בס״ד

פרויקט גמר תשפ״ה

שיפור ואופטימיזציה של ניהול מלאי באמצעות מודלים לחיזוי - 'סיון טכנולוגיות'

התמחות-כריית מידע

נפתלי כהן | 323591735

אליאור פולק | 318204567

מנחה | מר אורי גלבוע

המחלקה להנדסת תעשיה וניהול

מכון לב

המרכז האקדמי לב

שנת תשפ״ה

בס״ד

תודות

לאורך השנה ובעבודה על הפרויקט נעזרנו באנשים רבים לשאלות ויעוץ בשלל תחומים. אנו רוצים להודות להם על הליווי והסיוע במשך דרך זו.

מר אורי גלבוע, המנחה האקדמי, על הליווי המסור וההדרכה המקצועית במהלך העבודה על הפרויקט, יחד עם יחס חם ותומך. תודה על הגמישות הרבה בקביעת מועדי הפגישות ועל הזמינות במענה על שאלות. תודה על הדאגה, ההשקעה והאכפתיות, שמחנו לעבוד איתך וללמוד מניסיונך הרב.

מר יצחק דדוש, המנחה המקצועי, מהנדס תפ״י בשרשרת אספקה בחברת סיוון, על קבלת ייעוץ מקוצועי לה היינו זקוקים במהלך הפרויקט באופן מהיר, יעיל ומקצועי. על לימוד סבלני ומפורט של כל תהליכי העבודה לצורך ביצוע הפרויקט. על שיתוף הפעולה ועל ההכוונה שתרמה לנו רבות לאורך הדרך.

מר ניב בנוני, מהנדס מערכות מידע של חברת סיוון , על קבלת הנתונים להם היינו זקוקים לביצוע הפרוייקט ,על ההכוונה, הסיוע והיעוץ המקצועי לאורך שלבי הפרויקט.
על סקירה כוללת של תהליך העסקי בארגון, על הזמינות הרבה, הסבלנות, האדיבות והמקצועיות.

תודה לראשי החוג שלנו בשנים האחרונות , ד״ר דויד קאופמן ופרו׳ יהל גיאת,
תודה למזכירות החוג שלנו, הגב׳ מזל גאון על הכלים, ההתנסות, התמיכה והסיוע בכל בעיה.
תודה אחרונה למרכז האקדמי לב שבו למדנו והחכמנו בשנים האחרונות.

תקציר

חברת 'סיון טכנולוגיות' היא חברת סטארט-אפ בירושלים העוסקת בפיתוח מערכות לייזרים. עיקר הלקוחות של החברה הן לקוחות מהעולם התעשייה, מכוני מחקר וגורמים ביטחוניים. חברת 'סיון', בהיותה חברת סטארט-אפ, עובדת בצורה פרויקטאלית על מערכות שונים ולא באופן שוטף על מוצרים קבועים. ב'סיון', המחלקה שאחראית על תכנון וניהול החומר והרכש זו מחלקת 'שרשרת האספקה'. במסגרת האחריות שלה לתכנן אילו פריטים יהיו בשימוש בפרוייקטים הקרובים, ולפי זה לצאת לרכש ולהביא את הפריטים הנדרשים בזמן , בכמות ובאיכות הנדרשת לחברה. במערך ניהול המלאי מיושמת על חלק מן הפריטים מדיניות "מינימום–מקסימום" – שיטת בקרה המבוססת על קביעת גבולות מינמום מקסימום, שמטרתה להבטיח כי רמות המלאי תשמרנה באופן רציף בתוך טווח הערכים שנקבע מראש.

כיום, אין נוסחה סדורה לקביעת הגבולות ואין מנגנון בקרה תקופתי - דבר זה משפיע על המצב הקיים בצורה מובהקת: רק **55%** מכלל הפריטים נמצאים בין גבולות המינימום למקסימום כנדרש, **40%** חורגים מעל המקסימום ו**5%** חורגים מתחת המינימום. קבוצת פריטים זו היא גדולה וכוללת פריטים רבים מקטגוריות שונות, חלקם זולים וחלקם יקרים ולא כולם באותה רמת חשיבות.

אנחנו התמקדנו בקבוצת הפריטים מקטגוריית האופטיקות-זוהי קבוצת פריטים יקרה ויתר על כן גם חשובה וקריטית עבור הייצור והפיתוח-חוסר של פריט מקבוצה זו משמעותו עצירת קו הייצור, אך גם חריגה מגבולות המקסימום היא קריטית שכן לעיתים משנים את גרסת הפריט, וחריגה משמעותה גריטה משמעותית ויקרה של חומר והפסד כלכלי גדול. תמונת המצב הנוכחית בקטגוריה זו היא ש**44%** עומדים בין הגבולות, לעומת **36%** שחורגים מעל המקסימום ו**20%** החורגים מתחת למינימום. נוסף על כך, לחברה אין תחזית הוצאות מסודרת, עניין המכביד על ניהול תזרימי המזומנים.

הפרויקט נועד לספק תחזית צריכה שנתית לקבוצת הפריטים הנבחרת ובכך להשיג שתי תועלות מרכזיות: מתן צפי כספי לשנה הקרובה, והטמעת מודל ניהול מלאי שימליץ על נקודות הזמנה, זמני אספקה צפויים וגבולות מינימום-מקסימום מעודכנים, לשם הקטנת חוסרים ועודפים כאחד.

המדד שלנו להצלחה יהיה בבניית מודל תחזית ובבניית מודל לניהול מלאי טוב כך שנוכל לשפר את רמת המלאי שעומד בין הגבולות שבחברה לפחות ב20%, וכך לעזור לחברה לנהל טוב יותר את המלאי שלה ולחסוך כסף.

להלן שלבי הפרויקט: בשלב הראשון התחלנו באיסוף הנתונים כדי להכין אותם לכרייה. תהליך בניית בסיס הנתונים (DWH) התבצע בהתאם למתודולוגיית ETL .Extract – הנתונים נשלפו ממערכת ה ERP הארגונית באמצעות חיבור בין טבלאות רלוונטיות וכתיבת שאילתות בSQL. Transform - בוצעה טרנספורמציה של הנתונים, שכללה תיקוף, קידוד, סינון וביצוע בדיקות ולידציה על מנת להבטיח את שלמותם ואיכותם. בנוסף, הומרו הנתונים לפורמט המתאים למחסן הנתונים. Load- הנתונים שעברו את שלב העיבוד נטענו למחסן הנתונים אשר משמש כבסיס

לניתוחים והדמיות ב-Power BI וכן לתחזיות המבוססות על למידת מכונה. הזנו את בסיס הנתונים שבנינו לPower BI ובאמצעותו ביצענו ניתוחים על המצב הקיים, והבנו מה רמת החריגות. בכדי לממש את שלב התחזיות , השתמשנו בכלי AI מבוססי למידת מכונה. התחזית שרצינו לייצר היא תחזית המבוססת על למידה היסטורית של דפוסי צריכה שהיו בחברה, ולפי זה בחרנו את המודלים איתם נעבוד. לקחנו ארבע שיטות שונות של חיזויים שמתאימים לדפוס למידה עונתי. המודלים שהשתמשנו הם- PORPHET ARIMA, RANDOM FOREST, XGBOOST ובצענו את המטריקות לכל מודל. הנתונים שהיו לנו הינם משנת 2019 עד 2025. ביצענו חלוקה של הנתונים לפי ציר הזמן, כך ששנים 2019–2023 שימשו כסט האימון (Train Set), ואילו שנת 2024 שימשה כסט הבדיקה (Test Set). **מאחר ומדובר בסדרת זמן, השתמשנו בגישה של Time Series CrossValidation ולא ב-Cross-Validation רגיל**. מול התחזיות שיצאו לנו, השואנו לצריכות שהיו בפועל בשנת 2024, וככה יכולנו להעריך את הדיוק של כל מודל. לאחר קבלת תחזיות לצריכות מכל מודל השוואנו את השגיאות ובחרנו את המודל שהיה הטוב ביותר. על בסיס התוצאות בנינו מודל נוסף שמטרתו היא לחזות את המלאי על בסיס תחזית הצריכה, תוך התחשבות בגבולות המינימום והמקסימום. המודל כולל המלצות ליציאה לרכש וחישוב קליטה במחסן החברה (על בסיס חישובי LT מהיסטוריית המשלוחים), והמלצות לגבולות מינימום-מקסימום חדשים במידת הצורך. כדי לבנות את מודל ניהול המלאי השתמשנו בשיטת Cover-Time Simulation, ובשיטת Ss לניהול מלאי.

מודל Random Forest נבחר כמודל המוביל, לאחר שהציג את מדדי השגיאה הנמוכים ביותר בהרצה עבור כלל המקט"ים .על יסודו פותח דשבורד ניהולי המרכז את היסטוריית המלאי לכלל המקט"ים, מציג השוואה בין תחזיות לצריכה בפועל ברמת הפריט, ומעניק תחזית עלויות ברמת רבעון. הדשבורד משולב במודול ניהול מלאי הממליץ על נקודות הזמנה מיטביות, מחשב את מועד הגעת כל הזמנה תוך הקפדה על גבולות המינימום-מקסימום, ומאפשר התאמה דינמית של אותם גבולות בהתאם לתחזיות הצריכה במטרה לצמצם חריגות. בניית הדשבורד נעשתה תוך שמירה על שלושה עקרונות מרכזיים של UI\UX: Consistency and Standards-שמירה על שפה ויזואלית אחידה, סמלים מוכרים ומיקומים קבועים המבטיחים חווית משתמש נוחה. Visibility of System Status-הצגת מדדי KPI וחיווי מלאי בזמן אמת כך שהמשתמש יבין מיד היכן הוא עומד ; ו-אסתטיות ומינימליסטיות—עיצוב נקי החושף רק את המידע החיוני ומפחית עומס קוגניטיבי, כך שהנתונים עצמם תופסים את הבמה המרכזית. לאחר שיעלה לענן ויתחבר לשרת החברה (עם קונקטור בשם GeteWay), הדשבורד יעבור תהליך ETL מחזורי אחת לרבעון.

ישום ההמלצות צפוי לשפר משמעותית את התאמת רמות המלאי, הרבה למעלה מ20% שקבענו כמדד : שיעור הזמן שבו עמדו הפריטים בין גבולותיהם יעלה מ-**38% (2024) ל-87%**, ובחתך נקודת זמן של הבדיקה יטפס שיעור הפריטים העומדים בדרישות מינימום מקסימום מ-**44% ל-92%.**

בכך מממש הפרויקט את יעדיו – ייעול תהליכי הרכש והמלאי, צמצום הפסדי עודף ומניעת עצירות ייצור – ומבסס תשתית אנליטית תומכת החלטה המבוססת AI ו- BI.

Abstract

Civan Technologies is a Jerusalem-based startup engaged in the development of laser systems. The company's primary clients include those from the industrial sector, research institutes, and defense organizations. As a startup, Civan operates on a project-based model, developing a variety of products rather than working continuously on fixed product lines. At Civan, the department responsible for planning and managing materials and procurement is the Supply Chain department. This department is in charge of determining which items will be needed for upcoming projects, and based on that, initiating procurement to ensure the required items arrive on time, in the right quantity and quality.

For part of the inventory, a "Minimum–Maximum" policy is implemented—this is a control method based on setting lower and upper thresholds, with the goal of keeping stock levels continuously within a predefined range. Currently, there is no formal formula for setting these boundaries, and no mechanism for periodic review. This has a clear impact: only 55% of all items are within the defined minimum–maximum limits, 40% exceed the maximum, and 5% fall below the minimum. This group of items is large and includes components from various categories, some of them low-cost, others expensive, and not all of equal importance.

We focused on the optics category—a group of high-cost and, moreover, critical components for both production and development. A shortage of an item in this group means a production halt, but exceeding the maximum limit is also problematic, as items are sometimes revised, and excess stock may lead to costly scrapping and significant financial losses. The current situation in this category shows that 44% of the items are within limits, while 36% exceed the maximum and 20% fall below the minimum. In addition, the company lacks a structured expenditure forecast, which adds complexity to cash flow management.

The project aimed to provide an annual consumption forecast for the selected group of items, generating two key benefits: providing a financial outlook for the upcoming year and implementing an inventory management model that recommends reorder points, expected lead times, and updated minimum–maximum levels, in order to reduce both shortages and overstock. Our success metric was to build a forecasting and inventory management model that would improve the proportion of inventory within

bounds by at least 20%, thereby helping the company better manage its inventory and reduce costs.

The first step was data collection and preparation for analysis. The construction of the data warehouse (DWH) followed the ETL methodology. Extract: data was pulled from the organizational ERP system by joining relevant tables and writing SQL queries. Transform: the data underwent transformation, including validation, encoding, filtering, and cross-checks to ensure completeness and quality. The data was also converted into a format suitable for the data warehouse. Load: the processed data was loaded into the warehouse, which serves as the foundation for Power BI analysis and simulations, as well as machine learning-based forecasts. We loaded the database into Power BI and used it to analyze the current inventory situation and identify the level of deviation. To generate forecasts, we used AI tools based on machine learning. The forecast we aimed to produce was based on learning historical consumption patterns within the company, and accordingly we selected models suited for seasonal learning patterns. We used four different forecasting techniques: ARIMA, RANDOM FOREST, XGBOOST, and PROPHET, and evaluated each model using standard error metrics. The data available spanned the years 2019 through 2025. We split the data by time, using the years 2019–2023 as the training set, and 2024 as the test set. Since this is time-series data, we applied Time Series Cross-Validation instead of standard Cross-Validation. The forecasted values were compared to actual consumption in 2024 to evaluate the accuracy of each model. After generating consumption forecasts using each model, we compared the error metrics and selected the model that performed best. Based on the results, we built another model designed to predict inventory levels using the consumption forecast, while accounting for the minimum and maximum thresholds. The model includes recommendations for procurement timing and simulates arrivals to the company warehouse (based on historical delivery lead times), as well as recommendations for updated min–max boundaries if necessary. To construct the inventory management model, we applied the Cover-Time Simulation method and the Ss method for inventory control.

The Random Forest model was selected as the leading model after it achieved the lowest error metrics across all SKUs. Based on this model, we developed a managerial dashboard that integrates historical inventory for all items, compares forecasted and actual consumption at the item level, and provides quarterly cost projections. The dashboard also includes an inventory management module that recommends optimal reorder points, calculates expected arrival times based on lead times, and dynamically adjusts boundaries to reduce inventory deviations. The dashboard was built with three main UX/UI principles in mind: Consistency and Standards—maintaining a uniform visual language, familiar icons, and consistent layout to ensure a smooth user

experience; Visibility of System Status—displaying KPIs and inventory indicators in real-time so the user understands the current status at a glance; and Aesthetic and Minimalist Design—a clean interface that highlights only essential information, reducing cognitive load and keeping the focus on the data itself. Once connected to the company's server via a Gateway connector and deployed to the cloud, the dashboard will perform a recurring ETL process on a quarterly basis.

Implementing the recommendations is expected to significantly improve inventory control—exceeding the 20% improvement target: the percentage of time items remain within boundaries will increase from 38% in 2024 to 87%, and at the point-in-time test, the percentage of items meeting requirements will increase from 44% to 92%.

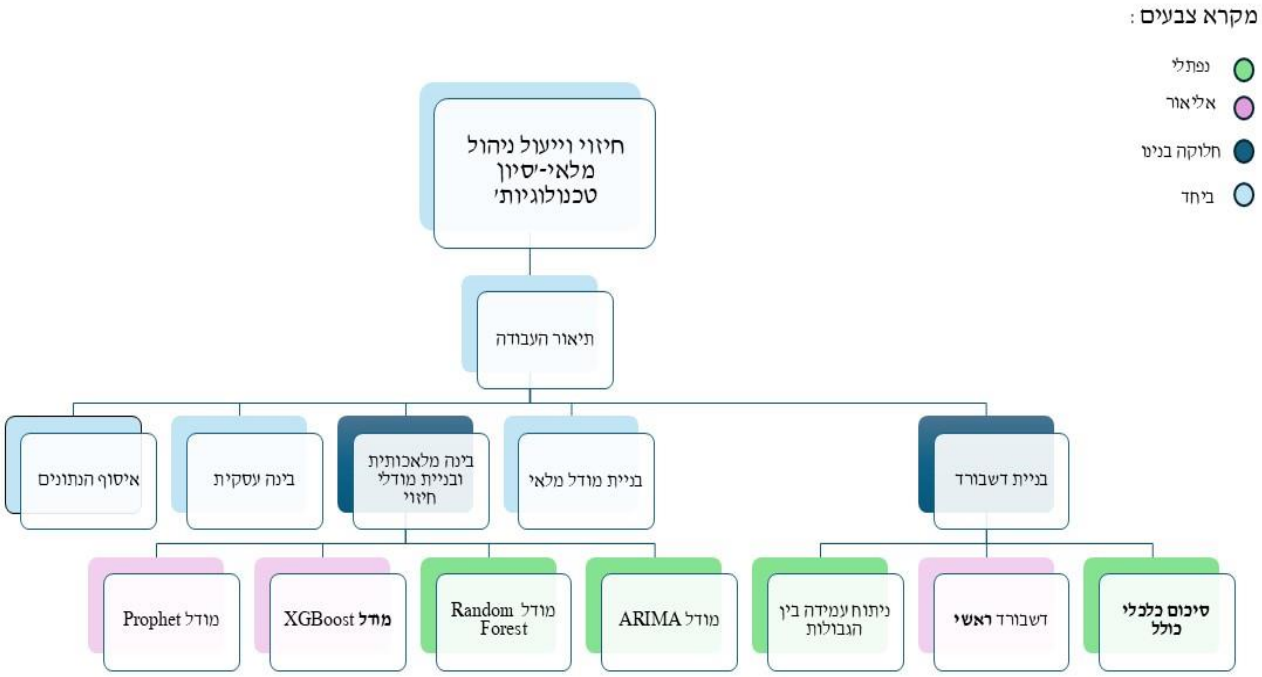
The project thus fulfills its objectives: improving procurement and inventory processes, reducing overstock-related losses and production stoppages, and establishing a decision-support infrastructure powered by AI.

תוכן עניינים

רשימת איורים

רשימת טבלאות

תרשים חלוקת עבודה



מבוא לפרוייקט

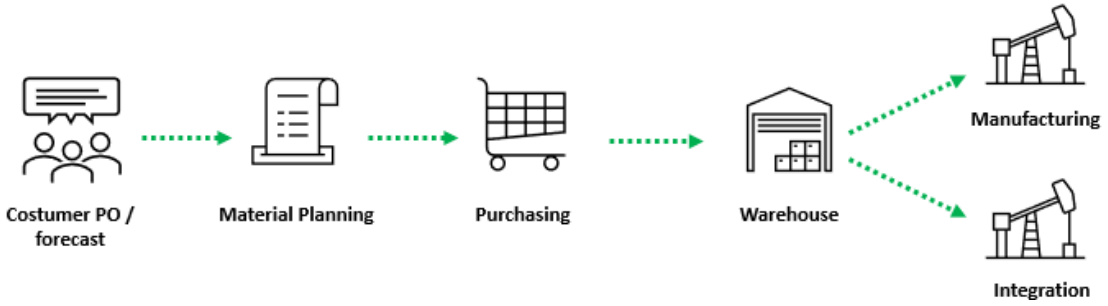
'סיון' הינה חברת היי-טק ירושלמית שהוקמה ע"י ד"ר אייל שקל (בוגר המרכז האקדמי לב) בשנת 2008. החברה מתמחה בפיתוח וייצור של לייזרים בעלי הספק גבוה הן לתעשיות אזרחיות והן לתעשיות ביטחוניות שונות. 'סיון' נחשבת לחברה בעלת טכנולוגיות פורצות דרך בתחומה ועל כך זכתה בשנים האחרונות בפרסים בינלאומיים שונים. חברת 'סיון' הינה חברה יצרנית ומתמקדת בייצור קרן לייזר דינאמית בעלת יכולת שליטה בצורת האלומה, עוצמת הקרן ועומק מיקוד במהירויות שונות. טכנולוגיה ייחודית זו הקנתה למנכ"ל החברה מעל 90 פטנטים עולמיים הרשומים על שמו.

בחברה יש 2 מחלקות שאחראיות באופן ישיר על ייצור הלייזר: מחלקת הייצור ומחלקת האינטגרציה.

במחלקת הייצור של החברה מייצרים כיום שני רכיבים מרכזיים – ראש אופטי ומגברים (AMP). לאחר סיום ייצור המוצרים הנ"ל במחלקת הייצור הם מועברים למחלקת האינטגרציה אשר תפקידה לתכלל אותם למערכת אחת גדולה.

כדי לספק את החומר למחלקות השונות, 'סיון' מחזיקה מחסן גדול שבו מאוחסנים כלל הפריטים והחומרים הרלוונטים לביצוע הייצור, ולפי הצורך מנפקים לקו היצור או האינטגרציה את החומרים הנדרשים להם.

מי שאחראי על החומרים ועל הזמנת חומר נדרש זו מחלקת 'שרשרת האספקה'-המורכבת ממחלקת התפ"י, שאחראית על תכנון וניהול החומר-כולל ניפוקים למחלקות השונות ומעקב אחר המלאים, ומחלקת הרכש שאחראית על הזמנת החומרים ותיאום ההגעה שלהם לחברה.



איור 1-תהליך זרימת החומר ב'סיון'

הבעיה

כיום בסיון, ישנו מחסן גדול המתספק את החברה.

חלק מהפריטים הנמצאים במחסן(כ-500 פריטים) מנוהלים בשיטת 'מינימום-מקסימום'.

'מינימום-מקסימום' זוהי שיטה לניהול מלאי שעובדת ע״י הגדרת גבולות מלאי קבועים לכל פריט הנמצא ברשימה. על הפריט להיות בכל מצב נתון בין הגבולות שהוגדרו לו. ברגע שפריט חורג מגבולות המינימום יש לצאת לרכש ולהזמין עוד מהפריט כדי לא להיות בחוסר.

הפריטים הנמצאים ברשימה הינם פריטים קריטיים וחיוניים עבור החברה, ובלעדיהם קו הייצור של החברה עלול להיעצר.

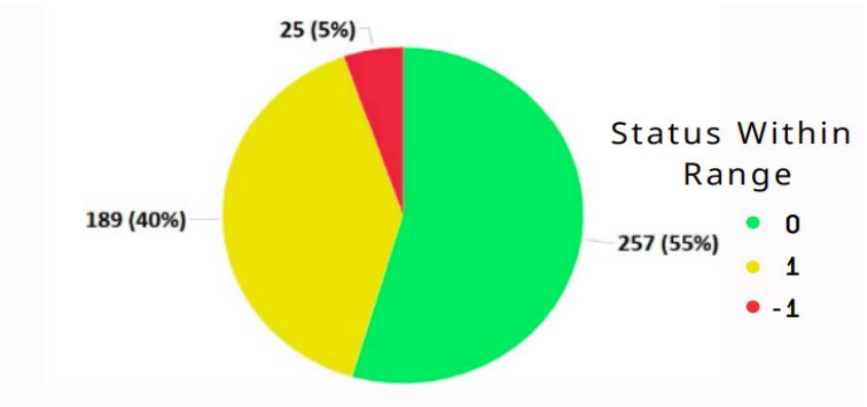
הפריטים האלו אינם ייחודים עבור פרויקט מסוים, אלא משמשים למגוון רחב של שימושים בחברה, ולכן אינם נרכשים כחלק מתהליך הצטיידות לפרויקט אלא הם תמיד צריכים להיות במלאי של המחסן, כדי שיהיו זמינים תמיד לשימוש.

יש לציין-בעת בניית הדו״ח עצמו, עמד שיקול מרכזי אחד-**הפריטים הקריטיים שבשימוש נרחב ייכנסו לדו״ח**, אך מעבר לכך לא היה חקר מעמיק על אילו כמויות וגבולות נכון עבור כל פריט.

כשאנו ניגשנו לחקור את המצב הקיים, רצינו להמליץ על הרשימה והדו״ח שלנו תוך התחשבות בפרטים חיוניים נוספים: עלות הפריט(לא נרצה שפריט יקר נחזיק מלאי גדול לעומת פריט זול שאין בעיה לאחזקה של עוד כמות), בזמן האספקה שלו(אם לפריט יש זמן אספקה רב, נחזיק את הגבול המינימלי שלו בטווח מספיק בטוח מ-0 כדי שלא נהיה בחוסר), בקצב הצריכה שלו(פריט שקצב הצריכה שלו גבוה נחזיק מלאי גדול יותר) וכמות מינימום להזמנה(יש פריטים שניתן להזמין רק בכמות מסוימת ועלינו להתחשב גם בזה).

באופן כללי, יש כמה בעיות שעליהם דיברו איתנו בחברה, שגילינו שיש קשר ביניהן ובין הניהול בשיטת 'מינימום-מקסימום'.

פילוח כמות ואחוז הפריטים מכלל הרשימה שעמדו בתוך הגבולות המוגדרים מובא בתרשים הבא:



איור 2 פילוח עמידה/חריגה מהגבולות של כלל הפריטים

ניתן לראות, כי רק 55% מן הפריטים עומדים בתוך הגבולות שהוגדרו להם-נתון המעיד כי יש בעיה במצב הקיים. ננסה להבין ממה זה נובע ומה הבעיות הנגרמות מכך.

בעיה ראשונה:

קו הייצור נעצר לעיתים תכופות מכיוון שיש חוסר בפריטים קריטיים.

אחת הבעיות המרכזיות שיכולות להתרחש כתוצאה מניהול בעייתי היא חריגה מגבולות המינימום של הפריט. חריגה מגבול המינימום משמעותה המיידית הוא חוסר מהפריט עצמו, דבר שעלול לגרום לחוסר כאשר יצטרכו את הפריט עבור הייצור.

ברגע שיש צורך בפריט והוא אינו זמין במלאי, המשמעות היא שלפחות עמדה אחת תהיה מושבתת עד אשר יגיע הפריט המתאים. עלות של עמדה אחת מושבתת הוא **לפחות 500$** ליום עבודה של הטכנאי. בנוסף יהיו עלויות נוספות-כמו קנייה של הפריט בכל מחיר מבלי אפשרות לניהול מו״מ על המחיר, שילוח מהיר בהטסה במקרה ומדובר בפריט שמגיע מספקים שנמצאים בחו״ל, שעלותה היא כ**800$** יותר ממשלוח רגיל שהיה מתבצע אם ההזמנה הייתה מתרחשת בזמן. בנוסף, ישנם חישובים כוללים יותר-כמו דחיית כלל הפרויקט, דבר שיכול להתרחש אם עמדה מסוימת לא תהיה פעילה במשך יום שלם, שהעלות של קנס ליום מוערכת במאות דולרים (תלוי בלקוח).

חריגה מגבולות המינימום היא הבעיה היותר ׳כואבת׳ מכיוון שהיא מורגשת בטווח הזמן המיידי-עצירת הקו, גרירה של הפרויקט מעבר לזמן שהוקצב ו׳הקפצת׳ כל המערכת למצב חירום כדי לנסות ולהביא את הפריטים החסרים במהירות המירבית.

בעיה שנייה:

חריגה מגבולות המקסימום גורמת לבעיה בקרב פריטים בעלי פגות תוקף.

חלק מהפריטים ברשימה, אלה חומרים מתכלים-ביניהם דבקים ומשחות תרמיות(יש בהם שימוש רב להדבקות ויצירת הפרדות בידודיות) . לחומרים מסוג זה יש פגות תוקף.

התהליך בדרך כלל, הוא שכשיוצאים לרכש של פריטים מסוג זה, משתדלים לקנות אותם כאשר פגות התוקף שלהם מקסימלית (בד״כ-פגת תוקף מקסימלית היא עד שנה). כאשר הפריט למגיע למחסן מסמנים את האצווה שלו וכותבים במערכת את התוקף שלו.

כאשר בפועל מזמינים כמות גדולה, שחורגת מגבולות המקסימום, התוקף של החומר יעבור והוא אינו ראוי לשימוש יותר-ויש כאן בעיה כפולה-**א.** הרבה חומר נזרק לפח(בזבוז של אלפי דולרים) **ב.** הצגה כאילו יש לנו כמות גדולה במלאי למרות שלכאורה אולי אין אף חומר שראוי לשימוש במלאי, ואז גם יכול להיווצר חוסר , למרות שבפועל יש כמות במלאי.

בעיה זו גומרת לבזבוז רב של כסף בכל שנה.

בעיה שלישית:

חריגה מגבולות המקסימום מובילה לחוסר מקום במחסן ומאלצת את החברה להשתמש במחסן חיצוני.

בד״כ, לכל פריט יש את המקום שלו במחסן, ואם יש כמות גדולה שחורגת זה יכול להיות שיקול לפנות לו מקום במקום פריט אחר-הבעיה שהמקום מוגבל, ומכיוון שלא ניתן לאחסן הכל, פריטים

עם כמות גדולה נשלחים לאחסון במחסן חוץ (מחסני חברת Flying Cargo בקרית גת). עלות זו עולה לחברה על כל משטח, ובנוסף עבור משלוח רגיל שיש לבצע בכל פעם מהחברה אל המחסן החיצוני, ומהמחסן החיצוני אל החברה(עלות של **400** ₪).

בעיה רביעית:

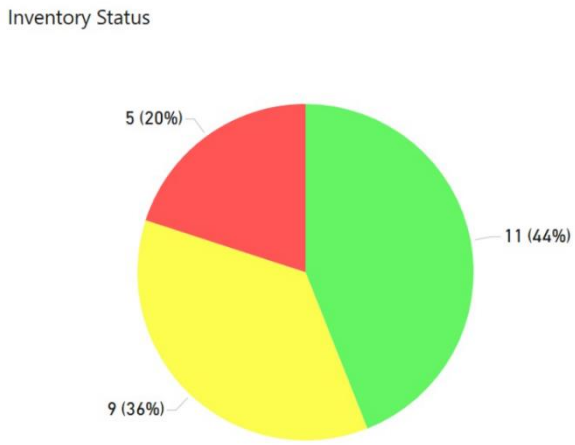
החברה איננה יודעת כיצד יש להיערך כלכלית ולוגיסטית לקראת השנה הבאה בכל הנוגע לרכש מרשימה זו.

כל חברה צריכה לעשות תחזית של הוצאות שנתיות-והחברה במקרה זה אינה יודעת מה יהיו ההערכות של ההוצאות שלה.

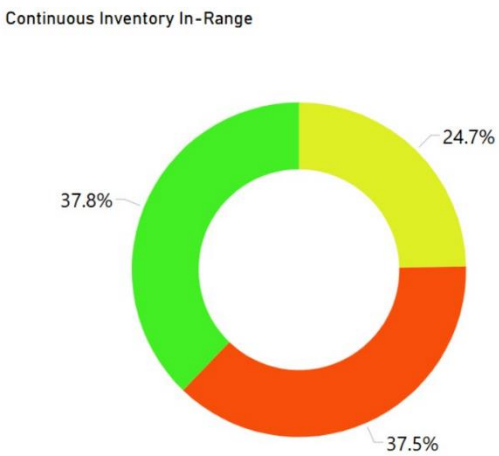
דבר נוסף שהחברה צריכה להיערך אליו-לוגיסטי. זה מתבטא בהכנת ופינוי מקום במדפי המחסן בעת הצורך, אך בנוסף בהכנת המחסנאים ופינוי "רצפת המחסן" שתהיה מוכנה לקליטה בזמנים מסוימים.

מתוך כלל הרשימה הגדולה של פריטי ה'מינימום-מקסימום', **אנו כאמור החלטנו להתמקד בקבוצת הפריטים השייכים לאופטיקות**-מתוך הבנה שפריטים אלו בייחוד הם יקרים וחשובים מאוד עבור הייצור והפיתוח-לא מדובר בעוד דבר פעוט ערך אלא בדברים שבלעדם קו הייצור יעמוד, וגם אנשי הפיתוח של החברה ישבתו מעבודה(עלות מוערכת של יום עבודה של מהנדס פיתוח-כ450$).

קבוצת פריטים אלו מונה כיום 25 פריטים, והתפלגות ה'עמידה בגבולות' שלהם מתפלגת כך :



איור 3-התפלגות כמות פריטים בין הגבולות-אופטיקות



איור 4-התפלגות בשבועות בין הגבולות-אופטיקות

בדקנו שני מדדים : כמות ואחוז הפריטים שהיו בין הגבולות בחתך של נקודת זמן מסויימת, ואחוז הפריטים שהיו בין הגבולות לאורך 5 שנים(בין שנת 2019 לשנת 2024). מצאנו כי אחוז הפריטים שעמדו בין הגבולות היה 44% , ואילו האחוז בשבועות היה גרוע יותר-כ38% בלבד עמדו בין הגבולות, לעומת כמות כמעט זהה של אחוזים שעמדו מתחת לגבול המינימלי. את המדדים האלה נרצה לשפר.

מטרת הפרויקט

מטרות הפרויקט הן לחקור ולהבין את תמונת המצב הקיימת, ולספק כלים מתקדמים מבוססי בינה מלאכותית ולמידת מכונה, לצורך פיתוח מודל תומך החלטה בתחום ניהול המלאי והרכש – במטרה לשפר את קבלת ההחלטות ולהביא לייעול תהליכים תפעוליים בחברה.

נממש את המטרות ע"י השלבים הבאים :

- ביצוע חקר מעמיק של המצב הקיים באמצעות כלי BI-
החברה עצמה אינה יודעת כמה 'חמור' המצב הנובע מהדו"ח, נבצע ניתוח מעמיק של ההשפעה של חריגות מהגבולות המוגדרים בדו"ח, תוך התחשבות בפרמטרים שלא נלקחו בחשבון עד כה (מעולם לא הוציאו דו"ח שמתעד כמה עלו הזריקות של החומרים עם הפגות תוקף לפח, לא בדקו כמה עולה בפועל עצירת קו היצור).
- ביצוע חקר אודות מודלים מבוססים למידת מכונה וכלי AI-
נמצא אילו מודלים יש עם תכונות שיתאימו למקרה המסוים שלנו, שמושפע מהיסטורית הזמנות וביקושים, אילוצי מחיר וזמני אספקה ועוד.
- מציאת מודל מתאים שיעזור לנו לבצע תחזיות של צריכות החברה-
נבחן מספר מודלים כדי שנוכל למצוא את התחזית הטובה ביותר עבור הצריכות העתידיות של החברה. נשווה בין המודלים השונים-בין הדיוק לשגיאות ואת המודל הטוב ביותר נבחר, תוך שנשאף למקסימום דיוק ומינימום שגיאות.
- פיתוח מודל לניהול מלאי המבוסס על מודל התחזיות שנבנה, עם המלצות לרכש – לאחר שנקבל תחזית לצריכות העתידיות של החברה(=ביקוש), נוכל לבנות מודל הממליץ על הזמנות. המודל יתחשב בצריכה החזויה, בגבולות המינימום והמקסימום שנרצה לשמור עליהם, על זמן האספקה ועל הביקוש בזמן האספקה-משתנה שעד היום לא החשיבו אותו כלל. נבנה דשבורד מסודר , בו ניתן יהיה לראות תחזית צריכה עבור כל פריט וצפי הזמנות לשנה הקרובה, כולל נקודות הזמנה וכמויות לכל הזמנה עבור כל פריט.
- כדי לענות על הצפי הכספי, נוסיף בדשבורד צפי הוצאות המתבסס על המודלים שנבנה כדי לתת להנהלה תמונת מצב סביב נושא זה.

אנחנו ניקח מהחברה את הנתונים על השנים 2019-2023, ועליהם נבנה את המודל שלנו, ואת האימותים נוכל לבצע אל מול מה שהיה בפועל בשנת 2024.

המדד שלנו להצלחה יהיה לשפר את אחוז הפריטים העומדים בין גבולות המינימום והמקסימום, שעומד כיום על כ38%, ב20% לפחות .

רקע תיאורטי

Supply Chain Demand Forecast Based on SSA-XGBoost Model

המאמר בוחן את הפוטנציאל של שילוב בין שיטות ניתוח סדרות זמן לבין אלגוריתם למידת מכונה מתקדם לצורך חיזוי ביקוש מדויק בשרשרת אספקה. המחקר מתמקד בשיפור איכות התחזיות על ידי הפחתת רעש מנתוני העבר וזיהוי דפוסים עונתיים ומגמתיים, אשר לעיתים מוסתרים בתוך תנודתיות הביקושים.

המאמר מציע מודל המשלב בין טכניקת SSA (Singular Spectrum Analysis), שמפרקת את סדרת הזמן לרכיבי מגמה, עונתיות ורעש, לבין אלגוריתם XGBoost – אלגוריתם בוסטים מהיר וחזק שמסוגל לטפל בקשרים לא ליניאריים בין משתנים. המודל נבנה כך שתחילה מיושם ניתוח סדרת הזמן באמצעות SSA, אשר מנקה את הנתונים ומבליט את התבניות המשמעותיות, ולאחר מכן נעשה שימוש ב-XGBoost לביצוע התחזית בפועל תוך שימוש גם במאפיינים נוספים כמו קטגוריות מוצר, יום בשבוע, חגים ומאפיינים עונתיים.

שאלת המחקר המרכזית שביקשו המחברים לבחון היא האם שילוב זה – SSA לצורך סינון הרעש ו-XGBoost לצורך למידת הדפוסים – יוכל להניב תחזית מדויקת יותר לעומת שיטות קיימות כגון ARIMA, GBDT או עצי החלטה רגילים. לצורך כך נאספו נתוני ביקוש ברמת SKU, והמודלים השונים הושוו ביניהם על בסיס מדדי ביצוע מקובלים כמו MAE, RMSE ו-R^2.

ממצאי המחקר מצביעים על יתרון ברור לשיטה המוצעת. מודל SSA-XGBoost הציג תוצאות מצוינות, עם דיוק גבוה במיוחד ברמת $R^2 = 0.988$, ושגיאות נמוכות לעומת שאר המודלים שנבדקו. השילוב בין ניקוי הסדרה מראש לבין למידת המכונה אפשר למודל לזהות מגמות עונתיות מורכבות, ולהגיב טוב יותר לשינויים לא צפויים בביקוש. מעבר לדיוק המספרי, המודל נמצא גם כיציב יותר בסביבות עתירות תנודתיות – יתרון חשוב בניהול מלאי דינמי.

מסקנות המאמר הן כי שילוב בין ניתוח סדרות זמן מתקדם לבין אלגוריתם למידת מכונה מדויק וזריז, כמו XGBoost, עשוי להוות פתרון אפקטיבי מאוד לחיזוי ביקושים בשרשראות אספקה. גישה זו יכולה לאפשר לארגונים לתכנן טוב יותר את המלאי, להפחית חוסרים ומלאים עודפים, ולהגיב בצורה מושכלת יותר לשינויים בביקוש. החוקרים ממליצים להמשיך ולבחון את יישום המודל בסביבות מגוונות נוספות – לרבות מוצרים עם דפוסי ביקוש שונים, ענפים תעשייתיים נוספים ואף שילוב משתנים חיצוניים נוספים בעתיד.

Decision rules-based method for dynamic adjustment of Min–Max ordering levels

המאמר עוסק בניתוח ביצועים של מדיניות מלאי מסוג Min–Max בשרשרת אספקה רב-שלבית, תוך בחינת האופן שבו תצורה זו משפיעה על רמות המלאי, זמינות השירות ועלויות התפעול לאורך שרשרת האספקה. המאמר מתמקד במיוחד במצבים שבהם יש תחנות ביניים רבות (כמו מרכזים לוגיסטיים, מחסנים ומוקדי הפצה), ומבקש לבחון את השפעת השימוש במדיניות זו על איזון בין עלויות החזקה, זמן תגובה לביקוש והסתברות לחוסרים.

בבסיס השיטה עומד הרעיון לקבוע עבור כל תחנה בשרשרת גבול תחתון (Min) וגבול עליון (Max) של מלאי, כך שכאשר המלאי יורד מתחת לרמת ה-Min מתבצעת הזמנה להשלמה עד ל-Max. מדובר בשיטה פשוטה יחסית ליישום ולניהול, ולכן היא נפוצה בארגונים רבים, במיוחד כאשר אין תחזית מדויקת או שהמערכת פועלת בצורה תגובתית. עם זאת, החוקרים מציינים כי על אף פשטותה, שיטה זו אינה תמיד יעילה מבחינת עלות או רמות שירות, בעיקר בסביבות עם אי-ודאות גבוהה, זמני הובלה משתנים או תלות הדדית בין התחנות בשרשרת.

שאלת המחקר המרכזית שנבדקה היא כיצד משפיעה מדיניות Min–Max על הביצועים התפעוליים הכוללים של שרשרת האספקה, במיוחד בהקשרים של זמן אספקה (Lead Time), רמות שירות ועלות כוללת. לצורך כך בוצעו סימולציות של מערכות שונות עם פרמטרים משתנים: מספר שלבים בשרשרת, שונות בביקושים, זמני אספקה משתנים, ורמות שונות של גבולות Min ו-Max. המדדים שנמדדו היו עלויות כוללות, אחוז החוסרים, כמות ההזמנות ותדירותן, וכן השפעה על תחנות מעלה הזרם.

הממצאים מצביעים על כך שמדיניות Min–Max עלולה לגרום לאפקט השוט (Bullwhip Effect) כאשר אין תיאום בין התחנות השונות, כלומר תנודות חריגות בביקוש שנובעות דווקא מהתגובות של התחנות להזמנות ולא מהביקוש האמיתי. בנוסף, נמצא כי במערכות מורכבות, שימוש אחיד בשיטת Min–Max לכל התחנות אינו אופטימלי, ויש להתאים את גבולות ה-Min וה-Max לפי תפקיד התחנה, קצב הספקה ורמת הביקוש. עוד נמצא כי תכנון לקוי של הגבולות עלול להוביל להזמנות תכופות מדי או לחוסר זמינות חוזר.

מסקנת המאמר היא כי אף ששיטת Min–Max פשוטה ונוחה ליישום, יש להשתמש בה בזהירות בסביבות לוגיסטיות מורכבות, ולשלב בה התאמות דינמיות לפי ביצועים בפועל, מגמות בביקוש ומאפייני כל תחנה בשרשרת. כמו כן, מומלץ לשלב בין Min–Max לבין מודלים תחזיתיים או שיטות בקרה מתקדמות כדי לשפר את היעילות ולהפחית עלויות. המאמר מציע גישה מבוססת סימולציה ככלי עזר חשוב בקבלת החלטות על מבנה וניהול מלאי בשרשראות מורכבות.

State-space ARIMA for supply-chain forecasting , International Journal of Production Research BY Ivan Svetunkov, John E. Boylan.

במאמרם ,"State-space ARIMA for Supply-Chain Forecasting" מציעים Svetanka ו-Boylan גישה מתקדמת לחיזוי סדרות זמן באמצעות הרחבה של מודל ARIMA הקלאסי, המתואמת במיוחד לאתגרים ייחודיים בשרשרת אספקה. המודל מבוסס על מסגרת state-space עם מקור שגיאה יחיד (Single Source of Error - SSOE) אשר מאפשר ייצוג מתמטי שקוף ונוח יותר לניתוח, תוך שמירה על גמישות ודיוק גבוהים. בניגוד לגישת Box-Jenkins המסורתית, הדורשת תהליך זיהוי מורכב והתאמה מדוקדקת של הפרמטרים (p,d,q), הגרסה המוצעת שואפת לפשט את השלבים תוך שיפור התוצאה.

החוקרים מציינים כי היתרון המרכזי של גישה זו טמון בהתאמתה למצבים בהם עומק סדרת הזמן מוגבל – מצב שכיח בארגונים תעשייתיים, במיוחד בעת ניתוח מוצרים עונתיים או השקות חדשות. הם מדגישים כי התחזיות המתקבלות מהמודל אינן רק מדויקות, אלא גם יציבות ונוחות לפרשנות עסקית ,יתרון חשוב בניהול מלאי, ביקושים ורכש.

במהלך המחקר, בוצעה השוואה מעמיקה בין המודל החדש לבין מודלים סטנדרטיים של ARIMA, תוך שימוש בנתוני אמת משרשראות אספקה שונות. תוצאות הניסוי הדגימו שיפור משמעותי בדיוק החיזוי, בעיקר בתחזיות קצרות טווח.

המאמר מסכם כי השימוש במודל ARIMA במבנה state-space מהווה פתרון ישים, אפקטיבי ואף מועדף במצבים תפעוליים שבהם סדרות הזמן מצומצמות או מקוטעות – תרחיש נפוץ במציאות העסקית.

מודל XGBOOST

XGBoost הוא אלגוריתם ללמידת מכונה ממשפחת ה־ Boosting, שנועד לשפר את ביצועי המודלים ע"י בנייה של עצים בצורה הדרגתית, כאשר כל עץ חדש מתקן את השגיאות של העצים הקודמים. האלגוריתם מבוסס על טכניקת Gradient Boosting ומשלב אופטימיזציה מתקדמת, עונש על מורכבות יתר (רגולריזציה), ותמיכה בפרלליזציה, מה שהופך אותו למהיר, מדויק ויעיל במיוחד בבעיות של סיווג (classification) ורגרסיה(regression) .

רכיבים עיקריים של XGBoost:

- Boosting : בניית סדרת עצים כאשר כל עץ מתקן את השגיאות של העץ הקודם.
- Gradient Descent : העצים מותאמים כך שימזערו את פונקציית האובדן (Loss) על-ידי גרדיאנט – השיפוע של השגיאה.
- Regularization (L1/L2) : מנגנון רגולריזציה שמונע overfitting ע"י הענשת עצים מורכבים מדי.
- Parallel Processing : תמיכה בהרצה מקבילית של חישובים, מה שמקצר את זמן האימון.
- Handling Missing Values : יכולת פנימית להתמודדות עם ערכים חסרים בצורה אוטומטית.
- Importance Calculation : חישוב חשיבות לכל תכונה באופן אוטומטי כחלק מתהליך האימון.

כיצד XGBoost פועל?

1. התחלה :יצירת עץ בסיסי שמבצע חיזוי גס.
2. איבוד שגיאה :חישוב השגיאות של העץ הנוכחי.
3. תיקון שגיאה :יצירת עץ חדש שמתמקד בשגיאות אלו, בשימוש בגזירה של פונקציית האובדן.
4. חזרה :חזרה על התהליך מספר פעמים עד שמושגת רמת דיוק מספקת או שמגיעים למספר מקסימלי של עצים.
5. תחזית סופית :סכימה של התחזיות מכל העצים לצורך קבלת התחזית הסופית.

יתרונות עיקריים של XGBoost:

- דיוק גבוה מאוד ביחס לאלגוריתמים אחרים.
- מהירות אימון גבוהה הודות למקביליות.
- יכולת להתמודד עם נתונים לא נקיים או חסרים.
- גמישות בשימוש : מתאים לרגרסיה, סיווג, בעיות דירוג ועוד.

- תמיכה פנימית בטכניקות של רגולריזציה ו־ early stopping.
- תומך במנגנוני early stopping ובקרת מורכבות פנימית.

חסרונות של מודל XGBoost

- נדרש כוונון פרמטרים (Hyperparameter Tuning) ידני ומדויק כדי להגיע לביצועים מיטביים, מה שדורש זמן וניסיון.
- קשה לפרש את תוצאות המודל בהשוואה למודלים פשוטים (כמו רגרסיה ליניארית), ולכן פחות מתאים כאשר נדרש הסבר ברור לתחזית.
- רגיש מאוד לנתונים רעשניים אם לא נעשית הכנה מוקדמת (preprocessing) איכותית.
- שימוש בזיכרון גבוה יחסית, במיוחד בבעיות עם כמויות נתונים גדולות מאוד.
- אינו תומך בעבודה טבעית עם משתנים סדרתיים לאורך זמן (כמו סדרות זמן) – יש להוסיף lag-ים ותכונות ידניות.

מודל Prophet

Prophet הוא מודל לתחזית סדרות זמן שפותח על ידי Facebook (Meta), ונועד להנגיש תחזיות גם למשתמשים ללא רקע סטטיסטי עמוק. המודל מבוסס על פירוק של סדרת הזמן לשלושה רכיבים עיקריים: מגמה (trend), עונתיות (seasonality), ואירועים חריגים (holidays/events). Prophet מתאים במיוחד לנתוני ביקוש עסקיים המאופיינים בדפוס עונתי ברור ובשינויים ארוכי טווח.

רכיבים עיקריים של מודל Prophet:

- מגמה: (Trend) תיאור שינויים ארוכי טווח בביקוש, עם אפשרות לשבירות (changepoints)או צמיחה לוגיסטית.
- עונתיות: (Seasonality) מחזורים כמו שבוע, חודש או שנה – מדוללים באמצעות פונקציות פורייה.
- אירועים: (Holidays/Events) ניתן להוסיף תאריכים חשובים כמו חגים, השקות, השבתות ייצור וכדומה, והמודל מחשב את השפעתם.
- טווחי חיזוי: (Uncertainty intervals) המודל מספק תחזיות הכוללות טווחי ביטחון הסתברותיים.

כיצד Prophet פועל:

1. פירוק הסדרה לרכיבים עיקריים.
2. התאמה נפרדת של כל רכיב(trend, seasonality, events) .
3. שילוב הרכיבים לתחזית כוללת.
4. הפקת תחזיות קדימה עם רצועות אי-ודאות.

יתרונות של מודל Prophet

- פשטות וקלות שימוש – מתאים גם למשתמשים ללא ניסיון קודם בתחזיות או למידת מכונה.
- מתמודד היטב עם נתונים חסרים או חריגים.
- תומך בעונתיות מרובה כמו שבועית, חודשית ושנתית.
- מאפשר הוספת השפעות של חגים ואירועים חיצוניים.
- מתאים במיוחד לתחזיות עסקיות עם דפוסים מחזוריים ברורים.
- מפיק תחזיות מהירות, גם על מספר סדרות זמן במקביל.

חסרונות של מודל Prophet

- מדויק פחות כאשר אין עונתיות ברורה או כאשר הדפוסים מורכבים ולא ליניאריים.
- מניח שעונתיות נשמרת קבועה לאורך זמן – מתקשה בזיהוי שינויים חדים בדפוס.
- מוגבל למודלים פרשניים בלבד – לא תומך באינטגרציה עם למידה עמוקה או נתונים מרובי ממדים.
- מבוסס בעיקר על הנחות ליניאריות, ולכן עשוי להיות פחות מדויק לעומת מודלים מתקדמים יותר כמו XGBoost או LSTM.
- אינו כולל גורמים חיצוניים אוטומטית – יש להוסיף אותם ידנית (כגון מבצעים, מחירים, מזג אוויר וכד׳).

מודל ARIMA

ARIMA (AutoRegressive Integrated Moving Average) הוא מודל סטטיסטי לחיזוי סדרות זמן (Time Series) . המודל מניח כי הערכים העתידיים של סדרה תלויים בערכים הסטוריים, בשינויים בין ערכים, ובשגיאות חיזוי קודמות (ממוצע נע של שגיאות). מטרת המודל הוא לזהות את הדפוסים המערכתיים בנתוני העבר – כמו מגמה (trend), עונתיות (seasonality) ורעש (noise) ולהשתמש בהם כדי לחזות ערכים עתידיים.

רכיבים מרכזיים של מודלARIMA

ARIM מוגדר על-פי שלושה פרמטרים : ARIMA (p, d, q).

תפקיד	תיאור	פרמטר
מתאר תלות בזמן	מספר הערכים הקודמים של הסדרה עליהם מבוססת התחזית.	**p** – AutoRegression
מתקן מגמה או עונתיות	כמה פעמים נבצע הבדלים (Differencing) על הסדרה כדי להפוך אותה לסטציונרית	**d** – Integration
מתאר רעש סטוכסטי	משתמש בשגיאות חיזוי קודמות כדי לשפר את התחזית	**q** – Moving Average

שלבי ביצוע של מודלARIMA

1. בדיקת סטציונריות : סדרת זמן צריכה להיות סטציונרית כלומר ממוצע, שונות ואוטוקורלציה קבועים לאורך זמן.
   מבצעים את מבחן ADF Test (Augmented Dickey-Fuller) , אם הסדרה אינה סטציונרית – משתמשים ב־ differencing (כלומר d>0) להפוך אותה לכזו.
2. קביעת ערכי p, q : שתמשים בגרפים של : ACF (Autocorrelation Function) כדי לאמוד את q . PACF (Partial Autocorrelation Function) כדי לאמוד p.
3. אימון המודל : מבצעים התאמה על סט האימון.
4. בדיקת ביצועים : בוחנים את שאריות המודל – (residuals) האם הן מתפלגות נורמלית ובנוסף מודדים שגיאה על סט הבדיקה MAE, RMSE, MAPE :וכו'.

יתרונות חסרונות

- פשטה יחסית : מבוסס על עקרונות סטטיסטיים ברורים ונוח להבנה.
- יעילות בסדרות זמן קצרות : מצליח להפיק תחזיות טובות גם מסטים קטנים יחסית.
- שליטה בפרמטרים : מאפשר שליטה ידנית על מרכיבי המודל ואופטימיזציה.
- דורש סטציונריות : סדרות זמן לא סטציונריות דורשות טיפול מקדים כמו differencing, ולעיתים גם זה לא מספיק.
- רגיש לרעש ועונתיות חזקה : לא מתאים טוב לסדרות עם עונתיות חזקה.
- מורכב לבחירת פרמטרים : בחירת p, d, q אופטימליים דורשת ניסיון וניתוח.

מודל Random Forest

Random Forest הוא אלגוריתם למידת מכונה מסוג *Ensemble Learning*, המבוסס על בניית קבוצה של עצי החלטה (Decision Trees) וביצוע תחזיות באמצעות ממוצע (לרגרסיה) או הצבעת רוב (לסיווג). המודל יוצר עצים שונים זה מזה באמצעות דגימה אקראית של נתונים ושל מאפיינים, וכך מפחית שונות ומונע *overfitting*. השילוב של תחזיות עצמאיות תורם לדיוק, יציבות ויכולת הכללה טובה יותר לעומת עץ יחיד.

רכיבים מרכזיים של Random Forest

- עצים (Decision Trees): כל עץ הוא מודל פשוט יחסית שמקבל החלטות על סמך שאלות. בעצי סיווג, העלים הם קטגוריות בעצי רגרסיה – ערכים מספריים.
- Bootstrap Aggregation (Bagging): עבור כל עץ, מבוצעת דגימה אקראית עם החזר (bootstrap sample) מתוך סט האימון. כך כל עץ מתאמן על תת־קבוצה שונה, מה שמכניס שונות למערכת ומונע התאמת יתר (Overfitting) .
- Random Feature Selection: בכל פיצול בעץ, נבחרת תת־קבוצה אקראית של משתנים (features) ולא כל המשתנים. זה מגביר את ה־ diversity בין העצים ומונע מצב שבו כל העצים מתמקדים באותם משתנים.
- (Aggregation) הצבעה /ממוצע: לאחר שכל העצים נבנו, השלב האחרון הוא שקלול התוצאות,
  - בסיווג: (Classification) כל עץ "מצביע" על קטגוריה, והתוצאה נקבעת לפי הרוב.
  - ברגרסיה: (Regression) נלקח ממוצע הערכים שמנבאים העצים.

שלבי ביצוע של Random Forest

1. הגדרת פרמטרים:

   n_estimators- מספר העצים ביער (ככל שיותר, לרוב מדויק יותר אך איטי יותר).

   max_depth, min_samples_split, max_feature - מגבלות על גודל העצים וגמישותם.

2. דגימת Bootstrap לכל עץ: נבחרים אקראית דגמים מהנתונים המקוריים, עם החזר, ליצירת סט אימון חדש לכל עץ.
3. בניית עץ החלטה לכל דגימה: בעץ נבחרת תת־קבוצה אקראית של משתנים בכל צומת. כל עץ נבנה עד לעומק מסוים או עד למיצוי תנאי העצירה.
4. ביצוע תחזית משוקללת: אם מדובר בסיווג – בודקים איזו קטגוריה הכי נפוצה. אם מדובר ברגרסיה – מחשבים את ממוצע התחזיות.

יתרונות / חסרונות

- דיוק גבוה : בזכות שילוב תוצאות של עצים רבים, המודל מצליח להתמודד עם מורכבות גבוהה ולהפחית שגיאות.
- עמידות בפניOverfitting - עצים בודדים נוטים ל־ overfit, אך שילוב של עצים מגוונים מקטין את הסיכון.
- התמודדות עם נתונים חסרים - אלגוריתם עצי ההחלטה יכול להתחשב בעובדה שחלק מהמידע חסר.
- זמן חישוב גבוה : כיוון שיש הרבה עצים ויש צורך לבצע דגימות חוזרות, המודל כבד יותר לעיבוד ודרוש זמן אימון ארוך.
- צורך בזיכרון גבוה : מודלים גדולים עם עצים מרובים עלולים להיות תובעניים מבחינת משאבי מערכת.
- תלות בפרמטרים : יש לבחור בקפידה את מספר העצים, גובהם, מספר המשתנים לכל פיצול ועוד – מה שדורש ניסוי וטעייה.

שיטת ומודלים לניהול מלאי

Min–Max בשיטת Time Cover

שיטת Min–Max היא אחת מהשיטות הפשוטות והנפוצות ביותר לניהול מלאי, שבה מגדירים רמת מלאי מינימלית (Min) ורמת מלאי מקסימלית (Max) לכל פריט. כאשר רמת המלאי יורדת אל מתחת לרף ה־Min, מתבצעת הזמנה שמטרתה להעלות את רמת המלאי חזרה עד לרף ה־Max.

בגישה הקלאסית, ערכי ה־Min וה־Max הם קבועים, אך בשיטת Time Cover הם מחושבים באופן דינמי, בהתאם לתחזית הביקוש ולזמן הכיסוי הנדרש. במקום לקבוע כמויות אבסולוטיות, השיטה מתמקדת בכיסוי של מספר שבועות קדימה, מה שמאפשר התאמה לתנודות בעונתיות, שינויים בקצב הצריכה, או זמני אספקה משתנים.

עקרונות הפעולה של Time Cover

הגדרת Time Cover : מספר השבועות קדימה שיש לכסות במלאי. לדוגמה, אם ה-Time Cover הוא 6 שבועות, ה-Max יהיה שווה לסך הצריכה הצפויה ל־6 השבועות הקרובים.

חישוב Max : מבוסס על תחזית הביקוש לטווח זמן מוגדר מראש (למשל : 6 שבועות).

חישוב Min : לרוב מייצג את צריכת הביניים הממוצעת בזמן האספקה (Lead Time), ולעיתים כולל גם מלאי ביטחון.

ברגע שרמת המלאי יורדת מתחת ל־Min, נבצע הזמנה בגודל של Max - רמת המלאי הנוכחית.

יתרונות של Time Cover

מאפשר ניהול מלאי דינמי המבוסס על תחזית ולא על ערכים קבועים.

מתאים במיוחד לפריטים עם עונתיות או תנודתיות בביקוש

מפחית עודפי מלאי בתקופות שפל, ומספק מענה טוב יותר בתקופות שיא.

ניתן להטמעה פשוטה באקסל או בכל מערכת ERP סטנדרטית.

תורם לייעול הרכש והורדת עלויות אחסון.

חסרונות של Time Cover

דורש תחזית ביקוש אמינה – כל שגיאה בתחזית עלולה לגרום לחוסרים או עודפים.

יש לעדכן את התחזית ואת ה-Time Cover באופן שוטף (למשל אחת לשבוע), אחרת המודל מאבד מהדיוק.

אינו מתאים לפריטים בעלי ביקוש נדיר (intermittent demand), שכן קשה לחזות עבורם את הצריכה השבועית.

אינו מתחשב בעלויות הזמנה קבועות או באילוצי לוגיסטיקה מורכבים (כמו מינימום הזמנה, קיבולת רכב, שילוח מרוכז וכו').

שיטת Safety Stock

שיטת Safety Stock (מלאי ביטחון) היא גישה מרכזית בניהול מלאי שמטרתה להתמודד עם חוסר הוודאות בביקוש ובזמני האספקה. השיטה קובעת רף של מלאי נוסף – מעבר לצריכה הצפויה – שנשמר במחסן כגיבוי למקרה של עיכובים באספקה או עלייה פתאומית בביקוש.

המלאי הבטוח מהווה מעין "כרית ביטחון" המפחיתה את הסיכון לחוסרים (stockouts) ובכך משפרת את רמת השירות ללקוח. גובה מלאי הביטחון מחושב לרוב בהתבסס על סטיית התקן של הביקוש, זמן האספקה, ורמת השירות הרצויה (Service Level).

עקרונות הפעולה של שיטת SS

חיזוי צריכה רגילה : תחזית הביקוש בזמן ההובלה (Lead Time).

הוספת מלאי ביטחון : תוספת מחושבת לפי רמת הסיכון שהעסק מוכן לספוג.

נקודת ההזמנה נקבעת כ :Reorder Point (ROP)

ROP = ביקוש ממוצע ב-LT + מלאי ביטחון

כאשר המלאי בפועל יורד מתחת ל־ROP, מתבצעת הזמנה.

יתרונות של שיטת SS

- מגדילה את אמינות האספקה ומפחיתה חוסרים.
- פשוטה להבנה ויישום – במיוחד בארגונים עם ביקוש יציב.
- גמישה – ניתן להתאים את רמת מלאי הביטחון לפי רמת השירות הרצויה.
- מתאימה כמעט לכל ענף – תעשייה, קמעונאות, רפואה, לוגיסטיקה ועוד.
- ניתן לשלב אותה עם שיטות אחרות (כמו EOQ או Time Cover).

חסרונות של שיטת SS

- עשויה להוביל לעודפי מלאי אם רמת הביטחון מוגזמת.
- מבוססת על סטטיסטיקות היסטוריות – ולכן עשויה להיכשל במצבים משתנים (שוק תנודתי, עונתיות פתאומית).
- אינה מתחשבת בעלויות רכש או עלויות אחסון – רק ברמת השירות.
- חישוב מדויק של SS דורש הבנה סטטיסטית ונתונים מהימנים על סטיית התקן של הביקוש וזמן האספקה.
- השיטה מגיבה – אך לא מונעת – חוסרים, ולכן יעילותה תלויה בדיוק התחזית.

תיאור העבודה

שלב א'-איסוף הנתונים

בשלב זה, המטרה הייתה להתחיל לאסוף את הנתונים שיהיו הכנה לכל המודלים שאותם נבנה בהמשך. חשוב לציין שבפרוייקט מסוג זה, בניית בסיס הנתונים זו אבן היסוד של כל הפרוייקט, שעליה בנויים כל המודלים שנבנה. במידה ובסיס הנתונים אינו אמין, מדוייק, נקי ומוכן להרצת מודלים- המודלים שנקבל לא יביאו לנו תוצאות חיזוי טובות.

במסגרת הפרויקט, תהליך בניית בסיס הנתונים התבצע בהתאם למתודולוגיית **ETL** (**Extract, Transform, Load**) הכוללת את השלבים הבאים:

- **חילוץ נתונים (Extract)** – הנתונים נשלפו ממערכת ה ERP-הארגונית באמצעות חיבור בין טבלאות רלוונטיות וכתיבת שאילתות בSQL , תוך מיקוד בפריטים המשפיעים על ניהול המלאי בשיטת מיני – מקס.
- **טרנספורמציה ועיבוד נתונים (Transform)** - בוצעה טרנספורמציה של הנתונים, שכללה תיקוף, קידוד, סינון וביצוע בדיקות ולידציה על מנת להבטיח את שלמותם ואיכותם. בנוסף, הומרו הנתונים לפורמט המתאים למחסן הנתונים.
- **טעינת הנתונים (Load)**- הנתונים שעברו את שלב העיבוד נטענו למחסן הנתונים אשר משמש כבסיס לניתוחים והדמיות ב Power BI,-וכן לתחזיות המבוססות על למידת מכונה.

כדי לבנות את בסיס הנתונים, השתמשנו במערכת הERP של החברה(החברה משתמשת בSAP) כדי להוציא את הנתונים. במערכת יש מאות טבלאות, והיינו צריכים להבין אילו טבלאות עלינו להשתמש ולחבר כדי לבנות את בסיס הנתונים הנדרש לשם ביצוע העבודה.

ביצענו מיפוי של הטבלאות בהן אנחנו צריכים להשתמש. הגענו ל8 טבלאות עיקריות שנרצה להוציא מהם נתונים:

- טבלת OITM: טבלת פרטי פריטים – כוללת את כל המאפיינים של כל פריט במערכת (מק"ט, תיאור, מלאי זמין, מדידה, ניהול באצוות וכו').
- טבלת OPOR: טבלת כותרות של הזמנות רכש – כוללת את פרטי ההזמנה הכלליים כגון מספר מסמך, תאריך, ספק, סטטוס וכו'.
- טבלת POR1: טבלת שורות של הזמנות רכש – מכילה את פרטי הפריטים בכל הזמנה, כולל מק"ט, כמות, מחיר ותיאור הפריט, תוך קישור להזמנת הרכש הראשית דרך DocEntry.
- טבלת OPDN: טבלת כותרות של מסמכי קליטת סחורה – מתעדת את פרטי הקבלה הכלליים של הסחורה מהספק, כולל תאריך, מספר מסמך, קוד הספק וסטטוס.
- טבלת PDN1: טבלת שורות של מסמכי קליטת סחורה – מפרטת את הפריטים שנקלטו בפועל לפי שורות, כולל מק"ט, כמות, מחיר, מיקום במחסן וקישור להזמנת הרכש המקורית.

- טבלת OIGE : טבלת כותרות של תנועות יציאה מהמלאי – מתעדת את פרטי המסמך של ההוצאה מהמחסן (לדוגמה : העברה, החזרה או גריעה).
- טבלת IGE1 : טבלת שורות של תנועות יציאה מהמלאי – מפרטת את הפריטים שיצאו בפועל, כולל מק"ט, כמות, מחיר, מיקום ועוד, תוך קישור למסמך הראשי.
- טבלת OITB : טבלת קטגוריות פריטים – מכילה את הקטגוריות (קבוצות) שאליהן משתייכים פריטים מתוך טבלת OITM.

מצורף פירוט המפרטים העיקריים של כל טבלה:

טבלה OPOR-1

OPOR	Data type	Discripion
DocNum	int	מספר המסמך כפי שמוצג למשתמש – מספר ההזמנה.
DocDate	datetime	תאריך המסמך (תאריך ההזמנה).
DocDueDate	datetime	תאריך אספקה משוער (Due Date).
DocEntry	int	מפתח פנימי של המסמך (Primary Key)
CardCode	int	קוד הספק (Business Partner Code).
DocStatus	nchar(1)	סטטוס המסמך (פתוח, סגור, מבוטל וכו').

טבלה POR1-2

POR1	Data type	Discripion
DocEntry	int	קישור להזמנת הרכש הראשית (OPOR).
BaseEntry	int	מפתח פנימי (DocEntry) של המסמך שממנו נוצרה שורת ההזמנה.
LineNum	int	מספר שורה במסמך.
ItemCode	nvarchar(50)	קוד המוצר או הפריט שנרכש.
Quantity	numeric(19,6)	כמות הפריטים שהוזמנה בשורה זו.
Price	numeric(19,6)	מחיר ליחידה של הפריט.
Dscription	nvarchar(100)	תיאור הפריט כפי שנרשם בהזמנה.

טבלה OITM-3

OITM	Data type	Discripion
ItemCode	nvarchar(20)	קוד ייחודי של הפריט (Primary Key).
ItemName	nvarchar(100)	שם הפריט כפי שמוצג למשתמשים.
ItmsGrpCod	int	קוד קבוצת פריטים (מקשר ל־OITB).
InvntryUom	nvarchar(20)	יחידת מידה ראשית לניהול מלאי (למשל : יח', ק"ג).
OnHand	numeric(19,6)	כמות נוכחית במלאי בכל המחסנים.
LastPurPrc	numeric(19,6)	מחיר אחרון ששולם ברכישה.
ManBtchNum	char(1)	האם הפריט מנוהל לפי מספרים סידוריים / אצוות (Batch Management).
OnOrder	numeric(19,6)	כמות בהזמנות רכש פתוחות לספקים.

טבלה OITB-4

OITB	Data type	Discripion
ItmsGrpCod	int	מזהה קבוצת פריטים (Primary Key)
ItmsGrpNam	nvarchar(100)	שם קבוצת הפריטים (שם תיאורי)

טבלה 5-OPDN

OPDN	Data type	Discripion
DocNum	int	מספר המסמך כפי שמוצג למשתמש – מספר קליטה.
DocDate	datetime	תאריך המסמך (תאריך קליטה).
CardCode	int	קוד הספק (Business Partner Code).
DocEntry	int	מפתח פנימי של המסמך (Primary Key)
DocStatus	nchar(1)	סטטוס המסמך (פתוח, סגור, מבוטל וכו').

טבלה 6-PDN1

PDN1	Data type	Discripion
DocEntry	int	קישור לקליטת רכש הראשית (OPDN).
BaseEntry	int	מפתח פנימי (DocEntry) של המסמך שממנו נוצרה שורת הקליטה.
LineNum	int	מספר שורה במסמך.
ItemCode	nvarchar(50)	קוד המוצר או הפריט שנקלט.
Quantity	numeric(19,6)	כמות הפריטים שנקלטו בשורה זו.
Price	numeric(19,6)	מחיר ליחידה של הפריט.
Dscription	nvarchar(100)	תיאור הפריט כפי שנרשם בהזמנה.
BinAbs	int	איתור במחסן (Bin Location).
WhsCode	int	קוד המחסן אליו יכנס הפריט.

טבלה 7-OIGE

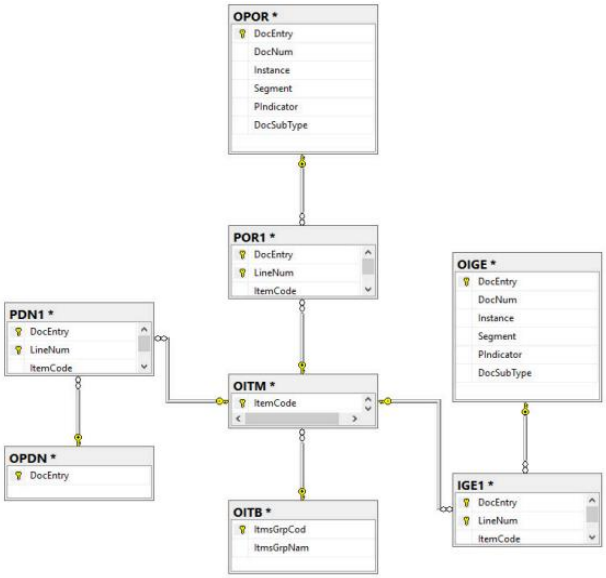
OIGE	Data type	Discripion
DocNum	int	מספר המסמך כפי שמוצג למשתמש – מספר ניפוק.
DocDate	datetime	תאריך המסמך (תאריך הניפוק).
UserSign	int	קוד המשתמש שיצר את המסמך.
DocEntry	int	מפתח פנימי של המסמך (Primary Key).
CardCode	int	קוד לקוח.
DocStatus	nchar(1)	סטטוס המסמך (פתוח, סגור, מבוטל וכו').

טבלה 8-IGE1

IGE1	Data type	Discripion
DocEntry	int	קישור להנפקת רכש הראשית (OIGE).
BaseEntry	int	מפתח פנימי (DocEntry) של המסמך שממנו נוצרה שורת הנפקה.
LineNum	int	מספר שורה במסמך.
ItemCode	nvarchar(50)	קוד המוצר או הפריט שהונפק.
Quantity	numeric(19,6)	כמות הפריטים שהונפקו בשורה זו.
Price	numeric(19,6)	מחיר ליחידה של הפריט.
Dscription	nvarchar(100)	תיאור הפריט כפי שנרשם בהזמנה.
BinAbs	int	איתור במחסן (Bin Location)
WhsCode	int	קוד המחסן שממנו יצא הפריט.

במהלך העבודה על בסיס הנתונים, בנינו מודל מבנה נתונים מסוג Snowflake Schema, בו טבלת המרכז היא OITM – טבלת העובדות, אשר מכילה את כלל המידע על הפריטים. סביב טבלה זו יצרנו קשרים עם מספר טבלאות ממד (Dimension Tables) שתומכות בניתוחים שונים לאורך תהליך הרכש והלוגיסטיקה.

במבנה שבנינו, קיימת הבחנה בין טבלאות כותרת (כגון OPOR, OPDN, OIGE) לבין טבלאות שורות (כגון POR1, PDN1, IGE1). טבלאות הכותרת מכילות את פרטי המסמך הכלליים (למשל: מספר מסמך, תאריך, ספק), ואילו טבלאות השורות מכילות את פירוט השורות בכל מסמך – כגון מק"ט, כמות ומחיר. כל טבלת שורות מקושרת גם לטבלת הכותרת שלה דרך השדה DocEntry, וגם לטבלת OITM דרך ItemCode בנוסף, חיברנו את טבלת OITM לטבלת OITB, המכילה את הקטגוריות של כל פריט (ItmsGrpCod), לצורך ביצוע פילוחים לפי קבוצות מוצרים. הקשרים שנבנו הם קשרי יחיד לרבים, דבר שמאפשר לנו לעקוב אחר כל פריט משלב ההזמנה עד לשלב ההגעה למחסן, כך נוכל לחשב בהמשך גם את זמן האספקה שלו.



איור 5-תרשים DSD

לאחר שלב ה־Extract, בו שלפנו את כלל הנתונים הדרושים ממקורות שונים, עברנו לשלב ה-Transform, אשר כלל מספר תהליכי ניקוי ועיבוד נתונים. בשלב זה ביצענו קידוד של ערכים טקסטואליים לשם האחדה, טיפול בערכים חסרים, והשוואת גדלים בין פריטים – תוך נרמול בין נפחים, משקלים ויחידות מידה שונות, על מנת לאפשר אנליזה עקבית ואחידה. כמו כן, בוצעה התאמה בין תאריכים ואירועים מתוך מערכות שונות (הזמנה, קליטה, צריכה, מלאי).

בשלב זה נדרשה גם בנייה של מדד רמות המלאי בפועל לאורך השנים, שכן לא הייתה לנו עמודה מוכנה לכך במקור. לצורך כך ביצענו פעולות אגרגציה (Aggregation) על מספר טבלאות – חיברנו בין טבלאות הקליטה והניפוקים לפי מק"ט ותאריך, וחשבנו את יתרת המלאי ברמה מצטברת לאורך זמן על פי סיכום של כמויות שנכנסו ויצאו מהמחסן. כך בנינו שדה חדש של InventoryBalance, המשקף את רמות המלאי בפועל בכל נקודת זמן.

במהלך בניית הטבלה המאוחדת, נתקלנו גם בדילמה מבנית: האם לבנות את הטבלה כך שלכל חודש תהיה עמודה נפרדת (כלומר: עמודות בשם Consumption_Jan22, Consumption_Feb22 וכו'), או לחלופין – להשתמש בעמודה אחת של תאריך (ConsumptionDate) ובשורה נפרדת לכל חודש של צריכה עבור כל פריט. לאחר עיון במקורות מקצועיים וסקירת שיטות עבודה נפוצות באנליזה ותחזיות, בחרנו באפשרות השנייה – מבנה ארוך (long format), שבו כל שורה מייצגת את הצריכה של פריט מסוים בחודש מסוים. מבנה זה מאפשר גמישות רבה יותר בניתוחים, חישובי ממוצעים עונתיים, וכן התאמה נוחה למודלים סטטיסטיים ולמידת מכונה.

בסיום שלב ה־Transform, עברנו לשלב ה־Load, ובו טענו את כל המידע המאוחד והמעובד לטבלה אחת מסכמת. טבלה זו שימשה אותנו להמשך שלבי התחזית והאנליזה, והיא כוללת את כלל השדות הרלוונטיים – החל ממק"ט ומחיר, דרך זמני אספקה ונתוני קליטה, ועד רמות מינימום-מקסימום וכמות המלאי לאורך השנים.

טבלה 9-בסיס נתונים סופי

Column Name	Data Type	Description
ItemCode	nvarchar(50)	קוד הפריט (מק"ט) הייחודי למוצר.
PRICE	numeric(19,6)	מחיר היחידה של הפריט בעת הרכישה.
PurchaseOrderNum	int	מספר הזמנת הרכש כפי שנרשם במערכת.
PurchaseOrderDate	datetime	תאריך יצירת הזמנת הרכש.
GoodsReceiptDate	datetime	תאריך קליטת הסחורה בפועל במחסן.
LT_Days	int	זמן אספקה בימים (Lead Time) בין ההזמנה לקבלה.
ConsumptionDate	datetime	תאריך הצריכה בפועל של הפריט.
ConsumptionQty	numeric(19,6)	כמות שנצרכה בפועל בתאריך הנתון.
InventoryDocDate	datetime	תאריך מסמך תנועת המלאי (ניפוק/קבלה).
InQty	numeric(19,6)	כמות פריטים שהתקבלה למלאי באותו היום.
OutQty	numeric(19,6)	כמות פריטים שיצאה מהמלאי באותו היום.
MinLevel	int	רמת המלאי המינימלית המוגדרת לפריט (סף תחתון).
MaxLevel	int	רמת המלאי המקסימלית המוגדרת לפריט (סף עליון).
InventoryBalance	numeric(19,6)	יתרת המלאי בפועל לאחר כל פעולה (כניסה או יציאה).

שלב ב-בינה עסקית

בשלב הזה, רצינו לעסוק בניתוח עסקי של המצב הקיים, באמצעות כלי BI. עיקר המטרה היא להמחיש לחברה כמה המצב חמור, לנסות להמחיש את כמות הפרוייקטים בהם יש חריגה ואת המשמעות הכלכלית של החריגות.

בינה עסקית (Business Intelligence) היינו כלי חיוני עבור קבלת החלטות מבוססות נתונים, במיוחד בסביבות עסקיות המאופיינות בשינויים תכופים ובמורכבות תפעולית. מערכות BI מאפשרות לסיוון לאסוף, לאחד ולנתח כמויות גדולות של מידע תפעולי, פיננסי ולוגיסטי, ובכך להפיק תובנות מבוססות על נתוני אמת. בדשבורד שנציג במשך, נעשה שימוש בכלים ויזואליים כמו טבלאות וגרפים לזיהוי חריגות במלאי ביחס לרמות סף שהוגדרו מראש. בעזרת ויזואליות אלו, ניתן יהיה לאתר בקלות פריטים החורגים מגבולות המינימום והמקסימום, דבר המאפשר תגובה מהירה לצורך תכנון חכם של רמות מלאי, צמצום בזבוז והפחתת עלויות אחסון מיותרות.

יתרה מכך, הבינה העסקית תורמת לא רק לזיהוי בעיות אלא גם למניעתן על ידי מתן תובנות פרואקטיביות. בדוגמה מתוך הדשבורד שנציג, ניתן לראות כיצד ניתוח פשוט של חריגות מלאי לפי קוד מוצר מסייע למנהלים לזהות סיכונים מידיים חוסרים שעלולים להוביל לעיכוב בייצור, או עודפים המחייבים אחסון חיצוני. מעבר לכך, הממשק הוויזואלי מאפשר לבעלי תפקידים שאינם אנליסטים להבין את הנתונים במהירות, ולהתמקד בפריטים הקריטיים ביותר מבחינה תפעולית וכלכלית. בדרך זו, BI הופכת להיות נדבך מרכזי במעבר מהתנהלות תגובתית להתנהלות אסטרטגית מבוססת נתונים.

התהליך המלא מבוסס על שרשרת טכנולוגית הנתמכת בכלים מגוונים, אך לבסוף מאפשרת למקבלי החלטות גישה מהירה, נגישה ומהימנה לנתונים לטובת תגובה מהירה, חיזוי מגמות ושיפור תהליכים עסקיים. בדומה לדשבורד שנציג בהמשך, השילוב בין איסוף נתונים נכון, ניתוח איכותי והצגה גרפית יעילה יוצר מערכת שלמה המאפשרת זיהוי חריגות, ניהול משאבים מדויק והובלת הארגון להחלטות מונחות מידע.

המחשת היתרונות של תהליך הבינה העסקית ניכרת היטב בדשבורד שנבנה לצורך ניתוח חריגות מלאי. הדשבורד מחולק לשני חלקים מרכזיים טבלה מספרית ותרשים עמודות חזותי. בטבלה מופיעים נתוני מלאי עבור פריטי אופטיקה שונים, כולל ערכי מלאי נוכחיים, גבולות מינימום ומקסימום, מחירי יחידה, ועמודה שמציגה את רמת החריגה מהמגבלות שהוגדרו. מתחת לטבלה מופיע תרשים עמודות צבעוני שממחיש באופן אינטואיטיבי את סטיית הפריטים מהטווחים הרצויים עמודות בצהוב עבור פריטים החורגים כלפי מעלה מהמקסימום, ואדום עבור פריטים הנמצאים מתחת לרמת המינימום.

במונחים ארגוניים, חריגה מהמינימום עלולה להצביע על סיכון להפרעה בתהליך הייצור או מתן השירות, ואילו חריגה מהמקסימום מצביעה על החזקת מלאי עודף הכרוכה בעלויות אחסון מיותרות ובשחיקת ערך. תובנות אלה אינן ניתנות לזיהוי אינטואיטיבי ממסדי נתונים גולמיים, אך הופכות לנגישות וברורות הודות לתהליך BI מדויק. הדשבורד מאפשר למנהלי הרכש, התפעול והכספים לקבל החלטות מושכלות בזמן אמת כגון הפחתת כמות ההזמנה לפריטים עם עודף, או תגבור אספקה לפריטים החורגים כלפי מטה.

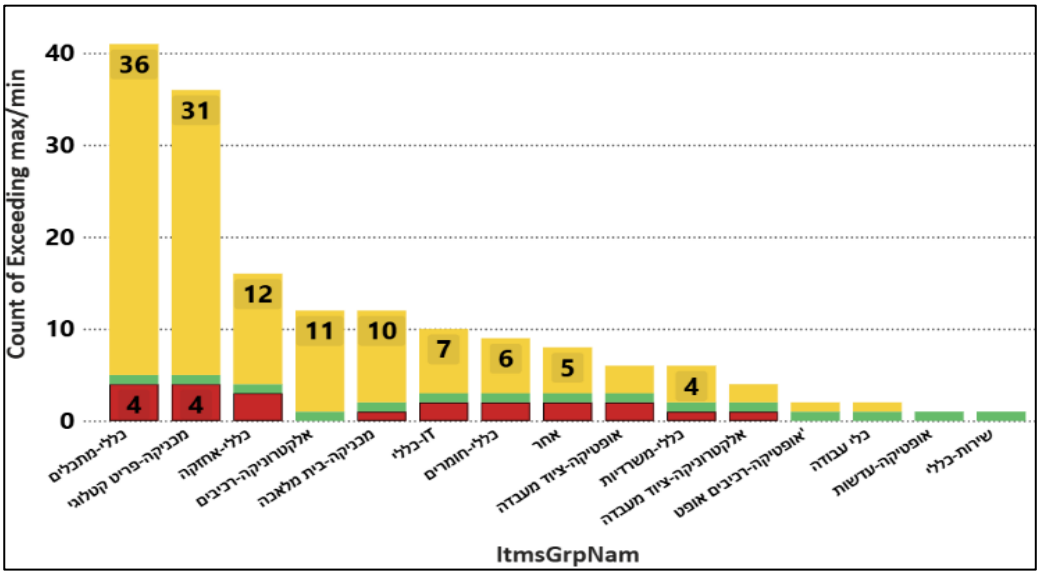
גליון ראשון-תיאור מצב קיים עבור כלל הפריטים

הבדיקה הראשונה שעשינו, היא כמה מכלל הפריטים חורגים מהגבולות המוגדרים להם.

כמו שהצגנו באיור מס' 2, 55% מכלל הפריטים עומדים בגבולות שלהם, ואילו 45% חורגים- מתוכם 40% חורגים מעל הגבול המקסימלי ו5% חורגים מהגבול המינימלי.

הוספנו לבדוק, האם החריגה שייכת לקטגוריה מסוימת, ואז ניתן לשייך סיבה מסוימת לחריגות(מחירים, זמני אספקה לקטגוריה זו וכו'), או שהחריגות נכונות לחלק נכבד מהקטגוריות.

גילינו, שכמעט בכל הקטגוריות יש חריגה,(הרוב חריגות מהמקסימום), כך שניתן להבין שבאופן רוחבי המצב אינו מנוהל כמו שצריך, ולא בגלל איזו סיבה מסוימת הקשורה לסוג הפריט.



איור 6-חריגה מהגבולות,לפי קטגוריה

כמו שציינו, בהתחלה חשבנו לעסוק באופן ישיר בכל הפריטים שנמצאים בכל דו"ח המינימום-מקסימום. לאחר שביצענו את הניתוח הכללי, הבנו שכדי להצליח ולבנות מודל שיבין דפוסי 'התנהגות' של צריכה והזמנה, עלינו להתמקד בקבוצת פריטים מסוימת ,ולאחר התייעצות עם המנחה מר אורי גלבוע ועם אנשי החברה, **החלטנו להתמקד בקבוצת הפריטים האופטיים**, מכמה סיבות: קבוצת פריטים זו היא יקרה , ולכן ממילא החשיבות שלה עולה. סיבה נוספת היא החיוניות של אותם פריטים – אלה פריטים נצרכים ברמה דחופה וקריטית, וחיסרון שלהם יהיה משמעותי מאוד. בנוסף, אלה פריטים משומשים בצורה יחסית תדירה לשאר הפריטים בדו"ח ולכן האמנו שנוכל לייצר להם תחזית ברמה אמינה ומדוייקת יותר מאשר פריטים אחרים בדו"ח או אם היינו מנסים לטפל בכל הפריטים במודל אחד.

גליון שני-תיאור מצב קיים;אופטיקות

לאחר שראינו את הפילוח הראשוני של כמות הפריטים שעומדים בגבולות בנקודת זמן מסוימת וגם את אחוז הזמן בשבועותשבו הפריטים בקטגורייה זו היו בין הגבולות, כמובא באיורים 3 ו-4, רצינו להכנס יותר לעומק של קטגורייה זו ולהבין את המאפיינים של החריגות שלה.

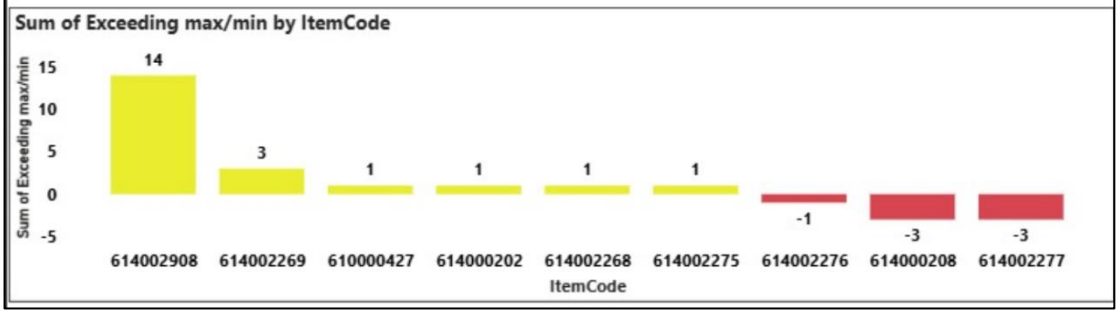
דשבורד זה מחולק לשני רכיבים ויזואליים עיקריים : טבלה ותרשים עמודות. הטבלה העליונה מציגה מידע פרטני עבור פריטי אופטיקה שונים, כאשר כל שורה מייצגת פריט בודד לפי קוד המוצר (ItemCode). לצד כל פריט מוצגים כמות במלאי הנוכחית (OnHand) ,רמות הסף שנקבעו מראש מינימום (MinLevel) ומקסימום (MaxLevel),מחיר היחידה (Price unit) ,ועמודת חישוב (Sum of Exceeding max/min) המבטאת את החריגה של הפריט מגבולות הסף כלומר, כמה פריטים יש מעבר למקסימום או מתחת למינימום.

מתחת לטבלה מוצג תרשים עמודות המשקף באופן ויזואלי את עמודת החריגה. עמודות בצבע צהוב מייצגות פריטים שהכמות שלהם במלאי גבוהה מהגבול העליון שהוגדר (עודף מלאי), עמודות בצבע אדום מייצגות פריטים שכמותם נמוכה מהגבול התחתון (חוסר במלאי).

התרשים מאפשר לזהות בקלות חריגות קיצוניות, כגון בפריט 614002908, שבו קיימת חריגה חיובית של 14 יחידות, או לחילופין בפריטים 614002277 ו614002208, שבהם נרשמה חריגה שלילית של 3 יחידות.

חריגה מתחת לרמת המינימום עשויה להוביל לעצירת תהליך הייצור עקב חוסר בזמינות פריט חיוני. לעומת זאת, חריגה מעל רמת המקסימום מצביעה על מלאי עודף, שעשוי לדרוש אחסון חיצוני ולהביא לעלויות תפעוליות מיותרות עבור הארגון. הדשבורד תורם לאיתור מוקדם של חריגות מסוג זה, תומך בקבלת החלטות מהירה ומאפשר תכנון אפקטיבי של מלאי ושרשרת האספקה.

ItemCode	OnHand	MaxLevel	MinLevel	Price unit	Sum of Exceeding max/min
614002908	16.00	2	1	46.19	14.00
614002269	7.00	4	2	1,471.33	3.00
610000427	5.00	4	2	3,120.09	1.00
614000202	31.00	30	15	352.19	1.00
614002268	4.00	3	1	2,151.04	1.00
614002275	4.00	3	2	814.78	1.00
614002276	1.00	3	2	815.00	-1.00
614000208	1.00	8	4	422.04	-3.00
614002277	0.00	6	3	814.74	-3.00



איור 7-חקר מצב קיים כללי;אופטיקות

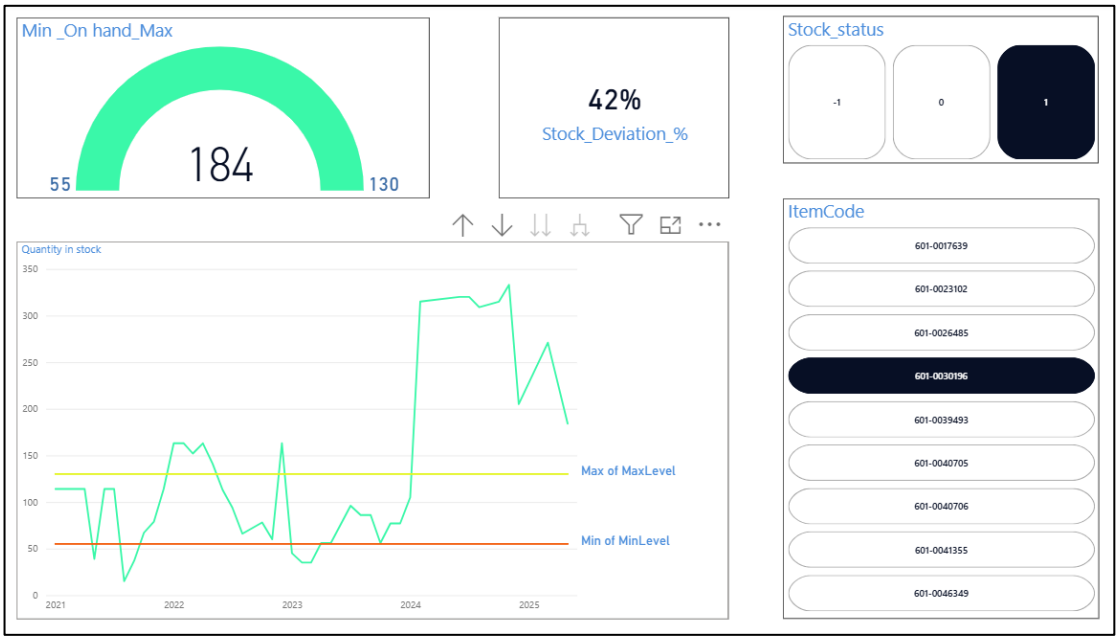
גליון שלישי-ניתוח פריט בודד;אופטיקות

בדשבורד זה מוצגת תצוגה אנליטית ממוקדת עבור פרטי אופטיקה ברמת פריט בודד. במרכז הדשבורד ניתן לראות גרף קווי המתאר את התפתחות כמות המלאי של הפריט לאורך זמן, החל משנת 2021 ועד אמצע 2025. הגרף מאפשר לבחון האם הכמות שנשמרה בפועל לאורך התקופה עמדה בגבולות הסף שנקבעו מראש גבול תחתון (מינימום) וגבול עליון (מקסימום) אשר מוצגים בקווים ישרים בצבעים בולטים (אדום וצהוב בהתאמה). ניתוח זה מסייע לזהות מגמות של חוסר מלאי או עודף מלאי לאורך זמן, ולקבל החלטות בהתאם.

בחלקו השמאלי העליון של הדשבורד מוצג מד מדידה אינטראקטיבי המציג את רמות הסף שנקבעו (Min / Max) לצד הכמות הקיימת בפועל בפריט (184 יחידות נכון לעת הצפייה). מד זה מאפשר לראות במבט אחד האם הפריט נמצא כיום בתוך גבולות הסף או חורג מהם.

בצידו הימני העליון של הדשבורד מוצג מד נוסף המציג את סטיית המלאי באחוזים (Stock Deviation %) המבטאת את רמת החריגה של הכמות בפועל ביחס לטווח התקני שנקבע.

לצד המדדים הכמותיים, מופיעים שני סלייסרים (מסננים אינטראקטיביים) חשובים לבחירת פריט (ItemCode) המאפשרת למשתמש לבחור את קוד המוצר הרצוי ולבחון את נתוניו באופן פרטני. סטטוס המלאי (Stock_status) המאפשר לסנן לפי מיקום הפריט ביחס לגבולות הסף: חוסר מלאי (-1), מלאי תקין (0), או עודף מלאי(1).



איור 8-חקר מצב פריט בודד;אופטיקות

שלב ג-בינה מלאכותית ובניית מודלי חיזוי

לאחר שבנינו את בסיס הנתונים הסופי והצלחנו למפות את עומק הבעיה הקיימת, עברנו לשלב הבא – בניית מודלי חיזוי אשר יאפשרו לנו להעריך את צריכת החברה עבור קבוצת הפריטים שנבחרה. מטרה זו דרשה גישה מתקדמת, מבוססת בינה מלאכותית, ולא כלים סטנדרטיים או סטטיסטיים בסיסיים.

בעולם ניהול המלאי והרכש, תחזית מדויקת היא תנאי קריטי להצלחה – נדרש לחזות לא רק את הביקוש עצמו, אלא גם את השפעות השוק, עיכובים פוטנציאליים באספקה, ועונתיות. שיטות מסורתיות אינן מספקות עוד את הגמישות והדיוק הנדרשים כדי להתמודד עם המורכבות הזו. לשם כך, שילבנו תהליכים מבוססי בינה מלאכותית ולמידת מכונה **(AI+ML)**, המאפשרים ניתוח של כמויות נתונים גדולות, זיהוי תבניות מורכבות ויצירת תחזיות שמסתגלות לתנאים משתנים.

היתרון המרכזי בגישה זו טמון ביכולת שלה לספק תובנות חכמות ופעולות מונחות נתונים, שמקדמות תכנון מושכל יותר, הפחתת עלויות תפעוליות ושיפור ביכולת להגיב לשינויים בלתי צפויים. תחזיות אלו לא עומדות בפני עצמן – הן מהוות תשתית לקבלת החלטות אופטימלית בניהול המלאי, בתזמון ההזמנות ובהקצאת המשאבים, דבר שיכול לגרום להפחתת עלויות בשל אחסון עודף וכן למנוע את עצירת קו הייצור בשל חוסרים.

הנתונים שנאספו מתעדים צריכה בפועל בין השנים 2019 ל-2024. לצורך אימון המודלים, הנתונים חולקו לסט אימון (Train Set) עבור השנים 2019–2023 ולסט בדיקה (Test Set) עבור שנת 2024. מאחר ומדובר בסדרות זמן, בוצע תהליך של Time Series Cross-Validation, המאפשר הערכת ביצועים אמינה לאורך רצף הזמן, בניגוד ל-Cross-Validation רגיל אשר מתעלם מממד הזמן.

לאחר גיבוש בסיס הנתונים הסופי, שכלל את כלל פריטי המלאי, נתוני צריכה היסטוריים, זמני אספקה ופרטים נוספים, התחלנו בתהליך בניית התחזית עבור שנת 2024. תחזית זו התבצעה בשיטה מדורגת, תוך שימוש במספר מודלים מבוססי בינה מלאכותית ולמידת מכונה. כל שלב בתהליך נבנה באופן עקבי עבור כל אחד מהמודלים שנבחנו על מנת לאפשר השוואה הוגנת ואחידה ביניהם. נציין פה שחילקנו ביננו את העבודה על בניית המודלים: נפתלי עבד על מודלים Arima+Random Forest, ואליאור עבד על Prophet+XGBoost. מהלך העבודה כלל את השלבים הבאים:

1. **הכנת משתני הקלט לכל מודל**

   עבור כל אחד מהמודלים, הוזנו משתנים רלוונטיים ממספר תחומים:

   - משתנים תפעוליים: מחיר הפריט (PRICE), זמן אספקה משוער (LT_Days), יתרת מלאי נוכחית (InventoryBalance).
   - משתני זמן: שנה, שבוע צריכה, ולעיתים ייצוגים סינוסיים וקוסינוסיים ללכידת עונתיות.
   - משתנים קטגוריים: מק"ט (ItemCode) ועוד, שהומרו לקידודים מספריים (Encoding) לפי הצורך.

   כל מודל הוזן בסט זהה של נתונים לאחר עיבוד מוקדם (preprocessing) וניקוי נתונים.

2. **חלוקת הנתונים לאימון ובדיקה**

הנתונים חולקו לערכת אימון (Training Set) שכללה נתונים עד לסוף 2023, ולערכת בדיקה (Test Set) שכללה את כל שנת 2024. המטרה הייתה לבדוק האם המודל מסוגל לבצע הכללה (generalization) מתוך ההיסטוריה אל העתיד.

3. **אימון המודלים והפקת התחזיות**

כל מודל אומן על נתוני העבר, והופעל לאחר מכן כדי להפיק תחזית שבועית לצריכה עתידית עבור כל פריט לשנת 2024. התהליך בוצע עבור כל הפריטים, כך שלכל פריט הייתה תחזית נפרדת.

4. **בחינת התחזית הכללית עבור כלל הפריטים**

בשלב ראשון נבחנה איכות התחזית ברמה מצרפית – כלומר, ניתוח כולל של תחזית מול צריכה בפועל לכלל המק"טים לאורך השנה. השוואה זו כללה תרשים מגמה כללי של צריכה בפועל מול תחזית שבועית, כדי לבחון האם המודל מצליח לשקף את מגמות העומק של הארגון לאורך זמן.

5. **חישוב מדדי שגיאה כוללים**

עבור כל מודל חושבו מדדי שגיאה מצרפיים על כלל התחזיות, בהם MAE (שגיאה מוחלטת ממוצעת), RMSE (שורש ממוצע ריבועי השגיאות) וכן R^2 (מקדם ההסבר), אשר מודד את יכולת המודל להסביר את השונות בנתונים. שלב זה סייע להעריך את הדיוק הכללי של כל מודל ואת יכולתו ללכוד מגמות משמעותיות בצריכה.

6. **ניתוח פיזור השגיאות**

לאחר ניתוח הממוצע, בוצע ניתוח גרפי של השגיאות. נעשה שימוש ב:

- תרשים פיזור (Predicted vs Actual) לבחינת ההתאמה בין התחזיות לבין הערכים האמיתיים.
- תרשים התפלגות השגיאה (Error Distribution) לבחינת מבנה השגיאות – האם הן מרוכזות סביב אפס, האם קיימת הטיה (bias), ומהו טווח השגיאות בפועל.

7. **ניתוח ברמת פריט בודד**

לאחר הניתוח הכללי, בוצע קידוד של כל אחד מהמודלים גם ברמת פריט יחיד. עבור מק"ט ספציפי, הוצגה השוואה בין צריכה בפועל לתחזית שבועית לאורך כל שנת 2024. בנוסף, חושבו גם מדדי שגיאה פרטניים (R^2,MAE, RMSE), וכן נבחנו גרפי הפיזור והשגיאה עבור אותו פריט, במטרה להבין את התנהגות המודל ברזולוציה פרטנית.

8. **השוואת המודלים והפקת מסקנות**

לבסוף, נאספו כלל הממצאים שנוצרו בכל שלב: התחזיות, השגיאות, מגמות הצריכה ויכולות הזיהוי של דפוסים חריגים או עונתיים – ונבחנו כבסיס לבחירת המודל המוביל. ההשוואה בוצעה הן מבחינה כמותית (מדדי שגיאה), והן איכותית (מגמות, קיצוניות, התאמה תפעולית).

שלבים אלו חזרו על עצמם עבור כל אחד מהמודלים שנבחנו. בכך נבנה מערך השוואתי מקיף, שהניח את הקרקע לניתוח הממצאים בפרק הבא.

שלב ד'-בניית מודל מלאי

לאחר שסיימנו את שלב התחזיות, בחרנו את המודל בו היו שגיאות המדדים הטובות ביותר, כפי שיפורט בפרק *'תוצאות'*, עברנו לשלב הבא בפרויקט – **בניית מודל לניהול מלאי** שמבוסס על התחזיות שהופקו. מודל זה נבנה על גבי תחזיות שבועיות לכל פריט (SKU), ומטרתו לתרגם את התחזיות לפעולות אוטומטיות של הזמנות, תוך שמירה על רמות מלאי תקינות ומניעת חוסרים או עודפים.

בחרנו ליישם את **שיטת Time Cover + Safety Stock**, שיטה המתאימה במיוחד למצבים בהם קיים מגוון רחב של פריטים עם דפוסי צריכה משתנים לאורך זמן. שיטה זו מבוססת על חישוב דינמי של נקודת ההזמנה והגבולות עבור כל פריט, בהתאם לצפי הביקוש האישי של אותו פריט.

האלגוריתם כולל מספר שלבים מרכזיים:

1. **הכנסת תחזית צריכה שבועית** לכל פריט (על בסיס המודל שנבחר).
2. **חישוב זמן האספקה (Lead Time)** בפועל לפי נתוני עבר – בממוצע של הזמן שעובר בין הזמנת פריט לקבלתו-מבוסס על סמך נתוני העבר שהוצאנו עבור כל פריט חישבנו את הזמן מרגע שיצאה ההזמנה ועד לשלב הגעתה למחסן החברה.
3. **קביעת כמות ההזמנה** לפי משך הכיסוי הרצוי (Cover Time), כלומר: כמה שבועות קדימה אנו רוצים שההזמנה תספיק, בהתאם לצריכה הצפויה..
4. **חישוב מלאי הביטחון (Safety Stock)** לפי הנוסחה:

$$\sqrt{LT} \times \sigma \times Z = SS$$

כאשר:

- Z הוא ערך ה'זי' של רמת השירות הרצויה (למשל, 1.65 עבור 95%)
- σ היא סטיית התקן של הצריכה השבועית
- LT הוא זמן האספקה בשבועות

5. **קביעת נקודת ההזמנה (ROP – Reorder Point):**

ROP =ביקוש חזוי בתקופת האספקה+מלאי ביטחון

בכל פעם שרמת המלאי החזויה יורדת מתחת לנקודה זו – נשלחת המלצה להזמנה חדשה.

במהלך הרצת המודל, **המערכת ממליצה על ביצוע הזמנה (Order)** ברגע שבו נמצא כי לפי התחזית – המלאי עתיד לרדת מתחת לרמת המינימום (MinLevel) **בתום זמן האספקה(LT) .** כלומר, לא מחכים לירידת המלאי בפועל, אלא מזהים מראש מתי הוא עלול לרדת אל מתחת לרמה הקריטית, ויוצאים להזמנה בזמן שיאפשר את הגעת הסחורה בדיוק ברגע הנכון. בהתאם לכך, **מחושבת גם נקודת הקליטה (Arrival)** הצפויה של כל הזמנה, בהתבסס על זמני האספקה ההיסטוריים הייחודיים לכל פריט.

שיטה זו מאפשרת תכנון פרואקטיבי שמונע חוסרים במלאי, שומר על זמינות גבוהה של פריטים קריטיים, ומייעל את ניהול ההזמנות והרכש לאורך זמן.

כחלק מן השינוי שאנחנו מציעים לחברה אנו גם נמליץ בחלק מהמקרים לשנות את הגבולות לפריטים, שכן גם אנשי החברה אמרו בעצמם שאין איזו נוסחה מתמטית שעל פיה בנו את הגבולות ולכן אנו גם באים לבדוק אם יש מקום לשפר את הגדרת הגבולות הקיימים.כדי שנוכל לשמור על הפריטים שיהיו כמה שיותר בין גבולות המינימום והמקסימום ולא יחרגו,קבענו **גבולות מלאי דינמיים** עבור כל פריט – גבול תחתון (MinLevel) וגבול עליון (MaxLevel) , שנועדו לוודא שרמות המלאי נשמרות בטווח תקין. הגבולות חושבו לפי תחזיות הצריכה ואופיינו כך :

- אם לאורך זמן המלאי נמצא פחות מ־80% מהשבועות בתוך התחום (בין מינימום למקסימום), הגבולות מתעדכנים אוטומטית.
- רמת השינוי המותרת נקבעה ל־15%± – כלומר, המודל מאפשר התאמה של הגבולות אם רמת התחזית משתנה בצורה עקבית, אך מונע קפיצות חדות שאינן מוצדקות.

שיטה זו עונה על הצורך של החברה לשפר את המצב הקיים בשמירה על הפריטים בין הגבולות המוגדרים. בנוסף, היא מאפשרת **שילוב אוטומטי בין תחזית מתקדמת לבין יישום פרקטי של ניהול מלאי** ,ומביאה לתכנון רכש יעיל, שמירה על זמינות מוצר גבוהה, והפחתת עלויות של מלאי עודף או חוסרים.

שלב ה׳-בניית דשבורד

לאחר שסיימנו את שלבי התחזית ובניית מודל ניהול המלאי, ביקשנו לגבש דרך אפקטיבית להעברת המידע והכלים לחברה, כך שניתן יהיה ליישם בפועל את מדיניות הרכש שהוגדרה בהתבסס על המודלים שפיתחנו. לצורך כך, זיהינו את הצורך בפלטפורמה אינטראקטיבית שתאפשר נגישות, בהירות והטמעה פשוטה גם עבור גורמים שאינם עוסקים ישירות באנליזה או בתחזיות.

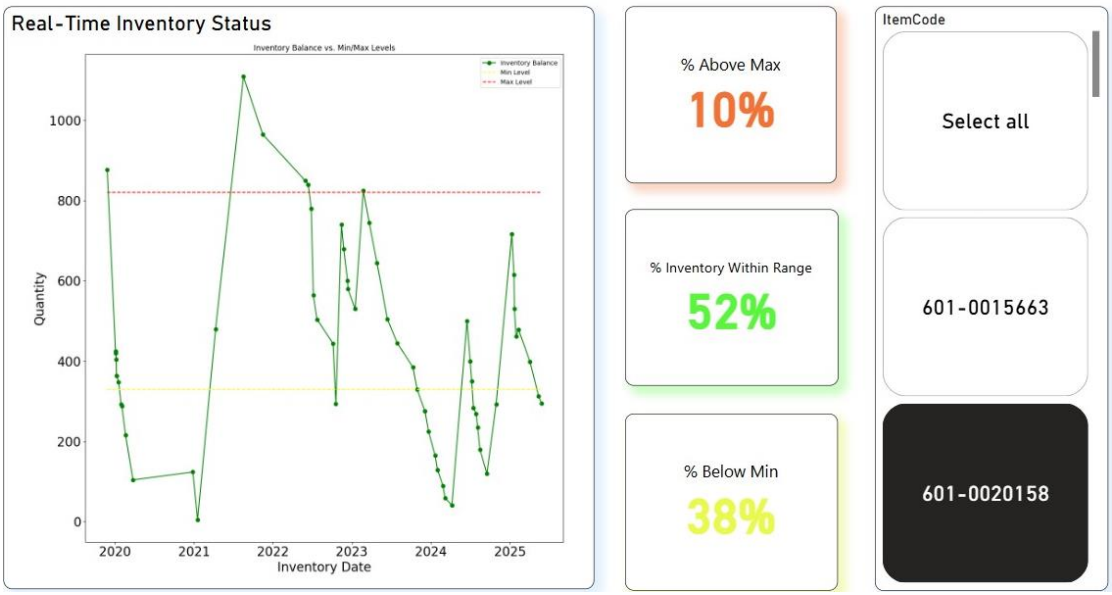
בחרנו להקים עבור החברה דשבורד ב־Power BI, מתוך מטרה להנגיש את תוצרי המודלים באופן ויזואלי, אינטואיטיבי וממוקד, תוך הדגשת התובנות המרכזיות והמלצות הרכש שנגזרות מהתחזיות. בשלב הראשוני, טענו את בסיס הנתונים לתוך הדשבורד והוספנו עליו שכבת קוד בפייתון, אשר מיישמת את לוגיקת התחזית וניהול המלאי המלאה. בהמשך, מתוכנן חיבור ישיר של המערכת לממשק ה־ERPשל הארגון, כך שהדשבורד יתעדכן באופן שוטף ויהפוך לחלק אינטגרלי מתהליך העבודה הקבוע של החברה.

בנינו בדשבורד שלנו מספר גליונות:

גליון ראשון-ניתוח עמידה בין הגבולות

גליון זה נועד לספק מבט היסטורי מקיף עבור כל פריט, במטרה לבחון כיצד התנהג המלאי לאורך זמן ביחס לגבולות שנקבעו לו – הן המקוריים והן אלו שהוגדרו מחדש במסגרת המודל שפיתחנו. הגליון מאפשר להבין את דפוסי ניהול המלאי בעבר, להעריך את מידת ההתאמה של הפריט לגבולות שנקבעו, ולזהות מגמות או חריגות חוזרות.

באמצעות הדשבורד ניתן לבצע סינון לפי מק״ט בודד, ולהציג עבורו את פילוח הזמנים – באחוזים – בהם המלאי עמד בתוך הגבולות שהוגדרו לו, לעומת חריגות כלפי מעלה (גבול מקסימלי) או מטה (גבול מינימלי). ניתוח זה מהווה כלי תומך החלטה חשוב, המסייע הן בהבנת המצב הקיים והן בהערכת האפקטיביות של מודל ניהול המלאי שהוטמע.

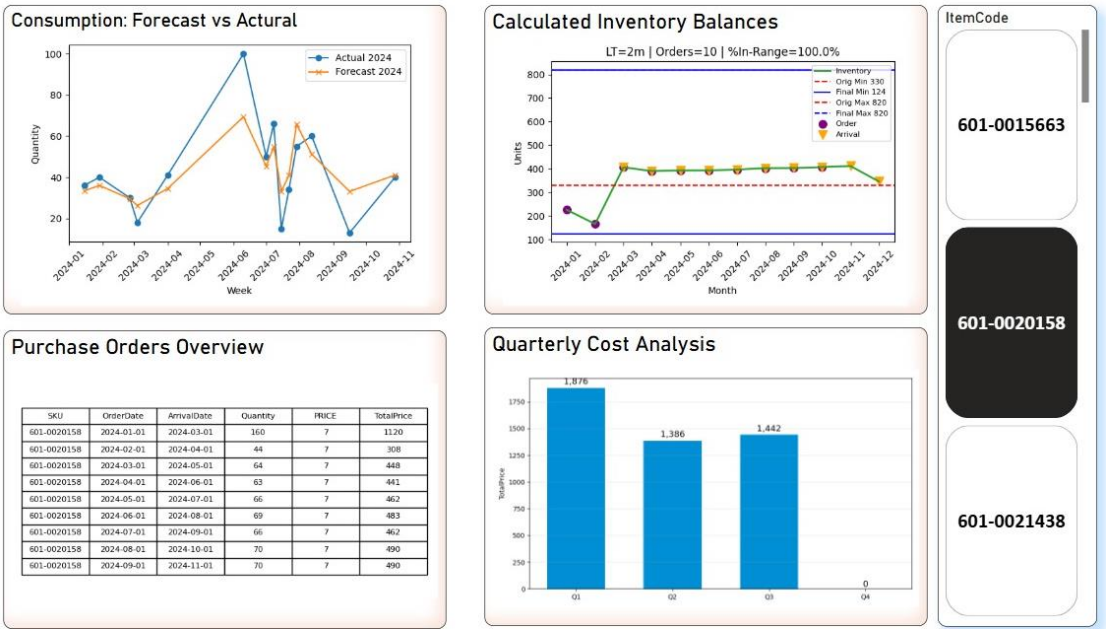


איור 9-דשבורד פילוח אחוזים בגבולות

גליון שני-דשבורד ראשי

בגליון זה, הכנסנו את עיקר המודלים שבנינו.

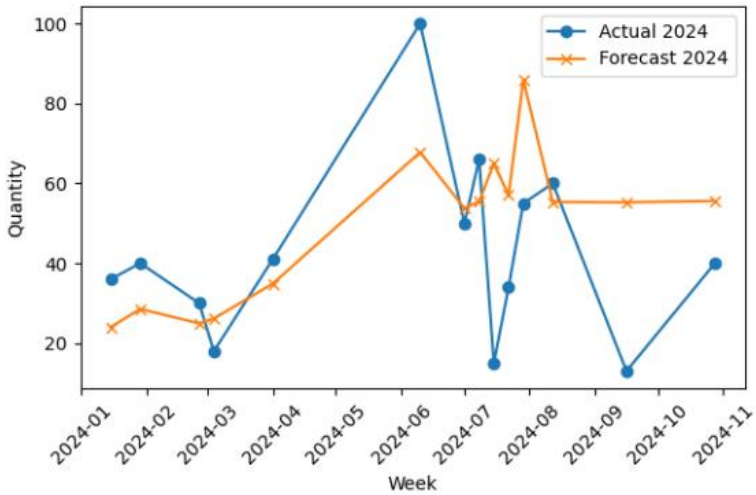
חילקנו את הגליון לארבע ויזואליות שונות, בה כל ויזואליה נותנת ערך אחר.



SKU	OrderDate	ArrivalDate	Quantity	PRICE	TotalPrice
601-0020158	2024-01-01	2024-03-01	160	7	1120
601-0020158	2024-02-01	2024-04-01	44	7	308
601-0020158	2024-03-01	2024-05-01	64	7	448
601-0020158	2024-04-01	2024-06-01	63	7	441
601-0020158	2024-05-01	2024-07-01	66	7	462
601-0020158	2024-06-01	2024-08-01	69	7	483
601-0020158	2024-07-01	2024-09-01	66	7	462
601-0020158	2024-08-01	2024-10-01	70	7	490
601-0020158	2024-09-01	2024-11-01	70	7	490

איור 10-דשבורד ראשי לניהול המלאי

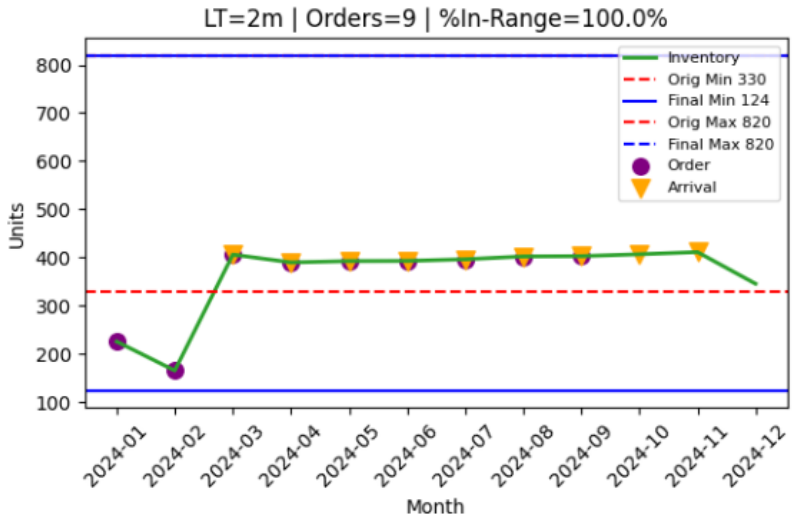
ויזואליה ראשונה-תחזית מול מה שהיה בפועל. לקחנו את המודל אותו בחרנו שישמש אותנו כמודל הנבחר, ואותו הכנסנו לדשבורד כמודל שעליו בונים את כל המדיניות. יש אפשרות לראות לפי כל פריט את תחזית הצריכה שלו לשנה הקרובה, לעומת מה שהיה בפועל. דבר זה מוסיף אמינות למודל, ומראה למשתמש את חיזוי העבר וכך יכול להתרשם מיכולות המודל ולהאמין בו יותר.



איור 11-ויזאוליה ראשונה;תחזית מול בפועל

ויזואליה שנייה-בניית תחזית המלאי. על בסיס תחזית הצריכה, הוספנו כעת את התרשים של ההמלצות להזמנות וההגעה הצפויה שלהן למחסן, על בסיס מודל **Time Cover + Ss**, כפי שהסברנו בחלק הקודם. כמובן שמצויין פה גם הגבולות המומלצים החדשים במידת הצורך.

כמו כן, מצויין מעל כל פריט את אחוז הזמן הצפוי שלו להיות בתוך הגבולות המוגדרים לו.



איור 12 ויזואליה שנייה;מעקב מלאי ונקודות הזמנה

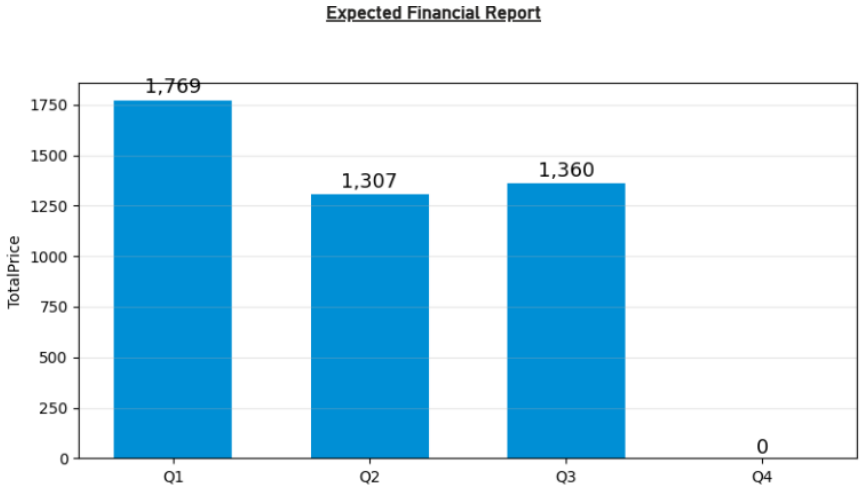
ויזואליה שלישית-טבלת הזמנות מסכמת: ויזואליה זו חשובה מאוד עבור הניהול השגרתי של כלל הפריטים. בטבלה זו מרוכזים עבור כל פריט כלל ההזמנות שצפויות לו לפי המלצת המודל בשנה הקרובה. בטבלה מצויין תאריך ההזמנה, כמות שיש להזמין, תאריך הגעה צפוי לחברה ומחיר של כלל ההזמנה. בעצם בטבלה זו יש את כל מה שנדרש כדי לשלוט באופן קל וברור בהזמנות, ועיקר העבודה עוברת להיות בקרה שאכן ההזמנות יצאו לפעול בלי תקלות, מאשר לתכנן מתי צריך לבצע את ההזמנות, לחשב כמויות נדרשות ועוד.

Orders table

SKU	OrderDate	ArrivalDate	Quantity	PRICE	TotalPrice
601-0020158	2024-01-01	2024-03-01	160	6.599999905	1055.9999848
601-0020158	2024-02-01	2024-04-01	44	6.599999905	290.39999582
601-0020158	2024-03-01	2024-05-01	64	6.599999905	422.39999392
601-0020158	2024-04-01	2024-06-01	63	6.599999905	415.799994015
601-0020158	2024-05-01	2024-07-01	66	6.599999905	435.59999373
601-0020158	2024-06-01	2024-08-01	69	6.599999905	455.399993445
601-0020158	2024-07-01	2024-09-01	66	6.599999905	435.59999373
601-0020158	2024-08-01	2024-10-01	70	6.599999905	461.99999335
601-0020158	2024-09-01	2024-11-01	70	6.599999905	461.99999335

איור 13-ויזואליה שלישית;טבלת הזמנות

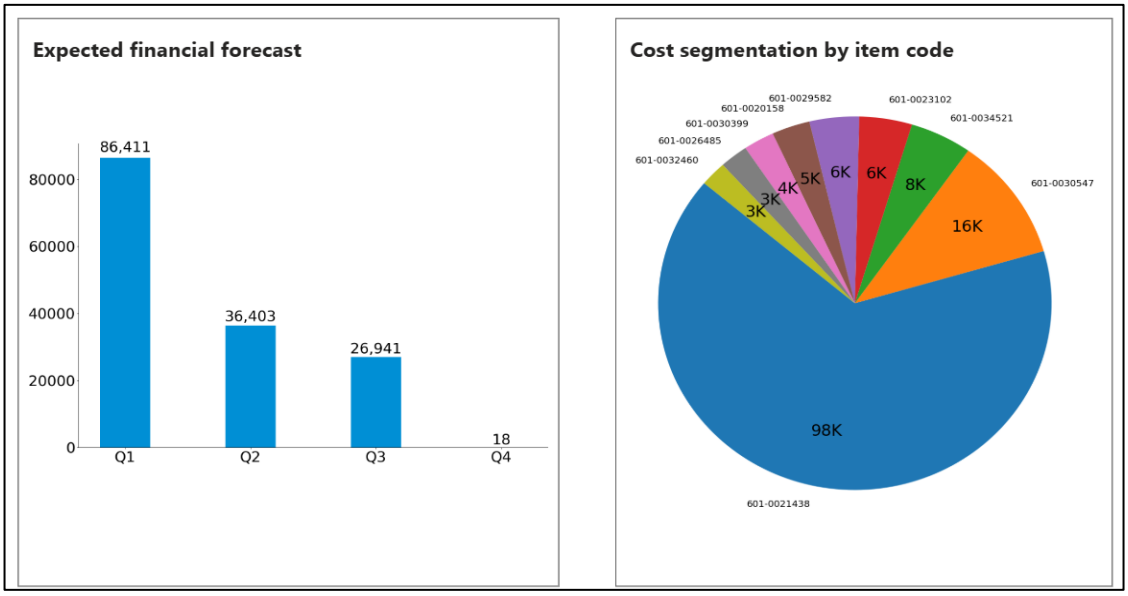
ויזואליה רביעית-סיכום כלכלי לפריט : ויזואליה זו באה לתת מענה להנהלה ולתת פירוט על כל פריט שנבחר, מה יהיו העלויות שלו לשנה הקרובה בחלוקה לפי רבעון. ויזואליה זו תעשה סדר ותציג באופן ברור מה הן ההוצאות הצפויות לכל פריט ולכן נדע להתמקד בפריטים יקרים יותר או פחות , לפי החלטה.



איור 14-ויזואליה רביעית;הוצאות לפי רבעון

גליון שלישי-סיכום כלכלי כולל

גליון זה בא לעשות את הסיכום הכולל של ההוצאות הצפויות לכלל הפריטים, ולתת ניתוחים בנוגע לחלוקה בין כלל הפריטים. בחלק שמאל של הגליון בנינו גרף עמודות מחולק לפי רבעון בו מפורטות העליויות של כלל הפריטים לשנה הקרובה, ובצד ימין של הגליון יש תרשים פאי שיש בו פילוח של כל ההוצאות הצפויות לשנה הקרובה. דבר זה יכול לשמש יסוד לניתוחים אנליטיים ושיפורים נוספים שניתן לעשות-כמו למשל ההבנה מה הפריט היקר ביותר והאם ניתן למצוא לו תחלופות זולות, הבנה האם דווקא הפריטים היקרים ביותר הם הפריטים שמוזמנים הכי הרבה והאם ניתן למצוא להם תחליף ועוד.



איור 15-פילוח הוצאות שנתי

תוצאות

כפי שציינו, עבור כל אחד מן המודלים עשינו את אותם המבחנים ואותם הבדיקות כדי שנוכל להגיע לכדי השוואה טובה ומדוייקת.

לצורך ההשוואה, לקחנו שלושה מדדים והשוונו בין המודלים השונים.

המדדים הם:

Mean Absolute Error (MAE):

מודד את ממוצע הפערים המוחלטים בין התחזיות לערכים בפועל, מבלי להתחשב בכיוון השגיאה. מדד זה מציין בכמה יחידות בממוצע התחזית "טעתה". ככל שהערך נמוך יותר – כך התחזית מדויקת יותר.

Root Mean Squared (RMSE):

מדד דומה ל-MAE, אך מעניק משקל גבוה יותר לשגיאות גדולות (על ידי ריבוע הפערים). לכן הוא רגיש יותר לערכים קיצוניים. ערך גבוה יכול להעיד על מקרים בודדים של תחזיות שגויות במיוחד.

R-squared (R^2):

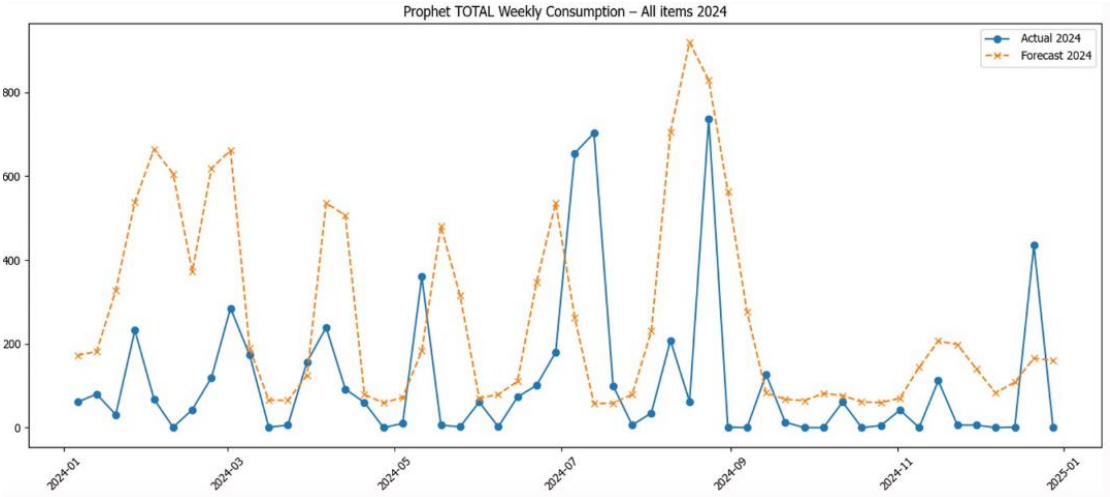
מציין את אחוז השונות בנתונים שהתחזית מצליחה להסביר. ערך של 1 מצביע על תחזית מושלמת, וערך של 0 מעיד שהמודל אינו מצליח להסביר את הנתונים כלל. מדד זה מאפשר להשוות בין מודלים שונים מבחינת איכות ההתאמה.

מודל Prophet

הערכת הביצועים של מודל Prophet בוצעה באמצעות מדדים סטטיסטיים מקובלים והשוואה ויזואלית בין התחזיות לביצועים בפועל. להלן התוצאות:

גרף 1: תחזית מול ביצוע – צריכה שבועית כללית לשנת 2024

בגרף זה מוצגת השוואה בין הביצוע בפועל (בכחול) לבין התחזית (בכתום) לאורך שבועות שנת 2024 עבור כלל הפריטים. ניתן לראות כי המודל מצליח לזהות היטב את המחזורים והעונתיות (בפרט חודשי שיא באמצע השנה), אך יש פערים בעוצמת התחזית – בעיקר בפיקים חריגים. ישנם מקרים בהם התחזית מגי בה מוקדם או מאוחר מהביצוע בפועל, דבר המצביע על מגבלות בזיהוי שינוי מגמה פתאומי.



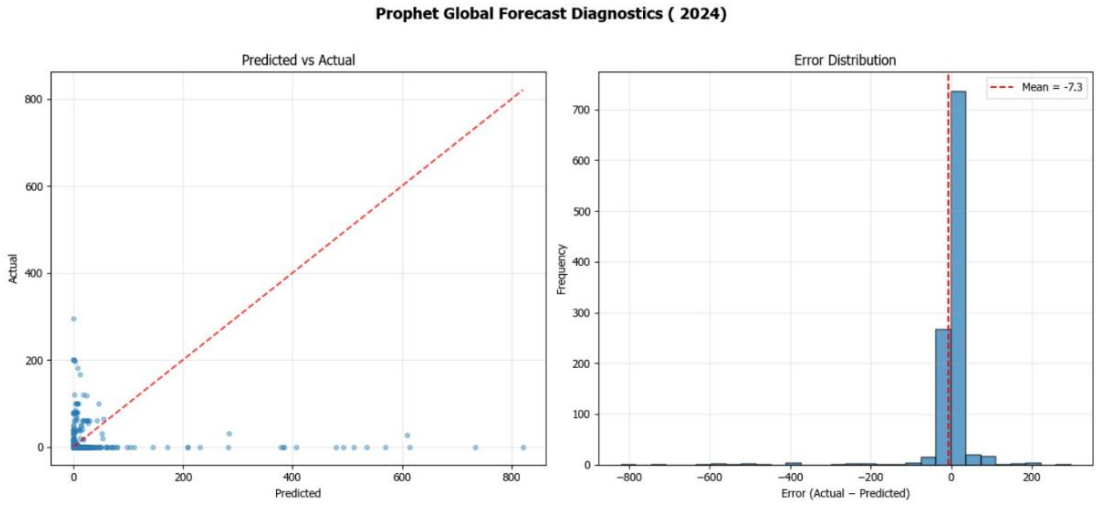
איור 16-Prophet-תחזית מול מציאות;כלל הפריטים

גרף 2- השוואת תחזית לעומת ערכים בפועל +פיזור השגיאות

בגרף השמאלי מוצגת השוואה בין התחזיות של המודל לבין הצריכה בפועל עבור כלל הפריטים. הקו האדום המקווקו מייצג את הקו האידאלי שבו תחזית הייתה זהה בדיוק לערך בפועל. ככל שהנקודות קרובות יותר לקו – כך התחזית מדויקת יותר. מהגרף ניתן לראות שרוב התחזיות מתרכזות בצריכות נמוכות יחסית, עם מספר תחזיות קיצון שלא קלעו לערכים בפועל. זה מצביע על מגבלה בדיוק בתחזיות עבור ערכים קיצוניים.

בגרף הימני מוצגת התפלגות השגיאות – כלומר, הפער בין התחזית לבין הערך בפועל. ניתן לראות שהרוב המוחלט של השגיאות מתרכז סביב האפס, **עם סטייה ממוצעת של כ-7.3- יחידות**, מה

שמצביע על כך שהמודל נוטה במעט להערכת חסר. זהו סימן לכך שהמודל מדויק יחסית אך עדיין קיים שיפור נדרש בתחום תחזיות-היתר עבור צריכות גבוהות .



איור 17 Prophet -פיזורי השגיאות; כלל הפריטים

גרף 3-תחזית לפריט בודד

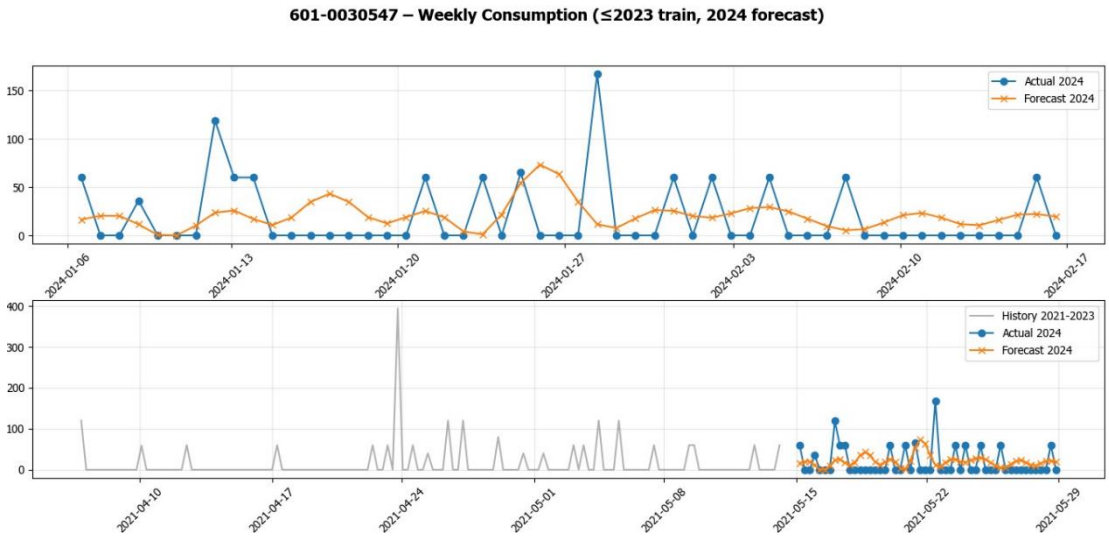
בשלב זה בחנו לעומק את דיוק התחזיות שהתקבלו ממודל התחזית הנבחר על בסיס נתוני ההיסטוריה עד סוף 2023, ותחזית עבור שנת 2024.
הבקרה בוצעה הן ברמה גרפית והן ברמה סטטיסטית :

- **תרשים התחזית מול בפועל (למעלה)** מציג את השוואת הצריכה החזויה לעומת הצריכה שהתבצעה בפועל במהלך 2024. ניתן לראות כי המודל מצליח לשקף בצורה סבירה את תנודות הביקוש – תוך נטייה קלה להחמצת קפיצות חריגות – אך כן מתאר את המגמות והעונתיות המרכזית.

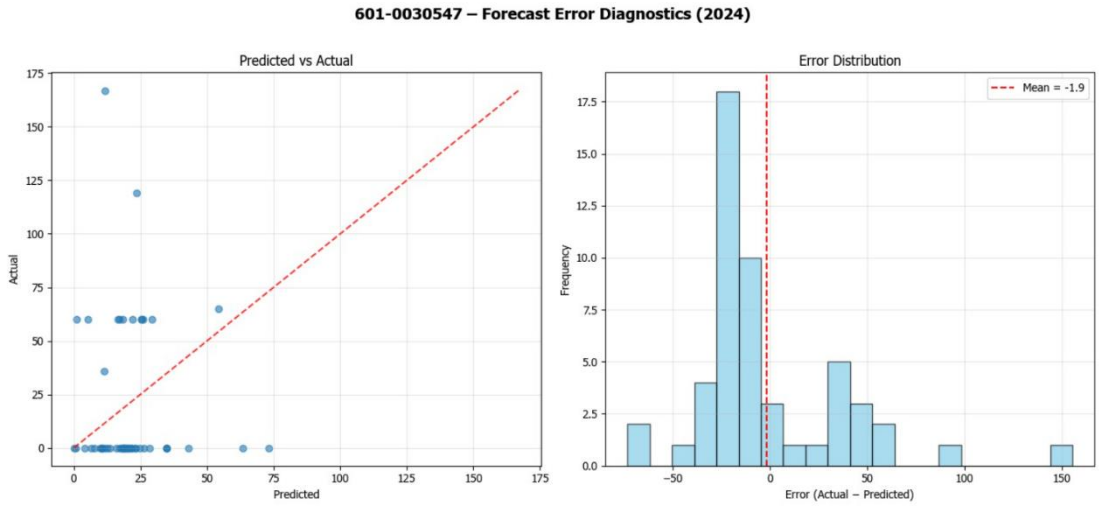
- **תרשים הפיזור (Predicted vs Actual)** ממחיש את איכות התחזית עבור כל תצפית. נקודות שהתקרבו לקו האדום מייצגות תחזיות מדויקות יותר, בעוד סטייה מהקו מסמלת שגיאה.

- **היסטוגרמת שגיאות (Error Distribution)** מצביעה על כך שפיזור הטעויות הוא יחסית סימטרי סביב 0, עם ממוצע שגיאה נמוך מאוד (כ־1.9 יחידות בלבד).
המשמעות : המודל אינו מוטה באופן עקבי כלפי מעלה או מטה, אלא טועה לעיתים לשני הכיוונים באופן מאוזן – תכונה מבורכת במודל חיזוי.

- **תרשים צריכה שבועית היסטורית** כולל חלוקה לשנים 2021–2023, ומראה כי המודל מצליח ללמוד תבניות עונתיות ולטפל ברמות רעש גבוהות יחסית.



איור 18-Prophet- תחזית לפריט הבודד



איור 19-Prophet-פיזור מול שגיאה למק"ט הבודד

מדדי שגיאות עבור כלל הפריטים למודל Prophet:

MAE =16.37
RMSE =67.02
R^2=0.3

מודל XGBoost

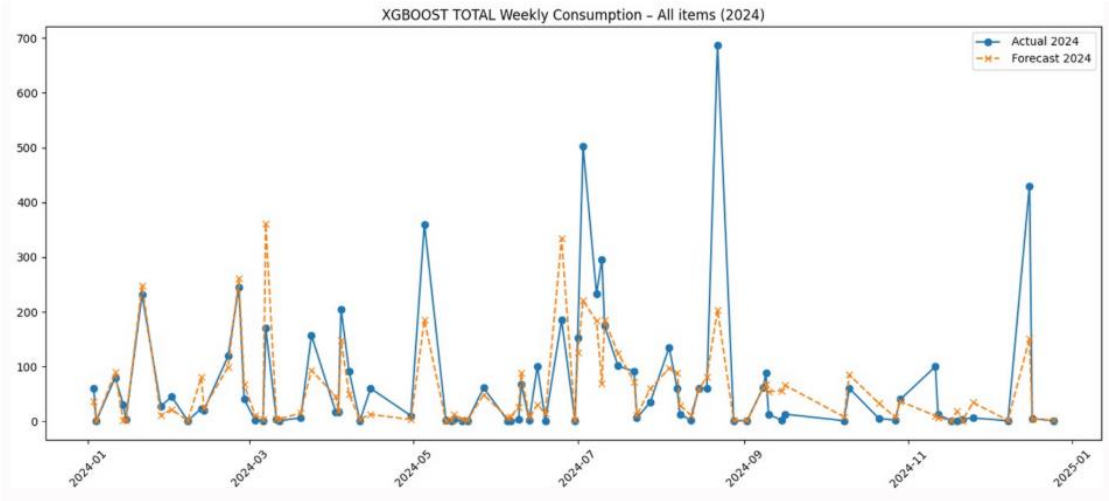
גרף 1 – תחזית מול צריכה בפועל

בתרשים זה מוצגת ההשוואה בין תחזית הצריכה השבועית של המודל לבין הצריכה בפועל של כלל הפריטים בשנת 2024.

הקו הכחול מייצג את הצריכה האמיתית (Actual 2024).

הקו הכתום מייצג את התחזית של המודל (Forecast 2024).

מהתרשים ניתן לראות שהמודל מצליח לזהות היטב את הדפוסים המרכזיים, כולל תקופות של שיאי צריכה (כגון בחודשי הקיץ) וירידות בביקוש. התחזית עקבית, קרובה מאוד למגמות בפועל, עם הבדלים נקודתיים בעיקר בשבועות שבהם התרחשו קפיצות לא שגרתיות בצריכה.



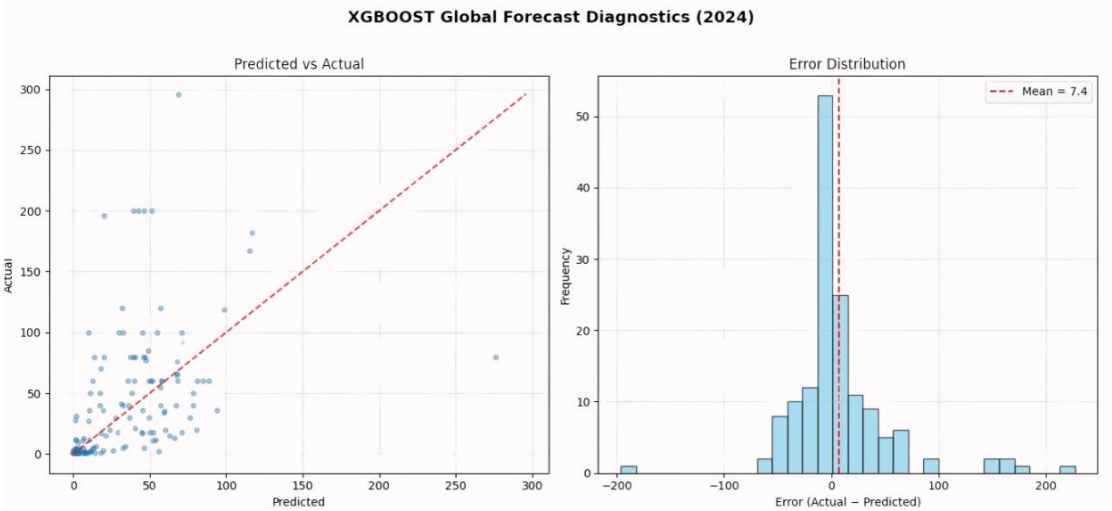
איור 20-XGBoost-תחזית מול מציאות;כלל הפריטים

גרף 2- השוואת תחזית לעומת ערכים בפועל +פיזור השגיאות

בתרשים זה ניתן לראות שתי תצוגות משלימות :

- **הגרף השמאלי: (Predicted vs Actual)**
מציג את הקשר בין התחזיות לבין הערכים בפועל. כל נקודה מייצגת זוג של תחזית מול תוצאה אמיתית. ריבוי נקודות בצמוד לקו האדום המקווקו (קו האידאל בו תחזית = אמת) מעיד על תחזיות מדויקות. ניתן לראות שרוב התחזיות אכן מרוכזות סביב הקו, בעיקר בטווחים הנמוכים והבינוניים של צריכה.

- **הגרף הימני: (Error Distribution)**
מציג את התפלגות השגיאות (כלומר, Actual – Predicted)
ההתפלגות מרוכזת סביב אפס, בצורה סימטרית יחסית, עם ממוצע שגיאה קטן של 7.4 מבנה זה מעיד על כך שהמודל אינו סובל מהטיה שיטתית (לא חוזה גבוה או נמוך באופן עקבי).



איור 21-XGBoost- פיזורי השגיאות; כלל הפריטים

גרף 3-תחזית לפריט בודד

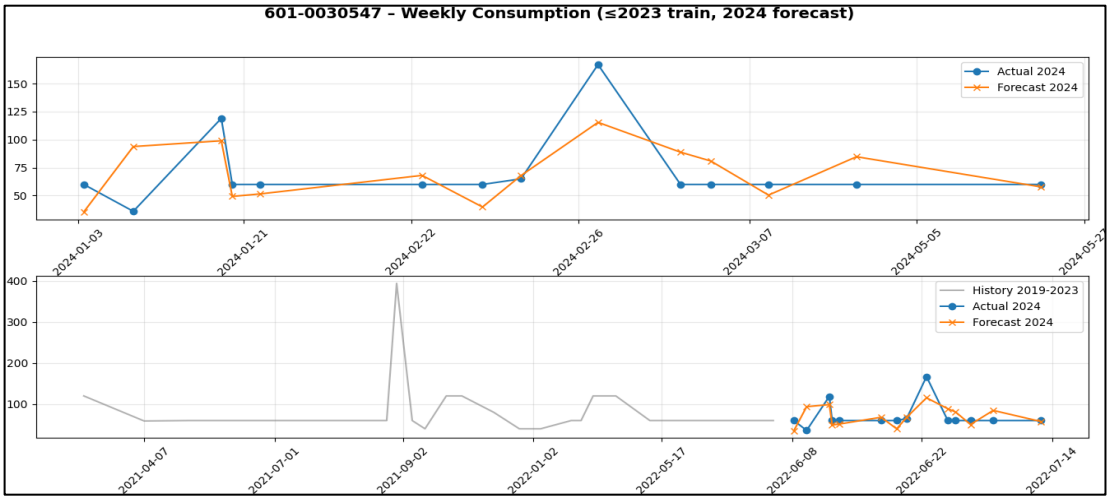
בתרשים זה אנו רואים את תחזית הצריכה השבועית לשנת 2024 עבור הפריט 601-0030547, בהתבסס על מידע היסטורי עד סוף 2023.

- הקו הכחול מייצג את הצריכה האמיתית בפועל בשנת 2024.
- הקו הכתום מייצג את תחזית הצריכה לשנת 2024 כפי שנחזה על ידי מודל XGBoost.

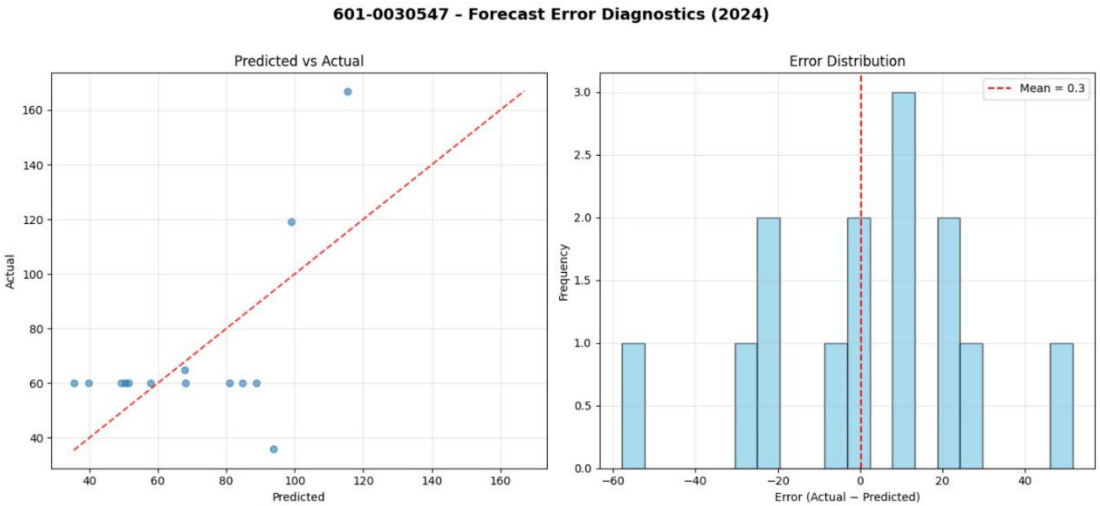
ניתן לראות כי המודל מצליח לקלוט בצורה טובה את המגמות הכלליות בפריט זה, לרבות עליות וירידות בצריכה לאורך השבועות. אמנם קיימות חריגות מסוימות (למשל בשבוע של סוף פברואר בו הצריכה בפועל הייתה גבוהה בהרבה מהתחזית), אך לאורך התקופה ניכרת יציבות יחסית וקרבה גבוהה בין הקווים.

המודל מצליח לזהות:

- תקופות של יציבות בצריכה (למשל באפריל-מאי).
- תקופות של עלייה חדה בביקוש, גם אם בעוצמה פחותה (לדוגמה, קפיצה בסוף פברואר).



איור 22-XGBoost;תחזית לפריט הבודד



איור 23-XGBoost-פיזור שגיאות למק"ט הבודד

מדדי שגיאות עבור כלל הפריטים למודל XGBoost :

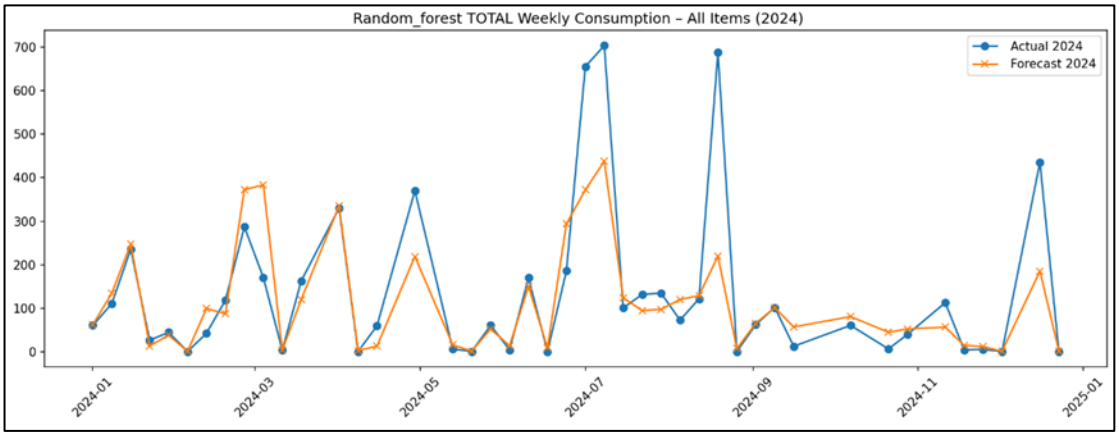
MAE =7.4
RMSE =9.94
R^2=0.43

מודל Random Forest

גרף 1 – תחזית מול צריכה בפועל

התרשים המצורף מציג את ההשוואה בין הצריכה בפועל (Actual 2024) לבין התחזית הצפויה על פי המודל (Forecast 2024) לאורך כל שבועות השנה. ניתן לראות כי המודל מצליח לשחזר בצורה טובה את המגמות הכלליות – כולל עונות שיא, תנודות חדות ונקודות קיצון. עם זאת, קיימים הבדלים נקודתיים בין ערכי התחזית לערכים בפועל, בעיקר בזמני צריכה קיצוניים תופעה צפויה בשימוש במודלים מבוססי עצים לניבוי סדרות זמן עם שינויים תקופתיים.

ניתוח גרפי זה מדגיש את תרומתו של מודל Random Forest ככלי עזר לקבלת החלטה בתכנון מלאי ותחזית צריכה, תוך שמירה על רמת דיוק גבוהה יחסית למורכבות הדפוסים שנצפו בפועל.

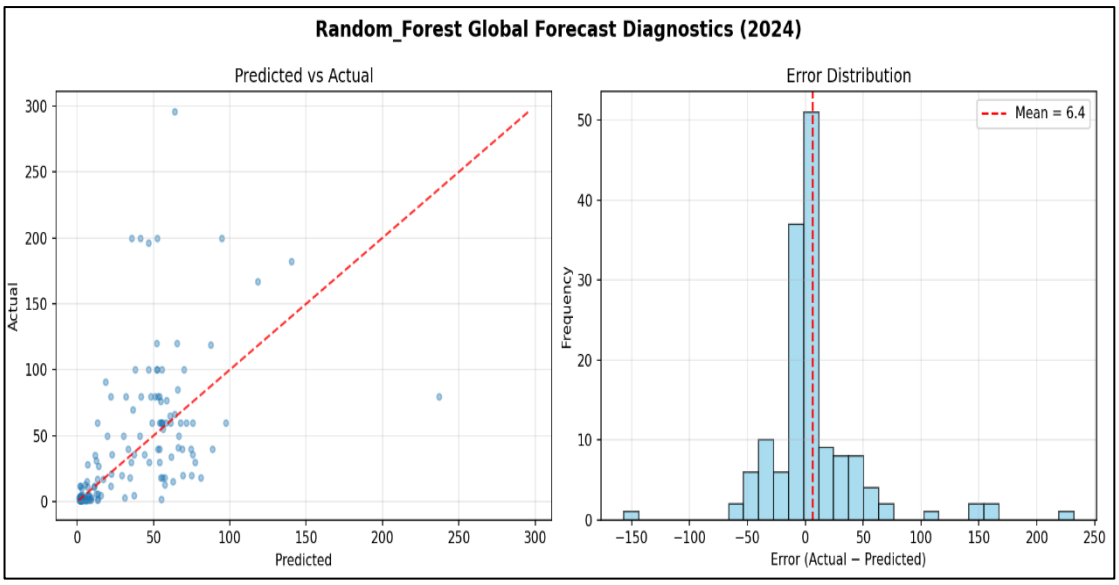


איור 24-RF-תחזית מול מציאות;כלל הפריטים

גרף 2- השוואת תחזית לעומת ערכים בפועל +פיזור השגיאות

לצורך הערכת ביצועי מודל ה־ Random Forest שבוצע במסגרת תחזית הצריכה לשנת 2024, הוצגו שני גרפים מרכזיים המדגישים את איכות התחזית ואת פרופיל השגיאה. הגרף השמאלי (Predicted vs Actual) מציג את הקשר בין ערכי התחזית שחזה המודל לבין הערכים האמיתיים שנמדדו בפועל. ניתן להבחין כי מרוב הנקודות מרוכזות סביב האלכסון (הקו האדום המקווקו), דבר המעיד על רמת דיוק טובה יחסית של המודל. עם זאת, ככל שהערכים עולים ניכרת נטייה להטיה קלה בתחזית, בעיקר כלפי חיזוי של ערכי קצה ,תופעה מקובלת במודלים מבוססי עצים שאינם מתמחים באופן ישיר בזיהוי נקודות קצה חריגות.

הגרף הימני (Error Distribution) מציג את התפלגות השגיאות (הפרש בין ערך בפועל לערך חזוי), כאשר רוב השגיאות מרוכזות סביב האפס, ובעלות ממוצע שגיאה חיובי של כ־6.4 יחידות.
התפלגות זו סימטרית יחסית, עם נטייה קטנה לשגיאות שליליות (שבהן התחזית גבוהה מהצריכה האמיתית), ומלמדת על כך שהמודל אינו מוטה באופן מהותי – יתרון חשוב בתחזיות תפעוליות.
פיזור השגיאות האופייני למודל זה תומך בצורה ברורה בתחזית הכללית, במיוחד כאשר מתמקדים במגמות ולא בערכים בודדים.



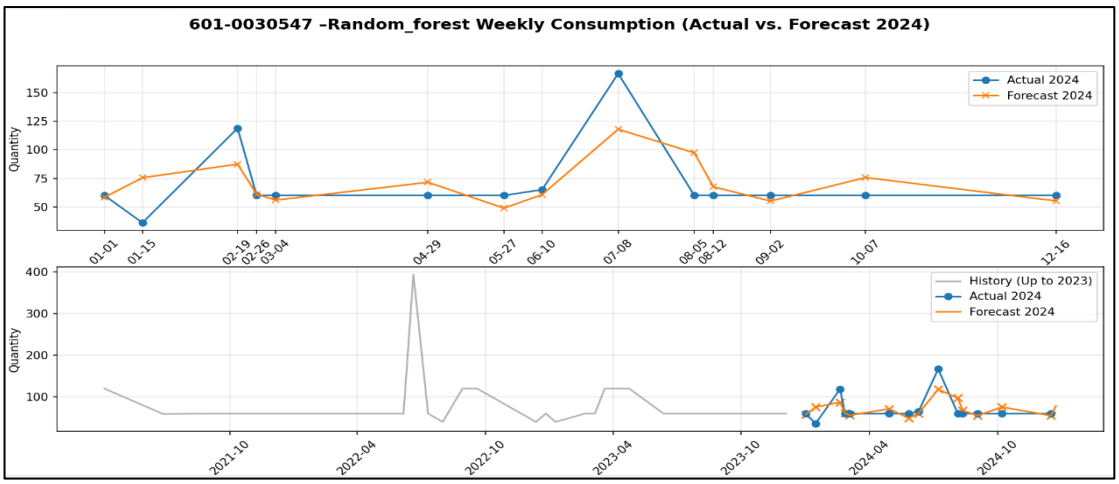
איור 25 -RF - פיזורי השגיאות; כלל הפריטים

גרף 3-תחזית לפריט בודד

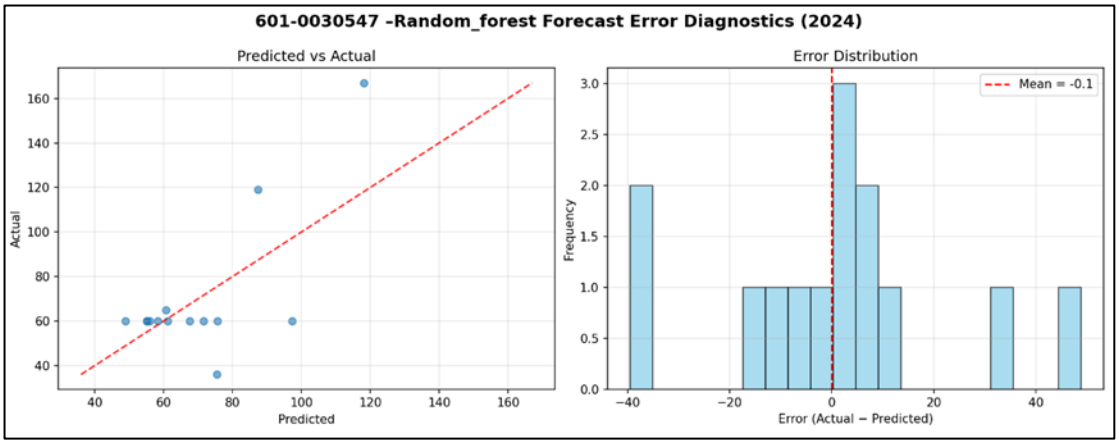
הגרף העליון בתמונה הנוכחית מציג את תחזית הצריכה השבועית לשנת 2024 עבור פריט מסוים (601-0030547), תוך השוואה בין נתוני אמת (בכחול) לבין תחזיות (כתום) במודל ה־ Random Forest. ניתן להבחין כי המודל מצליח לשחזר בצורה מדויקת יחסית את דפוסי הצריכה בטווחי זמן רבים לאורך השנה, בפרט באזורים שבהם הצריכה יציבה או בעלת תנודתיות קלה. עם זאת, קיימים חריגים נקודתיים למשל בחודש יולי בהם הצריכה בפועל קפצה לרמה חריגה שהמודל לא הצליח לחזות.

הגרף התחתון מאפשר מבט כללי המשלב את היסטוריית הצריכה עד סוף 2023 (באפור), ובכך מאפשר לזהות את המגמות שהמודל למד מהן. ניתן לראות כי המודל מתבסס על תבניות חוזרות מצריכה קודמת, אך ייתכן שעקב עוצמת התנודתיות ההיסטורית הנמוכה יחסית התחזית ל־2024 נותרה קבועה ולא הצליחה לקלוט אירועי קצה חדשים.

ניתוח זה נתמך גם על ידי גרף הפיזור המוצג (Predicted vs. Actual) וגרף התפלגות השגיאות (Error Distribution), בהם ניכר כי התחזית של המודל שומרת על יחסיות כללית סבירה מול ערכי האמת אך מתקשה בהתמודדות עם פיקים חדים בצריכה. ממוצע השגיאה) -0.1) נמוך יחסית, פיזור השגיאות סביב האפס יחד עם נקודות קצה קיצוניות מעידים על חוסר עקביות מסוימת במקרים של חריגות תפעוליות.



איור RF-26- תחזית לפריט הבודד



איור RF-27- פיזור מול שגיאה למק"ט הבודד

מדדי שגיאות עבור כלל המודלים עבור מודל Random Forest :

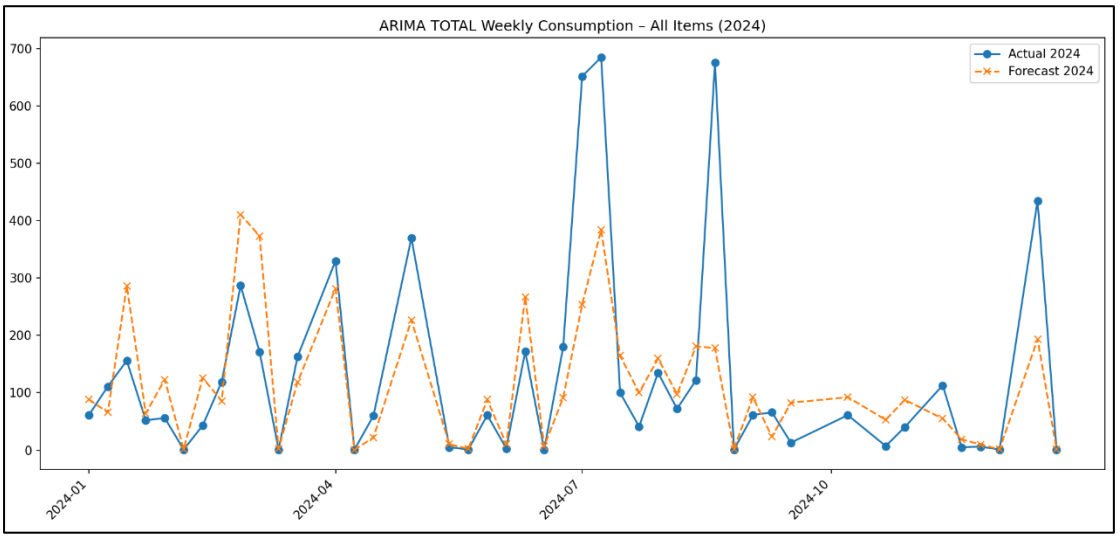
MAE = 5.6
RMSE = 7.97
R^2 = 0.6

מודל ARIMA

גרף 1 – תחזית מול צריכה בפועל

מהגרף ניכר כי המודל מצליח לשמר מגמות מסוימות של תנועה עונתית בסיסית, כגון עליות כלליות וירידות תקופתיות, אולם הוא מתקשה לשחזר את התנודתיות הבלתי צפויה שמאפיינת את חלק מהשבועות. בפרט, קיימת נטייה של המודל להחלקת סדרת הזמן ולטשטוש נקודות קצה כגון זינוקים חדים בצריכה (כפי שנראה במהלך חודשי הקיץ), שאינם משתקפים באופן מלא בתחזיות.

הפערים הבולטים ביותר מופיעים באזורים שבהם חלה צריכה קיצונית בפועל, ואילו התחזיות שומרות על ערכים מתונים יחסית. תופעה זו מעידה על מגבלתו של מודל ARIMA, שמבוסס על קשרים ליניאריים בין נקודות בזמן, במקרים של תנודתיות גבוהה או שינויים חדים שאינם צפויים מראש.



איור 28- ARIMA- תחזית מול מציאות;כלל הפריטים

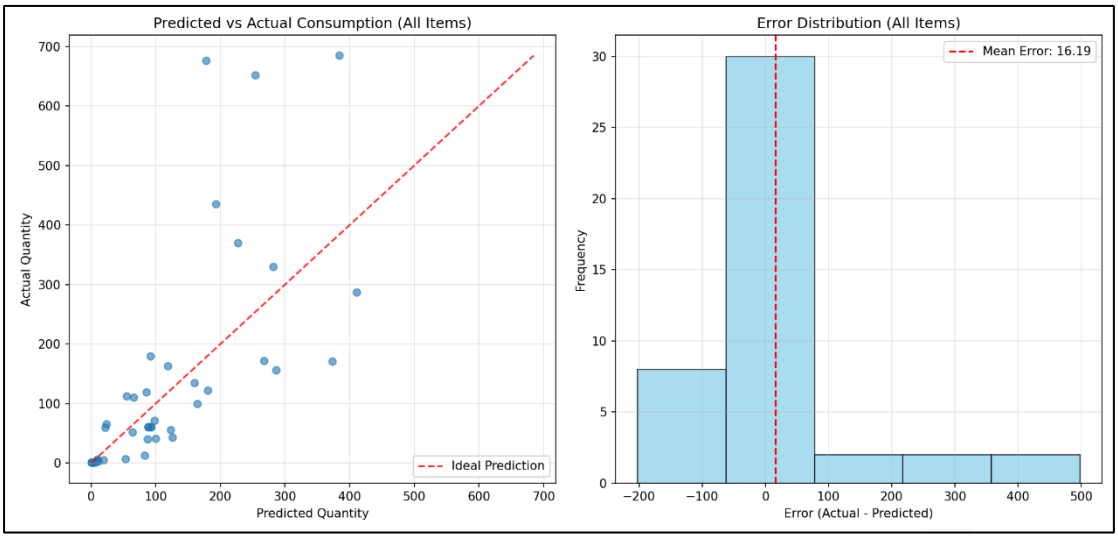
גרף 2- השוואת תחזית לעומת ערכים בפועל +פיזור השגיאות

במטרה להעריך את ביצועי מודל ARIMA בתחזית צריכה לשנת 2024, הוצגו שני גרפים מרכזיים, המספקים תובנות על רמת הדיוק של התחזית ועל מבנה השגיאות שנוצרו.

בגרף השמאלי, (*Predicted vs. Actual Consumption*) מתוארת ההשוואה בין הכמויות החזויות על ידי המודל לבין ערכי הצריכה בפועל לכלל הפריטים האופטיים. הקו האדום המקווקו מייצג חיזוי טוב שבו התחזית זהה לערך בפועל. רוב הנקודות מצויות במרחק ניכר מהקו, דבר המעיד על חוסר התאמה בין התחזית לצריכה בפועל. פיזור זה משקף על דיוק תחזיתי מוגבל של המודל, וכן על קושי לשקף בצורה טובה את דפוסי הצריכה האמיתיים.

בגרף הימני, (*Error Distribution*) נבחנת התפלגות השגיאות הפערים בין הצריכה בפועל לבין הערכים החזויים. ניתן להבחין כי ההתפלגות אינה סימטרית וניכרת נטייה לשגיאות שליליות, כלומר תחזיות גבוהות מהצריכה בפועל. ממוצע השגיאות עומד על 16.19 יחידות. אף על פי שממוצע חיובי העשוי לתת אינדיקציה על היעדר הטיה של המודל, אופן ההתפלגות מעיד על חוסר איזון וחזוי פחות מדויק.

שני הגרפים ממחישים את מגבלות המודל ככלי תחזיתי לצריכה. הפערים בין התחזית למציאות, לצד שגיאות מרובות ובלתי סימטריות, מגבירים את ההבנה לאפשרות להסתמך על מודל זה לצורך קבלת החלטות תפעוליות מדויקות. ממצאים אלו מדגישים את הצורך בשילוב מודלים מתקדמים היכולים ללתת חיזוי טוב יותר ולגשר על הפער בין החיזוי לצריכה בפועל .



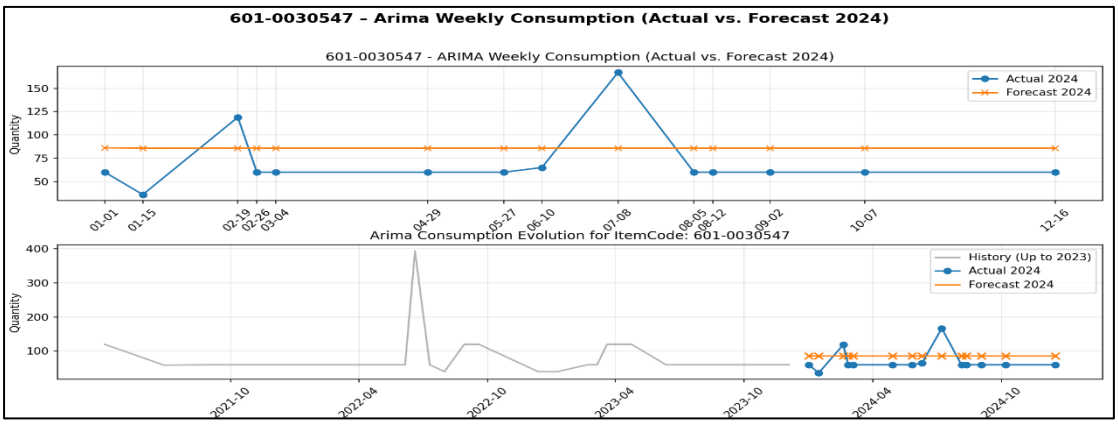
איור 29-ARIMA- פיזורי השגיאות; כלל הפריטים

גרף 3-תחזית לפריט בודד

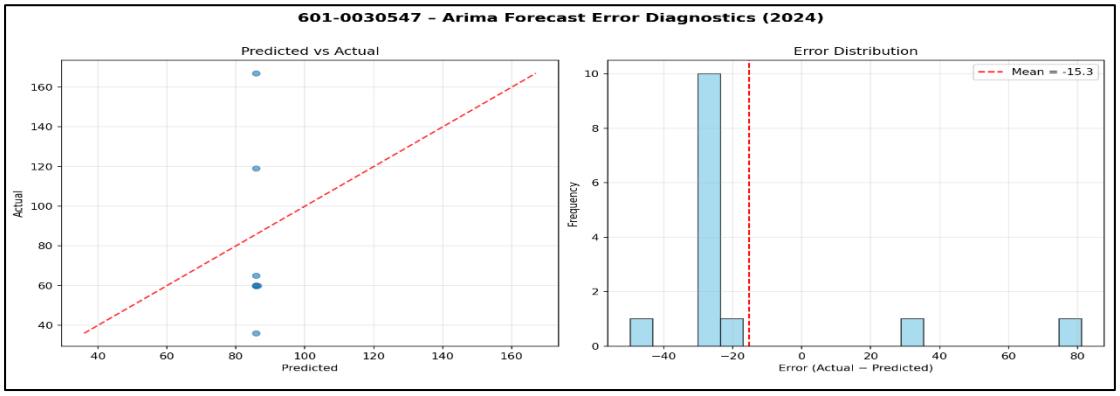
בגרף העליון מוצגת השוואה בין ערכי הצריכה בפועל במהלך שנת 2024 (קו כחול) לבין התחזית שחושבה על ידי המודל (קו כתום). מהשוואה זו עולה כי המודל מתקשה לשקף באופן מדויק את הצריכה בפועל. באופן בולט ניתן לראות בשבוע של תחילת יולי, שבו התרחש פיק חד בצריכה בפועל, בעוד שהמודל חזה ערך קבוע וללא תגובה לשינוי. גם בשבועות אחרים ניכרת סטייה בין התחזית לנתונים האמיתיים, התחזית נותרת קבועה, למרות שינויים בצריכה בפועל דבר המשקף על רגישות נמוכה של המודל לשינויים קצרי טווח.

הגרף התחתון מוסיף נדבך היסטורי לניתוח, ומציג את דפוסי הצריכה עד לסוף 2023 (קו אפור), לצד הערכים בפועל והתחזית לשנת 2024. גרף זה מאפשר להבין את בסיס הלמידה של המודל אשר ככל הנראה נשען על מגמות עונתיות או מחזוריות מן העבר.

בהתייחס לגרף הפיזור (Predicted vs. Actual) ולגרף התפלגות השגיאות (Error Distribution) נמצא כי קיים פיזור רחב של שגיאות, והתחזיות אינן שומרות על יחסיות סבירה מול הצריכה בפועל. ממוצע השגיאה השלילית (-15.3) מעיד על נטייה של המודל לחיזוי יתר-(Over prediction) בנוסף, התפלגות השגיאות אינה סימטרית סביב האפס, ומכילה ערכים קיצוניים המצביעים על חוסר עקביות וחוסר עמידות של המודל במקרים של חריגות תפעוליות.



איור 30-ARIMA- תחזית לפריט הבודד



איור 31- ARIMA- פיזורי השגיאות, כלל הפריטים

מדדי שגיאות עבור כלל המודלים עבור מודל ARIMA :

MAE = 23.18

RMSE = 42.17

R^2 = 0.3

לאחר שאספנו את נתוני המדד מכל המודלים, גילינו שהמודל עם השגיאות הנמוכות ביותר היה מודל ה**Random Forest**, ולכן בחרנו בו בתור המודל שאיתו נמשיך לבנות את סימולציית המלאי והרכש לחברה.

טבלת סיכום מדדים:

טבלה 10-סיכום מדדי שגיאה

	MAE	RMSE	R^2
Prophet	16.37	67.02	0.3
Random Forest	5.6	7.97	0.6
XGBoost	7.4	9.94	0.43
ARIMA	23.18	42.17	0.3

השפעת המודל והיישום על רמות המלאי והחריגות

לאחר בחירת מודל התחזית המיטבי (Random Forest) ויישום מודל ניהול המלאי שפיתחנו, ניתחנו את התרומה והשינויים בכלל המערכת– הכוללת תחזית צריכה, קביעת גבולות מלאי דינמיים, ותוכנית רכש אופטימלית – על ניהול המלאי בפועל.

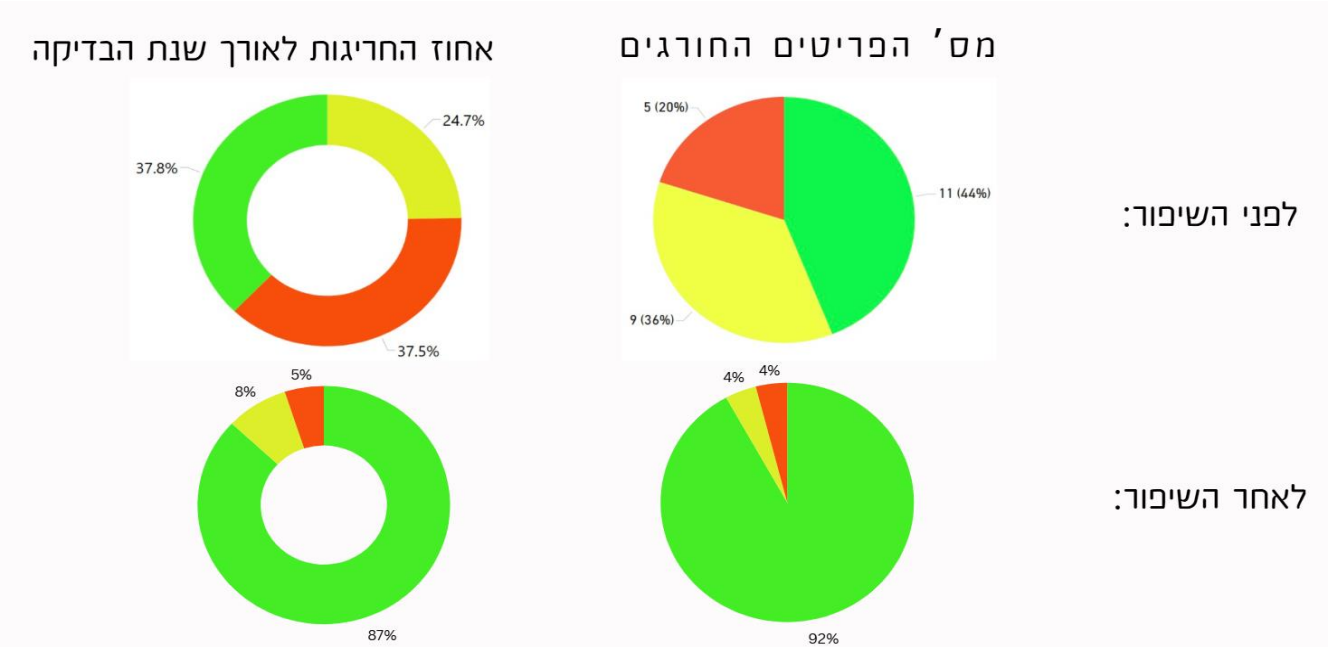
מהשוואת הביצועים לפני ואחרי השיפור עולה ניתן לראות :

- **שיעור הפריטים ששהו בתוך גבולות המלאי התקינים עלה מ־44% ל־92% (ביחס לאותה נקודת זמן שנבדקה בתחילת העבודה).**
- במקביל, **חלה ירידה גדולה בפריטים שחרגו מגבולות המלאי** –הן כלפי מעלה והן כלפי מטה.
- אם בתחילת התהליך מעל מחצית מהפריטים סבלו מחריגות תדירות לאורך זמן (חוסר או עודף), כעת , הפריטים יהיו בתוך הגבולות במשך **87% מהשבועות** (לעומת 38% שהיו לפני השינוי שלנו).

תוצאות אלו מדגימות את **התרומה המשמעותית של המודל לחיזוק היציבות התפעולית של הארגון**:

- הקטנת הסיכון לחוסרים והפסדי מכירה
- הפחתת עלויות עודפי מלאי
- התייעלות כוללת בתכנון הרכש

התרשימים הבאים ממחישים את השיפור באופן חזותי וברור :



איור 32-השוואת מדדים

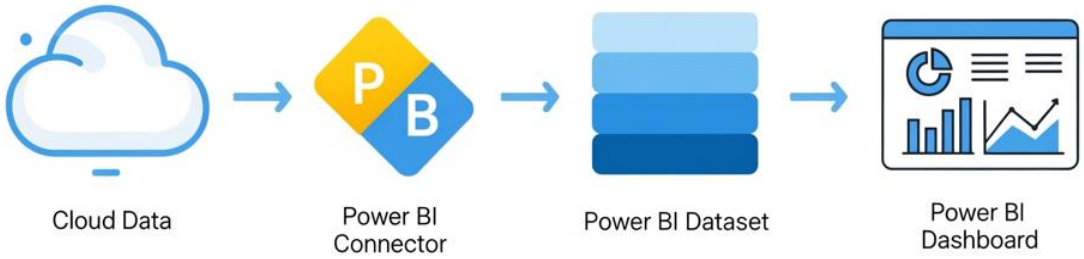
סיכום

מטרת הפרויקט הייתה לשפר את תהליכי ניהול המלאי והרכש בחברת 'סיון טכנולוגיות' באמצעות שילוב של תחזיות מבוססות למידת מכונה ויישום מודל מלאי אופטימלי. בפרט, ביקשנו להציע לחברה כלים חכמים לקבלת החלטות לגבי מתי וכמה להזמין, תוך שמירה על רמות מלאי תקינות ומינימום חריגות.

לאחר שניתחנו את התנהגות המלאי וצריכת הפריטים לאורך זמן, בנינו מודלים לחיזוי הביקושים העתידיים ובחרנו את המודל שהציג את רמת הדיוק הגבוהה ביותר(מודל Random Forest). את התחזיות שהתקבלו שילבנו במודל ניהול מלאי מסוג Cover-Time + Safety Stock , אשר מאפשר קביעת נקודות הזמנה אופטימליות ולפי זה בנינו המלצות לגבולות דינמיים לכל פריט, תוך התאמה לתנודות עתידיות בצריכה.

בשלב הסופי, יצרנו דשבורד אינטראקטיבי Power BI-שמרכז את התחזיות, ההמלצות לרכש, ניתוח חריגות והערכת עלויות – בצורה ברורה ונגישה לכל גורם בארגון. הדשבורד הוטמע במערכות של החברה ומתרענן בצורה אוטומטית חד לרבעון (Connect Power BI gateway) המערכת Power BI גוזרת את הנתנונים מהענן של סיוון לתוך הדשבורד האינטראקטיבי שיצרנו כך שהדשבורד נשאר מעודכן לצורך קבלת החלטות .

הפרויקט הניב תשתית ניהולית שיטתית, המפחיתה את חוסר הוודאות ומחזקת את יכולת קבלת ההחלטות בניהול הרכש והמלאי. השימוש בתחזיות מבוססות למידת מכונה הוביל לשיפור משמעותי בעמידה בגבולות המלאי וליעילות גבוהה יותר בתכנון ההזמנות.
הדשבורד שפיתחנו מהווה כלי עבודה מרכזי עבור החברה, וצפוי להפוך בהמשך לחלק בלתי נפרד ממערכת ה ERP-הארגונית.
מכאן והלאה, ניתן להמשיך ולחקור נושאים נוספים כגון : התאמת רמות מלאי לפי פרופיל סיכון של פריט, חישוב עלויות חסר מדויקות, שיפור תחזיות בעזרת שילוב של מגמות שוק חיצוניות, והרחבת המודלים לפריטים חדשים וקטגוריות שטרם נצבר עבורם מספיק מידע.



איור 33- זרימת הנתונים מהענן ל POWER BI בעמצאות gateway

בבילוגרפיה

Ni, S. F., Peng, Y., Peng, K., & Liu, Z. J. (2022). Supply Chain Demand Forecast Based on SSA-XGBoost Model. Journal of Computer and Communications, 10(12), 71–83 . https://doi.org/10.4236/jcc.2022.1012006

Fattahi, M., & Govindan, K. (2021). Min–Max inventory control policy for multi-echelon supply chain networks under uncertainty: A robust optimization approach. Applied Soft Computing, 107, 107343. https://doi.org/10.1016/j.asoc.2021.107343

Svetunkov, I., & Boylan, J. E. (2020). State-space ARIMA for supply-chain forecasting. International Journal of Production Research, 58(3), 818–827. https://doi.org/10.1080/00207543.2019.1600764

נספחים

Random Forest

```
import os
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
from sklearn.preprocessing import LabelEncoder
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from collections import deque
from statistics import NormalDist
import math


import matplotlib

matplotlib.use('TkAgg')


os.environ['TCL_LIBRARY'] = r'C:\\Program Files\\Python313\\tcl\\tcl8.6'
os.environ['TK_LIBRARY'] = r'C:\\Program Files\\Python313\\tcl\\tk8.6'


 cmp = cmp_df.copy()


   hist = hist_df.copy()


   future_forecast = future_forecast_df.copy()

   preds = cmp["PredictedQty"]
   actual = cmp["ConsumptionQty"]
   errors = actual.values - preds.values


   fig1 = plt.figure(figsize=(15, 8))
   fig1.suptitle(f"{sku} –Random_forest Weekly Consumption "
               f"(Actual vs. Forecast {test_year})",
               fontsize=14, fontweight="bold")


   ax1 = fig1.add_subplot(2, 1, 1)
```

```
    ax1.plot(cmp["SimDate"], actual, marker="o", label="Actual 2024")
    ax1.plot(cmp["SimDate"], preds, marker="x", label="Forecast 2024")
    ax1.legend();
    ax1.grid(alpha=0.3)
    ax1.set_ylabel('Quantity')

    if not cmp["SimDate"].empty:
        n_ticks = 8
        if len(cmp["SimDate"]) > n_ticks:
            step = len(cmp["SimDate"]) // n_ticks
            if step == 0: step = 1
            display_dates_upper = cmp["SimDate"].iloc[::step]
        else:
            display_dates_upper = cmp["SimDate"]

        ax1.set_xticks(display_dates_upper)
        ax1.set_xticklabels([d.strftime('%m-%d') for d in display_dates_upper],
rotation=45)


    ax2 = fig1.add_subplot(2, 1, 2)
    if not hist.empty:
        ax2.plot(hist["SimDate"], hist["ConsumptionQty"],
                 color="gray", alpha=0.6, label=f"History (Up to {test_year - 1})")

    ax2.plot(cmp["SimDate"], actual, marker="o", label=f"Actual {test_year}")

    forecast_2024_plot_segment = cmp[['SimDate', 'PredictedQty']].copy()


    future_plot_beyond_2024_segment = future_forecast[
        future_forecast['SimDate'] > forecast_2024_plot_segment['SimDate'].max()
        ].copy()


    combined_forecast_plot_df = pd.concat([forecast_2024_plot_segment,
future_plot_beyond_2024_segment]).sort_values(
        'SimDate')

    if not combined_forecast_plot_df.empty:

        ax2.plot(combined_forecast_plot_df['SimDate'],
combined_forecast_plot_df['PredictedQty'],
                 label='Forecast 2024', color='tab:orange', linestyle='-')  # No marker here
```

```
        ax2.scatter(cmp['SimDate'], cmp['PredictedQty'],
                    marker='x', color='tab:orange', s=50)

    ax2.legend();
    ax2.grid(alpha=0.3)
    ax2.set_ylabel('Quantity')


    all_dates_bottom_plot_list = []
    if not hist.empty:
        all_dates_bottom_plot_list.extend(hist['SimDate'].tolist())
    if not cmp.empty:
        all_dates_bottom_plot_list.extend(cmp['SimDate'].tolist())
    if not future_forecast.empty:
        all_dates_bottom_plot_list.extend(future_forecast['SimDate'].tolist())

    all_dates_bottom_plot =
pd.to_datetime(list(set(all_dates_bottom_plot_list))).sort_values()

    if len(all_dates_bottom_plot) > 1:
        start_plot_date = all_dates_bottom_plot.min()
        end_plot_date = all_dates_bottom_plot.max()

        date_range_for_ticks = pd.date_range(start=start_plot_date.replace(day=1),
                                             end=end_plot_date.replace(day=1) +
pd.DateOffset(months=6),
                                             freq='6MS')

        valid_ticks = [d for d in date_range_for_ticks if start_plot_date <= d <=
end_plot_date]

        ax2.set_xticks(valid_ticks)
        ax2.set_xticklabels([d.strftime('%Y-%m') for d in valid_ticks], rotation=45)

    plt.tight_layout(rect=[0, 0, 1, 0.94])
    plt.show()


    fig2 = plt.figure(figsize=(14, 5))
    fig2.suptitle(f"{sku} –Random_forest Forecast Error Diagnostics ({test_year})",
                  fontsize=14, fontweight="bold")


    ax3 = fig2.add_subplot(1, 2, 1)
    ax3.scatter(preds, actual, alpha=0.6)
    lims = [min(preds.min(), actual.min()),
```

```
            max(preds.max(), actual.max())]
    ax3.plot(lims, lims, "r--", alpha=0.8)
    ax3.set_xlabel("Predicted");
    ax3.set_ylabel("Actual")
    ax3.set_title("Predicted vs Actual");
    ax3.grid(alpha=0.3)


    ax4 = fig2.add_subplot(1, 2, 2)
    ax4.hist(errors, bins=20, alpha=0.7,
            color="skyblue", edgecolor="black")
    ax4.axvline(errors.mean(), color="red",
               ls="--", label=f"Mean = {errors.mean():.1f}")
    ax4.set_xlabel("Error (Actual − Predicted)")
    ax4.set_ylabel("Frequency")
    ax4.set_title("Error Distribution")
    ax4.legend();
    ax4.grid(alpha=0.3)

    plt.tight_layout()
    plt.show()


def plot_global_diagnostics(all_df: pd.DataFrame, title_year: int = 2024):

    gdf = all_df.copy()

    gdf['Date'] = gdf.apply(
        lambda row: datetime.fromisocalendar(int(row['ConsumptionYear']),
int(row['ConsumptionWeek']), 1), axis=1
    )
    gdf["Actual"] = gdf["ConsumptionQty"]
    gdf["Forecast"] = gdf["PredictedQty"]
    gdf = gdf[gdf["Date"].dt.year == title_year]


    weekly = (gdf.groupby("Date")[["Actual", "Forecast"]]
             .sum().reset_index())
    plt.figure(figsize=(14, 5))
    plt.plot(weekly["Date"], weekly["Actual"],
            marker="o", label=" Actual 2024")
    plt.plot(weekly["Date"], weekly["Forecast"],
            marker="x", label=" Forecast 2024")
    plt.title(f" Random_forest TOTAL Weekly Consumption – All Items
({title_year})")
    plt.xticks(rotation=45);
```

```
    #plt.grid(alpha=0.3);
    plt.legend();
    plt.tight_layout()
    plt.show()

    # --- B & C: Scatter + Histogram ----------------------------------------
    preds = gdf["Forecast"].values
    actual = gdf["Actual"].values
    errors = actual - preds

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))
    fig.suptitle(f"Random_Forest Global Forecast Diagnostics ({title_year})",
                 fontsize=14, fontweight="bold")

    # scatter
    ax1.scatter(preds, actual, alpha=0.4, s=15)
    lims = [min(preds.min(), actual.min()),
            max(preds.max(), actual.max())]
    ax1.plot(lims, lims, "r--", alpha=0.8)
    ax1.set_xlabel("Predicted");
    ax1.set_ylabel("Actual")
    ax1.set_title("Predicted vs Actual");
    ax1.grid(alpha=0.3)

    # histogram
    ax2.hist(errors, bins=30, alpha=0.7,
             color="skyblue", edgecolor="black")
    ax2.axvline(errors.mean(), color="red", ls="--",
                label=f"Mean = {errors.mean():.1f}")
    ax2.set_xlabel("Error (Actual − Predicted)")
    ax2.set_ylabel("Frequency")
    ax2.set_title("Error Distribution")
    ax2.legend();
    ax2.grid(alpha=0.3)

    plt.tight_layout()
    plt.show()


df = pd.read_csv("optical_db_update.csv", encoding='utf-8')
df['ConsumptionDate'] = pd.to_datetime(df['ConsumptionDate'], errors='coerce')
df = df.dropna(subset=['ConsumptionQty', 'ConsumptionDate'])

df['ConsumptionYear'] = df['ConsumptionDate'].dt.year
df['ConsumptionWeek'] = df['ConsumptionDate'].dt.isocalendar().week
df['ConsumptionWeekday'] = df['ConsumptionDate'].dt.weekday
```

```
df['YearWeek'] = df['ConsumptionDate'].dt.strftime('%Y-%U')


le = LabelEncoder()
df['ItemCode_encoded'] = le.fit_transform(df['ItemCode'].astype(str))


grouped = df.groupby(['ItemCode', 'YearWeek']).agg({
    'ConsumptionQty': 'sum',
    'PRICE': 'mean',
    'LT_Days': 'mean',
    'InventoryBalance': 'mean',
    'ConsumptionYear': 'first',
    'ConsumptionWeek': 'first',
    'ItemCode_encoded': 'first'
}).reset_index()


grouped.fillna(method='ffill', inplace=True)



features = ['PRICE', 'LT_Days', 'InventoryBalance', 'ConsumptionYear',
'ConsumptionWeek', 'ItemCode_encoded']
X = grouped[features]
y = grouped['ConsumptionQty']

train_data = grouped[grouped['ConsumptionYear'] <= 2023]
test_data = grouped[grouped['ConsumptionYear'] == 2024]

X_train = train_data[features]
y_train = train_data['ConsumptionQty']

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

item_codes = grouped['ItemCode'].unique()

last_date_training = df[df['ConsumptionYear'] < 2024]['ConsumptionDate'].max()
future_data = []

start_forecast_date = datetime(2024, 1, 1)

for item in item_codes:

    item_data_for_forecast_base = grouped[
```

```
        (grouped['ItemCode'] == item) & (grouped['ConsumptionYear'] < 2024)
        ].sort_values('YearWeek')

    if item_data_for_forecast_base.empty:
        print(f"אזהרה: אין נתוני אימון עבור מק"\ט {item} כדי ליצור תחזית עתידית. מדלג על תחזית(").
        continue


    last_item_data = item_data_for_forecast_base.iloc[-1]

    price = last_item_data['PRICE']
    lt_days_val = last_item_data['LT_Days']
    inventory = last_item_data['InventoryBalance']
    item_encoded = last_item_data['ItemCode_encoded']

    for i in range(52):
        future_week_date = start_forecast_date + timedelta(weeks=i)
        year = future_week_date.year
        week = future_week_date.isocalendar().week
        year_week = f"{year}-{str(week).zfill(2)}"

        future_data.append({
            'ItemCode': item,
            'YearWeek': year_week,
            'PRICE': price,
            'LT_Days': lt_days_val,
            'InventoryBalance': inventory,
            'ConsumptionYear': year,
            'ConsumptionWeek': week,
            'ItemCode_encoded': item_encoded
        })


future_df = pd.DataFrame(future_data)


X_future = future_df[features]
future_df['PredictedQty'] = model.predict(X_future)


future_forecast = future_df[['ItemCode', 'YearWeek', 'PredictedQty']]
print("--- תחזית ל 52-שבועות קדימה (דוגמה) ---")
print(future_forecast.head(20))
```

```
actual_2024 = grouped[grouped['ConsumptionYear'] == 2024].copy()


merged_2024_data = actual_2024.merge(
    future_df[['ItemCode', 'YearWeek', 'PredictedQty']],
    on=['ItemCode', 'YearWeek'],
    how='left'
)
X_2024 = merged_2024_data[features]
merged_2024_data['PredictedQty'] = model.predict(X_2024)

item_errors = []
summary_stats = []

plot_global_diagnostics(merged_2024_data, title_year=2024)

 (2024) ---"\n---הערכת מודל Random Forest ומדדי מלאי עבור כל מק"ט")print
for item in actual_2024['ItemCode'].unique():
    print(f"\n========== ItemCode: {item} ==========")
    try:
        item_df_2024 = merged_2024_data[merged_2024_data['ItemCode'] ==
item].sort_values('YearWeek').copy()

        if len(item_df_2024) < 2:
 אין מספיק נתוני צריכה בפועל עבור מק"ט {item} להערכת מודל או סימולציית מלאי ב 2024-מדלג(".")print(f"
            continue

        item_df_2024['SimDate'] = item_df_2024.apply(
            lambda row: datetime.fromisocalendar(int(row['ConsumptionYear']),
int(row['ConsumptionWeek']), 1), axis=1
        )
        item_df_2024 = item_df_2024.sort_values('SimDate')

        history_for_bottom_plot = grouped[
            (grouped['ItemCode'] == item) & (grouped['ConsumptionYear'] < 2024)
            ].copy()


        history_for_bottom_plot['SimDate'] = history_for_bottom_plot.apply(
            lambda row: datetime.fromisocalendar(int(row['ConsumptionYear']),
int(row['ConsumptionWeek']), 1), axis=1
        )
        history_for_bottom_plot = history_for_bottom_plot.sort_values('SimDate')

        future_item_df = future_df[future_df['ItemCode'] == item].copy()
```

```
        future_item_df['SimDate'] = future_item_df.apply(
            lambda row: datetime.fromisocalendar(int(row['ConsumptionYear']),
int(row['ConsumptionWeek']), 1), axis=1
        )
        future_item_df = future_item_df.sort_values('SimDate')


        plot_consumption(
            item,
            cmp_df=item_df_2024,
            hist_df=history_for_bottom_plot,
            future_forecast_df=future_item_df,
            test_year=2024
        )

        mae = mean_absolute_error(item_df_2024['ConsumptionQty'],
item_df_2024['PredictedQty'])
        mse = mean_squared_error(item_df_2024['ConsumptionQty'],
item_df_2024['PredictedQty'])
        rmse = np.sqrt(mse)
        r2 = r2_score(item_df_2024['ConsumptionQty'], item_df_2024['PredictedQty'])
        item_errors.append({
            'ItemCode': item,
            'MAE': mae,
            'RMSE': rmse,
            'R2': r2,
        })

        forecast_df_sim = pd.DataFrame({
            "Date": item_df_2024["SimDate"],
            "Forecast": item_df_2024["PredictedQty"]
        })
        actual_df_sim = pd.DataFrame({
            "Date": item_df_2024["SimDate"],
            "Actual": item_df_2024["ConsumptionQty"]
        })
```

Arima

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_absolute_error, mean_squared_error
from collections import deque
from statistics import NormalDist
import math
from datetime import datetime, timedelta

os.environ['TCL_LIBRARY'] = r'C:\\Program Files\\Python313\\tcl\\tcl8.6'
os.environ['TK_LIBRARY'] = r'C:\\Program Files\\Python313\\tcl\\tk8.6'
    cmp = cmp_df.copy()


    hist = hist_df.copy()


    future_forecast = future_forecast_df.copy()

    preds = cmp["PredictedQty"]
    actual = cmp["ConsumptionQty"]
    errors = actual.values - preds.values


    fig1 = plt.figure(figsize=(15, 8))
    fig1.suptitle(f"{sku} – Arima Weekly Consumption "
                  f"(Actual vs. Forecast {test_year})",
                  fontsize=14, fontweight="bold")


    ax1 = fig1.add_subplot(2, 1, 1)
    ax1.plot(cmp["SimDate"], actual, marker="o", label="Actual 2024")
    ax1.plot(cmp["SimDate"], preds, marker="x", label="Forecast 2024")
    ax1.set_title(f'{sku} - {model_name} Weekly Consumption (Actual vs. Forecast
{test_year})') # Dynamic title
    ax1.legend();
    ax1.grid(alpha=0.3)
    ax1.set_ylabel('Quantity')

    if not cmp["SimDate"].empty:
```

```
        n_ticks = 8
        if len(cmp["SimDate"]) > n_ticks:
            step = len(cmp["SimDate"]) // n_ticks
            if step == 0: step = 1
            display_dates_upper = cmp["SimDate"].iloc[::step]
        else:
            display_dates_upper = cmp["SimDate"]

        ax1.set_xticks(display_dates_upper)
        ax1.set_xticklabels([d.strftime('%m-%d') for d in display_dates_upper],
rotation=45)


    ax2 = fig1.add_subplot(2, 1, 2)
    if not hist.empty:
        ax2.plot(hist["SimDate"], hist["ConsumptionQty"],
                 color="gray", alpha=0.6, label=f"History (Up to {test_year - 1})")

    ax2.plot(cmp["SimDate"], actual, marker="o", label=f"Actual {test_year}",
color='tab:blue')


    forecast_2024_plot_segment = cmp[['SimDate', 'PredictedQty']].copy()

    future_plot_beyond_2024_segment = future_forecast[
        future_forecast['SimDate'] > forecast_2024_plot_segment['SimDate'].max()
    ].copy()

    combined_forecast_plot_df = pd.concat([forecast_2024_plot_segment,
future_plot_beyond_2024_segment]).sort_values('SimDate')

    if not combined_forecast_plot_df.empty:

        ax2.plot(combined_forecast_plot_df['SimDate'],
combined_forecast_plot_df['PredictedQty'],
                 label='Forecast 2024', color='tab:orange', linestyle='-') # No marker here

        ax2.scatter(cmp['SimDate'], cmp['PredictedQty'],
                    marker='x', color='tab:orange', s=50)


    ax2.legend();
    ax2.grid(alpha=0.3)
    ax2.set_ylabel('Quantity')
    ax2.set_title(f'Arima Consumption Evolution for ItemCode: {sku}')
```

```
    all_dates_bottom_plot_list = []
    if not hist.empty:
        all_dates_bottom_plot_list.extend(hist['SimDate'].tolist())
    if not cmp.empty:
        all_dates_bottom_plot_list.extend(cmp['SimDate'].tolist())
    if not future_forecast.empty: # Include entire forecast range for x-axis coverage
        all_dates_bottom_plot_list.extend(future_forecast['SimDate'].tolist())

    all_dates_bottom_plot =
pd.to_datetime(list(set(all_dates_bottom_plot_list))).sort_values()

    if len(all_dates_bottom_plot) > 1:
        start_plot_date = all_dates_bottom_plot.min()
        end_plot_date = all_dates_bottom_plot.max()

        date_range_for_ticks = pd.date_range(start=start_plot_date.replace(day=1),
                                             end=end_plot_date.replace(day=1) +
pd.DateOffset(months=6),
                                             freq='6MS')  # Every 6 months
        valid_ticks = [d for d in date_range_for_ticks if start_plot_date <= d <=
end_plot_date]

        ax2.set_xticks(valid_ticks)
        ax2.set_xticklabels([d.strftime('%Y-%m') for d in valid_ticks], rotation=45)

    plt.tight_layout(rect=[0, 0, 1, 0.94])
    plt.show()

    # ---------- Figure 2 ----------
    fig2 = plt.figure(figsize=(14, 5))
    fig2.suptitle(f"{sku} – Arima Forecast Error Diagnostics ({test_year})",
                  fontsize=14, fontweight="bold")

    # scatter
    ax3 = fig2.add_subplot(1, 2, 1)
    ax3.scatter(preds, actual, alpha=0.6)
    lims = [min(preds.min(), actual.min()),
            max(preds.max(), actual.max())]
    ax3.plot(lims, lims, "r--", alpha=0.8)
    ax3.set_xlabel("Predicted");
    ax3.set_ylabel("Actual")
    ax3.set_title("Predicted vs Actual");
    ax3.grid(alpha=0.3)

    # histogram
    ax4 = fig2.add_subplot(1, 2, 2)
```

```
    ax4.hist(errors, bins=20, alpha=0.7,
             color="skyblue", edgecolor="black")
    ax4.axvline(errors.mean(), color="red",
                ls="--", label=f"Mean = {errors.mean():.1f}")
    ax4.set_xlabel("Error (Actual − Predicted)")
    ax4.set_ylabel("Frequency")
    ax4.set_title("Error Distribution")
    ax4.legend();
    ax4.grid(alpha=0.3)

    plt.tight_layout()
    plt.show()


df = pd.read_csv('optical_db_update.csv', encoding='utf-8')

df['ConsumptionDate'] = pd.to_datetime(df['ConsumptionDate'], errors='coerce')
df['ConsumptionQty'] = pd.to_numeric(df['ConsumptionQty'], errors='coerce')
df.dropna(subset=['ConsumptionDate', 'ConsumptionQty'], inplace=True)
df['Week'] = df['ConsumptionDate'].dt.to_period('W')
df['ConsumptionYear'] = df['ConsumptionDate'].dt.year # Add ConsumptionYear for
consistency

valid_weeks = df[df['ConsumptionQty'] > 0].groupby('ItemCode')['Week'].nunique()
valid_items = valid_weeks[valid_weeks >= 2].index
df = df[df['ItemCode'].isin(valid_items)]

summary_stats = []

item_codes = df['ItemCode'].unique()
all_2024_predictions = pd.DataFrame() # To collect 2024 actuals and forecasts for
global plot

for item in item_codes:
    print(f"\n========== ItemCode: {item} ==========")
    item_df = df[df['ItemCode'] == item].copy()

    weekly = item_df.groupby('Week')['ConsumptionQty'].sum().to_timestamp()

    if len(weekly) <= 2:
.")מדלג ARIMA - לא מספיק נתונים שבועיים עבור מודל        print("⛔
        continue

    train = weekly[weekly.index.year <= 2023]
    test = weekly[weekly.index.year == 2024]
```

```
    if test.empty:
.")מדלג. זה ט"\מקבור עבור 2024 לשנת צריכה נתוני אין        print("⚠️
        continue

    if train.empty:
.")מדלג. זה ט"\מקבור עבור 2024) לפני (אימון נתוני אין        print("⚠️
        continue

    try:

        if len(train) < 3:
d=1.")מוגדר ADF. לא מספיק לבדיקת נתוני אימון. {len(train)} יש        print(f"
            d_value = 1
        else:
            adf_result = adfuller(train)
            p_value = adf_result[1]
            d_value = 0 if p_value < 0.05 else 1
            print(f"p-value: {p_value:.4f} → d = {d_value}")
    except Exception as e:
d=1.")מוגדר {item}: {e}. ט"\מקבור עבור (ADF) שגיאה בבדיקת תחנתיות        print(f"
        d_value = 1

    try:
        model = ARIMA(train, order=(1, d_value, 1))
        model_fit = model.fit()

        forecast_2024_series = model_fit.forecast(steps=len(test))
        forecast_2024_series.index = test.index

        mae = mean_absolute_error(test, forecast_2024_series)
        rmse = np.sqrt(mean_squared_error(test, forecast_2024_series))
        print(f"MAE: {mae:.2f}")
        print(f"RMSE: {rmse:.2f}")

        cmp_df_for_plot = pd.DataFrame({
            "SimDate": test.index,
            "ConsumptionQty": test.values,
            "PredictedQty": forecast_2024_series.values
        })
        cmp_df_for_plot['ItemCode'] = item
        cmp_df_for_plot['ConsumptionYear'] = 2024

        history_for_bottom_plot = pd.DataFrame({
            "SimDate": train.index,
            "ConsumptionQty": train.values
        })
```

```
        future_forecast_for_plot = pd.DataFrame({
            "SimDate": forecast_2024_series.index,
            "PredictedQty": forecast_2024_series.values
        })
        future_forecast_for_plot['ItemCode'] = item # Add ItemCode for consistent
merging later
        future_forecast_for_plot['ConsumptionYear'] =
future_forecast_for_plot['SimDate'].dt.year

        plot_consumption(
            sku=item,
            cmp_df=cmp_df_for_plot,
            hist_df=history_for_bottom_plot,
            future_forecast_df=future_forecast_for_plot, # This will currently only have
2024 forecast
            test_year=2024,
            model_name="ARIMA"
        )

summary_stats.append({
            "ItemCode": item,
            "MAE": round(mae, 2),
            "RMSE": round(rmse, 2),
            "Percent Within Range (Original Bounds)": round(pct_in_range_display, 2),
            "Min Level (Final)": int(sim_metrics["min_final"]),
            "Max Level (Final)": int(sim_metrics["max_final"]),
            "Orders Placed": sim_metrics["orders_n"],
            "End Year OK (Original Bounds)": sim_metrics["end_ok"]
        })

    except Exception as e:
 {item}: {e}")ט"\מק עבור בסימולציה או במודל קריטית שגיאה        print(f"

def plot_global_diagnostics(all_df: pd.DataFrame, title_year: int = 2024):

    gdf = all_df.copy()

    gdf['Date'] = gdf['SimDate']
    gdf["Actual"] = gdf["ConsumptionQty"]
    gdf["Forecast"] = gdf["PredictedQty"]
    gdf = gdf[gdf["Date"].dt.year == title_year]

    if gdf.empty:
.")מדלג {title_year}. לשנת גלובליים גרפים עבור נתונים מספיק אין        print(f"\n
        return
```

```
weekly = (gdf.groupby("Date")[["Actual", "Forecast"]]
          .sum().reset_index())
plt.figure(figsize=(14, 5))
plt.plot(weekly["Date"], weekly["Actual"],
         marker="o", label="Σ Actual")
plt.plot(weekly["Date"], weekly["Forecast"],
         marker="x", label="Σ Forecast")
plt.title(f"TOTAL Weekly Consumption – All SKUs ({title_year})")
plt.xticks(rotation=45);
plt.grid(alpha=0.3);
plt.legend();
plt.tight_layout()
plt.show()

preds = gdf["Forecast"].values
actual = gdf["Actual"].values
errors = actual - preds

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))
fig.suptitle(f"Global Forecast Diagnostics ({title_year})",
             fontsize=14, fontweight="bold")


ax1.scatter(preds, actual, alpha=0.4, s=15)

if len(preds) > 0 and len(actual) > 0:
    lims = [min(preds.min(), actual.min()),
            max(preds.max(), actual.max())]
    ax1.plot(lims, lims, "r--", alpha=0.8)
ax1.set_xlabel("Predicted");
ax1.set_ylabel("Actual")
ax1.set_title("Predicted vs Actual");
ax1.grid(alpha=0.3)

ax2.hist(errors, bins=30, alpha=0.7,
         color="skyblue", edgecolor="black")
ax2.axvline(errors.mean(), color="red", ls="--",
            label=f"Mean = {errors.mean():.1f}")
ax2.set_xlabel("Error (Actual − Predicted)")
ax2.set_ylabel("Frequency")
ax2.set_title("Error Distribution")
ax2.legend();
ax2.grid(alpha=0.3)

plt.tight_layout()
plt.show()
```

```
if not all_2024_predictions.empty:
    plot_global_diagnostics(all_2024_predictions, title_year=2024)
else:
    print("\n
```

לא נוצרו תחזיות 2024 עבור אף מק"\ט. לא ניתן להציג גרפים גלובליים(".

Prophet

```
from __future__ import annotations
import math, warnings
from collections import deque
from typing import Dict, List
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statistics import NormalDist
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import openpyxl

warnings.filterwarnings("ignore")
plt.rcParams["font.family"] = ["Arial Unicode MS", "Tahoma", "DejaVu Sans"]

def show_global_metrics_and_plot(results: dict[str, dict],
                                 year: int = 2024,
                                 title: str = "Prophet"):
    frames = [r["cmp"][["Date", "Actual", "Forecast"]].copy()
              for r in results.values()]
    gdf = pd.concat(frames, ignore_index=True)
    gdf["Date"] = pd.to_datetime(gdf["Date"])
    gdf = gdf[gdf["Date"].dt.year == year]
    actual, preds = gdf["Actual"].values, gdf["Forecast"].values
    errors  = actual - preds
    abs_err = np.abs(errors)
    mae  = mean_absolute_error(actual, preds)
    mse  = mean_squared_error(actual, preds)
    rmse = math.sqrt(mse)
```

```
    r2   = r2_score(actual, preds)
    mape = np.nanmean(abs_err / np.where(actual == 0, np.nan, actual)) * 100
    wmape = abs_err.sum() / actual.sum() * 100
    mean_d = actual.mean()
    print(f"\n=== {title} – global metrics {year} ====================")
    print(f"Mean weekly demand : {mean_d:,.2f}")
    print(f"MAE              : {mae:,.2f}  ({mae/mean_d*100:.1f} %)")
    print(f"MSE              : {mse:,.2f}")
    print(f"RMSE             : {rmse:,.2f}")
    print(f"R²               : {r2:.3f}")
    print(f"MAPE             : {mape:.2f} %")
    print(f"wMAPE            : {wmape:.2f} %")

print("============================================================
==\n")
    weekly = gdf.groupby("Date")[["Actual", "Forecast"]].sum().reset_index()
    plt.figure(figsize=(14, 4))
    plt.plot(weekly["Date"], weekly["Actual"],   marker="o", label="Actual")
    plt.plot(weekly["Date"], weekly["Forecast"], marker="x", ls="--",
label="Forecast")
    plt.title(f"{title} – TOTAL Weekly Consumption ({year})")
    plt.xticks(rotation=45)
    plt.legend(); plt.tight_layout(); plt.show()


def weeks_from_days(days: float | int) -> int:
    return max(1, math.ceil(days / 7))


def dynamic_bounds(series: pd.Series, q: float = 0.30, k: float = 1.5) -> tuple[float,
float]:
    s = series.dropna()
    if s.empty:
        return 1.0, 10.0
```

```
    min_l = np.percentile(s, q * 100)
    max_l = min_l + k * s.std(ddof=0)
    return float(max(min_l, 1)), float(max_l)

class ProphetInvCover:
    def __init__(self,
        date_col: str = "ConsumptionDate",
        qty_col: str = "ConsumptionQty",
        inv_col: str = "InventoryBalance",
        inv_date_col: str = "InventoryDocDate",
        lt_col: str = "LT_Days",
        sku_col: str = "ItemCode",
        test_year: int = 2024,
        min_raw_rows: int = 10,
        min_model_rows: int = 30):
        self.date_col = date_col
        self.qty_col = qty_col
        self.inv_col = inv_col
        self.inv_date_col = inv_date_col
        self.lt_col = lt_col
        self.sku_col = sku_col
        self.test_year = test_year
        self.min_raw_rows = min_raw_rows
        self.min_model_rows = min_model_rows
        self.data: Dict[str, Dict] = {}
        self.results: Dict[str, Dict] = {}

    @staticmethod
    def load(path: str | None = None, df: pd.DataFrame | None = None) ->
pd.DataFrame:
        if df is not None:
```

```
        return df.copy()
    return (
        pd.read_excel(path)
        if path.lower().endswith((".xls", ".xlsx"))
        else pd.read_csv(path, encoding="utf-8")
    )


def prepare(self, df: pd.DataFrame, q: float = 0.30, k: float = 1.5):
    df.columns = (
        df.columns.astype(str)
        .str.strip()
        .str.replace("\u00A0", "", regex=False)
        .str.replace("\t", "", regex=False)
    )
    df[self.date_col] = pd.to_datetime(df[self.date_col], dayfirst=True,
errors="coerce")
    df[self.inv_date_col] = pd.to_datetime(df[self.inv_date_col], dayfirst=True,
errors="coerce")
    df = df.dropna(subset=[self.date_col])
    df["_year_"] = df[self.date_col].dt.year
    tr_df = df[df["_year_"] < self.test_year]
    te_df = df[df["_year_"] == self.test_year]
    for sku, grp in tr_df.groupby(self.sku_col):
        if len(grp) < self.min_raw_rows:
            continue
        tr_w = grp.groupby(pd.Grouper(key=self.date_col, freq="W-
SAT"))[self.qty_col].sum().reset_index()
        te_w = te_df[te_df[self.sku_col] ==
sku].groupby(pd.Grouper(key=self.date_col, freq="W-
SAT"))[self.qty_col].sum().reset_index()
        min_dyn, max_dyn = dynamic_bounds(tr_w[self.qty_col], q, k)
        inv_cols = [self.inv_date_col, self.inv_col, self.lt_col]
```

```
        if {"MinLevel", "MaxLevel"}.issubset(df.columns):
            inv_cols += ["MinLevel", "MaxLevel"]
        inv_hist = df[df[self.sku_col] ==
sku].dropna(subset=[self.inv_date_col]).sort_values(self.inv_date_col)[inv_cols].rena
me(columns={self.inv_date_col: "Date"})
        old_min = inv_hist["MinLevel"].dropna().iloc[-1] if "MinLevel" in inv_hist
else np.nan
        old_max = inv_hist["MaxLevel"].dropna().iloc[-1] if "MaxLevel" in inv_hist
else np.nan
        min_use, max_use = (
            (old_min, old_max)
            if (not np.isnan(old_min) and old_min >= 1)
            else (min_dyn, max_dyn)
        )
        self.data[sku] = {
            "train": tr_w,
            "test": te_w,
            "inv": inv_hist,
            "min": min_use,
            "max": max_use,
            "old_min": old_min,
            "old_max": old_max,
        }

    def forecast(self, sku: str) -> bool:
        if len(self.data[sku]["train"]) < self.min_model_rows:
            return False
        tr = self.data[sku]["train"].rename(columns={self.date_col: "ds", self.qty_col:
"y"})
        tr["floor"] = 0
        m = Prophet(yearly_seasonality=True, weekly_seasonality=True,
daily_seasonality=False)
```

```
        m.fit(tr)
        first_sat = pd.Timestamp(f"{self.test_year}-01-01")
        first_sat += pd.DateOffset(days=(5 - first_sat.weekday()) % 7)
        future_dates = pd.date_range(first_sat, periods=52, freq="W-SAT")
        future = pd.DataFrame({"ds": future_dates, "floor": 0})
        fcst = m.predict(future)[["ds", "yhat"]].rename(columns={"ds": "Date", "yhat":
"Forecast"})
        fcst["Forecast"] = fcst["Forecast"].clip(lower=0)
        te = self.data[sku]["test"].rename(columns={self.date_col: "Date", self.qty_col:
"Actual"})
        cmp = fcst.merge(te, on="Date", how="left").fillna({"Actual":
0}).sort_values("Date")
        mae = mean_absolute_error(cmp["Actual"], cmp["Forecast"])
        mse = mean_squared_error(cmp["Actual"], cmp["Forecast"])
        rmse = math.sqrt(mse)
        r2 = r2_score(cmp["Actual"], cmp["Forecast"])
        mape = (np.mean(np.abs((cmp["Actual"] - cmp["Forecast"]) /
np.where(cmp["Actual"] == 0, np.nan, cmp["Actual"]))) * 100)
        self.results[sku] = {
            **self.data[sku],
            "fcst": fcst,
            "cmp": cmp,
            "mae": mae,
            "mse": mse,
            "rmse": rmse,
            "r2": r2,
            "mape": mape,
        }
        return True


    def simulate(self, sku: str, service_z: float = 2.05, review_weeks: int = 4,
cover_weeks: int = 8, stretch_pct: float = 0.15):
```

```
        r = self.results[sku]
        fcst, inv_hist = r["fcst"].copy(), r["inv"]
        minL, maxL = r["min"], r["max"]
        lt_days = (inv_hist[self.lt_col].dropna().quantile(0.95) if self.lt_col in inv_hist
else 7)
        L = weeks_from_days(lt_days)
        sigma_err = (r["cmp"]["Actual"] - r["cmp"]["Forecast"]).std(ddof=0)
        SS = service_z * (sigma_err if not np.isnan(sigma_err) else 0) * math.sqrt(L)
        inv0 = inv_hist[self.inv_col].dropna().iloc[-1] if not inv_hist.empty else maxL
        arrivals = deque([0.0] * L)
        sim = fcst.copy()
        for col in ["Inv_Begin", "OrderQty", "ArriveQty", "Inv_End"]:
            sim[col] = 0.0
        inv, orders_n = inv0, 0
        for i in range(len(sim)):
            arrived_now = arrivals.popleft()
            inv += arrived_now
            sim.at[i, "ArriveQty"] = arrived_now
            sim.at[i, "Inv_Begin"] = inv
            inv = max(inv - sim.at[i, "Forecast"], 0)
            sim.at[i, "Inv_End"] = inv
            if i % review_weeks:
                arrivals.append(0)
                arrivals.rotate(-1)
                continue
            demand_LT = sim["Forecast"].iloc[i : i + L].sum()
            demand_cov = sim["Forecast"].iloc[i + L : i + L + cover_weeks].sum()
            target_stock = demand_LT + demand_cov + SS
            inv_proj = inv + sum(arrivals) - demand_LT
            order_qty = max(0, target_stock - inv_proj)
            if inv_proj < (minL + SS):
```

```
            minL *= (1 - stretch_pct)
        if inv + order_qty > maxL:
            maxL *= (1 + stretch_pct)
        arrivals.append(order_qty)
        arrivals.rotate(-1)
        orders_n += int(order_qty > 0)
        sim.at[i, "OrderQty"] = order_qty
    pct_weeks = sim["Inv_End"].between(minL, maxL).mean()
    end_ok = int(minL <= sim["Inv_End"].iloc[-1] <= maxL)
    self.results[sku].update({
        "sim": sim,
        "pct_weeks": pct_weeks,
        "end_ok": end_ok,
        "orders_n": orders_n,
        "min_final": minL,
        "max_final": maxL,
        "L_weeks": L,
        "lt_days": lt_days,
    })

def plot_consumption(self, sku: str):
    cmp = self.results[sku]["cmp"].copy()
    test_year = self.test_year
    cmp = cmp[cmp["Date"].dt.year == test_year]
    preds = cmp["Forecast"]
    actual = cmp["Actual"]
    hist_df = self.data[sku]["train"].rename(columns={self.date_col: "Date",
self.qty_col: "y"})
    fig1 = plt.figure(figsize=(15, 8))
    fig1.suptitle(f"{sku} – Weekly Consumption (≤{test_year - 1} train, {test_year}
forecast)", fontsize=14, fontweight="bold")
```

```
ax1 = fig1.add_subplot(2, 1, 1)
ax1.plot(cmp["Date"], actual, marker="o", label=f"Actual {test_year}")
ax1.plot(cmp["Date"], preds, marker="x", label=f"Forecast {test_year}")
ax1.legend()
ax1.grid(alpha=0.3)
ax1.set_xticklabels(cmp["Date"].dt.strftime("%Y-%m-%d"), rotation=45)
ax2 = fig1.add_subplot(2, 1, 2)
ax2.plot(hist_df["Date"], hist_df["y"], color="gray", alpha=0.6, label=f"History
{hist_df['Date'].dt.year.min()}-{test_year - 1}")
ax2.plot(cmp["Date"], actual, marker="o", label=f"Actual {test_year}")
ax2.plot(cmp["Date"], preds, marker="x", label=f"Forecast {test_year}")
ax2.legend()
ax2.grid(alpha=0.3)
ax2.set_xticklabels(hist_df["Date"].dt.strftime("%Y-%m-%d"), rotation=45)
plt.tight_layout(rect=[0, 0, 1, 0.94])
plt.show()
errors = actual.values - preds.values
fig2 = plt.figure(figsize=(14, 5))
fig2.suptitle(f"{sku} – Forecast Error Diagnostics ({test_year})", fontsize=14,
fontweight="bold")
ax3 = fig2.add_subplot(1, 2, 1)
ax3.scatter(preds, actual, alpha=0.6)
lims = [min(preds.min(), actual.min()), max(preds.max(), actual.max())]
ax3.plot(lims, lims, "r--", alpha=0.8)
ax3.set_xlabel("Predicted")
ax3.set_ylabel("Actual")
ax3.set_title("Predicted vs Actual")
ax3.grid(alpha=0.3)
ax4 = fig2.add_subplot(1, 2, 2)
ax4.hist(errors, bins=20, alpha=0.7, color="skyblue", edgecolor="black")
```

```
        ax4.axvline(errors.mean(), color="red", ls="--", label=f"Mean =
{errors.mean():.1f}")
        ax4.set_xlabel("Error (Actual − Predicted)")
        ax4.set_ylabel("Frequency")
        ax4.set_title("Error Distribution")
        ax4.legend()
        ax4.grid(alpha=0.3)
        plt.tight_layout()
        plt.show()

    def plot_projection(self, sku: str):
        r = self.results[sku]
        sim = r["sim"]
        plt.figure(figsize=(10, 4))
        plt.plot(sim["Date"], sim["Inv_End"], label="Inventory", lw=1.6)
        plt.axhline(r["old_min"], ls="--", label="Old Min")
        plt.axhline(r["old_max"], ls="--")
        plt.axhline(r["min_final"], lw=2, label="New Min")
        plt.axhline(r["max_final"], lw=2, label="New Max")
        orders_mask = sim["OrderQty"] > 0
        plt.scatter(sim.loc[orders_mask, "Date"], sim.loc[orders_mask, "Inv_End"],
label="Order Placed", s=40)
        arrivals_mask = sim["ArriveQty"] > 0
        plt.scatter(sim.loc[arrivals_mask, "Date"], sim.loc[arrivals_mask, "Inv_End"],
label="Arrived", s=60)
        plt.title(f"{sku} – Weeks in range {r['pct_weeks']:.0%}")
        plt.grid(alpha=0.3)
        plt.legend()
        plt.tight_layout()
        plt.show()
```

```
    def run(self, plot: bool = False) -> pd.DataFrame:
        rows = []
        for idx, sku in enumerate(self.data, 1):
            if not self.forecast(sku):
                continue
            self.simulate(sku)
            res = self.results[sku]
            rows.append({
                "SKU": sku,
                "MAE": round(res["mae"], 2),
                "RMSE": round(res["rmse"], 2),
                "R2": round(res["r2"], 3),
                "MAPE%": round(res["mape"], 1),
                "PctWeeksInBounds": round(res["pct_weeks"] * 100, 1),
                "EndYearOK": res["end_ok"],
                "OrdersPlaced": res["orders_n"],
                "MinOrig": res["old_min"] if not np.isnan(res["old_min"]) else res["min"],
                "MaxOrig": res["old_max"] if not np.isnan(res["old_max"]) else
res["max"],
                "MinFinal": int(res["min_final"]),
                "MaxFinal": int(res["max_final"]),
            })
            if plot:
                self.plot_consumption(sku)
                self.plot_projection(sku)
        return pd.DataFrame(rows)


def dual_stage_pipeline(csv_path: str = "optical_fdb_dedup.csv",
    strict_raw: int = 10,
    strict_model: int = 30,
    loose_raw: int = 3,
```

```
    loose_model: int = 12,
    plot: bool = False,
    excel_path: str = "forecast_inventory_report.xlsx"):
    df_all = ProphetInvCover.load(path=csv_path)
    strict = ProphetInvCover(min_raw_rows=strict_raw,
min_model_rows=strict_model)
    strict.prepare(df_all)
    summary_strict = strict.run(plot=plot)
    passed = set(summary_strict["SKU"])
    loose = ProphetInvCover(min_raw_rows=loose_raw,
min_model_rows=loose_model)
    loose.prepare(df_all)
    loose.data = {sku: data for sku, data in loose.data.items() if sku not in passed}
    summary_loose = loose.run(plot=plot)
    summary_all = pd.concat([summary_strict, summary_loose], ignore_index=True)
    with pd.ExcelWriter(excel_path, engine="openpyxl") as writer:
        summary_all.to_excel(writer, sheet_name="Summary", index=False)
        for sku, r in strict.results.items():
            r["cmp"].to_excel(writer, sheet_name=f"{sku}_cmp", index=False)
            r["sim"].to_excel(writer, sheet_name=f"{sku}_sim", index=False)
        for sku, r in loose.results.items():
            r["cmp"].to_excel(writer, sheet_name=f"{sku}_cmp", index=False)
            r["sim"].to_excel(writer, sheet_name=f"{sku}_sim", index=False)
    print(f"\nReport saved to: {excel_path}")
    show_global_metrics_and_plot({**strict.results, **loose.results}, title="Prophet",
year=strict.test_year)


if __name__ == "__main__":
    dual_stage_pipeline(
        csv_path="optical_fdb_dedup.csv",
        strict_raw=10, strict_model=30,
```

```
    loose_raw=3, loose_model=12,
    plot=True,
    excel_path="forecast_inventory_report.xlsx"
)
```

XGBoost

```
import warnings
import pandas as pd
import numpy as np
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from math import sqrt

warnings.filterwarnings("ignore")

df = pd.read_csv("optical_fdb_dedup.csv", encoding="utf-8")
df["ConsumptionDate"] = pd.to_datetime(df["ConsumptionDate"], errors="coerce")
df = df.dropna(subset=["ConsumptionQty", "ConsumptionDate"])
df["ConsumptionYear"] = df["ConsumptionDate"].dt.year
df["ConsumptionWeek"] = df["ConsumptionDate"].dt.isocalendar().week
df["YearWeek"] = df["ConsumptionDate"].dt.strftime("%Y-%U")

le = LabelEncoder()
df["ItemCode_encoded"] = le.fit_transform(df["ItemCode"].astype(str))

grouped = df.groupby(["ItemCode", "YearWeek"]).agg({
    "ConsumptionQty": "sum",
    "PRICE": "mean",
    "LT_Days": "mean",
    "InventoryBalance": "mean",
    "ConsumptionYear": "first",
    "ConsumptionWeek": "first",
    "ItemCode_encoded": "first",
    "ConsumptionDate": "first"
}).reset_index()
```

```
train_df = grouped[grouped["ConsumptionYear"] < 2024]
test_df  = grouped[grouped["ConsumptionYear"] == 2024]

features = ["PRICE", "LT_Days", "InventoryBalance", "ConsumptionYear",
"ConsumptionWeek", "ItemCode_encoded"]
X_train, y_train = train_df[features], train_df["ConsumptionQty"]
X_test,  y_test  = test_df[features],  test_df["ConsumptionQty"]

model = XGBRegressor(
    objective="reg:squarederror",
    n_estimators=300, learning_rate=0.1,
    max_depth=6, subsample=0.8, colsample_bytree=0.8,
    random_state=42, n_jobs=-1
)
model.fit(X_train, y_train)
test_df["PredictedQty"] = model.predict(X_test)

def plot_global(df, year=2024):
    gdf = df.copy()
    gdf["Date"] = gdf["ConsumptionDate"]
    gdf["Actual"] = gdf["ConsumptionQty"]
    gdf["Forecast"] = gdf["PredictedQty"]
    gdf = gdf[gdf["Date"].dt.year == year]
    weekly = gdf.groupby("Date")[["Actual", "Forecast"]].sum().reset_index()
    plt.figure(figsize=(14, 5))
    plt.plot(weekly["Date"], weekly["Actual"], marker="o", label="Actual")
    plt.plot(weekly["Date"], weekly["Forecast"], marker="x", ls="--",
label="Forecast")
    plt.title(f"XGBoost – Total Weekly Consumption ({year})")
    plt.xticks(rotation=45)
    plt.legend(); plt.tight_layout()
    plt.show()
    preds = gdf["Forecast"].values
```

```
    actual = gdf["Actual"].values
    errors = actual - preds
    mae = mean_absolute_error(actual, preds)
    mse = mean_squared_error(actual, preds)
    rmse = sqrt(mse)
    r2 = r2_score(actual, preds)
    mape = np.nanmean(np.abs(errors) / np.where(actual == 0, np.nan, actual)) * 100
    print(f"MAE  : {mae:.2f}")
    print(f"MSE  : {mse:.2f}")
    print(f"RMSE : {rmse:.2f}")
    print(f"R²   : {r2:.3f}")
    print(f"MAPE : {mape:.2f}%")

plot_global(test_df, year=2024)

def plot_per_item(item, train_df, test_df, year=2024):
    cmp = test_df[test_df["ItemCode"] == item].copy()
    hist = train_df[train_df["ItemCode"] == item].copy()
    cmp["Date"] = cmp["ConsumptionDate"]
    hist_plot = hist.rename(columns={"ConsumptionQty": "y"})
    hist_plot["Date"] = hist_plot["ConsumptionDate"]
    cmp_plot = cmp.assign(
        Actual=cmp["ConsumptionQty"],
        Forecast=cmp["PredictedQty"]
    )[["Date", "Actual", "Forecast"]]
    fig = plt.figure(figsize=(15, 8))
    fig.suptitle(f"{item} – Weekly Consumption (≤{year-1} train, {year} forecast)",
fontsize=14, fontweight="bold")
    ax1 = fig.add_subplot(2, 1, 1)
    ax1.plot(cmp_plot["Date"], cmp_plot["Actual"], marker="o", label=f"Actual
{year}")
    ax1.plot(cmp_plot["Date"], cmp_plot["Forecast"], marker="x", label=f"Forecast
{year}")
```

```
    ax1.legend(); ax1.grid(alpha=0.3)
    ax1.set_xticklabels(cmp_plot["Date"].dt.strftime("%Y-%m-%d"), rotation=45)
    ax2 = fig.add_subplot(2, 1, 2)
    ax2.plot(hist_plot["Date"], hist_plot["y"], color="gray", alpha=0.6, label=f"History
{hist_plot['Date'].dt.year.min()}-{year-1}")
    ax2.plot(cmp_plot["Date"], cmp_plot["Actual"], marker="o", label=f"Actual
{year}")
    ax2.plot(cmp_plot["Date"], cmp_plot["Forecast"], marker="x", label=f"Forecast
{year}")
    ax2.legend(); ax2.grid(alpha=0.3)
    ax2.set_xticklabels(hist_plot["Date"].dt.strftime("%Y-%m-%d"), rotation=45)
    plt.tight_layout(rect=[0, 0, 1, 0.94])
    plt.show()

plot_per_item(test_df["ItemCode"].iloc[0], train_df, test_df, year=2024)
```

Power BI

Financial analysis

```
import math, warnings, numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from collections import deque
warnings.filterwarnings("ignore")

TEST_YEAR           = 2024
N_TREES             = 300
N_LAGS              = 12
MIN_HISTORY_MONTHS   = 6
REVIEW_M, COVER_M    = 1, 2
STRETCH_PCT         = 0.15
SERVICE_Z           = 2.05

def months_from_days(d): return max(1, math.ceil(d/30))
def dynamic_bounds(s, q=.3, k=1.5):
    s = s.dropna()
    if s.empty: return 1., 10.
    mn = np.percentile(s, q*100);  mx = mn + k*s.std(ddof=0)
    return max(mn, 1.), mx

def format_money_short(n):
    if n >= 1_000_000:
        return f"{n/1_000_000:.1f}M"
    elif n >= 1_000:
        return f"{n/1_000:.0f}K"
    else:
        return str(int(n))

df = dataset.copy()
```

```
df["ConsumptionDate"] = pd.to_datetime(df["ConsumptionDate"], errors="coerce")
df = df.dropna(subset=["ConsumptionDate"])
df = df[df["ConsumptionQty"] > 0]

orders_all = []

for sku, g in df.groupby("ItemCode"):
    g["Year"]      = g["ConsumptionDate"].dt.year
    g["Month"]     = g["ConsumptionDate"].dt.month
    g["YearMonth"] = g["ConsumptionDate"].dt.to_period("M").astype(str)

    mon = g.groupby(["Year","Month","YearMonth"],
as_index=False).agg({"ConsumptionQty":"sum"})
    train = mon[mon["Year"] < TEST_YEAR]
    if len(train) < MIN_HISTORY_MONTHS:
        continue

    def add_feats(d):
        d = d.copy()
        for l in range(1, N_LAGS+1):
            d[f"lag_{l}"] = d["ConsumptionQty"].shift(l)
        return d

    train_f = add_feats(train).dropna()
    if train_f.empty: continue

    rf = RandomForestRegressor(n_estimators=N_TREES, min_samples_leaf=2,
random_state=42, n_jobs=-1).fit(
        train_f.drop(columns=["YearMonth","ConsumptionQty"]),
        train_f["ConsumptionQty"]
    )

    hist = train_f[["Year","Month","ConsumptionQty"]].copy()
```

```
f_rows = []
for ym in pd.period_range("2024-01","2024-12",freq="M"):
    row = {"Year":ym.year,"Month":ym.month}
    for l in range(1, N_LAGS+1):
        row[f"lag_{l}"] = hist["ConsumptionQty"].iloc[-l] if len(hist)>=l else 0.0
    qty = max(rf.predict(pd.DataFrame([row]))[0], 0.0)
    f_rows.append({"YearMonth":str(ym), "Forecast":qty})
    hist.loc[len(hist)] = {"Year":ym.year,"Month":ym.month,"ConsumptionQty":qty}
fcst = pd.DataFrame(f_rows)

orig_min, orig_max = dynamic_bounds(train["ConsumptionQty"])
if "MinLevel" in g and g["MinLevel"].dropna().size:
    orig_min = g["MinLevel"].dropna().iloc[-1]
if "MaxLevel" in g and g["MaxLevel"].dropna().size:
    orig_max = g["MaxLevel"].dropna().iloc[-1]
minL, maxL = orig_min, orig_max

lt_days = g.get("LT_Days", pd.Series(dtype=float)).median()
L = months_from_days(lt_days) if not np.isnan(lt_days) else 1

sigma = fcst["Forecast"].std(ddof=0)
SS = SERVICE_Z * sigma * math.sqrt(L) if sigma > 0 else 0.0

inv0_series = g.get("InventoryBalance", pd.Series(dtype=float)).dropna()
inv = inv0_series.iloc[-1] if inv0_series.size else maxL

arrivals = deque([0.0]*L)
sim_rows = []
for i,row in fcst.iterrows():
    inv += arrivals.popleft()
    inv = max(inv - row["Forecast"], 0.0)
```

```
        demand_LT  = fcst["Forecast"].iloc[i:i+L].sum()
        demand_cov = fcst["Forecast"].iloc[i+L:i+L+COVER_M].sum()
        target     = demand_LT + demand_cov + SS
        projected  = inv + sum(arrivals) - demand_LT
        qty = max(0.0, target - projected)

        if projected < minL + SS: minL *= 1 - STRETCH_PCT
        if inv + qty > maxL:      maxL *= 1 + STRETCH_PCT

        arrivals.append(qty)
        sim_rows.append({"YearMonth":row["YearMonth"], "OrderQty":qty})

    sim = pd.DataFrame(sim_rows)

    ord_df = sim[sim["OrderQty"] > 0].copy()
    if ord_df.empty: continue

    ord_df.insert(0,"ItemCode",sku)
    ord_df["Quantity"] = ord_df["OrderQty"].round().astype(int)
    ord_df = ord_df[ord_df["Quantity"] > 0]

    unit_price = g["PRICE"].dropna().iloc[-1] if "PRICE" in g and g["PRICE"].dropna().size
else 0.0
    ord_df["PRICE"]      = unit_price
    ord_df["TotalPrice"] = ord_df["Quantity"] * unit_price

    ord_df["OrderDate"]   = pd.to_datetime(ord_df["YearMonth"]) + pd.offsets.MonthBegin(0)
    ord_df["ArrivalDate"] = ord_df["OrderDate"] + pd.DateOffset(months=L)

    orders_all.append(ord_df[["ItemCode","OrderDate","ArrivalDate","Quantity","PRICE","T
otalPrice"]])

if orders_all:
```

```
    tbl = pd.concat(orders_all, ignore_index=True)
    tbl = tbl[pd.to_datetime(tbl["OrderDate"]).dt.year == 2024]

    top10 = (tbl.groupby("ItemCode")["TotalPrice"]
             .sum()
             .sort_values(ascending=False)
             .head(10))

    fig, ax = plt.subplots(figsize=(10, 10))

    wedges, texts, autotexts = ax.pie(top10.values,
                                      labels=top10.index,
                                      autopct=lambda pct: "",
                                      startangle=140,
                                      colors=plt.cm.tab10.colors,
                                      textprops={'fontsize': 14})

    for i, wedge in enumerate(wedges):
        angle = (wedge.theta2 + wedge.theta1) / 2
        x = wedge.r * 0.7 * np.cos(np.deg2rad(angle))
        y = wedge.r * 0.7 * np.sin(np.deg2rad(angle))
        amount = format_money_short(top10.values[i])
        ax.text(x, y, amount, ha='center', va='center', fontsize=24, color='black')


    ax.axis("equal")
    plt.tight_layout()
    plt.show()

else:
    plt.text(0.5, 0.5, "no accepted orders in 2024",
             ha='center', va='center', fontsize=30)
```

```
    plt.axis("off")
    plt.show()


import math, warnings, numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from collections import deque
warnings.filterwarnings("ignore")

TEST_YEAR           = 2024
N_TREES             = 300
N_LAGS              = 12
MIN_HISTORY_MONTHS   = 6
REVIEW_M, COVER_M    = 1, 2
STRETCH_PCT         = 0.15
SERVICE_Z           = 2.05

def months_from_days(d): return max(1, math.ceil(d/30))
def dynamic_bounds(s, q=.3, k=1.5):
    s = s.dropna()
    if s.empty: return 1., 10.
    mn = np.percentile(s, q*100);  mx = mn + k*s.std(ddof=0)
    return max(mn, 1.), mx

df = dataset.copy()
df["ConsumptionDate"] = pd.to_datetime(df["ConsumptionDate"], errors="coerce")
df = df.dropna(subset=["ConsumptionDate"])
df = df[df["ConsumptionQty"] > 0]

orders_all = []

for sku, g in df.groupby("ItemCode"):
```

```
g["Year"]      = g["ConsumptionDate"].dt.year
g["Month"]     = g["ConsumptionDate"].dt.month
g["YearMonth"] = g["ConsumptionDate"].dt.to_period("M").astype(str)

mon = (g.groupby(["Year","Month","YearMonth"], as_index=False)
        .agg({"ConsumptionQty":"sum"}))

train = mon[mon["Year"] < TEST_YEAR]
if len(train) < MIN_HISTORY_MONTHS:
    continue

def add_feats(d):
    d = d.copy()
    for l in range(1, N_LAGS+1):
        d[f"lag_{l}"] = d["ConsumptionQty"].shift(l)
    return d

train_f = add_feats(train).dropna()
if train_f.empty: continue

rf = RandomForestRegressor(n_estimators=N_TREES,
                           min_samples_leaf=2,
                           random_state=42,
                           n_jobs=-1).fit(
    train_f.drop(columns=["YearMonth","ConsumptionQty"]),
    train_f["ConsumptionQty"])


hist = train_f[["Year","Month","ConsumptionQty"]].copy()
f_rows = []
for ym in pd.period_range("2024-01","2024-12",freq="M"):
```

```
    row = {"Year":ym.year,"Month":ym.month}
    for l in range(1, N_LAGS+1):
        row[f"lag_{l}"] = hist["ConsumptionQty"].iloc[-l] if len(hist)>=l else 0.0
    qty = max(rf.predict(pd.DataFrame([row]))[0], 0.0)
    f_rows.append({"YearMonth":str(ym), "Forecast":qty})
    hist.loc[len(hist)] = {"Year":ym.year,"Month":ym.month,"ConsumptionQty":qty}
fcst = pd.DataFrame(f_rows)

orig_min, orig_max = dynamic_bounds(train["ConsumptionQty"])
if "MinLevel" in g and g["MinLevel"].dropna().size:
    orig_min = g["MinLevel"].dropna().iloc[-1]
if "MaxLevel" in g and g["MaxLevel"].dropna().size:
    orig_max = g["MaxLevel"].dropna().iloc[-1]
minL, maxL = orig_min, orig_max

lt_days = g.get("LT_Days", pd.Series(dtype=float)).median()
L       = months_from_days(lt_days) if not np.isnan(lt_days) else 1

sigma = fcst["Forecast"].std(ddof=0)
SS    = SERVICE_Z * sigma * math.sqrt(L) if sigma>0 else 0.0

inv0_series = g.get("InventoryBalance", pd.Series(dtype=float)).dropna()
inv = inv0_series.iloc[-1] if inv0_series.size else maxL


arrivals = deque([0.0]*L)
sim_rows = []
for i,row in fcst.iterrows():
    inv += arrivals.popleft()
    inv  = max(inv - row["Forecast"], 0.0)

    demand_LT  = fcst["Forecast"].iloc[i:i+L].sum()
```

```
        demand_cov = fcst["Forecast"].iloc[i+L:i+L+COVER_M].sum()
        target     = demand_LT + demand_cov + SS
        projected  = inv + sum(arrivals) - demand_LT
        qty = max(0.0, target - projected)

        if projected < minL + SS: minL *= 1 - STRETCH_PCT
        if inv + qty > maxL:      maxL *= 1 + STRETCH_PCT

        arrivals.append(qty)
        sim_rows.append({"YearMonth":row["YearMonth"], "OrderQty":qty})

    sim = pd.DataFrame(sim_rows)

    ord_df = sim[sim["OrderQty"] > 0].copy()
    if ord_df.empty: continue

    ord_df.insert(0,"SKU",sku)
    ord_df["Quantity"] = ord_df["OrderQty"].round().astype(int)
    ord_df = ord_df[ord_df["Quantity"] > 0]

    unit_price = g["PRICE"].dropna().iloc[-1] if "PRICE" in g and g["PRICE"].dropna().size
else 0.0
    ord_df["PRICE"]      = unit_price
    ord_df["TotalPrice"] = ord_df["Quantity"] * unit_price

    ord_df["OrderDate"]   = pd.to_datetime(ord_df["YearMonth"]) + pd.offsets.MonthBegin(0)
    ord_df["ArrivalDate"] = ord_df["OrderDate"] + pd.DateOffset(months=L)

    orders_all.append(ord_df[["SKU","OrderDate","ArrivalDate","Quantity","PRICE","TotalP
rice"]])

if orders_all:
    tbl = pd.concat(orders_all, ignore_index=True)
```

```
    tbl["Quarter"] = tbl["OrderDate"].dt.quarter
    q_cost = (tbl.groupby("Quarter")["TotalPrice"]
              .sum()
              .reindex([1,2,3,4], fill_value=0))
    fig, ax = plt.subplots(figsize=(15, 10), dpi=100)
    for spine in ax.spines.values():
        spine.set_visible(False)
    bars = ax.bar(["Q1","Q2","Q3","Q4"], q_cost.values,
                  color="#008fd5", width=0.4)
    ax.tick_params(axis='x', labelsize=30)
    ax.tick_params(axis='y', labelsize=30)

    ax.spines["left"].set_visible(True)
    ax.spines["bottom"].set_visible(True)



    for bar in bars:
        h = bar.get_height()
        ax.text(bar.get_x()+bar.get_width()/2, h*1.01,
                f"{h:,.0f}", ha="center", va="bottom", fontsize=30)
    plt.tight_layout()
    plt.show()
else:
    plt.text(0.5,0.5,"no accepted orders in 2024",
             ha='center', va='center', fontsize=14)
    plt.axis("off"); plt.show()
```

Main Dashboard

```
# Monthly Random-Forest Forecast  +  Cover-Time Inventory Sim
# (2024 only • Inventory plot with dynamic bounds)
import math, warnings, numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from collections import deque
warnings.filterwarnings("ignore")

TEST_YEAR           = 2024
N_TREES             = 300         # Random-Forest trees
N_LAGS              = 12          # lags in months
MIN_HISTORY_MONTHS   = 6
REVIEW_M, COVER_M    = 1, 2        # Cover-Time policy (months)
STRETCH_PCT          = 0.15
SERVICE_Z            = 2.05        # z-score for Safety-Stock

# ---------- helpers ----------
def months_from_days(d):                # LT (days) → months
    return max(1, math.ceil(d/30))

def dynamic_bounds(s, q=.3, k=1.5):       # orig MIN/MAX
    s = s.dropna()
    if s.empty:
        return 1., 10.
    mn = np.percentile(s, q*100)
    mx = mn + k * s.std(ddof=0)
    return max(mn, 1.), mx

# ---------- dataset from Power BI ----------
df = dataset.copy()
```

```
df["ConsumptionDate"]  = pd.to_datetime(df["ConsumptionDate"],  errors="coerce")
df["InventoryDocDate"] = pd.to_datetime(df.get("InventoryDocDate"),
errors="coerce")
df = df.dropna(subset=["ConsumptionDate"])

sku, g = next(iter(df.groupby("ItemCode")))

# ---------- monthly aggregation ----------
g = g[g["ConsumptionQty"] > 0]
g["Year"]      = g["ConsumptionDate"].dt.year
g["Month"]     = g["ConsumptionDate"].dt.month
g["YearMonth"] = g["ConsumptionDate"].dt.to_period("M").astype(str)

mon = (g.groupby(["Year","Month","YearMonth"], as_index=False)
        .agg({"ConsumptionQty": "sum"}))

train = mon[mon["Year"] < TEST_YEAR]            # ≤ 2023
test  = mon[mon["Year"] == TEST_YEAR]           # 2024

if len(train) < MIN_HISTORY_MONTHS or test.empty:
    plt.text(0.5, 0.5,
             "Not enough monthly history for this ItemCode",
             ha="center", va="center", fontsize=14)
    plt.axis("off"); plt.show(); quit()

def add_feats(d: pd.DataFrame):
    d = d.copy()
    for l in range(1, N_LAGS+1):
        d[f"lag_{l}"] = d["ConsumptionQty"].shift(l)
    return d
```

```
train_f = add_feats(train).dropna()
X_train = train_f.drop(columns=["YearMonth","ConsumptionQty"])
y_train = train_f["ConsumptionQty"]

rf = RandomForestRegressor(n_estimators=N_TREES,
                           min_samples_leaf=2,
                           random_state=42,
                           n_jobs=-1).fit(X_train, y_train)

# ---------- Forecast Jan-Dec 2024 ----------
hist_ext = train_f[["Year","Month","ConsumptionQty"]].copy()
pred_rows = []
for ym in pd.period_range("2024-01", "2024-12", freq="M"):
    yr, mn = ym.year, ym.month
    row = {"Year": yr, "Month": mn}
    for l in range(1, N_LAGS+1):
        row[f"lag_{l}"] = hist_ext["ConsumptionQty"].iloc[-l] if len(hist_ext) >= l else
0.0
    fc_qty = max(rf.predict(pd.DataFrame([row]))[0], 0)
    pred_rows.append({"YearMonth": str(ym), "Forecast": fc_qty})
    hist_ext.loc[len(hist_ext)] = {"Year": yr, "Month": mn, "ConsumptionQty": fc_qty}

fcst   = pd.DataFrame(pred_rows)
actual =
test[["YearMonth","ConsumptionQty"]].rename(columns={"ConsumptionQty":"Actu
al"})
cmp    = (fcst.merge(actual, on="YearMonth", how="left")
              .fillna({"Actual": 0.0}))

# ---------- Inventory simulation ----------
orig_min, orig_max = dynamic_bounds(train["ConsumptionQty"])
```

```
if "MinLevel" in g and not g["MinLevel"].dropna().empty:
    orig_min = g["MinLevel"].dropna().iloc[-1]
if "MaxLevel" in g and not g["MaxLevel"].dropna().empty:
    orig_max = g["MaxLevel"].dropna().iloc[-1]
minL, maxL = orig_min, orig_max
L_days = g.get("LT_Days", pd.Series(dtype=float)).median()
L      = months_from_days(L_days) if not np.isnan(L_days) else 1
sigma = (cmp["Actual"] - cmp["Forecast"]).std(ddof=0)
SS    = SERVICE_Z * (sigma if not np.isnan(sigma) else 0) * math.sqrt(L)
inv0_series = g.get("InventoryBalance", pd.Series(dtype=float)).dropna()
inv = inv0_series.iloc[-1] if not inv0_series.empty else maxL
arrivals = deque([0.0]*L)
sim_rows = []
for i, row in cmp.iterrows():
    inv += arrivals.popleft()
    inv  = max(inv - row["Forecast"], 0.0)
    # Cover-Time review (Safety-Stock)
    demand_LT  = cmp["Forecast"].iloc[i:i+L].sum()
    demand_cov = cmp["Forecast"].iloc[i+L:i+L+COVER_M].sum()
    target     = demand_LT + demand_cov + SS
    projected  = inv + sum(arrivals) - demand_LT
    qty        = max(0.0, target - projected)
    if projected < minL + SS: minL *= 1 - STRETCH_PCT
    if inv + qty > maxL:      maxL *= 1 + STRETCH_PCT
    arrivals.append(qty)
    sim_rows.append({"YearMonth": row["YearMonth"],
                     "Inv_End": inv,
                     "OrderQty": qty})

sim = pd.DataFrame(sim_rows)
```

```
pct_in   = sim["Inv_End"].between(minL, maxL).mean()*100
orders_n = (sim["OrderQty"] > 0).sum()
# ---------- plot ----------
fig, ax = plt.subplots(figsize=(6, 4), dpi=100)
# Inventory curve
ax.plot(sim["YearMonth"], sim["Inv_End"], color="tab:green", lw=2,
label="Inventory")
ax.axhline(orig_min, color="red",   ls="--", lw=1.5, label=f"Orig Min
{orig_min:.0f}")
ax.axhline(orig_max, color="red",   ls="--", lw=1.5)
ax.axhline(minL,     color="blue", lw=1.5, label=f"Final Min {minL:.0f}")
ax.axhline(maxL,     color="blue", lw=1.5)
ax.axhline(orig_max, color="red",   ls="--", lw=1.5, label=f"Orig Max
{orig_max:.0f}")
ax.axhline(orig_max, color="blue",   ls="--", lw=1.5, label=f"Final Max {maxL:.0f}")
# Orders
ord_idx = np.where(sim["OrderQty"] > 0)[0]
ax.scatter(sim["YearMonth"].iloc[ord_idx],
           sim["Inv_End"].iloc[ord_idx],
           color="purple", marker="o", s=70, label="Order")
# Arrivals (L months after each order) – filter None
arr_dates, arr_inv = [], []
for i in ord_idx:
    j = i + L
    if j < len(sim):
        arr_dates.append(sim["YearMonth"].iloc[j])
        arr_inv.append(sim["Inv_End"].iloc[j])
ax.scatter(arr_dates, arr_inv,
           color="orange", marker="v", s=90, label="Arrival")
ax.set_title(f"LT={L}m | Orders={orders_n} | %In-Range={pct_in:.1f}%")
ax.set_ylabel("Units"); ax.set_xlabel("Month")
```

```
ax.legend(fontsize=8)
plt.xticks(rotation=45); plt.tight_layout()
plt.show()
```

Orders Table

```
import math, warnings, numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from collections import deque
warnings.filterwarnings("ignore")
.
.
.
# --- Cover-Time simulation (months)
    arrivals = deque([0.0]*L)
    sim_rows = []
    for i,row in fcst.iterrows():
        inv += arrivals.popleft()          # arrival
        inv  = max(inv - row["Forecast"],0)
        demand_LT  = fcst["Forecast"].iloc[i:i+L].sum()
        demand_cov = fcst["Forecast"].iloc[i+L:i+L+COVER_M].sum()
        target     = demand_LT + demand_cov + SS
        projected  = inv + sum(arrivals) - demand_LT
        qty = max(0.0, target - projected)

        if projected < minL + SS: minL *= 1-STRETCH_PCT
        if inv + qty > maxL:      maxL *= 1+STRETCH_PCT

        arrivals.append(qty)
        sim_rows.append({"YearMonth":row["YearMonth"], "OrderQty":qty})
    sim = pd.DataFrame(sim_rows)
    # --- collect orders
    ord_df = sim[sim["OrderQty"]>0].copy()
    if ord_df.empty: continue
    ord_df.insert(0,"SKU",sku)
```

```
    ord_df["Quantity"] = ord_df["OrderQty"].round().astype(int)
    ord_df = ord_df[ord_df["Quantity"] > 0]
    unit_price = g["PRICE"].dropna().iloc[-1] if "PRICE" in g and
g["PRICE"].dropna().size else 0.0
    ord_df["PRICE"]      = unit_price
    ord_df["TotalPrice"] = ord_df["Quantity"]*unit_price
    ord_df["OrderDate"]   = pd.to_datetime(ord_df["YearMonth"]) +
pd.offsets.MonthBegin(0)
    ord_df["ArrivalDate"] = ord_df["OrderDate"] + pd.DateOffset(months=L)
    orders_all.append(ord_df[["SKU","OrderDate","ArrivalDate","Quantity","PRICE",
"TotalPrice"]])
#  render table
if orders_all:
    tbl = pd.concat(orders_all, ignore_index=True)
    tbl["OrderDate"]   = tbl["OrderDate"  ].dt.strftime("%Y-%m-%d")
    tbl["ArrivalDate"] = tbl["ArrivalDate"].dt.strftime("%Y-%m-%d")
    fig_h = min(0.5 + 0.28 * len(tbl), 4)
    fig, ax = plt.subplots(figsize=(6, fig_h), dpi=100)
    ax.axis('off')
    t = ax.table(cellText=tbl.values,
                 colLabels=tbl.columns,
                 cellLoc='center',
                 loc='center')
    t.auto_set_font_size(False); t.set_fontsize(8); t.scale(1,1.2)
    plt.tight_layout()
else:
    fig, ax = plt.subplots(figsize=(6,1.5))
    ax.axis('off')
    ax.text(0.5,0.5,"no accepted orders in 2024",
            ha='center', va='center', fontsize=12, fontweight='bold')
plt.show()
```

Expected Financial Report

```
import math, warnings, numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from collections import deque
.
.
.
    # ----- collect orders (>0) -------------
    ord_df = sim[sim["OrderQty"] > 0].copy()
    if ord_df.empty: continue

    ord_df.insert(0,"SKU",sku)
    ord_df["Quantity"] = ord_df["OrderQty"].round().astype(int)
    ord_df = ord_df[ord_df["Quantity"] > 0]

    unit_price = g["PRICE"].dropna().iloc[-1] if "PRICE" in g and
g["PRICE"].dropna().size else 0.0
    ord_df["PRICE"]      = unit_price
    ord_df["TotalPrice"] = ord_df["Quantity"] * unit_price

    ord_df["OrderDate"]   = pd.to_datetime(ord_df["YearMonth"]) +
pd.offsets.MonthBegin(0)
    ord_df["ArrivalDate"] = ord_df["OrderDate"] + pd.DateOffset(months=L)

    orders_all.append(ord_df[["SKU","OrderDate","ArrivalDate","Quantity","PRICE",
"TotalPrice"]])

#
==================================================================
====
#  Quarter-level aggregation & bar chart
if orders_all:
```

```
    tbl = pd.concat(orders_all, ignore_index=True)
    # רבעון מתוך OrderDate
    tbl["Quarter"] = tbl["OrderDate"].dt.quarter
    q_cost = (tbl.groupby("Quarter")["TotalPrice"]
              .sum()
              .reindex([1,2,3,4], fill_value=0))
    # plot
    fig, ax = plt.subplots(figsize=(8, 4), dpi=100)
    bars = ax.bar(["Q1","Q2","Q3","Q4"], q_cost.values,
                  color="#008fd5", width=0.6)
    ax.set_ylabel("TotalPrice")
    ax.grid(axis="y", alpha=.3)

    for bar in bars:
        h = bar.get_height()
        ax.text(bar.get_x()+bar.get_width()/2, h*1.01,
                f"{h:,.0f}", ha="center", va="bottom", fontsize=13)

    plt.tight_layout()
    plt.show()

else:
    plt.text(0.5,0.5,"no accepted orders in 2024",
             ha='center', va='center', fontsize=14)
    plt.axis("off"); plt.show()
```

Stock Status

```
import pandas as pd

import matplotlib.pyplot as plt

dataset['InventoryDocDate'] = pd.to_datetime(dataset['InventoryDocDate'])

dataset = dataset.sort_values('InventoryDocDate')

within_limits = dataset[

    (dataset['InventoryBalance'] >= dataset['MinLevel']) &

    (dataset['InventoryBalance'] <= dataset['MaxLevel'])

]


percent_within_limits = len(within_limits) / len(dataset) * 100


print(f": {percent_within_limits:.2f}%")


plt.figure(figsize=(12, 12))


plt.plot(dataset['InventoryDocDate'], dataset['InventoryBalance'], label='Inventory Balance',
color='blue', marker='o')


plt.plot(dataset['InventoryDocDate'], dataset['MinLevel'], label='Min Level', color='red',
linestyle='--')


plt.plot(dataset['InventoryDocDate'], dataset['MaxLevel'], label='Max Level', color='green',
linestyle='--')


plt.xlabel('Inventory Date')

plt.ylabel('Quantity')

plt.title('Inventory Balance vs. Min/Max Levels')

plt.legend()

plt.tight_layout()

plt.show()
```