

## ארביטראז' קלנדרי בשוקי תחזיות – מחקר מעמיק

**מבוא:** שוקי תחזיות כמו Polymarket מאפשרים למסחר חוזים על אירועים עתידיים, כאשר מחיר חוזה "כן" (YES) או "לא" (NO) משקף את ההסתברות הנתפסת להתרחשות האירוע. במחקרים עדכניים נחשף כי בשווקים אלו קיימים כשלי תמחור כרוניים המנוצלים ע"י סוחרים מתוחכמים לארביטראז' – כלומר הפקת רווח חסר-סיכון מהפרשי מחירים <sup>1</sup>. <sup>2</sup> אחד מסוגי הארביטראז' הבולטים הוא **ארביטראז' קלנדרי (Calendar Arbitrage)** – ארביטראז' המבוסס על יחסי גומלין לוגיים בין שווקים שתוחמים אירוע בזמן. במסמך אופרטיבי ומעמיק זה נפרט את אסטרטגיית הארביטראז' הקלנדרי, נוכיח את תקפותה מתמטית בשלל תרחישים, נדון בהיבטים מעשיים (עמלות, נזילות, Slippage, סיכויי ביצוע), ונציע ארכיטקטורת מערכת מלאה לזיהוי והוצאה לפועל אוטומטיים של עסקאות כאלו. כמו כן נסקור שיטות NLP לזיהוי שווקים תלויים, מנוע חישוב כדאיות המבוסס על עומק **Order Book** ולא רק מחירי **Mid**, מנגנוני ניטור בזמן אמת ואסטרטגיות יציאה, נהלי ניהול סיכונים, מודולים מרכזיים בארכיטקטורה, טכנולוגיות מומלצות למימוש, ולבסוף – מקרים וניתוחים מדגמיים מאירועים בשנים 2024–2025 (בחירות, אירועים גיאופוליטיים, השקות מוצרים) עם תובנות גם מפלטפורמות נוספות מעבר ל-Polymarket.

### 1. אסטרטגיית Calendar Arbitrage: תנאי הארביטראז' ויחסי מחירים

**הגדרת האסטרטגיה:** ארביטראז' קלנדרי מתאפשר כאשר קיימים שני שווקים (או יותר) על אותו אירוע, אך בעלי מגבלת זמן שונה. למשל, שוק A שואל "האם אירוע X יקרה עד תאריך T1?" ושוק B שואל "האם אירוע X יקרה עד תאריך T2?" (כאשר T2 מאוחר יותר מ-T1). באופן לוגי, ההסתברות שאירוע יתרחש עד תאריך מאוחר (T2) צריכה להיות גדולה או שווה להסתברות שיתרחש עד תאריך מוקדם יותר (T1), משום שתנאי T2 כולל את T1 ועוד זמן נוסף. במונחי מחירי שוק, אם נסמן ב-P1 את מחיר חוזה ה"YES" בשוק עד T1, וב-P2 את מחיר חוזה ה"YES" עד T2, התנאי הלוגי הוא:  $P1 \leq P2$ . הפרת תנאי זה מייצרת עיוות מחירים שניתן לנצל לארביטראז' חסר-סיכון: אם בפועל  $P1 > P2$  (כלומר השוק בעל הדדליין הקצר מתמחר הסתברות גבוהה יותר מהשוק הארוך) – ניתן לבצע עסקה משולבת של מכירת חוזה ה"YES" בשוק הקצר (או קניית "NO" בשוק הקצר) וקניית חוזה ה"YES" בשוק הארוך, ובכך לנעול רווח ללא תלות בתוצאות העתידיות <sup>3</sup>. אסטרטגיה זו זהה במהותה לארביטראז' קלנדרי באופציות: מוכרים את החוזה קצר-הטווח היקר וקונים את החוזה הארוך-הטווח הזול באותו בסיס, לנעילת מרווח בטוח <sup>4</sup>.

**הגיון לוגי ויחסי מחירים:** התנאי  $P1 \leq P2$  נובע מהיחס "אם האירוע קרה עד T1, אז ודאי קרה עד T2". לכן, בלתי אפשרי מבחינה הגיונית ששוק "כן עד T1" יתמחר הסתברות גבוהה משוק "כן עד T2" – מצב כזה משמעו הזדמנות ארביטראז' ודאית. באופן דומה, בצד ה"לא": ההסתברות **לא-התרחשות** האירוע עד T1 צריכה להיות גדולה או שווה להסתברות לא-התרחשות עד T2 (שהוא פרק זמן ממושך יותר שבו האירוע עוד עלול לקרות). למעשה, העדר ארביטראז' מחייב שמחירי חוזה בינארי יעמדו במגבלת **מונוטוניות בזמן** – מחיר חוזה ככל שפג תוקפו רחוק יותר בזמן אמור לא לרדת מתחת למחיר חוזה זהה קצר טווח <sup>3</sup>. כאשר תנאי זה מופר, ניתן לבנות פורטפוליו של פוזיציות בשווקים הקצרים והארוכים שמבטיח **תשואה ללא סיכון** (Risk-Free Calendar Spread) <sup>4</sup>.

באופן מעשי, אם מזהים למשל ב-Polymarket שני שווקים על אותו אירוע עם מועדי סיום שונים, והחוזה הקצר יקר מהארוך, האסטרטגיה היא: - **למכור חוזה ה"YES" בשוק הקצר** (או לחילופין, לקנות את חוזה ה"NO" בשוק הקצר – שקול כלכלית למכירת ה"YES"), - **ולקנות חוזה ה"YES" בשוק הארוך**.

צעד זה מנצל את העיוות: אנו מוכרים ביוקר את ההסתברות בטווח הקצר וקונים בזול את אותה ההסתברות לטווח ארוך יותר. כעת ננתח מהי התשואה בכל תרחיש אפשרי להתרחשות האירוע.

## 2. הוכחה מתמטית לתקפות האסטרטגיה בשלושת התרחישים (Payoff מובטח)

כדי להוכיח שהאסטרטגיה אכן חסרת-סיכון, ננתח את **שלושת התרחישים ההדדיים** לגבי מועד התרחשות האירוע: 1. **התרחשות לפני T1**: האירוע קורה מוקדם (לפני או עד T1). 2. **התרחשות בין T1 ל-T2**: האירוע לא קרה עד T1, אך קורה מאוחר יותר לפני T2. 3. **אי-התרחשות עד T2**: האירוע אינו מתרחש כלל (לא עד T2 ולא אחריו).

נתייחס לפורטפוליו הארביטראז' שבנינו: פוזיציה 1: "Short YES" **בשוק T1** – מכירת חוזה "כן עד T1" במחיר P1 (שקול לרכישת "לא עד T1" במחיר  $(P1-1)$ ). פוזיציה 2: "Long YES" **בשוק T2** – רכישת חוזה "כן עד T2" במחיר P2.

ההשקעה ההתחלתית (עלות הפורטפוליו) היא:  $Cost = (1 - P1) + P2$ . ביטוי זה מייצג את הסכום ששילמנו: שילמנו  $(P1-1)$  דולר עבור חוזה ה"NO" בשוק הקצר, ו-P2 דולר עבור חוזה ה"YES" בשוק הארוך. נחשב כעת את התשלום (Payoff) המתקבל מכל פוזיציה בכל תרחיש, ואת הסכום הכולל:

תרחיש	תוצאת שוק T1	תוצאת שוק T2	סך התשלום מפורטפוליו הארביטראז'
1. אירוע קורה לפני T1	YES עד T1 = 1 (מתרחש) – אנחנו מכרנו Yes, לכן מפסידים $P1-1$ (שילמנו למשקיע הנגדי את ערך החוזה) ומנגד החוזה "NO" שקנינו שווה \$0.	YES עד T2 = 1 (מתרחש) – אנחנו קנינו Yes, מקבלים \$1.	\$1 דולר (שילמנו \$Cost\$, קיבלנו \$1\$ חזרה; ראה ניתוח בהמשך)
2. אירוע קורה בין T1 ל-T2	YES עד T1 = 0 (לא קרה עד T1) – שבידינו משלם \$1.	YES עד T2 = 1 (כן קרה עד T2) – החוזה Yes שבידינו משלם \$1.	\$2 דולר (קיבלנו \$1\$ משוק T1 ועוד \$1\$ משוק T2)
3. אירוע לא קורה כלל עד T2	YES עד T1 = 0 – החוזה "NO" עד T1 משלם \$1.	YES עד T2 = 0 – החוזה Yes שלנו שווה \$0.	\$1 דולר (קיבלנו \$1\$ משוק T1, \$0\$ משוק T2)

ניתוח התוצאות: בפורטפוליו שלנו, בכל תרחיש התשלום המינימלי שמתקבל הוא \$1. בתרחיש 2 אף מתקבלים \$2. מה הייתה ההשקעה ההתחלתית?  $Cost = (1 - P1) + P2$ . אם אכן היה עיוות מחירים כך ש- $P1 > P2$ , אזי  $Cost = 1$  – כלומר השקענו פחות מ-\$1 דולר. לפיכך: **תרחיש 1**: החזר \$1 דולר עלות – רווח \$1 – **Cost** (חיובי כי  $Cost < 1$ ). **תרחיש 2**: החזר \$2 דולר – רווח \$2 – **Cost** (בוודאי חיובי משמעותית). **תרחיש 3**: החזר \$1 דולר – רווח \$1 – **Cost** (חיובי).

בכל מקרה לא ספגנו הפסד, והרווח המובטח בכל התרחישים הוא לפחות  $(Cost - 1)$ , שהוא מספר חיובי כאשר  $Cost > 1$ . זהו **ארביטראז' חסר-סיכון**: לא משנה מתי (ואם בכלל) יתרחש האירוע X, הפורטפוליו יניב רווח מובטח. במקרה הגרוע ביותר נקבל בדיוק את הקרן בחזרה + רווח קטן, ובתרחישים אחרים נקבל אף יותר. מכאן נובע שתנאי  $P1 > P2$  **הוא תנאי לאפשרות ארביטראז'**, ובהכרח במצב שיווי משקל נטול ארביטראז' לא יתקיים  $P1 > P2$ .

חשוב לציין שגם במצב גבולי בו  $P1 = P2$ , עלות הפוזיציה  $Cost = 1$ . בתרחיש 1 ו-3 אז נקבל \$1 בדיוק – כלומר נקודת איזון (לא הפסד ולא רווח), ובתרחיש 2 נקבל \$2 (רווח \$1). מצב זה עדיין אטרקטיבי לסוחר משום שאין סיכון הפסד (Break-Even בתרחיש הגרוע, ורווח חיובי באחרים). לכן, בשווקים תחרותיים אפילו שוויון ייצור תמריץ למסחר עד להתכנסות ל- $P1 \leq P2$ . הדוקה עם פער מינימלי המשקף אולי עלויות עסקה. בפועל, קיום פער ממשי שבו  $P1 > P2$  משמעותית מ-P2 מהווה הזדמנות רווח אשר סוחרים "חכמים" צפויים לקפוץ עליה מיד <sup>2</sup>.

**דוגמה מספרית:** נניח שוק A ("כן עד 1 ביולי 2025") נסחר ב-0.60 (כלומר 60% הסתברות) ושוק B ("כן עד 1 בינואר 2026") נסחר ב-0.50 (50%). לפי הלוגיקה, היה על שוק B להיות לפחות 0.60. הארביטראז' ימכור חוזה "Yes" בשוק A

ב-0.60 (ובכך בעצם קונה "No" ב-0.40) ויקנה "Yes" בשוק B ב-0.50. העלות:  $0.50 + \$0.40 = \$0.90$  לכל יחידה. כעת ניתוח תרחישים: - אם האירוע קרה לפני יולי 2025: הוא קרה גם לפני 2026. מפוזציותיו הסותר יצטרך לשלם \$1 למי שקנה ממנו את חוזה ה-Yes בשוק A (הפסד \$0.40 כי הוא קיבל \$0.60 ומחזיר \$1) אך יקבל \$1 על חוזה ה-Yes שבידיו בשוק B. נטו קיבל \$1, השקיע \$0.90, רווח \$0.10. - אם קרה במחצית השנייה של 2025: שוק A פג כמפסיד (No) משלם \$1 - הוא הרוויח \$0.60 כי קיבל \$0.60 ממכירת כן ומחזיר \$0 (לזוכה ה-Yes), שוק B זוכה \$1. סה"כ קיבל \$2 על השקעה \$0.90, רווח \$1.10. - אם לא קרה עד סוף 2025: שוק A "לא" משלם \$1, שוק B מפסיד (\$0=Yes). קיבל \$1 על \$0.90, רווח \$0.10.

הרווח בכל מקרה חיובי. אילו  $P1 \leq P2$  כנדרש (נניח 50%-50%), אסטרטגיה כזו הייתה לפחות לא מפסידה (Break-even מינימלי), ולכן בשוק יעיל הפרש כזה לא אמור להחזיק מעמד ללא שינוי מחירים.

### 3. השפעת עמלות, נזילות, Slippage ו-Leg Risk על הרווחיות

הניתוח לעיל מניח עולם תאורטי ללא חיכוך - אך במציאות ישנם גורמים מעשיים המצמצמים את הרווח ואפילו יוצרים סיכון אם לא מנוהלים כראוי:

- **עמלות מסחר ודמי פלטפורמה:** עמלות עלולות "לאכול" חלק או את כל מרווח הארביטראז'. למשל, אם כל עסקה כרוכה בעמלה של 1%-2%, הפער  $P2 - P1$  צריך להיות גבוה מכך כדי שהרווח הנקי יישאר חיובי. למרבה המזל, פלטפורמת Polymarket נכון ל-2025 אינה גובה עמלת מסחר (0% fee) <sup>5</sup>, כך שעלות העסקה מתמצה בעיקר בעמלות רשת (גז) ותשלום מרווחי מחיר. בפלטפורמות אחרות (כגון Kalshi) יש עמלת מסחר או עמלת סיום (fee על הזכייה) שיש להביא בחשבון. לפיכך, במסגרת מנוע החישוב (ראו סעיף 5) יש להחסיר את העמלות מכל רווח תאורטי ולוודא שהרווח הנקי עומד בסף מינימלי רצוי.
- **עלויות גז ועסקה:** בפלטפורמות מבוססות כמו Polymarket (הפועלת על Polygon) קיים מחיר פעולה ברשת. הוצאה לפועל של שתי פקודות (קנייה ומכירה) כרוכה בתשלום גז. עלויות אלה **מקצות את הרווח** בעסקאות קטנות, ולכן יש לוודא שהארביטראז' "שווה את זה" גם לאחר הוצאות הגז. למשל, ב-Polymarket עסקה טיפוסית עשויה לעלות עשרות סנט (תלוית עומס רשת ומחיר מטיק) <sup>6</sup> <sup>7</sup>. בוט ארביטראז' חכם יחשב את הרווח הצפוי ויפחית ממנו את עלות הגז, ורק אם ה-ROI הנקי עולה על סף מסוים (נניח 2%) העסקה תוכרז כרווחית <sup>8</sup>. לדוגמה, קביעה מובנית: "דרוש לפחות 2% תשואה נקייה לאחר עלויות" <sup>9</sup>.
- **נזילות (Liquidity) וספר הפקודות:** הארביטראז' התאורטי מחשיב מחירי שוק כאילו אפשר לבצע את העסקאות בכמות הרצויה במחירים הללו. בפועל, ייתכן שרק כמות מוגבלת של חוזים זמינה במחיר היעד (בעומק ה-Order Book הקרוב). אם ננסה לבצע כמות גדולה, נאלץ "לאכול" פקודות נוספות פחות אטרקטיביות בספר הפקודות, ובכך המחיר הממוצע לקנייה/למכירה ירד - זהו **Slippage**. מקטין את מרווח הארביטראז' או אף מבטלו. למעשה, מחקרים מראים שרוב הזדמנויות הארביטראז' החוזרות (כמו בארביטראז' הקלנדרי בבחירות 2024) הופיעו **ברגעי נזילות נמוכה** ולפרקי זמן קצרים <sup>10</sup>, כי בשוק דליל ייתכנו סטיות מחיר עד שסוחרים מאזנים. אבל באותם רגעים גם קשה לנצל כמות גדולה בלי להזיז את המחיר בעצמך. לכן, סוחר ארביטראז' צריך להגביל את גודל הפקודה למינימום שלא ישפיע משמעותית על המחיר, או לחילופין לקבל שימלא רק חלק מהכמות (Partial Fill) לפי העומק הזמין.
- **סיכון רגליים (Leg Risk):** זהו אחד הסיכונים המהותיים בארביטראז' לא-אוטומטי: חוסר היכולת להוציא לפועל **בו-זמנית** את שתי רגלי העסקה. אם הסוחר הצליח לבצע את העסקה בשוק אחד ולא הספיק לבצע בשני לפני שהמחיר זז, הוא עלול להישאר עם פוזיציה לא מגודרת החשופה לסיכון כיוון. למשל, אם מכר "Yes" בשוק הקצר ולא הצליח לקנות "Yes" בשוק הארוך בזמן, כעת הוא חשוף לפוזיציה Short שאינה מובטחת רווח, וייתכן שהמחיר בשוק הארוך עלה בינתיים (או שהפער נסגר ע"י אחרים) כך שהעסקה השנייה כבר אינה אפשרית ברווח. סיכון זה בולט כשאין אפשרות לבצע **ביצוע אוטומי** של הפקודות (עסקה אחת הכוללת את שתיהן יחד - לא קיים כאן כי אלו שווקים שונים). למעשה, **אי-אוטומיות הביצוע** היא מגבלה קריטית: "רגל אחת עשויה להתמלא בעוד השנייה נכשלת, מה שיוצר חשיפה כיוונית" <sup>11</sup>. נרחיב בסעיף 6 על שיטות לצמצום סיכון זה (כגון פקודות מיוחדות, ביצוע סימולטני וכיו"ב), אך לצורך חישובי כדאיות יש להתייחס לכך שארביטראז' תאורטי עם פער קטנטן עלול לא להתממש בפועל בשל סיכון הביצוע. לכן לעיתים דורשים **מרווח ביטחון** גדול מהעלות

הטרנזאקציונית המינימלית, כדי שאפילו במקרה מילוי חלקי או תזוזת מחיר קלה – עדיין יישאר רווח. למשל, ייתכן שנציב סף שרווח הארביטראז' הגולמי יהיה לפחות 5% לפני עלויות, כך שגם אם המחירים זזים מעט במהלך הביצוע, יש שוליים לספוג זאת.

לסיכום, בעת זיהוי הזדמנות ארביטראז' קלנדר, יש "להכניס למציאות" את הגורמים הללו: לחשב רווח נטו אחרי עמלות וגז, לבדוק את עומק השוק הזמין לכמות הרצויה (ואף לחשב רווח כפונקציה של כמות – ייתכן שכדאי להגביל את היקף העסקה), ולהעריך האם המרווח מצדיק את סיכון הרגליים. צעדים אלו ימנעו כניסה לעסקאות שבהן הרווח התאורטי לא ייתרגם לרווח ממשי.

## 4. זיהוי אוטומטי של שווקים תלויים – שימוש ב-NLP ו-Clustering

בפלטפורמת תחזיות עלולים להיות מאות ואלפי שווקים פעילים, שחלקם למעשה דנים באותו אירוע באופן שונה (ניסוח שונה, תנאי זמן שונים, ספי ערכים שונים וכדומה). כדי לאתר הזדמנויות ארביטראז' לוגי באופן שיטתי, יש צורך בכלי אוטומטי לזיהוי שווקים תלויים או חופפים. כאן נכנסות לתמונה טכניקות **עיבוד שפה טבעית (Natural Language Processing, NLP)** ואלגוריתמי **Clustering** (אשכולות), שניתן להפעיל על טקסט שאלות השוק.

**אתגר הזיהוי:** לשני שווקים תלויים ייתכן ולא יהיה ניסוח זהה. למשל: - שוק אחד: "Will X happen by November 30", - שוק שני: "Will X happen in November 2025?"

למרות שאלו למעשה אותה שאלה (השני ניסוח מעט שונה – "במהלך נובמבר" שקול ל-"עד 30 בנובמבר"), נדרש מנגנון שיבין שהכוונה זהה או חופפת <sup>12</sup>. דוגמה נוספת: שוק אחד שואל "If Candidate Y wins the presidency, will they also win the popular vote?" ושוק אחר: "Will the popular vote winner be the same as the electoral college?" – ניסוחים שונים לאותה תלות לוגית. כמו כן קיימים שווקים עם תלות אימפליציטית: למשל "Will X win the election?" ו-"Will X's margin of victory exceed 5%?" – כאן אם X מנצח מעל 5%, הוא בוודאות ניצח, כלומר יש תלות מובנית (אחד מניח את השני) <sup>12</sup>. שווקים כאלו מכונים **סמי-תלויים** ועלולים להציע ארביטראז' קומבינטורי (לא בהכרח קלנדר, אך לוגי דומה).

### שלבי פתרון בזיהוי אוטומטי:

- 1. נרמול טקסטואלי (Text Normalization):** יש ללמד את המערכת לזהות ניסוחים שקולים. למשל, להמיר תאריכים בפורמטים שונים לצורה סטנדרטית (להפוך "by Nov 30" ו-"in November" שניהם ל-"30/11/2025"). ניתן להשתמש בספריות NLP לזיהוי תאריכים (Entities מסוג DATE בספריית spaCy או שימוש ב-dateparser לחילוץ תאריך). כמו כן לנרמל מונחים: "by end of 2025" ו-"before 2026" – להבין ששניהם אומרי אותו דבר (עד סוף 2025).
- 2. חילוף ישויות ומאפייני אירוע:** נרצה לפרק את טקסט השוק לרכיביו: מי/מה האירוע, פעולה/שאלה (מה נמדד), ותנאים (כמו תאריך, סף מספרי, מקום). למשל בשאלה "Will Apple release a new product line in 2025?" ישות עיקרית – **Apple**, אירוע – **השקת קו מוצר חדש**, זמן – **2025**. בשוק אחר: "Will Apple release a new product line before 2027?" – אותם ישות ואירוע, זמן – **2027**. לאחר חילוף כזה, ניתן ליצור "**טביעת אצבע**" **לאירוע** (Event Fingerprint) המתארת את הגרעין הלוגי ללא רכיב הזמן. בדוגמה זו הטביעת אצבע יכולה להיות: ("Apple", "launch new product line"). שווקים תלויים קלנדרית.
- 3. השוואת טביעות אצבע וקירוב סמנטי:** במקרים פשוטים, התאמת ישויות ושדה אירוע יספיקו – כלומר אם שני שווקים חולקים את אותם מאפיינים מלבד הזמן, נוכל ישירות לשך אותם כזוג תלוי (או לצרף אותם לקבוצה). עם זאת, שווקים תלויים לא תמיד מנוסחים סביב אותו ניסוח בדיוק. ייתכן הצורך בזיהוי סמנטי מתקדם: למשל זיהוי ששווקים "Will a Democrat win the popular vote and the Presidency?" ו-"Winning candidate also wins popular vote" קשורים, אף שהנוסח שונה לגמרי. כאן אפשר להיעזר ב**מודלי שפה** להבנת ההקשר. שתי גישות אפשריות:

4. **הטענת מודל שפה גדול (LLM)** על רשימת השווקים, ובקשתו לפלטר שווקים קשורים. מחקר עדכני אף מדגים סוכן LLM הקורא טקסטים של שווקים, מציע צמדי שווקים עם תלות זהה או הפוכה, ומקבץ אותם אוטומטית <sup>13</sup>. ניתן למשל לתת ל-LLM רשימת שווקים באשכול (למשל פוליטיקה) ולשאל: "Group these into clusters of related markets, and identify pairs where outcomes are logically linked (same outcome or opposite)" – גישה שהוכחה כבעלת דיוק סביר.
5. **הטענת משפטי השאלות לוווקטורים (Embedding)** והשוואת דמיון קוסינוסי/מרחק. מודלים ייעודיים כמו *sentence transformers* (SBERT) יכולים להמיר כל כותרת שוק לוווקטור סמנטי. לאחר מכן, ניתן לבצע **אלגוריתם Clustering** (למשל K-Means, Hierarchical Clustering או DBSCAN) על הוווקטורים כדי לזהות קבוצות של שווקים שדומים בשפה. לדוגמה, שווקים שכולם מזכירים "Apple" ו-"product line" יתכנסו לאותו אשכול. יש לשים לב לשלב נרמול התאריכים לפני ה-Embedding כדי ששני משפטים הנבדלים רק בתאריך לא ייחשבו שונים מהותית.
6. **כלים נוספים:** אפשר לשלב גם חוקים ידניים לזיהוי near-duplicates: למשל התעלמות מפיסוק, אותיות רישיות, והמרת מספרים למילים וכו'. בתחרויות ארביטראז' נצפו גם מקרים של שווקים כמעט זהים שנוסחו אחרת (למשל "Will X happen by November 30?" מול "Will X happen by Dec 1?") – שניהם שואלים אותו דבר למעשה, למעט הבדלי ניסוח זניחים של תאריך. כללים פשוטים יכולים לתפוס רבים מאלה.
7. **סינון וקיבוץ:** לאחר חישוב דמיון או זיהוי ישויות, נבנה **גרף או קבוצות** של שווקים תלויים. למשל, אם זהו 3 שווקים על "שיגור משימת SpaceX עד תאריכים 1, 2, 3 שונים", כולם יקובצו תחת אותו אירוע בסיס. ייתכנו גם זוגות תלויים לא קלנדריים (כמו הדוגמאות עם margin vs win), אותם ניתן לסווג בנפרד – למשל כקשר של "אותה תוצאה" או "תוצאות הופכיות" <sup>14</sup>. חשוב גם לזהות יחסים **בלעדיים**: לעיתים פלטפורמה מגדירה שמספר שווקים הם חלק מאותו אירוע רב-תוצאות Polymarket לדוגמה משתמשת במנגנון NegRisk שמבטיח שאם שוק אחד "Yes", האחרים "No" אוטומטית <sup>15</sup>. שווקים כאלה לא נותנים ארביטראז' (כי הפלטפורמה כבר דואגת לארביטראז' אפס דרך אפשרות המרה בין פוזיציות). הפוקוס שלנו הוא על שווקים נפרדים (לרוב אירועים שונים או נוסח שונה) שיש ביניהם קשר לוגי ולא בהכרח מנגנון פנימי שקושרם.
8. **אימות היחסים :** לאחר זיהוי אוטומטי, כדאי לבצע בדיקה לוגית לתלות. NLP עלול לטעות במקרים עדינים – למשל שוק אחד "אם X יקרה עד T1" ושני "Y יקרה עד T2". ייתכן שהם לא באמת תלויים (אם X ו-Y שונים), או תלות חלקית (אירוע אחד משפיע הסתברותית על אחר). מערכת זהירה יכולה לדרוש אישור ידני או חיווי confidence גבוה לפני שמכריזה על צמד כ-Arbitrage Pair. מחקרים שילבו גם בדיקה אמפירית – למשל בדיקת היסטוריית מחירים של שווקים שהמודל טען שהם תלויים, לראות אם אכן היה מתאם תנועות או הכרעות חופפות <sup>16</sup>. אנחנו יכולים להסתפק בכלי התוכנה אם אנו מגבילים לתבניות הלוגיות הפשוטות יותר (קלנדרי, על אותה שאלה; או שווקים שלכן ולא של אותו אירוע).
- השילוב של NLP, כללי נרמול, ו-Clustering מאפשר **סורק שווקים תלויים אוטומטי**. למשל, נוכל להריץ תהליך יומי שסורק את כל שאלות השוק ב-Polymarket (באמצעות API), משתמש בכלי NLP כדי לתייג כל שוק בישות/אירוע/זמן, ואז משווה את כל השווקים כדי לאסוף קבוצות. כך נתקבל מפת שווקים תלויים: כל קבוצה מייצגת אירוע בסיס (לדוגמה "השקת מוצר Apple חדש") וכל קבוצה מכילה את כל השאלות הנוגעות לאירוע (בזמנים שונים או תנאים שונים). מתוך אלה, המערכת תחפש **הזדמנויות ארביטראז'** – למשל בקבוצה שיש בה שני שווקים קלנדריים, נבדוק אם המחירים מפרים מונוטוניות; בקבוצה עם תנאי לוגי (כמו "X קורה" ו-"Y > Z במקרה X קורה"), נבדוק אם יש קומבינציית קנייה/מכירה שמבטיחה רווח (בדומה ללוגיקה שתוארה, רק עם יותר חישובים קומבינטוריים).
- יצוין שבמחקר אחד הודגם שימוש ב-Agent AI המבצע ממש את התהליך: Clustering של שווקים לנושאים, ואז **Relationship Discovery** בתוך כל אשכול למציאת זוגות שווקים עם תוצאה זהה או הפוכה <sup>13</sup>. גישה כזו (המשלבת הבנה סמנטית עמוקה ואימות סטטיסטי על שווקים שהסתיימו) הראתה שאפשר לנבא צמדי ארביטראז' מסוימים ולהשיג ביצועי מסחר חיוביים על בסיס אותם גילויים <sup>17</sup> <sup>18</sup>. זה מחזק את ההיתכנות הפתרון: NLP + Clustering הם כלים מרכזיים בבניית Scanner חכם שיזהה עבורנו באופן אוטומטי הזדמנויות ארביטראז' קלנדרי ולוגי.

## 5. מנוע חישוב כדאיות מבוסס Order Book (לא רק Mid-Price)

לאחר שזיהינו זוג (או קבוצה) של שווקים תלויים ופוטנציאל לארביטראז', עלינו להחליט אם וכיצד לבצע את העסקה. כאן נכנס **מנוע חישוב כדאיות** שתפקידו להעריך את הרווח הפוטנציאלי הנקי מעסקת ארביטראז' נתונה, בהינתן **מצב ספר הפקודות הנוכחי** בשווקים הרלוונטיים.

מדוע לא מספיק להסתכל על מחירי ה-Mid (אמצע המרווח)? מפני שמחיר mid משקף רק ממוצע גס בין ה-Bid ל-Ask, ועלול ליצור אשליית רווח במקום שאין בפועל. למשל, ייתכן שבשוק T1 ה-mid price הוא 0.60 ובשוק T2 הוא 0.55 (נראה  $P_1 > P_2$ , הפרש 0.05). אך אם בספר הפקודות עומד למעשה **היצע** (Ask) בשוק T1 ב-0.62 וקונים רק ב-0.58 (Bid), ובשוק T2 ההיצע ב-0.55 וה-Bid ב-0.53, הרי שסוחר ארביטראז' יצטרך **לקנות במחיר ההיצע ולמכור במחיר הביקוש** בפועל. ייתכן שהתמונה האמיתית היא שאין רווח: אולי כדי למכור Yes בשוק T1 נקבל רק 0.58, ולקנות Yes בשוק T2 נשלם 0.55 – הפער האפקטיבי 0.03 בלבד. לכן, חייבים להתבסס על **מחירי ה-Bid/Ask העדכניים בעומק הרלוונטי** כדי לחשב כדאיות אמיתית <sup>19</sup>.

### צעדי החישוב במנוע:

1. **שאיבת עומק שוק נוכחי:** עבור כל שוק בזוג הארביטראז', נשלף את **הצעת הקנייה הגבוהה ביותר (Highest Bid)** ואת **הצעת המכירה הנמוכה ביותר (Lowest Ask)**, וכן את הנפחים הזמינים ברמות מחיר אלו. למשל, בשוק T1: Bid=0.58 (ל-500 יח'), Ask=0.62 (ל-300 יח'). בשוק T2: Bid=0.53, Ask=0.55, כל אחד בנפח מסוים. הנתונים הללו זמינים לרוב דרך **API של ספר פקודות** או feed websocket רציף. Polymarket למשל מפעיל **CLOB (central limit order book)** היברידי עם הזנת WebSocket שנותנת עדכונים בזמן אמת על פקודות <sup>20</sup> <sup>21</sup>.

2. **בחירת תפקיד בכל שוק:** נחליט איזו פעולה לבצע בכל שוק. בדוגמת calendar arbitrage סטנדרטית: בשוק היקר קצר-הטווח נמכור Yes (כלומר נמכור במחיר ה-Bid הקיים, או נקנה No), ובשוק הארוך נקנה Yes (נשלם את ה-Ask). לחלופין, אם הזיהוי היה ש- $P_2 > P_1$  (שוק ארוך יקר מהקצר באופן לא הגיוני), אז להפך – נמכור Yes בשוק הארוך ונקנה Yes בשוק הקצר. בכל מקרה, נגדיר פונקציית רווח שתלויה במחירי הביצוע המשוערים:  $\text{Payout} = \text{Profit} - \text{Cost}$  למשל, לתרחיש  $P_1 > P_2$ : נקבל \$1 על החוזה שנמכור (אם הוא יפוג Yes) או על ה-No שקנינו (אם יפוג No), נקבל \$1 על החוזה שקנינו אם יפוג Yes, ונשווה לסכום ששילמנו upfront. אך חישוב זה נעשה כמו מקודם אבל עם מחירי Bid/Ask במקום  $P_1, P_2$  תאורטיים.

3. **התחשבות בנפחים וסכום עסקה:** ייתכן שההזדמנות קיימת רק לכמות מוגבלת. למשל, אם בשוק T1 ה-Bid 0.58 זמין רק ל-100 יחידות, ונרצה למכור 200, אז על 100 היחידות הנוספות כנראה נצטרך לפגוש Bids נמוכים יותר (נניח הבאים בתור 0.57 וכד'). מנוע חישוב חכם יבצע **אגרנציה של עומק**: הוא יכול לסכם כמה רווח יופק בהנחת מיליון X יחידות. דרך אחת: להתייחס לכמות מינימלית הדרושה – למשל לדרוש הזדמנות על לפחות \$500 ערך (כדי להצדיק מאמץ). דרך אחרת: לחשב ROI (תשואה יחסית) כפונקציה של גודל: לעתים עדיף לממש כמות קטנה עם ROI גבוה מאשר לנסות למקסם נפח על חשבון ROI.

המנוע יכול לבצע **סימולציית מילוי**: למשל, עבור כמות 100, לחשב שימולא מלא במחירים הנוכחיים (רווח  $R_1$ ). עבור 200, לחשב שחלק ימולא במחיר הבא (רווח  $R_2$ ), וכו'. ואז לבחור את הגודל האופטימלי (למשל שמקסם רווח נטו או לא יורד מתחת ל-ROI מינימלי).

בכל מקרה, המנוע צריך להחזיר את **הכמות המומלצת** לעסקה ואת **הרווח (נטו) הצפוי**. אם הרווח הצפוי קטן מהסף – ידחה את העסקה.

4. **הכללת עמלות ועלויות:** כחלק מהחישוב, יש להפחית עמלות מסחר (אם יש), גז וכל הוצאה ישירה. למשל, אם ידוע שכל טרייד גובה 1% מהסכום המושקע כעמלה – נוריד זאת מחישוב הרווח. בפלטפורמת בלוקצ'יין, יש להעריך את עלות הגז. דוגמה: בוט יכול לשלף את מחיר הגז הנוכחי ואת מחיר מטבע הרשת (MATIC) ולחשב שעלות שליחת פקודה  $\sim 150k \text{ gas} * \text{מחיר gas}$ . בוט ארביטראז' שהוצג בספרות מחשב את עלות העסקה בדולרים (ומציין שאין עמלות Polymarket) ומוודא שהתשואה לאחר עלות עדיין מעל מינימום <sup>6</sup> <sup>8</sup>.

5. **תשובה שנתית/פרק הזמן:** שיקול נוסף – **משך הזמן עד הכרעה**. עסקאות ארביטראז' קלנדר עשויות להינעל עד תאריך היעד האחרון (T2). אם T2 רחוק, המשמעות היא שההון מושקע תקופה ארוכה. ייתכן שהמערכת תרצה לדרוש ROI גבוה יותר עבור עסקאות ארוכות-טווח, בהשוואה לארביטראז' שננעל תוך ימים ספורים. זאת מפני שיכולים להיות עלות הזדמנות או סיכון מערכת לאורך זמן. למשל, ארביטראז' של 2% רווח הננעל רק בעוד שנה אולי אינו כדאי, בעוד 2% רווח ביום הוא מצוין. לכן, מנוע הכדאיות יכול לחשב גם **תשובה שנתית משוערת** (annualized ROI) ולסנן עסקאות מתחת לרף מסוים. בפועל אצל סוחרים מקצועיים, מסתכלים על "% לרבעון" או לימים שנותרו לארביטראז'. ראינו התייחסות של משתמשים לכך שחלק מההזדמנויות מניבות "מעל 20% לשנה" אם מחשבים היחס מול הזמן <sup>22</sup>.

דוגמה: אם נותרו 3 חודשים לפקיעה והארביטראז' הנקי הוא 1%, אולי זה לא מאוד מרשים (כ-4% שנתי). אם נותרו 3 ימים וניתן להרוויח 0.5%, זה עצום (פי כמה עשרות אחוזים שנתי). לכן, המנוע יכול לשקול גם מדד כזה, לפי העדפותינו.

**קביעת** סף כדאיות: על בסיס כל הגורמים לעיל, ההנהלה או המשתמש המפתח יקבעו מדיניות: למשל "לא להיכנס לארביטראז' עם רווח צפוי פחות מ-\$50 או ROI נטו  $> 3\%$ ". מדיניות זו יכולה להיות מקודדת. ואכן, באלגוריתם לדוגמה מגדירים `min_profit` או `min_roi` כפרמטר <sup>23</sup> <sup>24</sup>.

**מודל לחישוב כדאיות:** נסכם מודל פשוט: - קלט: שווקים A ו-B, כיוון העסקה (למשל "Short Yes" או "Long Yes"), כמות מקסימלית לניתוח (למשל עד 1000 יח'), ספי מינימום (ROI, % רווח \$) - תהליך: - שלוף Bid/Ask ועמוקים. - עבור כמות q מ-1 עד q\_max: חשב מחיר ממוצע להתמלאות q בכל רגל. נניח בשוק A (שמוכרים בו) נמלא q חלקו ב-Bid הטוב ואם נגמר נרד למחיר הבא; בשוק B (שקונים בו) נמלא q חלקו ב-Ask הנמוך וכן הלאה. אפשר לקבל סדרת מחירים  $p_A(q), p_B(q)$ . - חשב עלות  $\$ = [p_B(q) + (1 - p_A(q))] * \$q$  (לדוגמה אם קנינו q חוזי Yes בשוק B במחיר  $p_B$ , ומכרנו q חוזי Yes בשוק A במחיר  $p_A$ , העלות: שלמנו  $p_B q$  וקיבלנו  $p_A q$  מהמכירה, אך כדי למכור אולי יצרנו פוזיציה  $\$1$  each, או פשוט נאמר ששמרנו q ( $p_A - 1$ ) כהתחייבות). אפשר גם לחשב ישירות את  $payoff$  בכל תרחיש, אך כאומדן ניקח את  $worst-case\ payoff = q$  (כפי שראינו שזה מיני' קיבלנו), ואז הרווח הגרוע  $\$ = \$1 - cost * \$q$ . בעצם  $\$r(q) = cost(q) * (1 - q)$ . דרישה לארביטראז' היא  $\$r(q) \geq 0$  (שהיה  $0 \leq$  בכל תרחיש), אנחנו רוצים יותר מזה:  $\$r(q) > 0$  משמעותית. הפחת עמלות קבועות + גז (לפי q). - חשב  $roi = r(q) / cost(q)$ , ומשך  $(T2\ now) = 365 /$ . בדוק תנאי  $min\_profit, min\_roi$ , אולי אנואליזציה. - בחר q אופטימלי או קבע שההזדמנות לא כדאית אם לכל q הפרמטרים לא עומדים בסף.

פלט המנוע יהיה למשל: "Arbitrage detected between Market X and Market Y: buy Yes on Y at 0.55 and sell Yes on X at 0.58 (available 500). Suggested size: 300. Expected profit: \$12 (2.4% (available 300) ROI), annualized ~30%. Risk: low. Fees accounted

למעשה, בתשתית של בוט ארביטראז' אמיתי, המנוע הזה הוא הלוגיקה שמקבלת עדכוני מחירים ומחליטה **Trigger** (הפעלת ההוצאה לפועל) כאשר יש match לקריטריונים. בסעיף הבא ניגע בביצוע - שהוא השלב שאחרי זיהוי וכימות הכדאיות.

## 6. Execution Engine – מנוע ביצוע יעיל ומפחית סיכון

כאשר זוהתה עסקת ארביטראז' כדאית, יש לפעול מהר על מנת "לנעול" את הרווח לפני שהשוק ישתנה. מנוע הביצוע אחראי על שליחת פקודות לשווקים, תזמון ואסטרטגיית הביצוע, וטיפול במצבי כשל חלקי. הדרישות העיקריות: **מהירות, סינכרון, ואמינות.**

**ביצוע מקבילי vs סדרתי:** אידיאלית, נרצה לבצע את שתי רגלי הארביטראז' בו-זמנית ככל האפשר, כדי לצמצם סיכון חשיפה. בפועל, אי אפשר ממש לשלוח שתי עסקאות באותו פייקט רשת בדיק, אבל אפשר להתקרב לכך על-ידי שליחתן במקביל אסינכרוני. ב-Python למשל, ניתן בעזרת `asyncio.gather` לשלוח בקשות קנייה ומכירה סימולטנית <sup>25</sup>. כך, אם שתי הפקודות מוצאות מיד צד נגדי, שתיהן ימולאו כמעט בו-זמן. זה עדיף בהרבה על גישה סדרתית ("בצע A, המתן לאישור, בצע B"), שבה השוק השני עלול להשתנות בזמן הפעור.

**Immediate or Cancel / Fill or Kill:** כדי להימנע ממצב שממלאים אותנו חלקית בלבד, רצוי להשתמש בסוגי פקודות המגבילות זאת. - **IOC (Immediate-Or-Cancel)** – פקודה שאומרת: מלא מייד את מה שניתן עד הכמות שהוגדרה, וכל מה שלא מולא – בטל מייד. כך אם יש נזילות חלקית, נמנע לפחות מהמתנה. אבל עדיין ייתכן מילוי חלקי. - **FOK (Fill Or-Kill)** – אפילו טוב יותר: או שהפקודה תתמלא במלואה מייד, או שתבוטל כולה. כך מובטח שלא ניתקע עם חלקיק פוזיציה. לא כל זירת מסחר תומכת בפקודה כזו. אם Polymarket (ייתכן ולא ישירות דרך ה-UI אך אולי דרך פרוטוקול), זו אופציה טובה עבור פקודות גדולות כשחשוב לנו "הכל או כלום". - **Limit vs Market:** סוג פקודה **Limit** בעסקאות ארביטראז' נעדיף לרוב לשלוח פקודות **Limit** עם המחיר שזיהינו כעושה ארביטראז'. למשל, אם ראינו שיש **Ask** למכירה ב-0.55 על 100 יח', נשלח פקודת קניה **Limit** 0.55 בדיוק, וכך ניקח את אותה פקודה. אפשר גם לשלוח **Market** כדי לקחת כל מה שיש, אבל **Market** עלול למלא גם מעבר למה שרצינו ובמחירים גרועים יותר אם הנפח השתנה בן רגע. **Limit** מבטיח לא לקנות מעבר למחיר מסוים. עם זאת, שימוש ב-**Limit** דורש שתהיה הפקודה הנגדית – לשם כך קיבלנו את עומק השוק. - **משך תוקף (Time-In-Force):** כבר נגענו ב-**IOC/FOK**. **GTC (Good-Till-Cancelled)** – זה ברירת מחדל – השאר ספר עד שמבוטל, לא רצוי אצלנו כי לא נרצה להשאיר פקודה מארביטראז' תלויה.

לכן, **האסטרטגיה** היא: שלח שתי פקודות **Limit** (אחת מכירה בשוק A במחיר **Bid** הנוכחי, אחת קניה בשוק B במחיר **Ask** הנוכחי) עם **Time-In-Force = IOC**. כך כל פקודה תמלא מיידית מה שניתן ותבוטל את היתרה. אם שתיהן מתמלאות 100%, מעולה – הארביטראז' בוצע במלואו.

**ניהול Partial Fill:** אם קרה תרחיש שמפקדים עליו: אחת הפקודות מולאה חלקית או לא מולאה כלל, והשנייה מולאה (חלקית או מלאה). כעת אנו עם פוזיציה עודפת בצד אחד. מנוע הביצוע צריך **לטפל מיידית** במצב זה כדי לא להשאיר סיכון: - אם הייתה מילוי חלקי – למשל הצלחנו למכור 100 יחידות ב-A אבל קנינו 0 ב-B (כי אולי ה-**Ask** נעלם לפני שהפקודה הגיעה), כעת אנו **Short 100** יחידות ב-A. המנוע יכול לנסות שוב לקנות 100 ב-B במחיר מעט גבוה יותר (אולי יש **Ask** חדש קצת יותר יקר אך עדיין מתחת למחיר שמכרנו). אם המחיר עלה מדי (כבר אין רווח), אולי עדיף לקנות בחזרה את ה-**Yes** ב-A כדי לסגור את השורט (להודות בהפסד קטן) – זה בעצם ביטול העסקה אחרי מעשה. - אפשרות אוטומטית: ברגע שמתברר שרק צד אחד מולא, נשלח פקודת ביטול מיידית לצד השני (אם עוד לא בוטלה אוטומטית) <sup>26</sup>, ואז **נסגור את הפוזיציה באופן מידי** בשוק הראשון. "לסגור" אומר: אם נמכרו לנו X חוזי **Yes**, נקנה אותם חזרה בשוק A (אפילו במחיר מעט גרוע יותר) כדי לצאת נייטרלי, או אם קנינו יותר מדי בשוק B, נמכור אותם שם. זה כמובן עלול לגרום להפסד קטן (כי ארביטראז' נכשל והמחיר אולי זז), אבל זה חלק מניהול סיכונים – לקבל הפסד מזערי במקום להישאר חשוף.

מימוש לדוגמה בגישת קוד: המנוע שולח שני פקודות במקביל עם **timeout** של נניח 5 שניות. אם בתוך 5 שניות שתיהן לא אישרו מילוי, או אחת לא – הוא מבצע **cancel** למה שנשאר פתוח ומיד פועל לסגור מה שהתמלא <sup>26</sup> <sup>27</sup>. פרק זמן קצר (מספר שניות) חשוב כדי לא ליפול במצב של המתנה ארוכה מדי – נתונים מראים ש-75% מהפקודות בשווקים כאלו מתמלאות תוך כשעה (950 בלוקים בפוליגון) <sup>28</sup>, אבל אנו לא רוצים לחכות יותר מכמה שניות כשעושים ארביטראז' מהיר. פקודות שבסדר עדיפויות ראשונות בדרך כלל יתמלאו מייד, ואם לא – כנראה שמישהו הקדים אותנו או שהשוק זז, ואז עדיף לסגת מייד.

**יעילות ומהירות:** יש לוודא שלמנוע הביצוע יש גישה ישירה ומהירה ל-API המסחר: ב-Polymarket, קיימת ספריית **ClobClient Python** המאפשרת שליחת פקודות חתומות ישירות לשרשרת דרך שרת התאמה <sup>20</sup> <sup>29</sup>. יש להשתמש בחיבורים מהירים (למשל **WebSocket**) כדי לקבל אישור מייד על ביצוע או ביטול. בנוסף, ייתכן שכדאי לפצל את הביצוע למספר חלקים אם הכמות גדולה: במקום לשלוח 1000 יח' בפקודה אחת, אולי לשלוח 2 פקודות של 500 במקביל (אם אחת לא מתמלאת, לפחות סגרנו חצי וכו'). אך זה כבר אופטימיזציה מתקדמת.

**Retry Logic:** מה אם העסקה נכשלה בגלל עניין טכני – נניח הבוט שלח את הפקודות ונותק מהאינטרנט, או ה-**WebSocket** נפל, ולא קיבל אישור? מנוע הביצוע צריך להיות מחוסן: - עקוב אחרי מצב הפקודות: אם לא התקבלה תגובה, בצע שאילתה יזומה לסטטוס (פוליגון דרך **REST**) ולפי זה החלט. - אם בוצע **cancel** אך עדיין יש מילוי חלקי, וודא שסגרת. - במידה והארביטראז' עדיין קיים (נניח אחרי ביטול, המחירים חזרו לפער), אפשר לנסות שוב. אבל להיזהר לא להכנס ללופ אינסופי אם השוק זז מצד לצד.

**Parallel vs Serial:** ברוב המקרים, **Parallel** (סימולטני) הוא העדיפות, שכן כל מילי שנייה חשובה. עם זאת, יש מקרים גבוליים שאולי עדיף סדרתי: למשל, אם פער הארביטראז' גדול מאוד וברור, אפשר קודם לבצע שוק אחד ואז השני אם יש סיבות טכניות (כמו שוק אחד זמין יותר). אבל לרוב – לא.



**סוגי פקודה מתקדמים:** אם פלטפורמה תומכת, ניתן אף לשלוח פקודה מותנית: "מכור ב-A רק אם נקנה ב-B" - אך נדיר שיש API כזה, אלא אם משלבים חוזה חכם ייעודי לאטומיות (לא זמין כאן). לכן אנו נצמדים לניהול התוכנית.

לסיכום, מנוע הביצוע: - שולח פקודות Limit/IOC לשני השווקים במקביל <sup>25</sup> . - ממתיין חלון זמן קצר (שניות) <sup>27</sup> . - בודק סטטוס: אם שתי הפקודות מולאו מלא - הצלחה. - אם לא: מבטל את מה שבעוד פתוח <sup>26</sup> . - מחשב את המילוי החלקי בכל צד <sup>30</sup> , ומבצע **handle\_partial\_fill** - סגירה של הפוזיציה הפתוחה (למשל שליחה לשוק הנגדי כדי להתכסות). - מדווח חזרה הצלחה/כשל. אם כשל - יכול לסמן שלא מומש (אולי ינסה שוב כשהיה שינוי אם עדיין כדאי, תלוי לוגיקה).

גישה זו ממזערת למינימום את הסיכון שניתקע עם פוזיציה לא-מגודרת. היא טרייד-אוף: לפעמים פקודה הייתה מתמלאת אם חיכינו 30 שניות, אבל אנחנו ביטלנו אחרי 5 שניות. אולם כארביטראז'רים, אנחנו לא מהמרים - מעדיפים לא לקבל מילוי מאשר לקבל חצי מילוי.

**הערה:** Polymarket היא כאמור על רשת Polygon, אז הביצוע הסופי של סגירת העסקה הוא on-chain. זה אומר שאפילו אם שניהם מולאו, היישוב של הטרייד מתרחש בתוך בלוקים (לא מייד ברמת finality). אבל זה כבר שקוף - אנחנו סומכים על כך שאם קיבלנו אישור מילוי, אז הפוזיציה קיימת (הטוקנים אצלנו).

עוד טריק: אם הפלטפורמה הייתה מאפשרת **Atomic Swap** בין שווקים (למשל חוזה שתופר שני שווקים), זה היה פותר את הסיכון. Polymarket מאפשר *atomic swap* בין טוקני תוצאה לקולטרל עבור שוק יחיד <sup>21</sup> , אך לא בין שווקים שונים.

לסיום סעיף זה: מנוע ביצוע מהיר ומוגן הוא קריטי. בעזרת שימוש בפקודות IOC/FOK, זמן תגובה של חלקיקי שנייה, *Retry/Cancel* אוטומטי, נוכל להתמודד עם אופי השווקים הללו - **תנודתיים ומהירים**, בפרט סביב אירועים (לדוגמה: תוצאות סקרים או נאומים פוליטיים, שגורמים לתנודות ומצבים של אי-יעילות זמניים <sup>2</sup> ). ללא מנוע מבצע יעיל, הזדמנויות שנמשכות שניות עלולות להתפספס או להפוך לסיכון.

## 7. ניטור בזמן אמת ואסטרטגיות יציאה

**מנגנוני ניטור (Monitoring):** כדי שהמערכת "תדע" מתי להגיב, עליה להיות מחוברת בזמן אמת לשווקים. שימוש ב-**WebSocket** הוא הדרך המועדפת: Polymarket למשל מספקת ערוץ WS של עדכונים לכל שינוי בספר הפקודות ובמחרים <sup>31</sup> . המערכת שלנו תרשם לעדכונים על כל השווקים הרלוונטיים (או על כלל השווקים אם נרצה). כאשר מגיע שינוי (למשל פקודת קנייה חדשה שמרימה את ה-Bid בשוק אחד, או ביטול פקודה שמוריד את ה-Ask בשוק אחר), המערכת תחשב מיד אם נוצרה הזדמנות ארביטראז'. **זמן תגובה מהיר** קריטי: הארביטראז'ים הבטוחים לרוב נסתמים תוך שניות על-ידי אחרים, לכן על הבוט לזהות ולפעול מיידית <sup>32</sup> .

מבחינת מימוש, נוכל להחזיק תהליך **Market Monitor** הרץ בלולאה אסינכרונית, מאזין דרך ה-WebSocket, ומעדכן מבני נתונים פנימיים (למשל Map של Market->OrderBook state). בכל עדכון Bid/Ask חדש, הוא יכול לסמן את אותם שווקים כ"צריך בדיקת ארביטראז' מחדש". המערכת תבדוק את אותם שווקים מול בני הזוג שלהם (שבעזרת סעיף 4 יודעים מיהם).

בנוסף, ניטור כולל גם **מצב הפוזיציות שלנו**: אם נכנסנו לעסקת ארביטראז' (נניח מחזיקים כעת X חוזים פה ושם) - נרצה לעקוב אחרי שינויים שעלולים להצדיק תגובה (זה מתקשר לאסטרטגיית יציאה, בהמשך).

עוד אלמנט - **אירועים חיצוניים**: לפעמים יש הכרזות ידועות (כמו פרסום דו"ח, תוצאה) שיכולות לסגור ארביטראז' בן רגע. טוב שהמערכת תהיה מוכנה מבחינת נזילות (ואולי תימנע מכניסה ממש רגע לפני אירוע אם חוששים מתנודה חדה). ניתן לשלב ניטור חדשות או ל"ז" אירועים, אך זה מעבר לטווח הנוכחי.

**אסטרטגיות יציאה (Exit Strategy):** לאחר שביצענו עסקת ארביטראז', לכאורה הדרך הבטוחה היא **להחזיק את הפוזיציות עד הסיום (Settlement)**. במקרה של ארביטראז' קלנדרי, המשמעות להחזיק את שני החוזים עד שתוקפם יפוג והם ייושבו אוטומטית. מכיוון שבנינו את הפוזיציה כניטרלית סיכון, נוכל פשוט להמתין: - אם האירוע לא קורה עד

T1, נקבל תשלום על חוזה ה- No שלנו ב-T1, והמשיך להחזיק את חוזה ה- Yes של T2 עד T2 (אולי נקבל תשלום ב-T2 אם קרה בין לבין, או לא). - לא משנה מה, בסוף T2 נקבל את כל מה שמגיע לנו, וסכום זה יהיה  $\geq$  עלותנו (כמוכח).

**יתרון החזקה עד הסוף:** מובטח הרווח כפי שחישבנו, ללא תלות בשום דבר (מלבד שהפלטפורמה אכן משלמת). אין צורך בשום פעולה נוספת, ואין חשיפה למחיר שוק רגעי.

**חיסרון החזקה:** הכסף (USDC וכד') "נעול" בפוזיציות עד תום האירוע, שיכול להיות רחוק. למשל, ארביטראז' על בחירות 2024 שהתגלתה בנוב' 2023 – תאלץ להמתין עד נוב' 2024 לקבל את הרווח המלא. בזמן הזה אפשר היה להשתמש בהון לעסקאות נוספות. בנוסף, החזקות פוזיציות זמן רב דורשת אמון שאין תקלות (נדיר אך אפשרי – בהמשך נדון בסיכונים כמו שינוי תנאי שוק).

**יציאה מוקדמת כאשר הפער נסגר:** האלטרנטיבה היא **לצאת מהפוזיציה לפני הפירעון**, ברגע שכבר אין יתרון להחזקה. איך ומתי? - ברגע שהשוק מפנים את היחס הלוגי והפער במחירים נסגר (או התהפך). אם לדוגמה קנינו Yes ארוך ומכרנו Yes קצר, והנה לאחר זמן מה המחירים השתנו כך ש- $P1 \leq P2$  כעת – ייתכן שאין עוד סיבה מיוחדת להחזיק. למעשה, אם  $P1 \leq P2$  ירד מאוד ו- $P2 \leq P1$  עלה, ייתכן שאפילו תוכלו לסגור את שתי הפוזיציות ברווח גדול יותר מהמובטח. - לדוגמה, נגיד קנינו Yes\_T2 ב-0.50 ומכרנו Yes\_T1 ב-0.60. אם אחרי שבוע השוק התיישר ל- $P1=0.50$ ,  $P2=0.55$  (עכשיו סביר), נוכל לקנות חזרה Yes\_T1 ב-0.50 (לסגור השורט ב-Yes, או למכור את ה-Yes ששיצרנו שם) ולמכור את Yes\_T2 ב-0.55. בכך נסגור את שתי הפוזיציות במזומן: קיבלנו 0.60 בהתחלה ממכירת Yes\_T1, הוצאנו 0.50 כעת לקנות אותו חזרה – רווח 0.10 שם; קנינו T2 ב-0.50, מכרנו כעת ב-0.55 – רווח 0.05 שם; סה"כ 0.15 רווח, לעומת 0.10 שהיה מובטח אם היינו מחכים לתרחיש הגרוע. אמנם בתרחיש האמצע (האירוע קורה בין T1 ל-T2) היינו מרוויחים יותר (0.10+0.90 אפשרי), אבל זה לא בטוח שיקרה. - **ניהול סיכונים:** יציאה מוקדמת מסירה אפילו את הסיכון המערכתי של תקלות (כי כבר אין פוזיציה).

מתי הגיוני לצאת? - **אם הפער נסגר לגמרי:** כלומר עכשיו הפוזיציות שאתה מחזיק נסחרות במחיר שמשקף בדיוק את ארביטראז'. אז הפוזיציה שלך כנראה שווה בדיוק כמה ששילמת, או מעט יותר, אז אתה יכול לצאת לפחות ללא הפסד. אולי תעדיף להישאר רק אם יש עדיין רווח עתידי. - **אם הפער התהפך או over-corrected:** יכול להיות אפילו מקרה שבו  $P1 < P2$  כעת משמעותית (כלומר היה overshoot). אז למעשה אתה יכול לסגור ברווח גדול מהמצופה. זה כמובן האידיאל לצאת. בשוק יעיל, אם אתה היה הגורם שתיקן את המחיר, כנראה הפער פשוט נסגר הדוק ולא מעבר. אבל לעתים תנודתיות יכולה לגרום להיפוך קל. - **לפני אירוע מידע גדול:** ייתכן ותחליט לצאת מוקדם אם מתקרב אירוע גדול שעלול לגרום לתנודה חריגה במחירים (למרות שבכל תרחיש אתה מוגן, אבל לפעמים ספקות עלולים לגרום לדברים משונים – כמו עצירת מסחר, שינוי תנאים, וכו'). למשל, חלק מהסוחרים היו יוצאים מפוזיציות cross-platform לפני תוצאת אירוע כדי לא להיות תלויים בהכרעות אורקל שונות (ראו סעיף 8 על מקרה כזה).

**הגדרת לוגיקת יציאה במערכת:** - ניתן לקבוע שהמערכת תנסה **לסגור פוזיציות פתוחות** אם המחירים בשוק השתנו כך שרווח הסגירה הנוכחי  $\geq$  רווח מובטח מסוים. בפועל, קל: אחרי פתיחת ארביטראז', להמשיך לנטר את אותם שווקים. אם מתקיים  $P1_{\text{new}} \leq P2_{\text{new}}$  (שיווי משקל), אולי פשוט סגור. - לחילופין, אפשר לקבוע "נצא אם נוכל לנעול לפחות X% מהרווח המקסימלי כרגע". - או "נצא אם נותר Y זמן לפקיעה והפער נסגר, כדי לפנות הון".

יש גם אפשרות של **יציאה חלקית**: למשל אם היו שלושה שווקים, אולי נסגור חלק אם אחד נותר. אבל ברוב המקרים זה זוג.

לבסוף, יש אסטרטגיה של **"הפוך כיוון"**: נניח לאחר תקופה, השוק התהפך כל כך שעכשיו יש הזדמנות ארביטראז' בכיוון ההפוך. אפשר תאורטית לבצע ארביטראז' הפוך (Long הפוזיציות ההפוכות) שיבטל את הקודמות ויוצר פוזיציה חדשה. אך זה בגדול אותו דבר כמו לצאת ואז להיכנס שוב.

**דוגמה פרקטית:** בבחירות 2024 צוין שקיימים צמדי שווקים תלויים שהארביטראז' בהם הופיע פעמים רבות (6,630 פעמים בזוג מסוים!) אך ברווחיות נמוכה <sup>10</sup>. המשמעות היא שהפער היה נסגר מהר או קטן. ייתכן שסוחרים לקחו פוזיציה ויצאו כעבור דקות כשנסגר. המספר הגבוה מעיד ש**כניסה ויציאה מהירה** (תוך שעה או פחות) התרחשה שוב ושוב <sup>10</sup>. זה מתאים לגישה אקטיבית: לנצל את הפער, וכשהמחירים מתיישרים (אפילו לפני ההכרעה הסופית חודשים אחר כך) – לממש רווח ולחפש את ההזדמנות הבאה.

בשורה התחתונה, **אין חובה להחזיק עד הסוף** אם אפשר להרוויח מוקדם. החזקה עד הסוף נועדה להבטיח את הרווח המינימלי. אם השוק נותן הזדמנות לצאת ברווח – לא נהסס, כך נשחרר את ההון לעסקאות נוספות. ההחלטה תלויה גם במידת הביטחון שלנו בסגירת פוזיציה. אם השוק עמוק וקל למכור/לקנות חזרה, נוטה לסגור. אם חששנו מזילות נמוכה (קשה לצאת) אולי מלכתחילה לא היינו נכנסים.

המנוע בפועל יכול להיות מתוכנת: - לעדכן כל הזמן את ה-"unrealized P/L" של כל פוזיצית ארביטראז' פתוחה לפי מחירי השוק, - אם התנאי מתקיים ( $ROI > X$  או  $P/L \geq \text{threshold}$ ), לשלוח פקודות סגירה (sell מה שקנינו, buy back מה שמכרנו). - אחרת, אם מתקרבים לפקיעה, אולי לחכות לפקיעה.

כמובן, תמיד יכולה להיות אפשרות של **כשל שוק** או ארוע חיצוני (למשל השוק לפתע מוקפא או נסגר מוקדם), ולכן לא תמיד נרצה להישאר עד הרגע האחרון – אך נושא זה נכלל בדיון ניהול סיכונים הבא.

## 8. ניהול סיכונים ותרחישי קצה

למרות שארביטראז' מתיימר להיות "ללא סיכון", במציאות מערכות מסחר ושווקים יכולים להציב סיכונים בלתי צפויים. נפרט כמה מקרים ודרכי התמודדות:

- **שווקים עם תנאים שונים או ניסוח מבלבל:** לא תמיד שני שווקים "דומים" הם למעשה תלויים מלא. ייתכן שוני סמוי. למשל: שוק "Will candidate X win the presidency by T1" – A? ושוק "Will candidate X be President on T2" – B בשני המקרים X צריך להיות נשיא, אך ייתכן הבדל: הראשון שואל על ניצחון בבחירות עד תאריך (אולי יום הבחירות), והשני שואל אם יחזיק בנשיאות בפועל עד תאריך (כולל אפשרות מוות/פרישה בינתיים). זה דק – ומי שלא קורא את כללי השוק עלול להניח תלות מוחלטת כשאינה. **ניהול סיכון:** המערכת צריכה להשוות לא רק את טקסט הכותרת אלא גם את **החוקים (Rules)** אם זמינים ב-API. לעתים, שם נמצא פרט מבדיל. אפשר למשל לדרוש ששני השווקים יאזכרו אותו "Resolver" או מקור הכרעה, לבדוק אם יש סעיפי הגדרות שונים. במידה ויש ספק, **עדיף לארביטראז' האנושי לבדוק** או פשוט להימנע. בשלב הראשון, אולי נמקד לארביטראז'ים קלנדריים ברורים (אותו שאלה בדיוק, תאריך שונה). אם בכל זאת נמצאה הזדמנות על שווקים שנראים תלויים אך עם שונות קטנה, ניתן לסמן אותה להחלטה ידנית או לבדיקה מוגברת. דוגמה ממחקר: נמצאו **13 זוגות שווקים תלויים** בבחירות 2024, כולל יחסים מורכבים (כמו פער בין פופולר ווט לאלקטורל קולג' וכן "איזו מפלגה מנצחת באיזו רמה")<sup>33</sup> <sup>34</sup>. שם ננקט משנה זהירות באימות תלות. חלק מהשווקים הוגדרו "weak dependency" – תלות חלשה (למשל LLM זיהה תלות אך בפועל לא תמיד נכון)<sup>35</sup>. המערכת צריכה לדעת לזהות ולסנן כאלה. עוד דוגמה: **שווקים מותנים** – נניח שוק אחד הוא "Will X happen if Y wins election" ושני "Will X happen"? – פה התלות היא הסתברותית ולא דטרמיניסטית כי תלוי אם Y באמת זכה. אלו מצבים מורכבים – כנראה לא ניגע כי אין ארביטראז' בטוח, רק שוני הערכת הסתברות.

- **תקלות API/WebSocket:** תשתית התוכנה שלנו עלולה להיתקל בבעיות: ניתוק מהרשת, קריסת חיבור WebSocket, או תקיעת התהליך. אם בזמן כזה הופיעה הזדמנות, נפספס אותה. גרוע מזה, אם זה קורה באמצע **ביצוע עסקה** (נניח שלחנו פקודות ולא קיבלנו אישור), המערכת עלולה לא לדעת שיש לה פוזיציה פתוחה. **ניהול סיכון:**

- שימוש בחיבורים עם *Auto-reconnect* ו-*Heartbeat*. למשל, ספריית `websockets` מאפשרת לגלות ניתוק ולנסות להתחבר מחדש מיד. בינתיים, אפשר לעבור ל**Polling** (שאלת ה-API כל שנייה) עד שה-WS חוזר.
- ריבוי ערוצים: אולי חיבור WebSocket נוסף כגיבוי.
- לוגים וניטור חיצוני: להריץ את הבוט עם השגחה (למשל מערכת שידועת להתריע אם הוא לא שולח הודעות זמן מה).
- בעת כשל, המערכת צריכה **לא לבצע פעולות מסחר חדשות** אלא אולי רק לדאוג לסגור פוזיציות במידת הצורך. הגיוני להקפיא מסחר אוטומטי אם מזהים שזרימת הנתונים לא אמינה.
- לגבי ביצוע: לאחר שליחה, המנוע יכול לפתוח *Timer*: אם לא שמעתי כלום 5 שניות, נניח כשל ולבטל. כמו כן, יש לשמור את ה-Order IDs שנשלחו ולוודא בסופו של דבר (דרך קריאת API מצב הזמנות) שלא נשארו פתוחות.
- בקוד לדוגמה רואים שימוש ב-`asyncio` `timeout` של 5 שניות<sup>36</sup> <sup>27</sup> והגעה ל-`except` במצב `TimeoutError` שמבטל הכל<sup>37</sup>.

- **שינויים פתאומיים בשוק:** זה יכול להיות
- **תנועה חדה במחיר:** למשל, מישהו עם מידע חדש נכנס וקנה הרבה חוזים, המחירים זזים משמעותית. אם קרה בדיוק כשאנו מבצעים, כבר דיברנו איך לבטל ולסגת. אבל גם לאחר שאנו מוחזקים: לדוגמה, קנינו Yes ארוך ומכרנו Yes קצר, ואז פתאום הגיע מידע שהופך את האירוע לכמעט ודאי שלא יקרה – שני השווקים ירדו במחיר חזק. לכאורה אין לנו מה לחשוש כי החזקתנו מבטחת עד הסוף, אך אולי השוק הופסק זמנית עקב עדכון? (Polymarket לפעמים עוצרת מסחר אם אירוע מוכרע מוקדם ואז מחכה לאורקל).
- **השהיית מסחר/Low liquidity:** אם מסיבה כלשהי אחד השווקים מוקפא (אולי admin intervention, או תקופת מחלוקת אורקל), אנחנו לא יכולים לסגור פוזיציות או לגדר עוד.
- **שינוי בשאלה/תנאים:** ייתכן נדיר – אם הפלטפורמה מגלה שנוסח השוק היה שגוי ומתקנת/מעדכנת אותו.
- למשל, "תאריך T2" הוגדר כ-31/12/2025 ובטעות תקנו ל-01/01/2026. שינוי כזה יכול לשנות מעט את ההיקף.
- **ביטול שוק (Invalid Market):** חלק מהפלטפורמות, כולל Polymarket, שומרות לעצמן (דרך מנגנון אורקל) את הזכות לסגור שוק כאילו "לא היה" (למשל אם השאלה לא הכריעה ברור). במצב כזה, שני השווקים אולי לא יתיישבו כפי שחשבנו. לדוגמה, היה שוק "Will X happen by Dec 2025?" – ננעל כלא רלוונטי כי האירוע בוטל, אז הכריזו על Invalid (החזירו 50-50 לכל מחזיק). אם השוק האחר עד 2026 נשאר רגיל – האסטרטגיה שלנו נהרסת: במקום \$1 או \$0, נקבל החזר \$0.5 על אחד. **ניהול סיכון:**
- לעבור היטב על חוקי השווקים ולוודא שאין סעיף "If unclear or event cancelled, market resolves N/A".
- "invalid". אם יש סיכון כזה, אולי לא להיכנס או להיות מודע.
- לפזר סיכונים: לא לשים את כל ההון בארביטראז' אחד, במיוחד אם יש חשש משפטי/טכני.
- להשתמש ב-**Position Tracker** חכם: לתעד כל פוזיציה שיש לנו (כמות ונכס בכל פלטפורמה), ולעקוב אם יש ארוע שעלול להשפיע. למשל, בוט הארביטראז' cross-platform שמר מבנה של כל פוזיציותיו ובדק אם אחת תלויה באורקל שונה <sup>38</sup> <sup>39</sup>. הוא אף זיהה אם יש פוזיציות חוצות פלטפורמות שעלולות להיות מועד למניפולציית אורקל <sup>40</sup> <sup>41</sup>.
- במקרה של Polymarket, לארועי settlement יש תהליך (UMA אורקל) שיכול להיות במחלוקת. היו מקרים, כמו במרץ 2025, שמשתמש רב-השפעה התערב בהצבעה וגרם לפתרון שגוי בשוק מסוים <sup>42</sup>. סוחר שגידר אותו מול Kalshi (ששם נפסק נכון) הפסיד. דרך להקטין סיכון כזה: להימנע מארביטראז' cross-platform אלא אם הפער עצום (כגיבוי לסיכון) <sup>43</sup>. בתוך אותה פלטפורמה, סיכון כזה קטן יותר (כי שני השווקים היו מוכרעים יחד לרוב אם משהו קורה).
- **הימנעות שווקים "מועדים לפורענות":** אם זיהינו שוק עם ניסוח בעייתי, או נושא שנוי במחלוקת (העלול לגרום לויכוח על התוצאה) – אולי עדיף שלא להיות שם.
- **מעקב אקטיבי:** אם אנו מוחזקים זמן רב, רצוי שמישהו (או תהליך) יבדוק באופן תקופתי שהשוק עדיין חי, לא הוכרז כ-invalid וכו'.
- **תוכניות גיבוי במערכת:**
- אם רכיב אחד קורס (נגיד ה-NLP Clustering התבלבל ונתן קשר שגוי) – רצוי הגנה מפני ביצוע עסקה חסרת-תלות באמת. אפשר למשל לפני כל ביצוע להשוות מחירי שוק: במצב הגיוני, אם  $P1 > P2$ , הארביטראז' יהיה רווחי. אם קיבלנו סימן ארביטראז' אך למעשה  $P1 < P2$  (אולי המידע הישיר היה שגוי) – לא לבצע.
- אם ה-Executor מחזיר כשל – המערכת תסמן הזדמנות זו כנכשלת ולא תנסה שוב מיד (אולי תחכה לתזוזת מחיר משמעותית לפני ניסיון).
- **Rate limiting:** כדי להימנע ממצב שהבוט מציף את ה-API בפעולות (מה שעלול לגרום חסימה או עמלות מיותרות), נגדיר תקרות: למשל לא יותר מ-X עסקאות בשנייה. נתון: Kalshi מגביל 20 קריאות/שנייה (קריאה) ו-10 כתיבות/שנייה בבסיס <sup>44</sup>. Polymarket אולי פחות מגביל אך יש עלות גז – בכל מקרה נשלב בקרת קצב.
- **סיכון מערכת/רגולטורי:** אולי פחות טכני – אם שוקי התחזיות משתנים (לדוגמה רגולטור מגביל שווקים מסוימים, פתאום Kalshi נאלץ לסגור שוק...), זה אתגר. אין פיתרון ברור בקוד, אלא מודעות ותגובה (ניח יציאה יזומה לפני תאריך סגירה צפוי).

לסיכום חלק זה: ניהול הסיכונים משלים את האלגוריתם כך שלא נסמוך עיוור על "רווח ללא סיכון". הוא כולל: - בדיקות עקביות ולוגיות טרם כניסה (מניעת false arbitrage), - מעקב קפדני אחרי ביצוע (חירום במידת חלקיות), - פיזור ואיזוני כניסה (לא הכל בהזדמנות אחת), - ניטור מתמשך של פוזיציות פתוחות ושינויים בשווקים (כולל מנגנונים יחודיים כמו invalid), - ותוכניות יציאה מהירות בעת תקלה.

בגישה שמרנית, גם לאחר הזנת האלגוריתם, נרצה לפעמים פיקוח אנושי, לפחות בשלבים הראשונים, כדי לוודא שהמערכת לא לוקחת סיכון חריג עקב מצב שלא שקלה.

## 9. ארכיטקטורה מודולרית מוצעת (Scanner, Normalizer, Pricing, Detector, Executor, Risk Manager וכו') וזרימת מידע

כעת נאגד את הכל למבנה ארכיטקטוני כולל, מודולרי, שבו כל רכיב אחראי על חלק מהמערכת. ארכיטקטורה מודולרית מקלה על פיתוח, בדיקה ותחזוקה – ניתן לשפר רכיב מבלי להשפיע על אחרים, והפרדת האחריות ברורה. להלן המודולים העיקריים וזרימת הנתונים ביניהם:

- **Scanner (סורק שווקים):** מודול שאוסף מידע גולמי מהפלטפורמות. הוא מתממשק ל-API הציבורי של שוקי התחזיות כדי לקבל את רשימת כל השווקים הפעילים, את הפרטים שלהם (כותרת, תיאור, זמן סיום, נזילות, מחירים נוכחיים וכו'). הוא מרענן באופן תקופתי (נניח כל שעה) את רשימת השווקים החדשים או את המידע הסטטי (לא משתנה בתדירות גבוהה). ה-Scanner מספק את הפלט הנתונים ל-**Normalizer**.
- **Normalizer (מנרמל ויוצר Fingerprint):** מקבל את רשימת השווקים ופרטיהם מהסורק. עבור כל שוק, מבצע עיבוד שפה: חילוץ ישויות (שמות, תאריכים, מקומות), נרמול תאריכים לפורמט אחיד, זיהוי מילים שקולות (synonyms). לאחר מכן, יוצר ייצוג מובנה לכל שוק – למשל אובייקט עם שדות: `event_subject`, `event_action`, `event_threshold` (אם יש כמו ערך מספרי), ו-`event_time`. גם ייתכן שדה `fingerprint_id` או כך, המהווה מזהה קבוצת אירוע (למשל צירוף subject+action+context ללא זמן). את התוצרים הוא שומר במאגר או מעביר ישירות ל-**Detector**.
- **Clustering & Pairing (מקבץ ומאתר זוגות):** זה חלק מה-Detector או שלב נפרד. הרכיב לוקח את כל השווקים המנומרים ומבצע קיבוץ. אפשר לממש בשני שלבים:
  - **Clustering לאשכולות אירוע:** על בסיס ה-fingerprint (או embedding), מחלק לקבוצות כך שבכל קבוצה שווקים שכנראה נוגעים לאותו אירוע.
  - **Relation Detector:** בתוך כל קבוצה, בודק אילו שווקים מקיימים קשר לוגי. למשל, אם יש 3 תאריכים שונים, ידע שכל צמד עוקב (T1 vs T2, T2 vs T3 וכו') מהווים מועמדים לארביטראז' קלנדרי. גם עשוי לבדוק סוגי תלות אחרים (אם היו, אך הפוקוס שלנו קלנדרי). פלט של רכיב זה: רשימת **Pairs** (זוגות שוק) או **Groups** עם ציון סוג הקשר (כגון "Market A implies Market B" או "same event different times").
- **Pricing Engine (מנוע תמחור/כדאיות):** עבור כל זוג/קבוצה שמצאנו, מנוע זה מושך **בזמן אמת** את נתוני ספר הפקודות (Bids, Asks) העדכניים של אותם שווקים (כאן נכנס שימוש ב-WebSocket/עדכוני שוק). הוא מחשב האם יש פער מחירים מספיק (כמו בסעיף 5): לדוגמה מחשב על בסיס ה-Best Bid/Ask את הרווח הפוטנציאלי ו-%תשואה, בהתחשב בנפחים זמינים, בעמלות, בעלויות גז וכו'. מנוע זה יכול להיות כל הזמן running, או מופעל באירוע:
- אפשרות 1: להריץ אותו באופן מתמשך לכל זוג שסומן, כך שכל שינוי קטן בעומק השוק הוא מחשב מחדש פוטנציאל.
- אפשרות 2: להפעיל אותו **On-Demand** כאשר מגיע אירוע WebSocket. למשל, המוניתור קיבל שינוי ב-market X, הוא יודע ש-X שייך לזוג (X,Y), אז מפעיל את Pricing calc על X-Y. מנוע התמחור מספק כתוצאה **אינדיקציה**:

האם יש ארביטראז' זמין מעל הסף, ואם כן, מה הפרטים (איזה כיוון – למכור באיזה ולקנות באיזה, כמה כמות, רווח צפוי). מידע זה הולך ל-Detector/Decision Maker.

- **Arbitrage Detector / Decision Maker**: רכיב שמקבל את תוצאות Pricing. אם זוהתה הזדמנות העומדת בקריטריונים, הוא מחליט "לפעול". ייתכן שהוא גם יבדוק שלא פעלנו כבר על הזדמנות זו לפני רגע (מניעת כפל), או שאין לנו כבר פוזיציה פתוחה זהה (אולי לא נעשה שוב אותה עסקה אם כבר מומשה). ברגע שהחליט – הוא מעביר **Order Instructions** למודול ה-Executor.

- **Execution Engine (מבצע פקודות)**: כפי שפורט בסעיף 6, זה הרכיב שמנהל את ביצוע העסקה בפועל מול זירות המסחר. הוא מקבל הוראה למשל: "Arbitrage Trade: Sell 300 YES on Market X @0.58, Buy 300 YES on Market Y @0.55". הוא מתזמן את שליחת הפקודות הנדרשות (דרך SDK או API מתאימים), מטפל ב-Partial/Timeout וכו'. ה-Executor פועל באופן אסינכרוני (תוך כדי ששאר המערכת ממשיכה לנטר, אבל אולי ננהג בזירות ונמנע מלבצע 5 ארביטראז'ים בו-זמנית – נשלב תור ביצוע, או מספר אשכולות thread אם מערכת גדולה). בסיום, הוא מודיע על תוצאה:

- הצלחה: העסקה בוצעה במלואה, הפוזיציה כעת פתוחה (הכוונה: אנחנו מחזיקים כמות X בחוזה הארוך ו-X ב-"No" בחוזה הקצר, למשל).

- כשל: בוטל, לא מומש (אז אין שינוי במצבנו) או מומש חלקית עם סגירה (אולי הפסד קטן). דיווח זה הולך ל-Risk Manager/Position Tracker.

- **Position Tracker & Risk Manager (מעקב פוזיציות ומנהל סיכונים)**: מודול שרושם כל פוזיציה שנכנסה אליה. הוא ישמור טבלא של פוזיציות פתוחות: עבור כל ארביטראז' שביצענו – מה מצבנו (כמה חוזים Long/Short בכל שוק, מה עלות הכניסה, מה הרווח הצפוי בפירעון).

- ה-Tracker משתלב עם המוניטור: הוא מנטר את מחיר השווקים של כל פוזיציה.
- ה-Risk Manager מגדיר חוקים מתי להתערב: למשל אם מזהה שינוי קיצוני במחיר באחד הצדדים, אולי שוקל hedge נוסף (לא שכית, כי כבר מגודר). יותר רלוונטי – אסטרטגיית יציאה: הוא יבדוק אם ניתן לצאת כעת ברווח סביר. אם כן, הוא יאותר ל-Execution Engine לבצע **Exit Trade** (הפוך מהכניסה: לקנות בחזרה/למכור).
- בנוסף, risk manager יפקח על חריגות: נניח שוק פתאום מסומן "Suspended" – הוא יטריגר התראה או טיפול מיוחד (אולי לסגור בצד השני אם אפשר).
- ה-Risk Manager יכול גם לרכז סטטיסטיקות: למשל לחשב כמה הון בסיכון (מבחינת locked collateral) יש לנו. אם הרבה, אולי נפחית פעילות.
- אם הפלטפורמה היא cross-platform (Polymarket+Kalshi), הוא ודאי יעשה הצלבות לוודא שאין לנו חשיפה לאורקל שונה <sup>40</sup>. במקרה שלנו (נניח התחלנו רק Polymarket), פחות.

- חלק חשוב: **Alerting/Logging** – מנהל הסיכונים יכול להתחבר למערכת התראות (Slack/Email) כדי לדווח לאדמינס על אירועים חשובים: "שוק XYZ סומן כ-Invalid, פוזיציה נסגרת ידנית!" או "WebSocket reconnected after 10s downtime" וכד'.

- **Database / Storage**: בכל הארכיטקטורה, נשתמש בבסיס נתונים על מנת לשמור מידע:

- רשימת שווקים (סטטי או מתעדכן) – כדי לא להריץ NLP כל פעם מאפס, אפשר לשמור תוצאות Normalizer ולרענן רק שינויים.
- תוצאות clustering – אולי נשמור mapping של market\_id לקבוצה/זוג כדי להשתמש בזמן אמת.
- עסקאות שביצענו – לתיעוד P&L ולהימנע מכפילויות. גם למעקב מיסוי/דיווח כמובן.
- אפשר שה-Position Tracker יהיה חלק מה-DB: טבלת positions.
- לוגים לכל פעולה – debugging וניתוח (למשל כמה ארביטראז'ים תפסנו).
- DB יכול להיות פשוט SQLite או Postgres, תלוי כמה נתונים. יתרון Postgres: אפשר לשלב גם חיפוש טקסט (לשימוש NLP בסיסי) וקל לשימוש בריבוי תהליכים.

**User Interface / Monitoring Dashboard (לא חובה אבל טוב):** אפשר לבנות דשבורד המציג בזמן אמת את מצב המערכת – אילו זוגות שווקים זוהו, איפה יש פערים כרגע, כמה רווח פוטנציאלי, מה ביצענו וכו'. זה לא חיוני לליבה אבל מאוד מועיל לניטור אנושי. ניתן להשתמש בגרף גרפי או אפילו CLI שמדפיס סטטוס.

**זרימת הנתונים (Data Flow):** 1. **אתחול:** Scanner שולף את נתוני כל השווקים → Normalizer מעבד ומפיק Fingerprints → Clusterer יוצר קבוצות אירוע ותתי-יחסים. תוצאות אלו נשמרות (ב-DB או בזיכרון) לשימוש במהלך המסחר. 2. **ניטור שוטף:** מודול WebSocket מנוי לעדכוני Order Book עבור כל השווקים שזיהינו כקשורים (למעשה, אפשר להתחבר לכולם – Polymarket WS מאפשר סאבסקריפצ'ן לפי רשימת 3. markets). **אירוע שינוי:** כאשר מגיע אירוע (למשל "Market X: Best Ask dropped to 0.52 with 200 liquidity"), ה-Monitor מעדכן את הנתון במבנה הפנימי, ואז: - מזהה שוק X שייך לזוג (X,Y) ואולי לקבוצה. - קורא למנוע התמחר: "חשב כדאיות עבור זוג Pricing X-Y". - Engine לוקח את Bid/Ask הנוכחי של X ו-Y (מהמבנה המעודכן, או מבקש גם Y אם לא עודכן זה עתה) → מחשב רווח אפשרי. - Detector מקבל: אם "profit\_expected >= threshold", מעביר ל-Executor. אחרת לא עושה כלום. 4. **ביצוע:** Executor מקבל הוראת "Trade on X-Y", שולח מיד את הפקודות לשווקים דרך REST API או SDK או WebSocket (אם תומך). - הוא מחכה תגובה (ייתכן כמה תגובות: filled/partial for each). - מעדכן על הצלחה/כשל. - במקביל, ה-Order Book Monitor גם יקבל את אותן פקודות כעידכונים (למשל השפיעו על ה-Bid/Ask). אם הוא חכם, אולי מתעלם מעדכונים שנובעים מהפעולה שלנו (אפשר על בסיס user id או זיהוי עצמי). 5. **אחרי ביצוע:** אם הצליח → Position Tracker נעדכן: הוספנו פוזיציה ב-X,Y. Risk Manager יתחיל לעקוב. - אם נכשל → אולי לא קורה כלום (הזדמנות כנראה פספסנו), אולי יסמן את הזוג כ"לא לנסות שוב מיד". 6. **מעקב פוזיציה:** בעוד הפוזיציה פתוחה, כל אירוע מחיר על X או Y ייגרום ל-Risk Manager לחשב unrealized P/L. - אם תנאי יציאה מתקיים (למשל פער נסגר) → שולח ל-Executor פקודות הפוכות (sell מה שקנינו, buy מה שמכרנו). - Executor מבצע יציאה דומה (גם כאן רלוונטי שימוש ב-Position Tracker - IOC) מסמן פוזיציה כסגורה, רושם רווח סופי. 7. **סיום טבעי:** אם הגענו לפקיעת T2 בלי שיצאנו, אז: - Polymarket תסגור את השווקים. אנו כנראה נקבל את התשלומים דרך החזקת הטוקנים (יצטרך לממש redeem, אבל Polymarket אולי עושה אוטומטית). - Position Tracker יעדכן רווח שהתקבל. - אפשר לרשום בהיסטוריה הצלחה. 8. **שגרה ותיחזוק:** - ה-Scanner יכול לרוץ אולי פעם ביום כדי לזהות שווקים חדשים (ואז Normalizer+Clusterer ירוצו שוב, להוסיף אותם). - אפשרי גם לשלב מודול לומד: למשל, רישום איזה סוגי ארביטראז' הכי משתלמים כדי לכוון התאמות בעתיד (לא הכרחי). - ה-Alert System: הבוט יכול לשלוח הודעות על הזדמנויות גדולות במיוחד או שגיאות. לדוגמה, אם ROI > 10% או profit > \$100, סמן כ-"High urgency" 45 46 ולשלוח נטיפיקציה. ניתן לשלב זאת.

ניתן לסכם את המודולים העיקריים בטבלה:

מודול	תפקיד מרכזי	טכנולוגיות/הערות
<b>Market Scanner</b>	איסוף רשימת שווקים ונתונים (מטא-דאטה, תגיות, זמן)	REST API / SDK של הפלטפורמה, DB
<b>&amp; Normalizer NLP</b>	נרמול טקסט, חילוף אירוע/ישיויות, יצירת מזהה אירוע (Fingerprint)	ספריית NLP (spaCy), custom code
<b>&amp; Cluster Relations</b>	קיבוץ שווקים לאירועים, זיהוי זוגות תלויים לוגית	מודל embedding (SBERT), אלגוריתם אשכול (KMeans) או LLM agent
<b>Order Book Monitor</b>	ניטור בזמן אמת של Bid/Ask לכל שוק רלוונטי	WebSocket, מבנה נתוני OrderBook
<b>Pricing Engine</b>	חישוב רווח ארביטראז' פוטנציאלי על בסיס עומק נוכחי	לוגיקת חישוב (Python), גישה לנתוני OrderBook, חישוב עמלות/גז
<b>Opportunity Detector</b>	החלטה על כניסה לעסקה אם הרווח מעל סף	חוקים מוגדרים, בדיקת כפילויות
<b>Execution Engine</b>	הוצאה לפועל של פקודות קנייה/מכירה סימולטניות, ניהול מילוי	SDK למסחר (למשל ClobClient), asyncio למקביליות, ניהול שגיאות

מודול	תפקיד מרכזי	טכנולוגיות/הערות
Position Tracker	שמירת פוזיציות פתוחות, חישוב P/L שוטף, מעקב סטטוס	שמירה ב-DB, חישובי P/L (Python)
Risk Manager	מדיניות יציאה (Exit), טיפול תקלות (ביטולי שווקים, ניתוקים), התראות	חוקים (Rule engine), אינטגרציה להתראות (Slack/Email)
Database	אחסון נתוני שווקים, פוזיציות, לוגים	PostgreSQL / SQLite
Alert/Logging	רישום אירועים ויידוע מפעילים	ספריית לוגים (loguru וכו'), API לשליחת הודעות

מבנה זה מודולרי, מאפשר למשל להחליף את מודול ה-NLP במודל חזק יותר בעתיד (נניח לעבור לשימוש ב-LLM), או להוסיף תמיכה בפלטפורמה נוספת (פשוט להרחיב את ה-Scanner ושאר השרשרת קצת, אבל העיקרון נשמר).

חשוב על **scalability** : עם ארכיטקטורה זו, ניתן להריץ רכיבים שונים במקביל (כפיסות microservices). למשל, Monitor, Detector, Executor יכולים לפעול כשרשרת נפרד או אפילו תהליך. אפשר להשתמש ב-message queue (כמו RabbitMQ) לתקשורת ביניהם (למשל Monitor דוחף "opportunity" ל-queue, ה-Executor מקשיב). כך אפשר להתאים עומס: אם יש המון עדכונים, לשים backpressure.

בהמשך, בחרנו בטכנולוגיות (סעיף 10) שתומכות ארכיטקטורה אסינכרונית רבת-משתתפים זו היטב.

## 10. טכנולוגיות מומלצות למימוש (Python async, ספריות NLP, בסיסי נתונים, תורים, ניטור)

בבחירת טכנולוגיות, שקלנו אמינות, ביצועים וקלות שילוב. המטרה היא לבנות מערכת יציבה, בזמן-אמת, שניתן לפתח ולתחזק יחסית בקלות. להלן הבחירות העיקריות:

- **שפת תכנות: Python (אסינכרוני)** – פייתון נבחר כיוון שיש לו אקו-סיסטם עשיר גם לפיננסים (web3, API, clients וגם ל-NLP (spaCy, transformers), והפיתוח בו מהיר. ספציפית נשתמש ביכולות **asyncio** של Python כדי לנהל מקביליות (כגון האזנה במקביל ל-WS ושליחת פקודות בזמן-אמת) בצורה יעילה. פייתון אמנם איטי בשטף בודד, אך צווארי הבקבוק פה הם I/O (תקשורת רשת) ולא חישובים כבדים, ולכן **asyncio** הוא בחירה טובה. אכן, קיים כבר ClobClient עבור Polymarket בפיתון <sup>20</sup>, וספריות WebSocket נפוצות (**websockets** או **aiohttp**). בנוסף, השימוש בפייתון מקל על שילוב מודולי NLP ומאפשר ניסויים/שינויים מהירים – חשוב בעולם שמשתנה.
- **גישה לפלטפורמה (API/SDK):**
- **Polymarket**: כפי שמודגם בספרות, Polymarket מפעילה **CLOB היברידי** עם API. נשתמש בספריית **py\_clob\_client** (כמו בקוד <sup>20</sup>) כדי להתחבר לשרתי Polymarket. ספרייה זו תטפל בחתימות עסקאות על Polygon, ניהול Keys וכו'. היא מספקת גם ממשק נוח ל-WS (כמו בקוד <sup>31</sup>).
  - **Kalshi** או אחרות: אם נרצה להרחיב לפלטפורמות אחרות, אפשר להשתמש ב-REST שלהן (Kalshi יש REST, מוגבל קצב <sup>44</sup>). ייתכן שנקדיש תהליך נפרד לכך, אך ניתן.
- **Web3 (Blockchain)**: ה-ClobClient משתמש כנראה ב-web3.py כדי לתקשר עם Polygon <sup>47</sup>. נשתמש בו עבור פעולות כמו `fetch gas price`.
- **ספריות NLP:**



- נשתמש ב-**spaCy** (עם מודל אנגלי מתקדם) לחילוץ ישויות (PERSON, ORG, GPE, DATE, CARDINAL וכו'). spaCy נוחה ומדויקת יחסית, וניתן להוסיף לה כללים. למשל, אפשר לאמן קצת זיהוי סוגי שאלות ספציפיות (עם או בלי, אך כנראה מספיק שימוש במנוע הקיים).
- עבור נרמול תאריכים נשתמש ב-**dateparser** שיכול לקחת מחרוזת ("by end of 2025") ולהחזירה כתאריך סטנדרטי (31/12/2025).
- אם נבצע embedding לטקסט, נוכל להשתמש בספריית **sentence-transformers** (המספקת מודלים pre-trained כמו all-MiniLM-L6-v2, שהוא מהיר ונותן וקטורים 384-מימד) לחישוב וקטור לכל שאלה. זו ספרייה פייטונית קלה לשימוש.
- ייתכן שניעזר גם ב-**NLTK** או **regex** פשוטים עבור כמה דפוסים (למשל חיפוש "{4}" כדי לתפוס שנה).
- במידה ונלך לכיוון LLM agent, אפשר לשלב למשל API של OpenAI GPT-4 למטלות clustering/relational. אבל זה כנראה overkill ולא הכרחי בהתחשב שאפשר פתרון זול יותר. מעבר לכך, LLMs כבדים יכולים להוסיף תלות חיצונית ופחות אמינות.

## • בסיס נתונים (Database):

- נבחר ב-**PostgreSQL** כמסד נתונים רלציוני עיקרי. הוא אמין, ידוע ומטפל היטב בטרנזקציות. ניצור טבלאות עבור Markets, Relationships, Positions, Trades, Logs וכו'.
- אם נרצה מהירות בפעולות קריאה רבות (כמו בדיקות תדירות על מחירים), אפשר להשתמש ב-Redis כ-cache, אבל כנראה לא נחוץ כי המידע החי בא מה-WS בכל מקרה.
- שימוש ב-Postgres מקל גם שילוב למערכות ניהול (ניתן לצרף Grafana או tools אחרים).
- **async DB access**: נשתמש ב-async driver כמו **asyncpg** כדי לא לחסום loop כאשר כותבים ללוג/DB.

## • מנגנון תורים (Message Queue):

- להפרדת מודולים, ייתכן שימוש ב-**RabbitMQ** או **Redis Streams**. למשל, Monitor מכניס אובייקט "Opportunity" לתור, וה-Executor צורך אותו. זה מוסיף איתנות – אם ה-Executor איטי, התור נעצר; אם קורס, ההודעה נשמרת אולי.
- אפשר גם בפשטות, בתוך Python asyncio, להשתמש ב-**asyncio.Queue** עבור תקשורת בין קורוטונים. אם אנו בפלטרומה אחת ובוט אחד, internal queue עשוי להספיק.
- אמינות: עבור מערכת production mission-critical, RabbitMQ (עם persistent delivery) מבטיח שלא נפספס הזדמנויות גם אם התהליך restart. זה שיקול כבד יותר.
- לאור מורכבות, אפשר להתחיל ללא, ואם רואים צורך, להוסיף RabbitMQ.

## • ניטור ותצוגה:

- **Logging**: נשתמש בספריית logging של Python (או משודרגת כמו **loguru**) כדי לרשום קבצים עם כל אירוע חשוב.
- **Metrics**: ניתן לשלב חשיפה של metrics (Prometheus) – למשל קצב עסקאות, רווח יומי, latency וכו'.
- **Alerting**: לחיבור מהיר, אפשר להגדיר את Risk Manager שישלח התראה במייל או Slack כאשר אירוע מסוים קורה (נניח Loss חריג, או הזדמנות ROI>20%). הספרות מראה דוגמה של AlertSystem שנותן תיעדוף ודירוג דחיפות <sup>45</sup>.
- **Dashboard**: אפשר ליצור ממשק web בסיסי (אולי Flask + some charts) כדי לצפות במצב. לא קריטי אבל מועיל. כטכנולוגיה אפשר גם Streamlit (מאפשר בניית GUI מהירה בפיתוח) – אם רוצים לראות רשימת הזדמנויות בזמן אמת.

## • Scaling & Performance

- Python מסוגל לטפל ב-100+ שווקים במקביל (כל עוד רוב הזמן מבילים בהמתנה ל-WS).

- אם נרחיב לפלטפורמות רבות או לאלפי שווקים, ייתכן צורך בריבוי תהליכים. אפשר להשתמש ב- **multiprocessing** או לשבץ מודולים שונים כמיקרו-שירותים (למשל Service אחד עושה NLP clustering אחת לתקופה, Service אחר מקבל updates ומחשב).
- גם אפשר להיעזר ב-GPU ל-NLP אם עושים embedding לכמויות, אך זה כנראה overkill – מספר השווקים לרוב לא עצום (Polymarket ~ מאות עד אלפים) וניתן לבצע embedding חד פעמי לכל שוק ולהתעדכן עבור חדשים.
- **אמינות (Reliability):**
  - נריץ את הבוט בסביבת שרת יציבה (למשל Linux VM).
  - להשתמש בכלי ניהול תהליכים (systemd / docker container with restart policy) שיבטיחו חזרה לעבודה אם נופל.
  - יחידות בדיקה (Unit tests) אפשר לכתוב עבור לוגיקת החישוב (לבדוק שפונקציית profit מחשבת נכון וכד').
  - *Polygon/Blockchain concerns*: לוודא שימוש ב-API מהיר (ייתכן להחזיק Node עצמי או להשתמש ב-endpoint מהיר כדי שהפקודות ישודרו מהר).
  - ה-ClobClient ככל הנראה פותר הרבה, אבל אולי נשתמש גם ב-web3 לעניין הגז וכו' <sup>6</sup>.
- **דוגמאות טכנולוגיות אלטרנטיביות:**
  - אפשר היה לשקול Node.js/TypeScript – בוודאי אפשרי עם ספריות WS ו-web3, אבל אז NLP נהיה פחות זמין (יש, אך Python מוביל בזה).
  - C++/Rust – מהיר אך פיתוח איטי, לא צריך לרובד זה.
  - בסיס, Python async עונה על הדרישות באופן מאוזן. ואכן, פרויקטים בפועל דיווחו על שימוש בו – למשל Polymarket Agents toolkit הוא בפייתון <sup>48</sup>, וגם בוטים (Navnoor Bawa) המוזכר הריץ כנראה בפייתון, לפי הקוד שהציג).
- **ספריות/טכנולוגיות נוספות:**
  - **websockets (python)** – לטיפול בקשר WebSocket.
  - **pandas** – אולי לניתוח נתונים/לוגים אם נאסוף, אך לא נדרש בזמן אמת.
  - **dataclasses** – להגדרת מבני נתונים (למשל מחלקה ל-ArbitrageOpportunity עם שדות) כדי שיהיה קל לנהלם.
  - **asyncio tasks** – לניהול משימות מתוזמנות (למשל scanner כל שעה).
  - **Sentry** – כלי שירותי למעקב אחר שגיאות runtime.
- הטכנולוגיות האלו יוצרות יחד מערכת אינטגרטיבית: - (asyncio) Reactor Async pattern לניטור וביצוע בו-זמנית. - NLP ו-Data Science libs לניתוח סמנטי. - בסיס נתונים ו-Queue לאמינות וארגון. - Integration עם API של Polymarket (שכבר בנויה). - Logging/Monitoring לשקיפות.
- כל הרכיבים הללו ניתנים לחיבור בפייתון, ויש ניסיון מוצלח בקהילה: Polymarket Agents למשל מספק "scaffolding for trading agents" עם שימוש ב-Async, web3, וכו' <sup>48</sup>. אנחנו בעצם בונים סוכן כזה ממוקד אסטרטגיית Calendar Arbitrage.

## 11. ניתוחים ומקרי בוחן 2024-2025 (בחירות, גיאופוליטיקה, מוצרי טכנולוגיה, פלטפורמות נוספות)

נסקור כעת מספר מקרים אמיתיים מן השנים האחרונות המדגימים את השימוש בארביטראז' קלנדרי ולוגי, ואת הביצועים של מערכות דוגמת זו שתוארה. חלק מהנתונים מגיעים ממחקרים שסקרו פעילות ארביטראז'רים ב-Polymarket בתקופה זו:

### א. בחירות 2024 בארה"ב - ארביטראז' בין שווקים תלויי-תוצאה וטווחי זמן:

הבחירות לנשיאות ארה"ב 2024 יצרו שפע שווקים על אירועים חופפים: הן שווקים ישירים ("מי ינצח את הנשיאות?") והן שווקים נלווים ("מי ינצח בפופולר ווט?", "מה יהיה הרכב הסנאט?"), וכן שווקים עם טווחי זמן שונים (למשל חלקם נסגרו בנובמבר 2024, חלקם בינואר 2025 לאישור הסופי). מחקר מצא **13 זוגות שווקים תלויים** הקשורים לבחירות 2024 <sup>33</sup>. אחד המקרים הבולטים (שכונה "Pair 4") היה: - שוק A: "המנצח בפופולר ווט יהיה גם הנשיא?" (במילים אחרות, האם לא יהיה פער בין הצבעת העם לקולות האלקטורים). - שוק B: "מי ינצח בפופולר ווט?" (מועמד דמוקרטי או רפובליקני).

קשר לוגי: אם שוק A טוען שלא יהיה פער (Popular Vote winner = President), ניתן לגזור הסתברות משוק B (מי הפופולר ווט) ומי הנשיאות. למעשה, זוג שווקים אלו אפשרו לארביטראז'ר להרכיב פוזיציות שקולות להימור על "יהיה פער". נמצא שבמהלך הקמפיין היו **אלפי רגעים** בהם השווקים האלו לא היו מתואמים, וניתן היה להרוויח על ידי קנייה/מכירת שילוב מתאים <sup>49</sup>. Pair 4 הנ"ל הציג לא פחות מ-6,630 הזדמנויות ארביטראז' במרוצת הזמן - מספר מדהים <sup>49</sup>. אמנם רובן היו קטנות (פערי מחיר של סנטים בודדים) בשל פעילות גבוהה. אך סוחר אוטומטי זריז היה יכול לנצל רבות מהן. המחקר ציין שהרווח המקסימלי הממוצע בארביטראז' בין-שווקים אלה היה סביב \$100 בלבד <sup>50</sup> (מוגבל בנזילות של כ-\$2,000 באותם רגעים), כלומר כדי להרוויח משמעותית צריך היה לבצע שוב ושוב (מה שאכן נעשה: סוחר מוביל ביצע 4,049 עסקאות ארביטראז' בתקופה והרוויח מעל \$2 מיליון! <sup>2</sup>).

דוגמה מספרית: נניח לפני הבחירות, שוק A (אין פער) היה ב-0.8 (80% סיכוי שלא יהיה פער), ושוק B העריך 60% סיכוי שהדמוקרט ינצח בפופולר ווט. שילוב הגיוני היה צריך לקיים יחסים מסוימים (מורכב מעט לחישוב ידני, אבל אם הדמוקרט מוביל בפופולר והפופולר=נשיאות סביר, אז הדמוקרט גם לנשיאות  $0.8 \times 0.6 = 0.48$ ). אם מצאו אי התאמה - קנו/מכרו.

### ב. שווקים קלנדריים על השבעתו של הנשיא, 2024-2025:

Polymarket אירחה שווקים כגון "האם ג'ו ביידן יהיה נשיא ב-31/12/2024?" לצד "האם ג'ו ביידן יהיה נשיא ב-01/06/2025?" (מועדים שרלוונטיים לבריאותו, אולי). קשר לוגי: אם הוא מכהן בסוף 2024, סביר שיהיה גם באמצע 2025, אלא אם קורה משהו ברבעון הראשון. אם השוק הקצר (2024) נסחר גבוה מהארוך (2025), היה ניתן לעשות Calendar Arbitrage. ואכן, ייתכן רגעים בהם חשש קצר-טווח (למשל שמועה על פרישה) העלה את ההסתברות שלא ישרוד עד 2025 יותר מהסתברות שלא ישרוד עד סוף 2024, באופן לא הגיוני. ארביטראז' כזה היה כרוך במכירת חוזים קצרים וקניית ארוכים ולהיפך לפי הצורך. סוחרים שעקבו אחר מצב בריאותם של מנהיגים השתמשו בזה. (אין לנו מספרים מדויקים פומביים, אך זה תרחיש סביר שקרה).

### ג. אירועים גיאופוליטיים - דוגמת המלחמה באוקראינה:

בתחילת 2022 היו שווקים: "האם רוסיה תפלוש לאוקראינה לפני 1 במרץ 2022?" ואולי "האם רוסיה תפלוש לפני 1 באפריל 2022?" (תאריך מאוחר יותר). כשטווחי הזמן חופפים (מרץ ואפריל), ברור שאם הסיכוי עד מרץ היה בטעות גבוה מזה עד אפריל, יש ארביטראז'. בתחילת פברואר 2022, למשל, השוק קצר-הטווח אולי היה בתנודתיות גבוהה (חדשות יומיות), וייתכן נסחר בפרמיה מול הארוך. סוחרים מקצועיים ניצלו עיוותים כאלה - מוכרים את הקצר וקונים את הארוך. אם הפלישה לא קרתה עד מרץ, הקצר פג ב-sch (הם זכו שם), אך הארוך עוד אפשרות לפלישה עד אפריל (אולי בסוף קרה בפברואר אז שניהם כן - אך אז הפסידו בקצר, זכו בארוך, כנראה עדיין רווחיים אם יחס היה נכון). מחקר לא ציין ספציפית את המקרה הזה, אך כן הוזכר שמאורעות תנודתיים (כמו מלחמה) גרמו **לאי-יעילות** זמנית במחירים <sup>2</sup>. סוחרים הפיקו רווחים משמעותיים בסמוך לאירועים כמו פלישות או הסכמי שלום, כי השוק הכללי הגיב לאט או בצורה לא אחידה.

#### ד. השקת מוצרי טכנולוגיה (Apple example):

Polymarket הריצה שוק "האם Apple תשיק קו מוצר חדש עד סוף 2025?" (שהסתיים ב-No, >1%)<sup>51</sup> <sup>52</sup>, וגם שוק ארוך טווח יותר: "האם Apple תשיק קו מוצר חדש לפני 2027?" – נכון לעכשיו נסחר בסביבות 83%<sup>53</sup>. בתקופת 2025, אם השוק קצר הטווח (עד 2025) היה מחירו גבוה מהשוק הארוך (עד 2027), זו הייתה הזדמנות ברורה. בפועל, סביר שהשוק הארוך תמיד היה מעל (כי עד 2027 כמעט ודאי Apple תעשה משהו חדש). ואכן כרגע: לפני 2027 עומד ~83%. לו נניח לפני שנה השוק 2025 היה 10% וה-2027 70%, לארביטראז' יש "כסף על הרצפה": מוכר 2025 @10% Yes (או קונה 2025 @90% No), וקונה 2027 @70% Yes. עלות:  $1.60\$ = 0.90 + 0.70$ . על שתי יחידות? זה יותר מ-1, רגע – כאן יש יותר מורכבות כי צריך כנראה לשלב עם עוד משהו (אולי עוד שוק ב-2026). אבל אם היה 10% מול 70%, כנראה השוק תיקן מהר. סצנה סבירה יותר: באמצע 2024, שמועות על מוצר AR/VR (שבסוף הוכרז 2023). אולי "עד 2025" נסחר 40%, "עד 2027" 60%. אם משהו חשב ש-אם לא עד 2025 אז כן עד 2027 (די הגיוני), אולי פרופיל מחירים היה אחרת. בכל מקרה, ארביטראז' יכול היה לעקוב. דוגמה נוספת: שוק על "Will SpaceX reach Mars orbit by 2024?" לעומת "by 2025". פערים כאלו לפעמים מתרחשים כי קהל הסוחרים בכל שוק שונה (אופטימיסטים קצרי טווח מול ריאליסטים ארוכי טווח).

#### ה. ארביטראז' בין פלטפורמות (Kalshi vs Polymarket):

לא בדיוק Calendar Arbitrage, אבל קשור ל-"Logical Arbitrage". לעתים Polymarket וקאלישי הציעו את אותו שוק בדיוק (למשל תוצאה של בחירות, או מדד כלכלי). למשל בספט' 2025, הבחירות בארה"ב גרמו למחירי שווקים של Polymarket vs Kalshi להיות שונים. מחקרים ציינו שפיצול נזילות בין פלטפורמות הביא להפרישי מחירים ניכרים<sup>54</sup>. זה נוצל ע"י בוטים של "Cross-market arbitrage". לדוגמה, שוק על "Republicans win House in 2024" היה 55% ב-Kalshi ו-60% ב-Polymarket – סוחר היה קונה ב-55% בקאלישי, מוכר ב-60% בפולימרקט, ונועל פער (ברגע האמת שתייה משלמות \$1 אם נכון, אז רווח 5 סנט מינוס עמלות). ארביטראז' כזה דורש כמובן גם בוט ומתודולוגיה דומה (זיהוי שווקים מקבילים בניסוח שונה – NLP יכול לעזור כאן מאוד גם, כי הפלטפורמות מנסחות אחרת). גם דורש התחשבות בעמלות שונות (Kalshi גובה fee על רווח). המערכת שלנו ניתנת להרחבה לזה: ה-Scanner/Normalizer היו קוראים משתי פלטפורמות, מקבצים cross-platform. מנוע התמחור היה בודק גם שם. בפועל, ב-2024-2025 היו כבר כלים (כמו שהשתמש Reddit ציין<sup>55</sup>) שסורקים קאלישי ופולימרקט ומציגים הזדמנויות.

#### ו. תוצאות בפועל:

מקורות שונים מעריכים שסוחר ארביטראז' גרפו **עשרות מיליוני דולר** בתקופה הזו מריבוי הארביטראז'ים<sup>1</sup>. זה לא מספר מבוטל, ומדגיש עד כמה השוק היה לא-יעיל בזמנים סוערים. עוד צוין ש"עוקרי ארביטראז'" ("bot-like bettors") הפכו את Polymarket לסוג של "מכונת ארביטראז'" ולא קזינו" לטענת כתבה<sup>56</sup> – דהיינו, במקום שאנשים יהמרו באקראי, הרבה פעילות נהייתה ניצול שיטתי של פערים.

**סיכום מקרי הבוחן:** הם מראים שהמערכת שהגדרנו אינה תאורטית בלבד. שילוב של NLP לאיתור שווקים תלויים, וחיבור ל-Order Books עם בוט ביצוע מהיר, הוא בדיוק מה שסוחרים מוצלחים עשו בתקופה האחרונה: - בבחירות, זיהוי תלות פופולר ווט ונשיאות והפעלת בוט שעם כל תנועת סקר ישר מנצל פער בין השווקים<sup>49</sup>. - בזירות מרובות, השוואת מחיר וניצול שיהיו או הבדלי משמשים<sup>54</sup>. - בזמן אירועים דרמטיים, היכולת להגיב תוך שניות במקביל (להכניס פקודות לפני שהפער נסגר ידנית).

הארכיטקטורה המודולרית, הטכנולוגיות (Python Async, NLP וכו'), והבנת הסיכונים – כל אלה היוו ככל הנראה את בסיס מערכות ה-HFT (מסחר בתדירות גבוהה) שפעלו בפולימרקט. כפי שמציין אחד המקורות, **"טובי הארביטראז'ים ביצעו אלפי עסקאות בוטים בתקופת בחירות 2024"**<sup>2</sup>, וניצלו את זמני התנודתיות (Debate nights, polling) – זה בדיוק התרחיש שלעיל: הבוט מאזין (אולי אף קיבל חדשות בזמן אמת) ומכה בשוק שבו מחירים "לא הסתדרו" לפני שהקהל הרחב מעדכן.

עם התבססות השוק והצטרפות גופים מקצועיים (למשל השקעת ICE בקאלישי, וצפי כניסת שחקנים גדולים ב-2025), הפערים יצטמצמו<sup>57</sup>. אך כפי שנאמר, נכון ל-2024/25, זו הייתה שעתם היפה של ארביטראז'ים זריזים.

**סיכום:** ארביטראז' קלנדרי ולוגי בשוקי תחזיות הוא תחום מרתק שבו הנחות מתמטיות ולוגיות מתורגמות לאסטרטגיות מסחר אלגוריתמיות. באמצעות הבנה מעמיקה של יחסים בין אירועים, בניית הוכחות לנכונות הארביטראז', ותכנון הנדסי

מוקפד של מערכת אוטונומית – ניתן היה (ועדיין ניתן) לקצור רווחים כמעט חסרי-סיכון משווקי התחזיות הצומחים. תכננו מסמך אופרטיבי שלם: מהיסודות התאורטיים, דרך ההיבטים המעשיים (עמלות, נזילות), ועד לארכיטקטורת תוכנה מבוזרת ומודולרית, שנתמכת בטכנולוגיות עדכניות. דוגמאות מהשטח הדגינו שהתיאוריה פגשה מציאות – והמערכות הללו אכן עובדות. Implementing לארביטראז' קלנדרי הוא אתגר אך בעל תגמול גבוה, וכשהוא מבוצע נכון, הוא שיטתי, סקלאבילי, ומציג כיצד מידע זמן הופכים לכסף בעולם שוקי התחזיות <sup>56</sup>.

ואולי הדבר המרגש ביותר: בעוד משקיעים רגילים "מהמרים" על תוצאות, המערכת שלנו מממשת תפיסה הנדסית – להרוויח ללא קשר לתוצאה, רק בזכות חוסר השלמות של שוקי ההמונים. זהו הכוח של ארביטראז' לוגי, וכפי שהראינו, ניתן לרתום אותו בהצלחה בשוקי התחזיות של השנים האחרונות.

:Sources

1. Navnoor Bawa, "Building a Prediction Market Arbitrage Bot: Technical Implementation", Substack, Nov 2025 <sup>31 29 20 2 1</sup>
2. Oriol Saguillo et al, "Unravelling the Probabilistic Forest: Arbitrage in Prediction Markets", arXiv 2025 <sup>50 49 33</sup>
3. Bocconi Students Investment Club, "Well, CAN THEY PREDICT?": Crypto Prediction Contracts", Oct 2023 <sup>4 3</sup>
4. Navnoor Bawa, "Arbitrage Bot Code Snippets", 2025 <sup>46 45 43 40 42 8 5 28 27 26 25</sup>
5. Probutkickerz Reddit user, "Polymarket x Kalshi Arbitrage Finder", Nov 2025 <sup>19 55</sup>
6. arXiv 2025, "Semantic Trading: Agentic AI for Clustering and Relationship Discovery" <sup>21 13</sup>
7. Arxiv, "Appendix Unravelling the Probabilistic Forest", <sup>35 34</sup>
8. Polymarket Market Page, "Apple product line 2025", 2025 <sup>53 51</sup>

- 
- <sup>44 43 42 41 40 39 38 37 36 32 31 30 29 28 27 26 25 24 23 20 11 9 8 7 6 5 2 1</sup>  
 Building a Prediction Market Arbitrage Bot: Technical Implementation <sup>57 54 47 46 45</sup>  
<https://navnoorbawa.substack.com/p/building-a-prediction-market-arbitrage>
- Well, CAN THEY PREDICT?": An investigation of Crypto Prediction Contracts' – BSIC | Bocconi" <sup>4 3</sup>  
 Students Investment Club  
[/https://bsic.it/well-can-they-predict-an-investigation-of-crypto-prediction-contracts](https://bsic.it/well-can-they-predict-an-investigation-of-crypto-prediction-contracts)
- Unravelling the Probabilistic Forest: Arbitrage in Prediction Markets <sup>50 49 35 34 33 10</sup>  
<https://arxiv.org/html/2508.03474v1>
- Semantic Trading: Agentic AI for Clustering and Relationship <sup>48 21 18 17 16 15 14 13 12</sup>  
 Discovery in Prediction Markets  
<https://arxiv.org/html/2512.02436v1>
- Polymarket x Kalshi Arbitrage Finder : r/Kalshi <sup>55 22 19</sup>  
[/https://www.reddit.com/r/Kalshi/comments/1parkbp/polymarket\\_x\\_kalshi\\_arbitrage\\_finder](https://www.reddit.com/r/Kalshi/comments/1parkbp/polymarket_x_kalshi_arbitrage_finder)
- Will Apple release a new product line in 2025? Prediction... | Polymarket <sup>53 52 51</sup>  
<https://polymarket.com/event/will-apple-release-a-new-product-line-in-2025>

People making silent profits through arbitrage on Polymarket <sup>56</sup>

<https://www.chaincatcher.com/en/article/2212288>