets

Sports markets are among the most popular markets on prediction market platforms. They offer money if you can predict the outcome of a sporting event; for example, the winner of a "Magic vs 76ers." It will pay 1 per share in the case that the 76ers win, and 0 if they lose, and the opposite is true for the "Magic win" security. In the image above, we limited the games to those starting at approximately 0.5 probability.

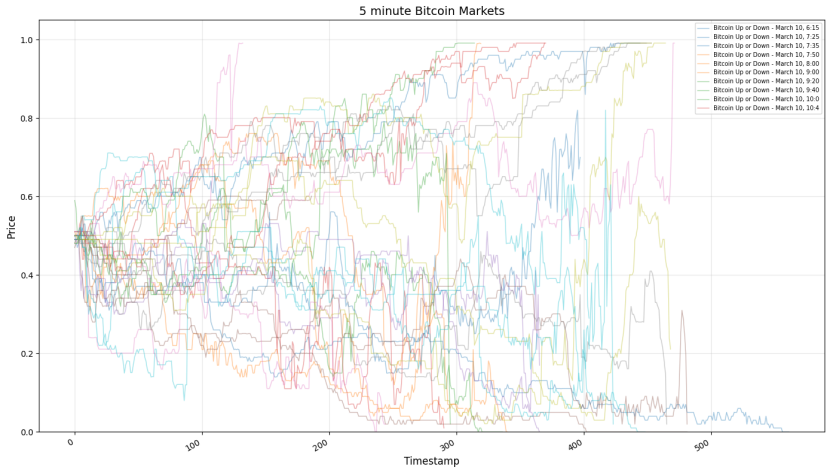


Figure 6: 5-minute Bitcoin Markets Polymarket

Bitcoin markets offer the up and down security, which pays you according

to whether the price of Bitcoin is above or below the starting price of the 5-minute interval. The up security will pay 1 dollar per share if the price is above the starting point, 0 otherwise. This is what was used to generate the above illustration.

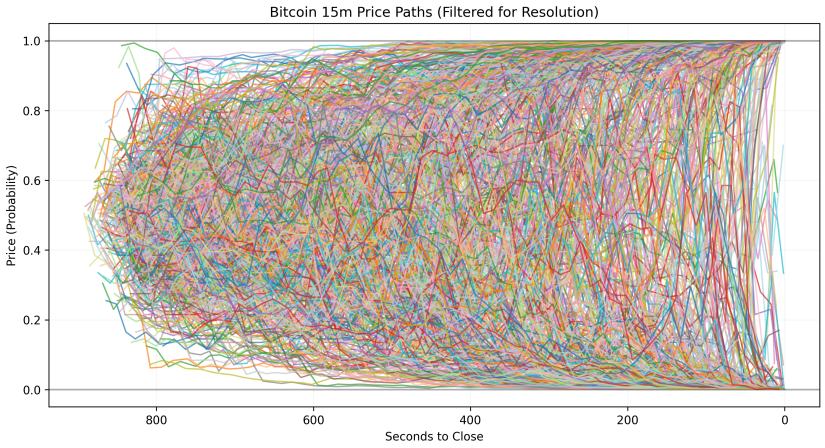


Figure 7: 15-minute Bitcoin Markets Kalshi

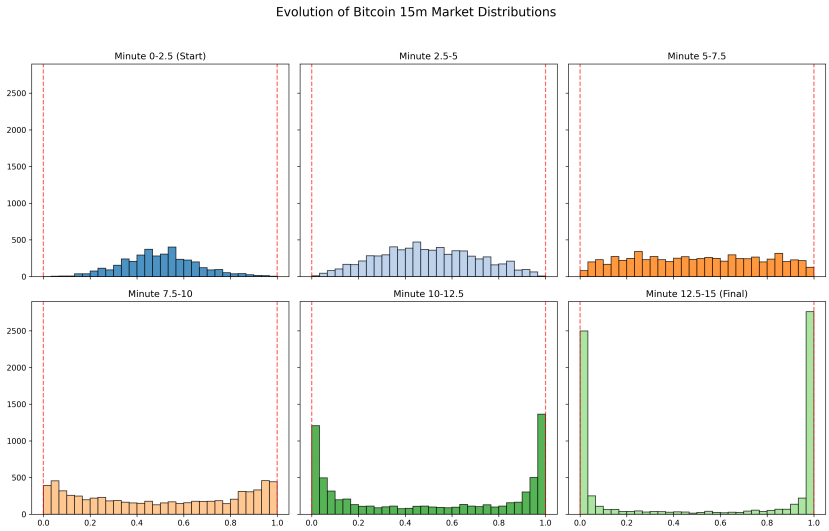


Figure 8: 15-minute Bitcoin Markets Distribution Kalshi

Lastly, we have the 15-minute Bitcoin markets derived from Kalshi, which are exactly the same as the 5-minute markets but with a longer time period.

The illustration below demostrates the price paths of 720 bitcoin markets. We can see that the distribution approximately follows the same pattern as the one we observed in our Brownian motion simulation, with the distribution becoming more bimodal as time passes and starting with a normal distribution around 0.5.

References

[1] Y. Chen and J. W. Vaughan, "A new understanding of prediction markets via no-regret learning," arXiv preprint arXiv:1003.0034, 2010.

[2] O. Saguillo, V. Ghafouri, L. Kiffer, and G. Suarez-Tangil, "Unravelling the Probabilistic Forest: Arbitrage in Prediction Markets," arXiv preprint arXiv:2508.03474, 2025.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*, Lecture Notes, Course 6.041-6.431, Massachusetts Institute of Technology, Fall 2000.

[4] D. J. Aldous, "Using prediction market data to illustrate undergraduate probability," 2012.

[5] D. Klein, "Prediction market prices as martingales: Theory and analysis," Statistics 157.

[6] F. C. Klebaner, *Introduction to Stochastic Calculus with Applications*, Imperial College Press, Monash University, Australia.

[7] G. Berkolaiko, *Undergraduate Introduction to Mathematics of Option Pricing*.

A Python Implementation

A.1 Geometric Binomial Tree

```
import matplotlib.pyplot as plt
import numpy as np
from pathlib import Path

OUTPUT_DIR = Path(__file__).resolve().parents[1] / "figures"

levels = 15
Time = 15
deltaTime = Time / levels
sigma = 0.08
u = 1.05
d = 0.95
previous = {0.5}


plt.figure(figsize=(10, 6))


for i in range(levels):
    current = set()
    for p in previous:
        p = round(p, 5)
        if p == 0 or p == 1:
            current.add(p)
            plt.plot([i, i + 1], [p, p], color="black", alpha=0.5)
            continue
        if i == levels - 1:
            plt.plot([i, i + 1], [p, 1], color="black", alpha=0.5)
            plt.plot([i, i + 1], [p, 0], color="black", alpha=0.5)
            continue

        up = p * u
        down = p * d
        if up >= 1:
            up = 1
        if down <= 0:
            down = 0
        current.add(up)
        current.add(down)

        plt.plot([i, i + 1], [p, up], color="black", alpha=0.5)
        plt.plot([i, i + 1], [p, down], color="black", alpha=0.5)

    previous = current

plt.savefig(OUTPUT_DIR / "GeometricBinomialTree.pdf", bbox_inches="tight")
plt.savefig(OUTPUT_DIR / "GeometricBinomialTree.png", dpi=150, bbox_inches="tight")
```

A.2 Arithmetic Binomial Tree

```
import matplotlib.pyplot as plt
import numpy as np
from pathlib import Path

OUTPUT_DIR = Path(__file__).resolve().parents[1] / "figures"

levels = 20
Time = 20
deltaTime = Time / levels
sigma = 0.07
r = 0.00
previous = {0.5}


plt.figure(figsize=(10, 6))


for i in range(levels):
    current = set()
    for p in previous:
        p = round(p, 5)
        if p == 0 or p == 1:
            current.add(p)
            plt.plot([i, i + 1], [p, p], color="black", alpha=0.5)
            continue
        if i == levels - 1:
            plt.plot([i, i + 1], [p, 1], color="black", alpha=0.5)
            plt.plot([i, i + 1], [p, 0], color="black", alpha=0.5)
            continue

        up = p + sigma * np.sqrt(deltaTime) + r * deltaTime
        down = p - sigma * np.sqrt(deltaTime) + r * deltaTime
        if up >= 1:
            up = 1
        if down <= 0:
            down = 0
        current.add(up)
        current.add(down)

        plt.plot([i, i + 1], [p, up], color="black", alpha=0.5)
        plt.plot([i, i + 1], [p, down], color="black", alpha=0.5)

    previous = current


plt.title(f"Arithmetic Probability Tree")
plt.xlabel("Time Steps")
plt.ylabel("Market Probability (Price)")
plt.axhline(0.5, color="gray", linestyle="--", alpha=0.5)
plt.ylim(-0.05, 1.05)
plt.grid(True, alpha=0.2)
plt.savefig(OUTPUT_DIR / "ArithmeticBinomialTree.pdf", bbox_inches="tight")
plt.savefig(OUTPUT_DIR / "ArithmeticBinomialTree.png", dpi=150, bbox_inches="tight")
```

A.3 Brownian Motion for Prediction Markets

```python
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path

OUTPUT_DIR = Path(__file__).resolve().parents[1] / "figures"


def Pred_Mkt_Bronwian_Motion(dW, d):
    steps = dW.shape[1] + 1
    B = np.zeros((d, steps))
    B[:, 0] = 0.5

    for i in range(d):
        for j in range(1, len(B[i])):
            B[i][j] = B[i][j - 1] + dW[i][j - 1]
            if B[i][j] >= 1 or B[i][j - 1] >= 1:
                B[i][j] = 1
            elif B[i][j] <= 0 or B[i][j - 1] <= 0:
                B[i][j] = 0

            if j == steps - 1:
                if B[i][j] >= 0.5:
                    B[i][j] = 1
                elif B[i][j] < 0.5:
                    B[i][j] = 0

    return B


def plot_distribution_grid(B, cols=3):
    # Select 6 snapshots from start to finish
    num_snapshots = 6
    indices = np.linspace(30, B.shape[1] - 1, num_snapshots, dtype=int)
    rows = (num_snapshots + cols - 1) // cols

    fig, axes = plt.subplots(rows, cols, figsize=(15, 10), sharex=True, sharey=True)
    axes = axes.flatten()
    cmap = plt.get_cmap("tab20")

    for i, idx in enumerate(indices):
        ax = axes[i]
        data = B[:, idx]

        # Plot histogram with color from colormap
        ax.hist(
            data,
            bins=30,
            range=(0, 1),
            color=cmap(i % 20),
            edgecolor="black",
            alpha=0.8,
        )

        # Add reference lines for 0 and 1
        ax.axvline(0, color="red", linestyle="--", alpha=0.6)
        ax.axvline(1, color="red", linestyle="--", alpha=0.6)

        ax.set_title(f"Time Step: {idx}")
        ax.set_xlim(-0.05, 1.05)

    plt.suptitle("Evolution of Brownian Motion Market Distributions", fontsize=16)
    plt.tight_layout(rect=[0, 0.03, 1, 0.95])
    plt.savefig(OUTPUT_DIR / "Brown_PM_Distributions_Evolution.png", dpi=300, bbox_inches="tight")


def main():
    n = 5000
    d = 710
    sigma = 0.3
    T = 1
    time = np.linspace(0, T, n)
    dt = time[1] - time[0]
    dW = sigma * np.sqrt(dt) * np.random.normal(size=(d, n - 1))
    B = Pred_Mkt_Bronwian_Motion(dW, d)
```

```
75      plot_distribution_grid(B)
76      plt.show()
77
78      plt.plot(time, B.T)
79      plt.savefig(OUTPUT_DIR / "Bronwian_Prediciton_Markets.png", dpi=300, bbox_inches="tight")
80      plt.show()
81
82
83  if __name__ == "__main__":
84      main()
```