

FENS402-S04-2023.2 ENGINEERING DESIGN PROJECT 2

**DESIGN OF AN IN-SEASON INVENTORY ALLOCATION
SYSTEM FOR APPAREL RETAILERS**

Submitted by:

Saliha Beyza Çakıcı (IE) - 20191704014

Salih Aladdin Sarıhan (IE) - 20191704006

Doruk Gül (IE) - 20191704072

Elif Nihan Koç (IE) - 20191704019

Project Supervisor: Associate Professor Mustafa Hekimoğlu

Faculty of Engineering and Natural Sciences

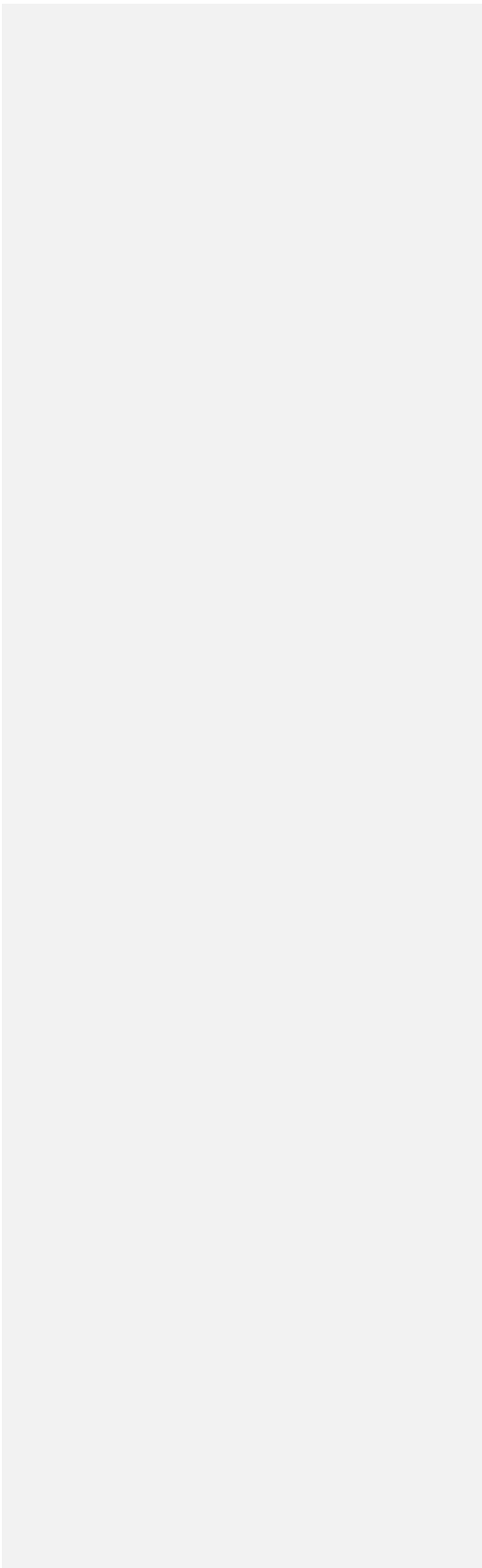
Kadir Has University

Spring 2024

LIST OF FIGURES

Figure 1.1 Process Map	8
Figure 3.1 Total Sales and Percentages of Product Groups	21
Figure 4.1 R-square Values of Regression Models for Each Product Group	23
Figure 5.1 Sales Season Chart	25
Figure 5.2 Transformation of Dynamic Programming.....	28
Figure 6.1 K-Folds 5 Cross Validation	29
Figure 6.2 Demand Estimation from Cross Validation.....	30
Figure 6.3 Roadmap of Parameter Estimation	30
Figure 6.4 Cross-Validation Results: MAPE Values for Each Product	31
Figure 6.5 $\hat{\sigma}_i^2$ Values for Each Product Group.....	33
Figure 7.1 $V(v_{ij}^2, q_{ij}^2)$	39
Figure 7.2 Optimum Second Period Scenarios	40
Figure 7.3 Optimum First Period Scenarios	41
Figure 7.4 Optimum Targets for Each Product	42
Figure 7.5 Map of Selected Cities.....	43
Figure 7.6 Inventory Allocation Rates in Adana.....	44
Figure 7.7 Inventory Allocation Rates in Ankara	44
Figure 7.8 Inventory Allocation Rates in İzmir	45
Figure 7.9 Inventory Allocation Rates in İstanbul	45
Figure 7.10 Inventory Allocation Rates in Other 9 Cities.....	46
Figure 7.11 First Period Allocation Proportion to Cities	46
Figure 7.12 Second Period Allocation Proportion to Cities.....	47
Figure 8.1 Login Page	51
Figure 8.2 Logo	51
Figure 8.3 Username and Password	52
Figure 8.4 Log in Button	53
Figure 8.5 Error Message	54
Figure 8.6 Main Page	55
Figure 8.7 Main Page-Loading Excel File	55
Figure 8.8 Main Page with Excel File	56
Figure 8.9 Main Page-Calculate Button	57
Figure 8.10 Main Page-Filter Options.....	58

Figure 8.11 Main Page-Download Button 58



LIST OF TABLES

Table 3.1 Metadata	12
Table 3.2 Description of Data Columns	13
Table 3.3 10 Sample Rows from Raw Data	13
Table 3.4 Detection of Abnormal Sales Values in the Dataset	15
Table 3.5 Exemplary Sample from Dataset	15
Table 3.6 Top 10 Best Selling Models of 2023 Season	16
Table 3.7 Basic Statistical Analysis of Total Sales Data	17
Table 3.8 Samples of Used Columns	18
Table 3.9 Samples of Independent Variable Columns for Regression	19
Table 3.10 10 Sample of Dummy Variable Table	19
Table 3.11 Product Categories with Corresponding Product Types and Dummy Variables ...	20
Table 5.1 Description of Random Variables in The Mathematical Model	24
Table 5.2 Description of State Variables in The Mathematical Model	24
Table 5.3 Description of Parameters in The Mathematical Model	25
Table 7.1 Example of Cross Validation Results	35
Table 7.2 Estimated Parameters	37
Table 7.3 Variables and Parameters For Validation	48
Table 7.4 Comparison of Second Period Cost Calculated with Excel and Python	48
Table 7.5 Comparison of Total Cost Calculated with Excel and Python	48

TABLE OF CONTENTS

<i>ABSTRACT</i>	5
<i>1 INTRODUCTION</i>	6
1.1 Motivational Case Study	6
<i>2 LITERATURE REVIEW</i>	9
<i>3 DATASET</i>	12
3.1 Raw Data	12
3.2 Preprocessing of Dataset	14
3.3 Dummy Variables	17
<i>4 DEMAND PREDICTION STUDY</i>	22
<i>5 MATHEMATICAL MODEL</i>	24
5.1 Variables and Parameters	24
5.1.1 Random Variables	24
5.1.2 State Variables	24
5.1.3 Parameters	25
5.3 Calculation of Expected Cost for the 2nd Period	26
5.4 Dynamic Programming	27
<i>6 APPLICATIONS OF MODELS</i>	29
6.1 Cross Validation	29
6.2 Mean Absolute Percentage Error	31
6.3 Parameter Estimation for Demand Distribution	32
6.3.1 Demand Distinction Using Sales Period Ratios	32
6.4 Mean Absolute Deviation and Variance Estimation	32
6.5 Utilization of Negative Binomial for Demand Distribution in the Dynamic Programming Model	33
<i>7 NUMERICAL RESULTS</i>	35
7.1 Cross Validation Results for Sweater Product Group	35

7.2 Parameter Estimation for a Product Group with Cross Validation.....	37
7.2.1 Upper Limit Calculation for State Variables.....	38
7.3 Calculation of Second Period Costs with Using Parameter Estimation.....	39
7.4 q_{ij}^2 Values That Minimize the Expected Cost According to Different v_{ij}^2 Values.....	40
7.5 Expected Cost Calculation with Dynamic Programming	41
7.6 Inventory Distribution Rates of Product Groups by Cities	43
7.7 Validation for Python Code	47
8 IMPLEMENTATION.....	50
9 CONCLUSION.....	60
APPENDIX A: DATA PROCESSING	61
A.1 Separation of Product Groups by Using Raw Data.....	61
A.2 Regression Code.....	63
A.3 Dummy Variable Code	64
A.4 Cross Validation Code	65
APPENDIX B: FIRST AND SECOND PERIOD COST CALCULATION	68
APPENDIX C: IMPLEMENTATION.....	73
C.1 Login Page.....	73
C.2 Main Page.....	75
REFERENCES	86

ABSTRACT

In today's consumer world, large-mode retail brands in the rapidly growing apparel sector may face time management and product stock problems in their inventory allocation. Stocking the products in the stores by considering the demands in the stores, especially in period-based shipments, provides flexibility in the shipments to be made after the first sales. This study aims to improve the inventory allocation practices of chain clothing brands in the retail sector. In this process, mathematical models are developed, and methodologies and decision support systems described in the existing literature are utilized. The study offers more efficient inventory allocation by optimizing demand forecasting using a data-driven dynamic programming model. In general, this study shows how to find scenarios that minimize the costs of inventory allocation and under what conditions these scenarios will occur.

Keywords: *Inventory Allocation, Data-driven Optimization, Demand Forecasting, Retail Sector, Apparel Industry*

1 INTRODUCTION

The textile retail industry is characterized by rapid shifts in consumer preferences, seasonal variations in demand, and competitive pressures. With the rise of fast fashion and e-commerce, retailers are under increased pressure to distribute products quickly and efficiently. This is why, for leading textile retailers, the initial distribution of products to stores at the beginning of a season has considerable strategic importance. This issue is particularly critical because many top companies allocate a portion of their inventories to their stores at the start of the season. Consequently, the first shipment involves varying quantities for each store the company operates. This approach is important for minimizing losses, and companies must make these strategic decisions accurately to determine the optimal proportion of the first shipment for each store. However, determining the shipment sizes is complex due to the need to consider numerous parameters and uncertainties, with changing customer demands being the most prominent. Although companies could theoretically solve this problem by evaluating each store individually and calculating the expected demand, this method is time-consuming. As a result, many demands at the beginning of the season might go unmet, leading to significant losses both early on and throughout the sales season. To address this, instead of assessing each store separately, companies should adopt an optimization strategy that considers all stores collectively. This approach enables more efficient and timely decision-making, ultimately reducing potential losses.

1.1 Motivational Case Study

Effective inventory allocation and demand forecasting are critical in today's apparel industry, where adapting to rapidly changing fashion trends and consumer preferences is essential. Continuous inventory management and planning processes aim to ensure that customer expectations are met, and a competitive advantage is maintained. Efficient planning through accurate periodic inventory demand forecasting also enhances store management and logistics processes.

Despite advancements, many companies still struggle with accurate demand forecasting, resulting in excess inventory or stockouts. This highlights a significant gap in current methods tailored to the apparel industry's unique challenges [1]. Our study aims to address this gap by

improving period-based demand forecasting for LC Waikiki, enhancing the efficiency of the inventory process and competitive advantage. Specifically, our objectives are to optimize store-based demand forecasting to save time and reduce expenditures by dynamically forecasting future demand using sales data from different product groups, stores, and sizes for the summer season of 2023.

In our model, we utilize data from LC Waikiki, employing statistical analysis and dynamic programming methods. The significance of our study lies in its contributions to both academic literature and practical applications. Academically, it addresses the gap in effective demand forecasting methods in the apparel industry. Practically, it offers companies a robust model to enhance inventory allocation, reducing costs associated with overstocking and understocking and improving overall operational efficiency. As a result, our study aims to increase the profitability, customer satisfaction, and competitiveness of companies.

Accordingly, Figure 1.1 depicts the roadmap of our study's processes and the applications used in these processes. The figure illustrates in detail how stages from data processing, mathematical model development and optimization, to application processes are integrated and interact with each other. This visual clearly outlines the logical flow of our model and what tools and techniques are used at each step, allowing readers to better understand the scope and methodology of our work.

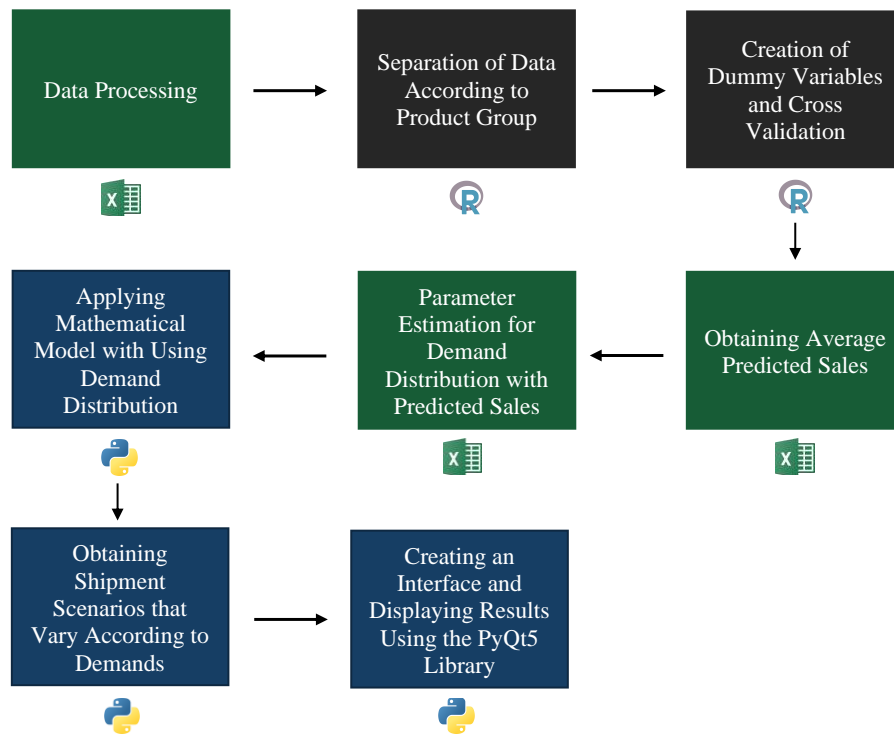


Figure 11+.1 Process Map

2 LITERATURE REVIEW

This section provides comprehensive literature research by considering statistical models for inventory allocation in the apparel industry, serving multiple sales points. The increase in fast consumption in the apparel business has made retail inventory allocation more crucial. Due to maximum demand uncertainty, initial inventory shipments to stores are strategically important in this process. Moreover, initial shipment optimization plays a role in improving inventory allocation practices and increasing sales for leading textile retailers like LC Waikiki.

[2] presented the design, implementation, and testing of a new system to help Zara improve first shipments to stores for approximately 8,000 new product introductions each year. In the study, data-driven methods were utilized to generate distributional forecasts for the demand for new products across all stores and to measure the reduction in demand uncertainty stemming from the initial few days of sales. Optimization models were presented that capture demand learning dynamics, the pooling role of store inventory, and the interactions between store display stock and product sizes. As a result of this study, it is estimated that the seasonal sales by this system will increase by approximately 2.2%, and the number of unsold products at the end of the regular sales season will be reduced by 4.3%. The study highlights the importance of efficiently distributing products from stores to stores in initial shipments, reflecting companies' commitment to minimum loss policies and deciding which stores should have how much initial stock.

[3] consider the inventory stocking problem in a 2-stage distribution system where one store serves retailers simultaneously. The risk pooling effect is analytically defined in the scenario of parallel stores with independent demands. Risk pooling is a supply chain management strategy that reduces uncertainty and unpredictability in supply and demand by combining inventory or capacity across multiple locations, commodities, or periods.

The study by [4] adopts a heuristic method based on solving a lower-bound problem by analyzing a distribution system with preliminary demand information. This study demonstrates that systems with advanced demand information have lower inventory levels and associated costs. Furthermore, the model developed in this study provides a framework for measuring system performance based on varying factors such as risk pool, pre-demand information, store location, and shipment lead times.

Their research [5] shed light on the fact that the demand quantities of products sold in retail stores vary among all stores based on criteria such as cultural differences, demographic structure,

geographical conditions, and local economy. To address this, they developed a dynamic stochastic optimization model that calculates the changing demand quantities depending on these parameters and determines the optimal size of inventory allocation to different stores. This research provides valuable insights into the importance of tailoring inventory allocation strategies to the specific characteristics of each store, which can significantly improve inventory management in the apparel industry.

[6] developed a customer demand test for the new season by collaborating with one women's clothing retailer and two large footwear retailers specialized in their fields and sending new products to the market to some selected stores before the first sales season. However, in this study, no study was conducted on the inventory management system of retailers; only the issue of pre-determination of customer demand was addressed.

The study by [7] examines the multi-period inventory allocation problem in a single store and the multi-retailer environment with lost sales for an associated demand environment. The study aims to develop a model that can minimize expected lost sales and holding costs. The study also shows that the Lagrangian relaxation heuristic of the inventory allocation problem in the context of demand forecasting can reduce the computationally challenging two-stage inventory allocation problem to separate single-retailer inventory ordering problems.

[8] reported that to obtain the latest market information on products in the fashion industry, which has a short life cycle and demand uncertainty, many fashion companies apply allocation methods, including multiple replenishment and initial allocation. In this study, the authors model the allocation process as a multi-stage system with multiple inputs and outputs to measure efficiency, one of the most essential criteria for performance evaluation in supply chain management.

The subsequent study by [9] tackled the common challenge of rebalancing inventory across the retail network to better align with supply and demand. The problem was formulated as a complex mixed integer linear program, and the primal-dual approach in the Lagrangian relaxation method was used to establish upper bounds. The research results demonstrated that the meta-heuristic method used in the study outperformed commercial optimizers, which can often produce sub-optimal solutions with much more significant optimality gaps of up to 300%. Instead, the meta-heuristic method consistently produced solutions with optimality gaps of around 7%. This research not only provides a practical solution to the inventory rebalancing challenge but also offers a series

of numerical tests to illustrate how operational procedures can affect system performance, thereby providing valuable management information.

3 DATASET

3.1 Raw Data

Within our study, we collaborated with a major retail chain in the fashion industry to obtain a comprehensive dataset for the 2023 season. As detailed in Table 3.1, this dataset comprises 334,709 rows and 9 columns. These columns include Store Name, Buyer Group, Class Group, Class Name, Model Name, Color Description, Corrected Average Daily Department Stock, Sales Quantity, and Department Turnover Rate.

The column names and contents of our dataset are presented in detail in Table 3.2. This table explains the data type contained in each column. To better illustrate the diversity and scope of our dataset, Table 3.3 presents a random sample of 10 rows. This table visually demonstrates the breadth and detail of our dataset, showcasing the variety of data upon which our analyses are based.

This sample table clearly shows the various components of our dataset and how these components come together to form a comprehensive analysis base. This sample table clearly shows the various components of our dataset and how these components come together to form a comprehensive analysis base

Table 3.1 Metadata

Number of Rows	334.709
Number of Columns	9
Number of Unique Store	542
Number of Unique Buyer Group	9
Number of Unique Class Group	19
Number of Unique Class	31
Number of Unique Model	443
Number of Unique Color	181

Table 3.2 Description of Data Columns

<i>Column_Name</i>	<i>Definition</i>
Store_Name	Names of stores and stores in Turkey by province
Buyer_Group	Contains the group or category to which the recipient belongs
Class_Group	Represents a more detailed grouping used to organize products within the company
Class_Name	Provides more detailed names or identifiers for classification groups specified in the Class_Group column
Model_Name	Contains model names or identifiers of products
Color_Description	Defines the colors of the products
ADDD_Ave_Stock	The weighted average of the number of shares purchased per hour during the day
Sales_Quantity	The seasonal sales number of products
Dep_T_R	Dividing the average stock by sales

Table 3.3 10 Sample Rows from Raw Data

<i>Store_ Name</i>	<i>Buyer_ Group</i>	<i>Class_ Group</i>	<i>Class_ Name</i>	<i>Model_ Name</i>	<i>Color_ Description</i>	<i>ADDD_ Ave_Stock</i>	<i>Sales_ Quantity</i>	<i>DEP - T_R</i>
ADN DAM BARRIER ROAD	OUTER WEAR	WOVEN VEST	VEST	YLK, KAGOSA -TR	BEIGE	5	21	4
AYD KUSADASI MALL	OUTER WEAR	WOVEN JACKET	PU THIN JACKET	MNT, KOMBER -TR	TOBACCO	5	2	0
BLC CENTER	KNIT WEAR	KNITTED BODY- TSHI7RT S. S	BASIC KNITTED POLO NECK T-SHIRT	KK, TSH, ROTKI -MEL	LIGHT GREY MELAN	20	6	0

CNK BIGA PAKTPLUS MALL	BLAZER	WOVEN BLAZER	BLAZER	CKT, DOK -ENKA	GREY	5	4	1
DEN CINAR	OUTER WEAR	WOVEN JACKET	PU THIN JACKET	MNT, KOMBER- TR	TOBACCO	6	9	1
ZNG MARKET	KNIT WEAR	KNITTED SWEATER	KEY KNITTED SWEATER	KZK, VALE	NEW BLACK	4	6	1
ELZ GAZI AVENUE	TIE/ BOWTIE	ACCESSORY TIE/BOWTIE	KEY ACCESSORY TIE/BOWTIE SMART	KRV, ERAS- INCE-KRVT	BLACK	2	7	3
YLV KIPA	KNIT WEAR	KNITTED BODY- TSHIRT S. S	BASIC KNITTED TANK NECK T-SHIRT	KK, TSH, DUBAR-POCKET	ECRU	1	-1	-1
IST AIRPORT	OUT WEAR	WOVEN JACKET	SYNTHETIC THIN JACKET	MNT, KACHET- TR	NAVY	2	4	2
KON KULU	KNIT WEAR	KNITTED VEST	KEY KNITTED VEST	YLK, SOFI-Y	NAVY	6	4	1

3.2 Preprocessing of Dataset

In the data analysis process, the columns of the dataset were first analyzed by heading. Then, the analysis continued by focusing on the *Model_Names* and *Sales_Quantity* columns. These columns indicated the unique identities of the products and their sales performance. Python programming language and Panda's library were used for data analysis.

In our dataset, we identified significant omissions and anomalies in the sales counts of some products. As shown in detail in Table 3.4, we detected sales value anomalies and their percentages in the data. In our dataset, a total of 14.826 products did not have sales counts recorded, and there are 13.341 products with a sales count of -1, 873 products with a sales count of -2, 123 products

with a sales count of -3, 13 products with a sales count of -4, 3 products each with a sales count of -5. These negative and nan values at the columns are data anomalies that can seriously affect data integrity and accuracy during analysis. To prevent the negative effects of these anomalies, in the analysis, we replaced the data entries with nan and negative values in the sales column with a value of 0. This made the dataset more consistent and reliable. The diversity of the sales columns of the updated dataset is shown in detail in Table 3.5 below. With this adjustment, we aim to obtain more reliable results in our analysis.

Table 3.4 Detection of Abnormal Sales Values in the Dataset

<i>Sales Quantity</i>	<i>Total Count</i>	<i>%</i>
Blank	14.826	4.42 %
-1	13.341	3.98 %
-2	837	0.25 %
-3	123	0.03 %
-4	13	0.003 %
-5	3	0.0009 %

Table 3.5 Exemplary Sample from Dataset

<i>Model Name</i>	<i>Sales Quantity</i>
CKT, DOK-BUTTER	6
KK.TSH, DUBAR	430
PNT, VENUS-2	53
KK.TSH, HATO	-9
PNT, RELATE	30
ATL, B.Y, BAYRA	1
MNT, KALIBU-YD	0
CKT, DOK-ROBE	NaN

CKT, DOK-ROBE	1
KK.TSH, B.Y, BEJA-23S	216

The primary reason for filling these values with 0 is the potential impact of nan and negative-valued rows on the analysis process, which could lead to misleading results. Therefore, the initial aim is to ensure data integrity by replacing nan values with 0. Secondly, negative sales figures may not be realistic and could indicate possible data entry errors or incomplete information. Thus, by replacing these negative values with 0, the accuracy of the analysis can be enhanced.

As a result, columns with gaps and negative sales numbers were converted to 0 to increase the consistency of the dataset and the reliability of the analysis results. In line with these adjustments, the following analyses were performed.

In Table 3.6, an analysis was conducted to identify the product segments that were the most and least demanded during the summer season. According to the analysis, among the segments that reached the highest sales figures in the summer season, the models of the KNITTED BODY-TSHIRT SHORT-SLEEVE (KK.TSH) and WOVEN PANTS (PNT.) CHINO classification groups stand out. At the same time, the total sales quantity of the top 10 best-selling products for the entire season constitutes 37.74% of the total products sold during that season.

Table 3.6 Top 10 Best Selling Models of 2023 Season

<i>Model Name</i>	<i>Total Number of Sales</i>
KK.TSH, DUBAR	830568
KK.TSH, ROTKI-MEL	347039
KK.TSH, ROTKI	332960
KK.TSH, SONO	278459
KK.TSH, B.Y, BEJA-23S	238855
KK.TSH, BEDAR	232591
PNT, MARLO	215566
PNT, LEVEL	191694
KK.TSH, GOLVIP	144166
PNT, VENUS	137280

In another step, the statistical properties of the dataset were examined, and an analysis of the data distribution was performed. This analysis considered critical statistical properties of the dataset, such as minimum, maximum, quartile values, median, and mean; as seen in Table 3.7, the minimum value of the dataset is 0, and the maximum value is 830568. The median value was 4993, and the mean was 17639.208. This significant difference between the median and the mean indicates that the distributions in the dataset are asymmetric and right skewed. According to the quartile values, the first quartile is 1680, and the third quartile is 14902. In this case, the interquartile range is calculated as 13222. In this context, a detailed evaluation of the statistical properties of the dataset has been made.

Table 3.7 Basic Statistical Analysis of Total Sales Data

Minimum Value	0
First Quartile (25th Percentile)	1680
Median Value (50th Percentile)	4993
Third Quartile (75th Percentile)	14902
Maximum Value	830568
Interquartile Range (Q3 – Q1)	13222
Average	17639.208

3.3 Dummy Variables

After analyzing the baseline data, our study continued with regression modeling to explore relationships among features. For this aim, the dummy variable method was used and applied to the prepared dataset. A dummy variable, also known as an Indicator Variable, is a synthetic variable represented by 0s and 1s that denotes a characteristic with multiple categories or levels [10]. The purpose of using this method was to perform regression analysis by transforming categorical data into a numerical format using dummy variables.

When we were creating dummy variables; first identified dependent and independent variables. In our study, the total sales column was identified as the dependent variable; independent variables were created by store_name, model_name, and color_description columns. Table 3.8 below provides examples of the columns used. At this stage, we decided to combine the

model_name and color_description columns with the store_name column since the store_name column has a dominant diversity ratio compared to the other columns. Otherwise, we anticipated that this dominance could potentially reduce the impact of the model's name and color description variables on the model. Therefore, to alleviate this problem and ensure model efficiency, we integrated store names with model names and color descriptions as combined categories. Secondly, while providing this integration, we have divided the products into 12 product groups as T-shirt (TSH), sweater (KZK), sweat (SWT), shorts (ŞRT), undershirt (ATL), vest (YLK), bermuda (BRD), pants (PNT), coat (MNT), jacket (CKT), shirt (GMK) and necktie (KRV) by using the model_name column.

Table 3.8 Samples of Used Columns

<i>Store_Name</i>	<i>Model_Name</i>	<i>Color_Description</i>
ANT_KEMER	KZK, LETHER	NEW BLACK
ANT KEMER	KZK, BARON	DARK PETROL
ANT KEMER	KZK, BARON	DULL GREEN
ANT KEMER	KZK, BARON	STONE MELANGE
ANT KEMER	KZK, OLIVER	BEIGE

As can be seen in Table 3.8, the model_name column contains product group and model definitions together. In our study, we created separate columns for these definitions and categorized our analysis according to product groups. KZK expression while delivering the product group; the expressions Leather, Baron, and Oliver describe the model. In our study, we created separate columns for these definitions and categorized our analyses according to product groups. Then, we created our first independent variable, store_name_product_grp, by combining the representations of the product groups with our store_name column. Next, we created the Wareh_Name_Model_Grp column by concatenating the model definitions and the store_name column and the Wareh_Name_Color_Desc column by concatenating the color definitions and the store_name column. Table 3.9 contains an example of the independent variables we used for the regression.

Table 3.9 Samples of Independent Variable Columns for Regression

<i>Store_Name_Product_Grp</i>	<i>Store_Name_Model_Grp</i>	<i>Store_Name_Color_Desc</i>
ANT_KEMER_KZK	ANT_KEMER_LETHER	ANT_KEMER_NEW_BLACK
ANT_KEMER_KZK	ANT_KEMER_BARON	ANT_KEMER_DARK_PETROL
ANT_KEMER_KZK	ANT_KEMER_BARON	ANT_KEMER_DULL_GREEN
ANT_KEMER_KZK	ANT_KEMER_BARON	ANT_KEMER_STONE_MELANGE
ANT_KEMER_KZK	ANT_KEMER_OLIVER	ANT_KEMER_BEIGE

Once all our variables were defined, we converted each definition created in the argument columns into a property for the dummy variable table, including products that accounted for 80% of each product group's total sales to manage categorical data effectively. Thanks to this approach, we ensured that products with little or no sales did not compromise the model's efficiency or cause significant deviations in the forecasts.

Finally, as seen in Table 3.10, we created our dummy variable table by checking each intersection cell in each column and labeling it with 1 if the feature is present in the row and 0 if it is not.

Table 3.10 10 Sample of Dummy Variable Table

<i>Store_Name_Product_Grp</i>	<i>Store_Name_Model_Grp</i>	<i>Store_Name_Color_Desc</i>	<i>Total_Sales</i>	<i>ANT_KEMER_SWT</i>	<i>ANT_KEMER_KANPO</i>	<i>ANT_KEMER_ZERO</i>	<i>ANT_KEMER_DARK_BEIGE</i>	<i>ANT_KEMER_BUXE_WHITE</i>
ANT_KEMER_SWT	ANT_KEMER_KANPO	ANT_KEMER_DARK_BEIGE	27	1	1	0	1	0
ANT_KEMER_SWT	ANT_KEMER_ZERO	ANT_KEMER_BUXE_WHITE	26	1	0	1	0	1

In the table 3.11, we examined the dummy table we created according to the product type and number of dummy variables in the product categories. In this regard, the product category with the most product types is T-Shirt; the product category with the most dummy variables is Shirt.

This shows us that shirts are the most diverse product category when we consider the store, color and model together.

Table 3.11 Product Categories with Corresponding Product Types and Dummy Variables

<i>Product Category</i>	<i>Number of Product Types in the Category</i>	<i>Number of Dummy Variables in Category</i>
UNDERSHIRT	2450	2371
BERMUDA	4565	4454
JACKET	1130	1899
SWEATER	395	751
SHIRT	8500	11609
NECKTIE	624	1467
COAT	4060	5115
PANTS	9979	7334
SHORTS	5185	5966
SWEAT	2615	3852
T-SHIRT	10917	9977
VEST	1135	2287



<i>Product Groups</i>	<i>Total Sales</i>	<i>%</i>
T-Shirt	1469052	44.46%
Pants	619058	18.73%
Shirt	315739	9.55 %
Sort	252602	7.64 %
Bermuda	186255	5.64%
Coat	159478	4.83%
Undershirt	102519	3.10 %
Sweat	97518	2.95 %
Vest	48969	1.48%
Jacket	34463	1.04 %
Sweater	11790	0.36 %
Necktie	6995	0.22 %

Figure 3.1 Total Sales and Percentages of Product Groups

Figure 3.1 presents the total sales numbers and percentages of the product groups. According to this table, the highest-selling product group is T-Shirts, accounting for 44.6% of total sales. In contrast, the product group with the lowest sales for the season is neckties, comprising only 0.22% of total sales.

4 DEMAND PREDICTION STUDY

4.1 Regression

Regression is a statistical technique that helps us understand the relationship between a dependent variable and one or more independent variables. It helps us measure the statistical significance of the predicted relationships between variables. Statistical significance assesses the predicted accuracy of the relationship between variables, providing us with a level of confidence in determining how well the actual relationship fits the predicted relationship [11].

In this study, a logarithmic regression model was employed using the R programming language to measure the relationship between Total_Sales, which was determined as the dependent variable in our study, and the independent variables. For this purpose, first, the natural logarithm (ln) of the dependent variable was taken. However, one point to note is that the logarithms of products with zero Total_Sales are undefined because the logarithm of zero is undefined. To address this issue, we increased the Sales_Quantity values by 1 for models with zero Total_Sales, thereby resolving the problem of undefined logarithms. This adjustment allowed the model to make more reliable predictions and assess the relationships between variables more accurately [12].

In the subsequent analysis, the model's performance was evaluated statistically by measuring the relationship between independent variables and the dependent variable. We determined the statistical significance of each independent variable using regression results. The residual standard error indicates the extent to which model predictions deviate from actual values. R^2 and adjusted R^2 values show the proportion of variance in the dependent variable explained by the model. The F-test evaluates the overall performance of the model, and the p-value of each marginal hypothesis test of regression covariates determines the statistical significance of each coefficient within the regression model. Analyzing these values provides important information about the reliability and predictive ability of the model, helping to evaluate the effectiveness of the regression analysis.

In the analysis, the Multiple R-squared values, which measure the model's compatibility with real data, were observed to be above 0.5 across all product groups. These values indicate that a large proportion of the variation in the dependent variable can be predicted from the independent variables, demonstrating a high explanatory power. Additionally, the fact that these values are not

equal to 1 indicates that there is no overfitting. This shows that the model does not overfit the training data. From this, we understand that the model maintains its generalizability and works correctly [13].

Similarly, the p-values for each dataset were observed to be less than 0.05, indicating that the model is statistically significant for each dataset. This implies that the ability of the independent variables to explain the dependent variable is not due to chance, confirming their importance.

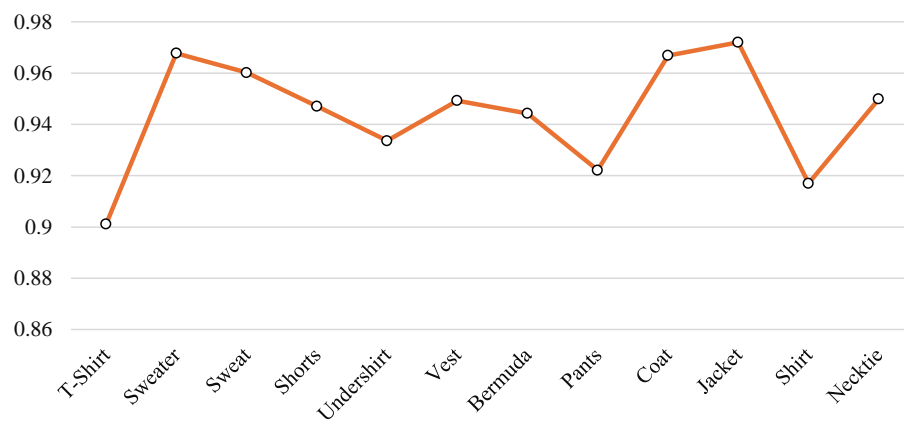


Figure 4.1 R-square Values of Regression Models for Each Product Group

5 MATHEMATICAL MODEL

5.1 Variables and Parameters

5.1.1 Random Variables

Table 5.1 defines the demand variables used in our mathematical model. These demands were used as random variables with Negative Binomial Distribution in the model. \widehat{D}_{ij}^1 represents the demand quantity of product i at Store j during the 1st period. Similarly, \widehat{D}_{ij}^2 represents the demand quantity of product i at Store j during the 2nd period.

Table 5.1 Description of Random Variables in The Mathematical Model

\widehat{D}_{ij}^1	<i>Random demand at first period</i>
\widehat{D}_{ij}^2	<i>Random demand at second period</i>

5.1.2 State Variables

Table 5.2 introduces v_{ij}^1 , indicating initial quantity of product i at Store j in the 1st period, and q_{ij}^1 , the quantity of product allocated to Store j during that period. Similarly, v_{ij}^2 and q_{ij}^2 represent these variables for the 2nd period.

Table 5.2 Description of State Variables in The Mathematical Model

q_{ij}^1	<i>Initial shipment to each store</i>
q_{ij}^2	<i>Second shipment to each store</i>
v_{ij}^1	<i>Units of inventory are carried in stores at the beginning of the first period</i>
v_{ij}^2	<i>Units of inventory are carried in stores at the end of the first period</i>

5.1.3 Parameters

The variables in Table 5.3 denote the cost for loss and holding cost parameters. These parameters are considered constant for all products and all stores. For the 1st period, c^1 represents the cost for loss, and h^1 represents cost assessed per unit of unsold inventory. Similarly, c^2 and h^2 denote the cost for loss and holding cost for the 2nd period, respectively. Parameters a^1 and a^2 are also used to represent the cost corresponding to the shipment sizes in the 1st and 2nd periods.

Table 5.3 Description of Parameters in The Mathematical Model

h^1	<i>Cost assessed per unit of unsold inventory remaining in the first period</i>
h^2	<i>Cost assessed per unit of unsold inventory remaining after the second period</i>
c^1	<i>Unit cost assessed per lost sale in the first period</i>
c^2	<i>Unit cost assessed per lost sale in the second period</i>
a^1	<i>Acquisition cost for first period shipment</i>
a^2	<i>Acquisition cost for second period shipment</i>

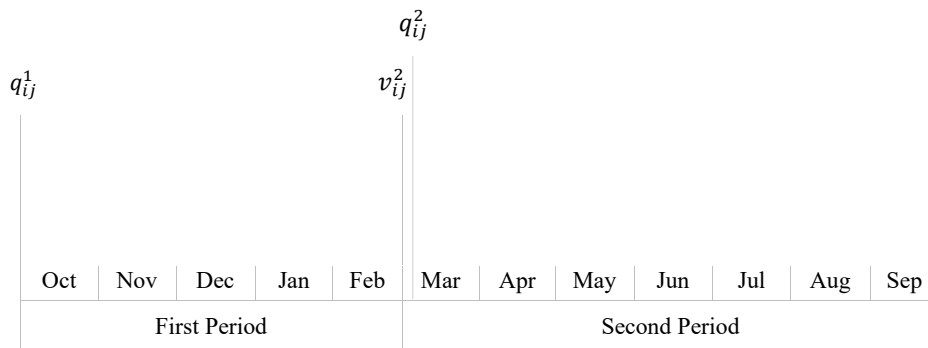


Figure 5.1 Sales Season Chart

Figure 5.1 illustrates the sales seasons during the 1st and 2nd periods and demonstrates the application of variables within the mathematical model. The 1st period spans from October to Mar, totaling 5 months, while the 2nd period covers the remaining 7 months of the year.

Eqn. (5.1) is used to calculate the total expected cost values for 1st period and determine the q_{ij}^1 quantity that minimizes the expected cost for different v_{ij}^1 values. Thus, for product i at Store j at the start of the 1st period represented by v_{ij}^1 , the shipment quantity q_{ij}^1 that minimizes the expected cost for the period is found. However, to perform this calculation, it is first necessary to compute the expected value for the 2nd period, as the initial inventory for the 2nd period depends on the v_{ij}^1 , q_{ij}^1 , and \widehat{D}_{ij}^1 values from the 1st period, thereby influencing the 2nd period based on the product inventory and demand from the 1st period. Therefore, initially, the expected cost value for the 2nd Period is calculated, and an optimum scenario is obtained from the beginning of the season using (5.1) through dynamic programming.

$$\begin{aligned} \text{Min E } [c^1 \sum_{j=1}^n (\widehat{D}_{ij}^1 - q_{ij}^1 - v_{ij}^1)^+ + h^1 \sum_{j=1}^n (q_{ij}^1 + v_{ij}^1 - \widehat{D}_{ij}^1)^+ \\ + V(v_{ij}^2, q_{ij}^2)] + a^1 q_{ij}^1 \end{aligned} \quad (5.1)$$

5.3 Calculation of Expected Cost for the 2nd Period

$$\begin{aligned} V(v_{ij}^2, q_{ij}^2) = \text{Min E } [c^2 \sum_{j=1}^n (\widehat{D}_{ij}^2 - v_{ij}^2 - q_{ij}^2)^+ \\ + h^2 \sum_{j=1}^n (v_{ij}^2 + q_{ij}^2 - \widehat{D}_{ij}^2)^+] \\ + a^2 q_{ij}^2 \end{aligned} \quad (5.2)$$

Eqn. (5.2) calculates the expected cost for the 2nd period, and it is also the detailed version of the part representing the 2nd period in (5.1). The variable v_{ij}^2 represents the quantity of product i available at Store j at the beginning of the 2nd period. Based on different values of the

v_{ij}^2 variable, the q_{ij}^2 quantity is determined, representing the shipment amount in the 2nd period and minimizing the expected cost. Therefore, the expected cost values for product i in store j are found according to the optimum scenarios in the 2nd Period.

5.4 Dynamic Programming

Dynamic programming, pioneered by Richard Bellman in 1957, is a mathematical technique designed to address complex, multi-stage decision problems by determining the optimal strategy. Bellman asserted that an optimal policy exhibits the property that, irrespective of the initial state and decisions made, subsequent decisions must conform to an optimal policy with respect to the state resulting from the initial decision. In essence, given knowledge of the current state and intended decision, any future optimal policy remains independent of prior policy decisions. Consequently, dynamic programming is primarily employed in tackling multi-stage, sequential decision problems [14].

We used the dynamic programming method to integrate the expected cost values found in the 2nd period into (5.1). The state variables v_{ij}^1 and q_{ij}^1 for the 1st period undergo a dynamic programming based on the values of the random variable \hat{D}_{ij}^1 using (5.3). The expected values for the 2nd period are then added to the calculation in (5.1). This process determines the 1st period shipment quantity that minimizes the expected cost based on the initial product quantities at the stores at the start of the season. As a result of these operations, the optimal q_{ij}^1 variable representing the shipment quantity that minimizes the expected cost for the first period is determined based on the v_{ij}^1 values, which represent the initial product quantities for product i at Store j at the beginning of the 1st season [15].

$$v_{ij}^2 = (v_{ij}^1 + q_{ij}^1 - \hat{D}_{ij}^1)^+ \quad (5.3)$$

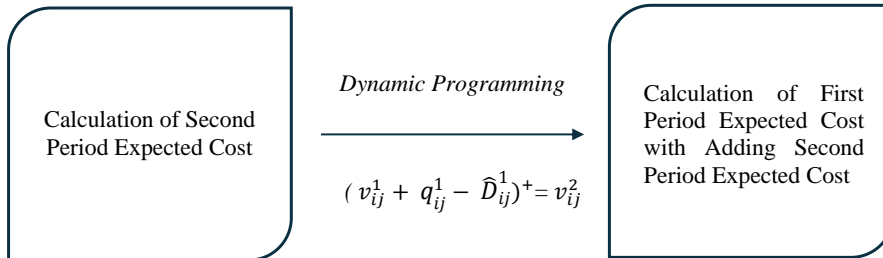


Figure 5.2 Value Iteration with Dynamic Programming

Figure 5.2 outlines the sequence of operations conducted within our mathematical model. Initially, the expected costs for the 2nd period are computed, followed by the calculation of expected values for the 1st period using dynamic programming techniques.

6 APPLICATIONS OF MODELS

6.1 Cross Validation

After making our data processable as explained in the data editing section, we first created the dummy by inserting our data into the code we wrote in R programming language. Our data was an Excel file with 12 different sheets representing 12 different product groups. We created a dummy thanks to an R program code that covers 12 different sheets of this data, and our data is ready for Cross Validation thanks to its new version that includes dummy variables. While performing Cross Validation, we again took our sales data as the dependent variable of the log linear regression and the remaining dummy variables as independent variables. We carried out all these processes for our 12 different product groups. While doing Cross Validation, we used the k-folds 5 method, that is, we divided our 12 product groups into 5 different slices [16]. We performed the same operations for all slices, considering 1 slice as test and the remaining 4 slices as train.

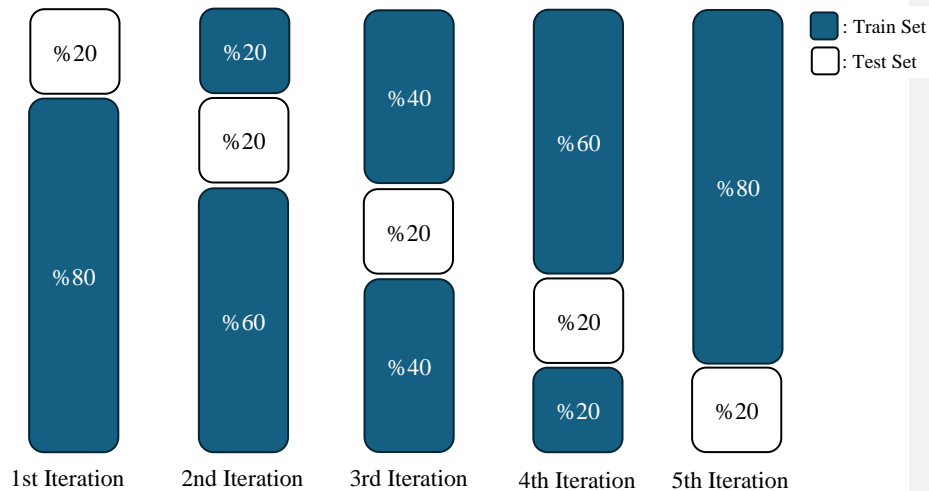


Figure 6.1 K-Folds 5 Cross Validation

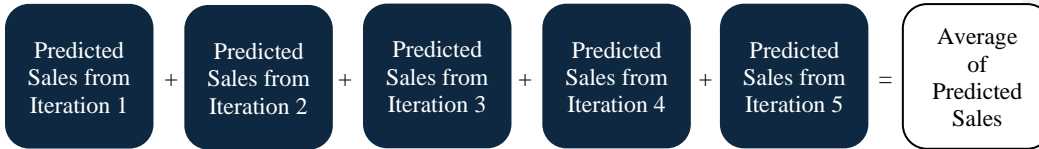


Figure 6.2 Demand Estimation from Cross Validation

Through the process shown in Figure 6.2, we obtained the expected demand forecast quantities for all products within 12 product groups. In Figure 6.3, we demonstrated how we obtained the negative binomial distribution parameters p and r for the \hat{D}_{ij}^1 and \hat{D}_{ij}^2 variables for all products using the predicted demand quantities. First, using our demand forecasts, we calculated 12 different Mean Absolute Deviation (MAD) values for the 12 different product groups. We calculated variance for product groups using these 12 different MAD values. Then, using these variances, we determined the negative binomial distribution parameters for each product.

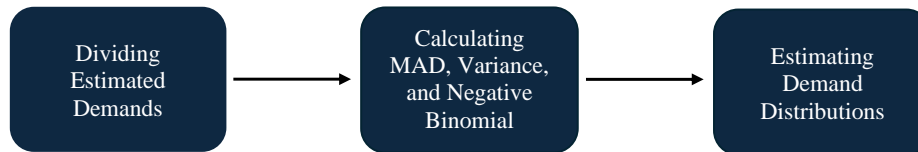


Figure 6.3 Roadmap of Parameter Estimation

As depicted in the figure 6.3, we divided the estimated demands for each product group based on historical sales data, dividing the year into two periods: the first five months as the 1st Period and the remaining seven months as the 2nd Period. For both periods, we calculated the parameters of the negative binomial distribution, denoted as r and p , using the Mean Absolute Deviation (MAD) and Variance values of the demand data. Specifically, these parameters were determined for the demand during the 1st Period \hat{D}_{ij}^1 and the 2nd Period \hat{D}_{ij}^2 . With these negative binomial parameters, we derived the distributions for the random demand variables, which were then incorporated into the Mathematical Model to accurately represent and predict demand variability.

6.2 Mean Absolute Percentage Error

We performed cross-validation on each product group by dividing them into 5 different slices using the R programming language. Through cross-validation, using past season data for each of the 12 different product groups, we obtained prediction values using a log-linear regression model. We found the predicted values of sales for all product groups. We calculated Mean Absolute Percentage Error (MAPE). We did this purely for control purposes to see that the predicted values we obtained were within acceptable deviation ranges. Eqn. (6.1) shows how we calculated the Mean Absolute Percentage Error (MAPE) value. In this calculation, the index i represents the product, and j represents the store. MAPE results for all product groups are shown in Figure 6.4 .

$$\frac{1}{n} \sum_{i=0}^n \frac{|Actual\ Sale_{ij} - Predicted\ Sale_{ij}|}{Actual\ Sale_{ij}} \quad (6.1)$$

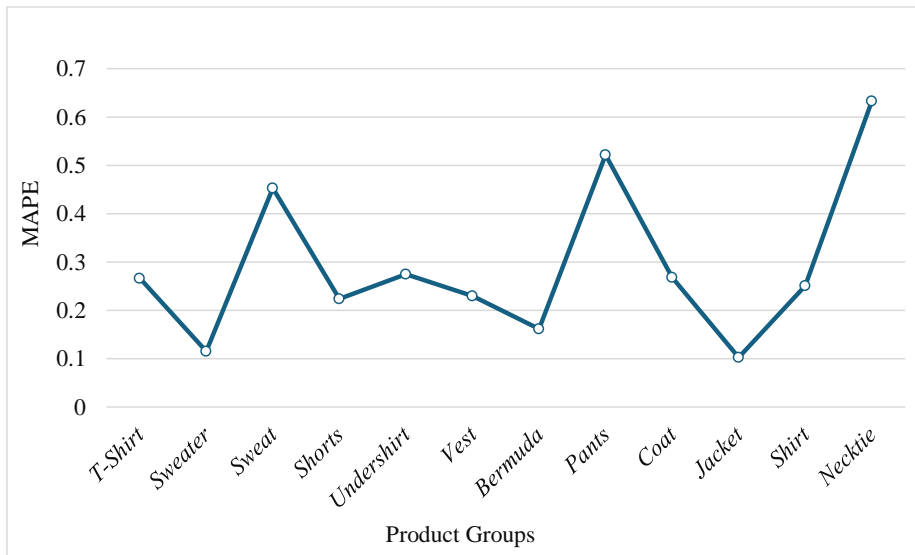


Figure 6.4 Cross-Validation Results: MAPE Values for Each Product

6.3 Parameter Estimation for Demand Distribution

Thanks to the estimated demands we obtained, we determined the negative binomial parameters of \widehat{D}_{ij}^1 , representing the demand in period 1, and \widehat{D}_{ij}^2 , representing the demand in period 2, which we used in the Mathematical Model. We obtained these parameters using Mean Absolute Deviation and Variances of the estimated Demands. Since Estimated Demands represent the products sold throughout the season, it was necessary to separate the estimated demands into two different periods, \widehat{D}_{ij}^1 and \widehat{D}_{ij}^2 .

6.3.1 Demand Distinction Using Sales Period Ratios

To make this distinction, we developed an approach for estimating demands. Using past season sales data for one product group owned by LC Waikiki, we determined the ratio of sales in the 1st period to the total sales for the entire season, as well as the ratio of sales in the 2nd period to the total sales for the whole season. By multiplying these ratios with the demand forecasts obtained from cross-validation for the products, we determined the values of \widehat{D}_{ij}^1 and \widehat{D}_{ij}^2 variables. So, with using this approach we separated \widehat{D}_{ij}^1 and \widehat{D}_{ij}^2 with past sales ratios. In following parts, we will use this approach as $q_{1,2}$. Indices 1 represents period 1 ratio and 2 represents period 2 ratio.

6.4 Mean Absolute Deviation and Variance Estimation

To estimate a demand distribution for each product group we assume that the expectation of the demand distribution is equal to the demand forecast. To estimate the variance, we utilize Mean Absolute Deviation (MAD) of each product obtained from our K-fold cross validation study, where K is set to 5. To this end, we utilize demand forecasts from cross-validation, 12 MAD values were calculated for 12 different product groups using the formula in (6.2). These MAD values were then used to calculate the variance for the 12 different product groups using (6.3) as shown in Figure 6.5. With the obtained variance values, the negative binomial distribution parameters p and r for the \widehat{D}_{ij}^1 and \widehat{D}_{ij}^2 variables were calculated for product group $i \in \wp$ at store $j \in S$, where \wp is the set of product groups and S denotes set of stores.

$$MAD_i = \frac{1}{|S|} \sum_{j=0}^{|S|} |Actual\ Sale_{ij} - Predicted\ Sale_{ij}|, \quad (6.2)$$

where $|S|$ denotes the cardinality of the store set. The estimated demand variance ($\hat{\sigma}_i^2$) of the product group i is estimated using (6.3) below. Note that this approach is also suggested for variance estimation from demand forecasts [17].

$$\hat{\sigma}_i^2 = 1.25 MAD_i \quad (6.3)$$

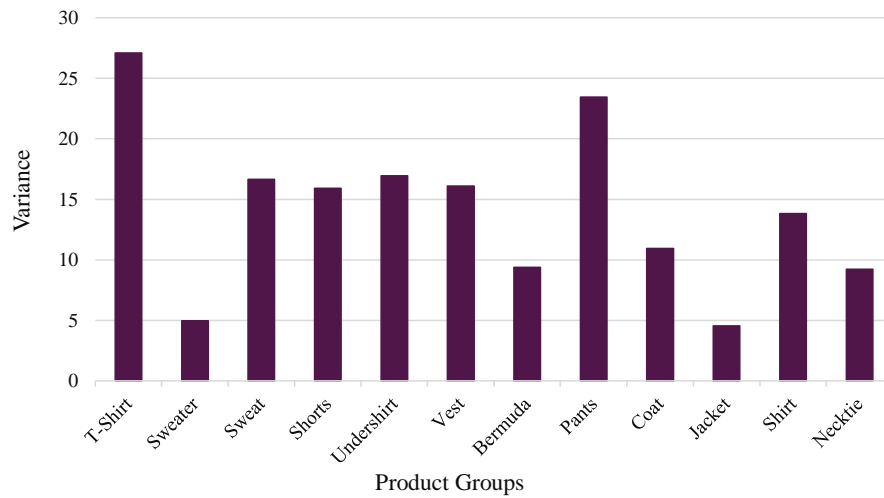


Figure 6.5 $\hat{\sigma}_i^2$ Values for Each Product Group

6.5 Utilization of Negative Binomial for Demand Distribution in the Dynamic Programming Model

As previously explained, using the Mean Absolute Deviation (MAD) and variance values obtained across 12 different product groups, we determined the parameters of the negative binomial

Formatted: Heading 2, Left, Indent: First line: 0 cm, Line spacing: single

distribution for the random variables \widehat{D}_{ij}^1 and \widehat{D}_{ij}^2 , which represent the demand quantities in the 1st and 2nd periods for a product i at store j . The parameter p in the negative binomial distribution represents the probability of success, while r represents the number of successes. Using (6.4), we calculated the predicted sales value from cross-validation and the parameter p for product i at store j . During this calculation process, we also used the variance of the product group to which the product belongs to determine the value of p [18].

The obtained p value, along with the ratios derived from the approach used to distinguish between product group variance and demand, was used in (6.5) to calculate the parameter r for product i at store j .

$$\frac{\text{Predicted Sale}_{ij}}{\widehat{\sigma}_i^2} \quad (6.4)$$

$$\frac{(q_{1,2})(\widehat{\sigma}_i^2)(p_{ij}^2)}{(1 - p_{ij})} \quad (6.5)$$

7 NUMERICAL RESULTS

7.1 Cross Validation Results for Sweater Product Group

In the Cross Validation section, we obtained the *Predicted Sales* values for all products in 12 different product groups as shown in Table 7.1. The *Product Name* column consists of the Store Name, Product Group, Model Name, and Color Description attributes. The "Product Code" section represents the unique codes in the system for each product. For example, the code "#0200006" in the system represents the product "ADN_OPTIMUM_AVM, KZK, VALE, NEW_BLACK." The *Actual Sales* column represents the total sales quantities made in the previous season for the products.

In the Numerical Result section, the KZK, VALE, NEW_BLACK product was considered, and the results were shared. These operations were also performed for other products, and the results were obtained accordingly.

Table 7.1 Example of Cross Validation Results

<i>Product Name</i>	<i>Product Code</i>	<i>Actual Sales</i>	<i>Predicted Sales</i>	<i>MAPE</i>	<i>/ Predicted Sales- Actual Sales /</i>
ADN_OPTIMUM_AVM, KZK, VALE, NEW_BLACK	#0200006	23	22.71	0.0126	0.29
ANK_ANKAMALL_2, KZK, VALE, NEW_BLACK	#0200012	23	23.02	0.0009	0.02
ANK_GMK_BULVARI, KZK, VALE, NEW_BLACK	#0200024	21	24.00	0.1429	3
ANK_NATA_VEGA_AVM, KZK, VALE, NEW_BLACK	#0200035	21	22.71	0.0814	1.71
ANK_SINCAN_CARSI, KZK, VALE, NEW_BLACK	#0200037	27	22.71	0.1589	4.29
ANT_MANAVGAT_NOVAMALL, KZK, VALE, NEW_BLACK	#0200063	21	23.02	0.0962	2.02

ANT_MARKANTALYA_AVM, KZK, VALE, NEW_BLACK	#0200072	20	23.02	0.1510	3.02
ANT_TERRACITY, KZK, VALE, NEW_BLACK	#0200083	26	26.95	0.0365	0.95
BRD_MERKEZ, KZK, VALE, NEW_BLACK	#0200089	22	22.46	0.0209	0.46
BRS_KORUPARK, KZK, VALE, NEW_BLACK	#0200093	26	26.07	0.0027	0.07
EDR_MARGI_AVM, KZK, VALE, NEW_BLACK	#0200101	23	23.22	0.0096	0.22
ESK_ESPARK_AVM, KZK, VALE, NEW_BLACK	#0200104	24	24.81	0.0337	0.81
ESK_HAMAMYOLU, KZK, VALE, NEW_BLACK	#0200105	21	22.46	0.0695	1.46
IGD_MERKEZ, KZK, VALE, NEW_BLACK	#0200117	28	22.84	0.1843	5.16
INT_INTERNET_SATIS, KZK, VALE, NEW_BLACK	#0200147	27	22.84	0.1541	4.16
IST_212_AVM, KZK, VALE, NEW_BLACK	#0200154	26	22.84	0.1215	3.16
IST_AKASYA_MALL, KZK, VALE, NEW_BLACK	#0200156	26	29.42	0.1315	3.42
IST_AKBATI_AVM, KZK, VALE, NEW_BLACK	#0200158	26	27.26	0.0485	1.26
IST_ARMONIPARK_LCW, KZK, VALE, NEW_BLACK	#0200164	22	22.68	0.0309	0.68

7.2 Parameter Estimation for a Product Group with Cross Validation

Using the obtained *Predicted Sales* values, we first calculated the Mean Absolute Deviation and Variance values for all product groups. Using these values, we determined the Negative Binomial Distribution Parameters as shown in Table 7.2. Using these parameters, we obtained the random variables \hat{D}_{ij}^1 and \hat{D}_{ij}^2 , and the state variables v_{ij}^1 , v_{ij}^2 , q_{ij}^1 and q_{ij}^2 .

Table 7.2 Estimated Parameters

<i>Product Code</i>	p_{ij}^1	p_{ij}^2	r_{ij}^1	r_{ij}^2	\hat{D}_j^1	\hat{D}_j^2	q_j^1	q_j^2	v_j^1	v_j^2
#0200006	0.91	0.91	97.28	146.35	9.07	13.64	19	28	10	29
#0200012	0.93	0.93	117.64	176.98	9.19	13.83	19	28	10	29
#0200024	0.97	0.97	280.10	421.39	9.58	14.42	20	29	10	30
#0200035	0.91	0.91	97.28	146.35	9.07	13.64	19	28	10	29
#0200037	0.91	0.91	97.28	146.35	9.07	13.64	19	28	10	29
#0200063	0.93	0.93	117.64	176.98	9.19	13.83	19	28	10	29
#0200072	0.93	0.93	117.64	176.98	9.19	13.83	19	28	10	29
#0200083	0.99	0.99	1065.16	1602.44	10.76	16.19	22	33	10	32
#0200089	0.91	0.91	85.48	128.59	8.97	13.49	18	27	10	28
#0200093	0.99	0.99	1030.50	1550.31	10.41	15.66	21	32	10	31
#0200101	0.94	0.94	134.39	202.17	9.27	13.95	19	28	10	29
#0200104	0.99	0.99	980.63	1475.27	9.91	14.90	20	30	10	30
#0200105	0.91	0.91	85.48	128.59	8.97	13.49	18	27	10	28
#0200117	0.92	0.92	105.04	158.03	9.12	13.72	19	28	10	29
#0200147	0.92	0.92	105.04	158.03	9.12	13.72	19	28	10	29

#0200154	0.92	0.92	105.04	158.03	9.12	13.72	19	28	10	29
#0200156	0.99	0.99	1163.14	1749.85	11.75	17.68	24	36	10	34
#0200158	0.99	0.99	1077.66	1621.25	10.89	16.38	22	33	10	32
#0200164	0.91	0.91	95.85	144.20	9.06	13.62	19	28	10	29

In Table 7.2, r_{ij}^1 and p_{ij}^1 represent the number of successes and the probability of success in the first period, respectively. Similarly, r_{ij}^2 and p_{ij}^2 represent the number of successes and the probability of success in the 2nd period. \widehat{D}_{ij}^1 and \widehat{D}_{ij}^2 , in order, represent the predicted sales quantities for the 1st and 2nd periods.

7.2.1 Upper Limit Calculation for State Variables

We had to develop an approach to determine the boundaries of the variables v_{ij}^1 , v_{ij}^2 , q_{ij}^1 and q_{ij}^2 . First of all, for the variables q_{ij}^1 and q_{ij}^2 representing shipment sizes, we determined the upper limits of shipment sizes by taking twice the values of \widehat{D}_{ij}^1 ve \widehat{D}_{ij}^2 in the 1st and 2nd Periods as is (7.1) and (7.2). We determined the variable v_{ij}^1 , which represents the number of products i in store j at the beginning of Period 1, by taking the difference between the actual sales number and the predicted sales number. Because this number represents the amount of product remaining at the end of the season, that is, the amount of product transferred to the new season. To increase the number of scenarios, we set the v_{ij}^1 variable to be at least 10 as in (7.3). The variable v_{ij}^2 represents the product i in store j at the beginning of the 2nd Period. We took the upper limit of this variable according to the result of $v_{ij}^1 + q_{ij}^1$. Because when the 2nd period begins, the maximum amount of product available is in the scenario where no sales were made in the 1st period. Therefore, thanks to this process, we maximized the number of possible scenarios for v_{ij}^2 as in (7.4).

$$\text{Upper Limit of } q_j^1 = \text{RoundUp}(\widehat{D}_{ij}^1 * 2, 0) \quad (7.1)$$

$$\text{Upper Limit of } q_j^2 = \text{RoundUp}(\widehat{D}_{ij}^2 * 2, 0) \quad (7.2)$$

$$\text{Upper Limit of } v_{ij}^1 = \text{Max}(\text{Actual Sales}_{ij} - \text{Predicted Sales}_{ij}, 10) \quad (7.3)$$

$$\text{Upper Limit of } v_{ij}^2 = v_{ij}^1 + q_{ij}^1 \quad (7.4)$$

Note: Table 7.2, \widehat{D}_{ij}^1 , \widehat{D}_{ij}^2 , v_{ij}^1 , v_{ij}^2 , q_{ij}^1 and q_{ij}^2 columns represent the boundaries of these variables. For example, when $q_{ij}^2 = 28$, it represents variable q_{ij}^2 can take values between (0,28).

7.3 Calculation of Second Period Costs with Using Parameter Estimation

At this stage, expected cost values for all scenarios in the second period have been calculated using the cartesian products of v_{ij}^2 and q_{ij}^2 values for each product, utilizing parameters and variables $p2$, $r2$, v_{ij}^2 , q_{ij}^2 , and \widehat{D}_{ij}^2 as shown in Table 7.2. Figure 7.1 includes only products with the KZK, VALE, and NEW_BLACK product group-model-color definitions from the "ADN_OPTIMUM_AVM" store.

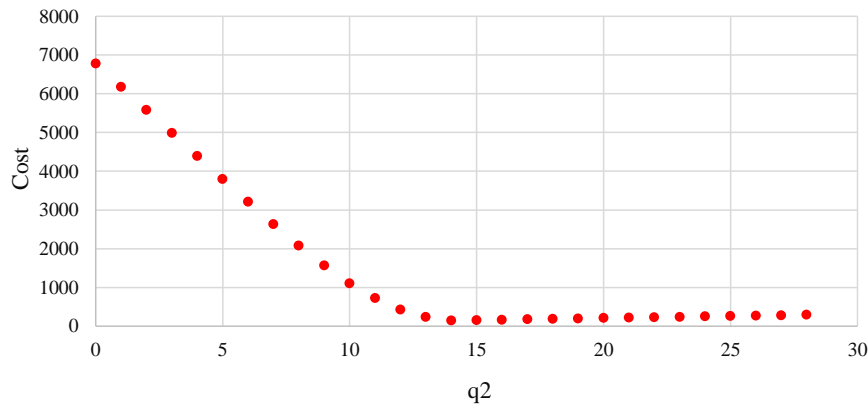


Figure 7.1 $V(v_{ij}^2, q_{ij}^2)$

Figure 7.1 is given to better explain the implementation phase of the mathematical model. This Figure shows the q_{ij}^2 values that minimize the cost in the 2nd Period when the v_{ij}^2 value of the product in the ADN_OPTIMUM_AVM store with the KZK, VALE and NEW_BLACK product group-model-color definitions is 0. This process is done by considering all scenarios according to the cartesian product of v_{ij}^2 and q_{ij}^2 , and q_{ij}^2 values that minimize the cost are found according to different v_{ij}^2 values. Figure 7.1 shows the q_{ij}^2 values that minimize the cost when v_{ij}^2 is 0.

7.4 q_{ij}^2 Values That Minimize the Expected Cost According to Different v_{ij}^2 Values

In this section, q_{ij}^2 values that minimize the cost of the 2nd period for the product with the product group-model-color definitions KZK, VALE, and NEW_BLACK at the ADN_OPTIMUM_AVM store have been determined corresponding to different v_{ij}^2 values.

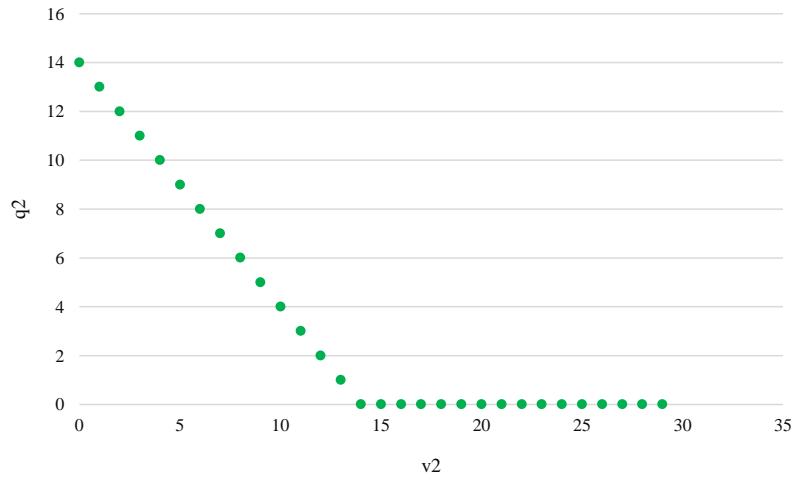


Figure 7.2 Optimum Second Period Scenarios

To provide an explanation of the values in Figure 7.2, consider the example of the product "ADN_OPTIMUM_AVM, KZK, VALE, NEW_BLACK". When $v_{ij}^2 = 7$, the corresponding q_{ij}^2 value that results in the minimum cost is also 7. In this way, the q_{ij}^2 values that minimize the cost for different v_{ij}^2 values have been determined for all products from among all scenarios generated by the Cartesian product of v_{ij}^2 and q_{ij}^2 values in Figure 7.2.

7.5 Expected Cost Calculation with Dynamic Programming

As explained earlier in the Mathematical Model section, the expected cost values from the 2nd period were integrated into (5.1) using dynamic programming through (5.3). In this way, by calculating the expected value for the 1st period at the beginning of the season, we determine the q_{ij}^1 values required to allocate from inventory according to different v_{ij}^1 values, thereby minimizing costs in the 1st period. In Figure 7.3, the optimal q_{ij}^1 quantities required to allocate from the inventory based on v_{ij}^1 values representing the initial stock of KZK, VALE, and NEW_BLACK product group-model-color definitions in the ADN_OPTIMUM_AVM store have been determined. These quantities indicate the amounts needed for effective inventory distribution in the store.

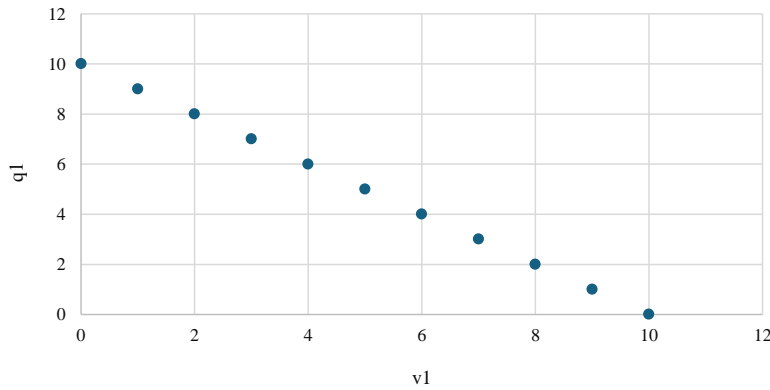


Figure 7.3 Optimum First Period Scenarios

In summary, Figure 7.3 enabled us to find the q_{ij}^1 values that minimize costs for the first period across all products, based on different v_{ij}^1 values.

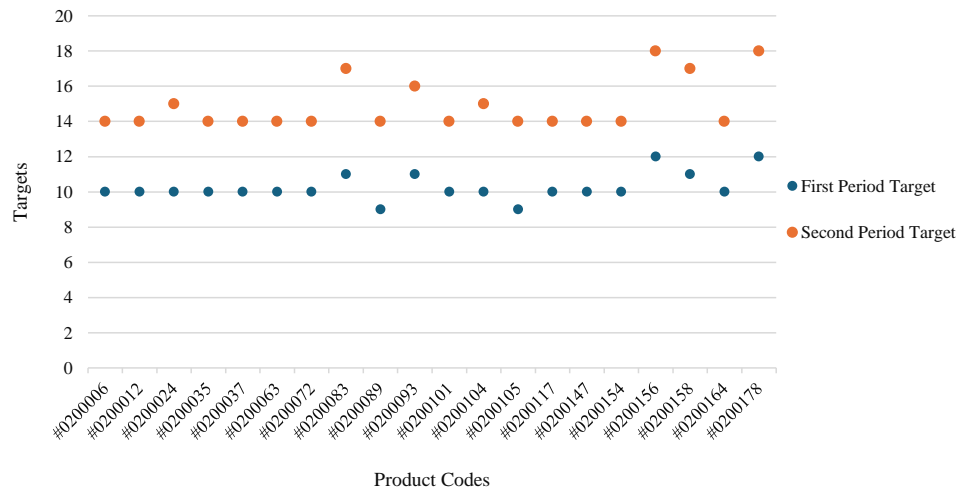


Figure 7.4 Optimum Targets for Each Product

Using the mathematical model, we calculated the optimal q_{ij}^1 quantity based on v_{ij}^1 , representing the inventory level in the store for the 1st period, and determined the optimal q_{ij}^2 quantity based on v_{ij}^2 , representing the inventory level for the 2nd period. The $v_{ij}^1 + q_{ij}^1$ and $v_{ij}^2 + q_{ij}^2$ values in the periods represent how many units of the product should be in the store in the 1st and 2nd periods, respectively. Figure 7.4 shows how many units of certain products should be in the store in the 1st and 2nd periods

7.6 Inventory Distribution Rates of Product Groups by Cities

These values represent the quantities of products that should be present in stores during the 1st and 2nd periods. Each value was determined through costs that minimize the expected value for their respective period.

In this section, we compared the values and ratios of targeted product quantities for the 1st and 2nd periods across 14 major cities for each of the 12 distinct product groups. This study aims to answer the question of how inventories across the 14 cities should be distributed between the two periods based on their total available products.

Figure 7.5 shows 14 city's that we selected: Adana, Ankara, Antalya, Bursa, Diyarbakır, Edirne, Eskişehir, Gaziantep, İstanbul, İzmir, Kayseri, Konya Muğla ve Ordu. Our selection criteria were based on specific considerations. Firstly, each selected city represents its region's most densely populated city. Another criterion was the availability of historical sales data across all product categories in these cities. In other words, there was sales data for the 14 cities we determined for all product groups. This enabled us to collect outputs for all product groups and make comparisons accordingly. While carrying out this study, we evaluated the cities of ADANA, ANKARA, ISTANBUL and IZMIR separately due to their data size and customer capacities. The sales numbers of the remaining 9 cities in the 1st and 2nd periods are added together to give the allocation percentages for period 1 and period 2 for each product group.

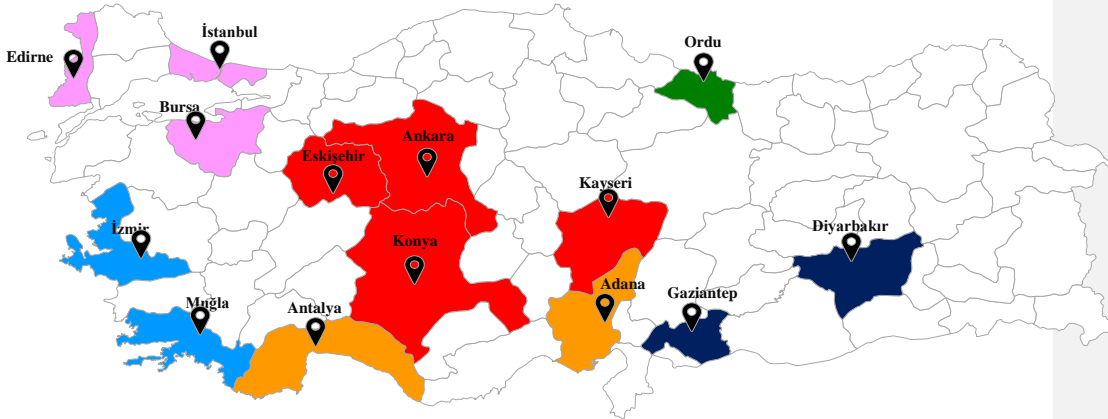


Figure 7.5 Map of Selected Cities

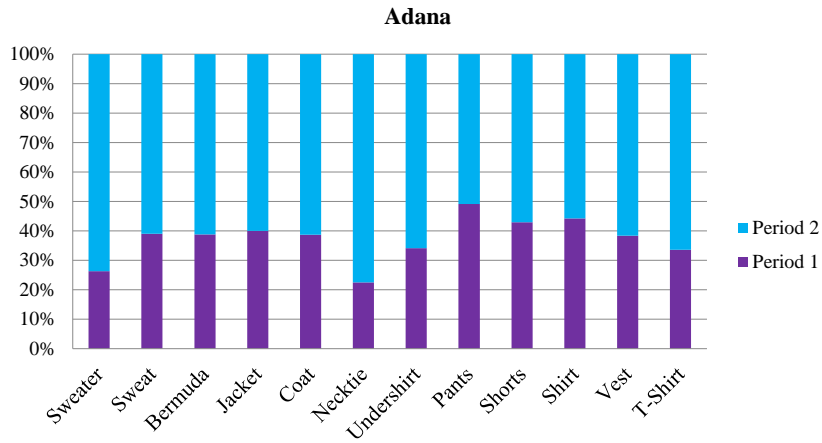


Figure 7.6 Inventory Allocation Rates in Adana

In Figure 7.6, we observe the allocation rates in the 1st and 2nd periods from the inventory in the city of Adana. Thanks to these results, allocation decisions can be made to create minimum costs for each product group in the next season.

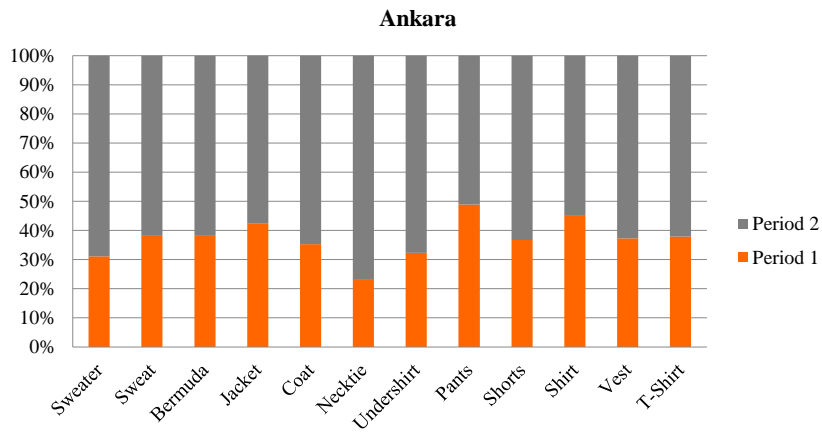


Figure 7.7 Inventory Allocation Rates in Ankara

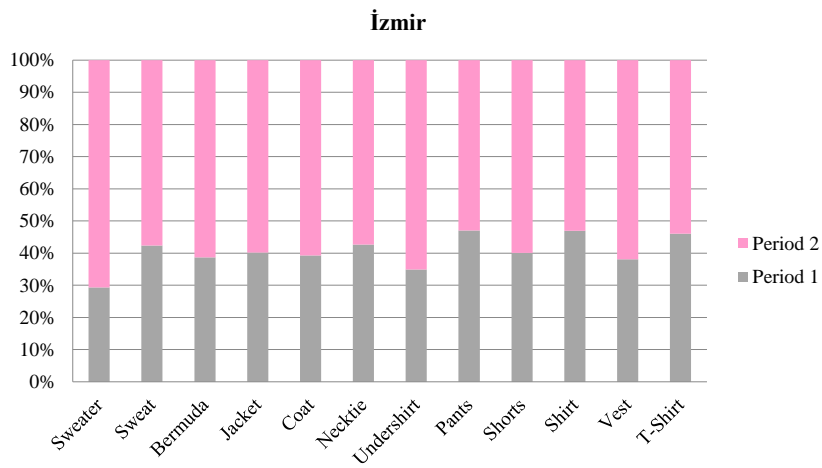


Figure 7.8 Inventory Allocation Rates in İzmir

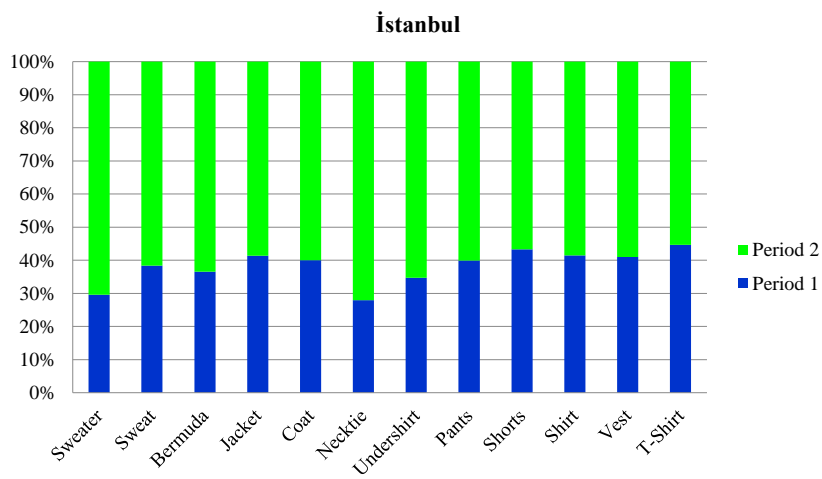


Figure 7.9 Inventory Allocation Rates in İstanbul

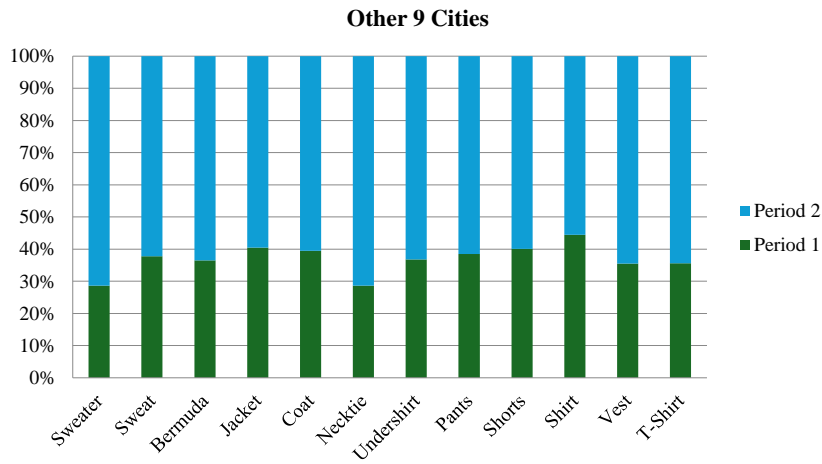


Figure 7.10 Inventory Allocation Rates in Other 9 Cities

Likewise, by using the results in Figure 7.7, Figure 7.8, Figure 7.9 and Figure 7.10, inventory allocation decisions can be made to create optimum scenarios for all cities and each product group in the next season.

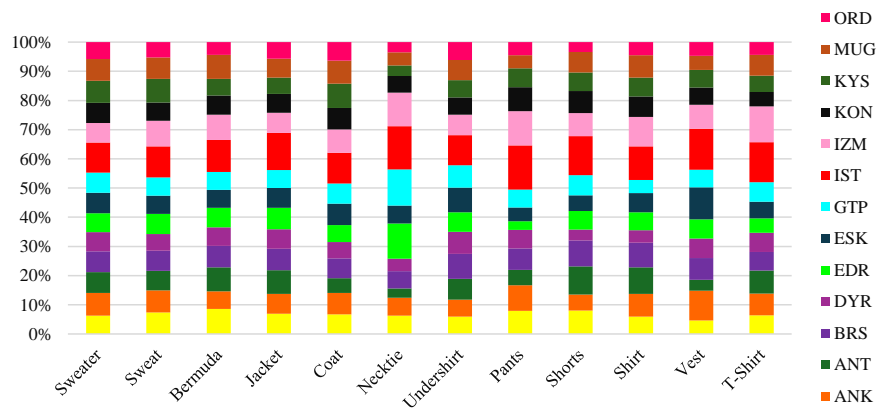


Figure 7.11 First Period Allocation Proportion to Cities

Figure 7.11 shows the proportion of product groups distributed to cities from the General Inventory in the 1st Period. For example, 12.7% of all jacket products to be sent to cities for sale in the 1st Period were sent to Istanbul.

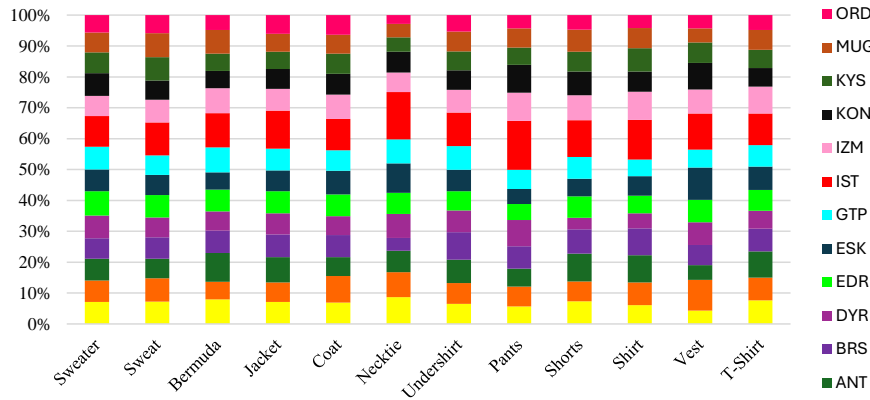


Figure 7.12 Second Period Allocation Proportion to Cities

Figure 7.12 shows the proportion of product groups distributed to cities from the General Inventory in the 2nd Period. For example, 15.8% of all trouser products to be sent to cities in the 2nd period were sent to Istanbul.

7.7 Validation for Python Code

Following cross-validation, we used the predicted values obtained for all products in 12 different product groups. With these values, we first determined the negative binomial parameters for the products, and then calculated the values of \hat{D}_{ij}^1 , \hat{D}_{ij}^2 , v_{ij}^1 , v_{ij}^2 , q_{ij}^1 , and q_{ij}^2 . Using these variables, we applied our Python-based mathematical model to derive optimal scenarios for both the 1st and 2nd periods. However, to verify the optimality of these scenarios and the accuracy of the calculated cost values, we compared the results from the Python code with those from the manually implemented mathematical model in Excel. The ranges and values of our variables and parameters were defined as described in Table 7.3 below.

Table 7.3 Variables and Parameters for Validation

\bar{D}_{ij}^1	\bar{D}_{ij}^2	v_{ij}^1	v_{ij}^2	q_{ij}^1	q_{ij}^2	h^1	h^2	c^1	c^2	a^1	a^2	p_{ij}^1	p_{ij}^2	r_{ij}^1	r_{ij}^2
(0,8)	(0,10)	(0,5)	(0,5)	(0,5)	(0,5)	15.3	17.2	100	200	10	10	0.72	0.66	3	2

Table 7.4 Comparison of Second Period Cost Calculated with Excel and Python

v_j^2	q_j^2	<i>2nd Period Cost (Excel)</i>	v_j^2	q_j^2	<i>2nd Period Cost (Python)</i>
0	6	54.46	0	6	54.47
1	5	44.46	1	5	44.46
2	4	34.46	2	4	34.46
3	3	37.96	3	3	37.96
4	2	48.30	4	2	48.30
5	1	61.55	5	1	61.55
6	0	75.99	6	0	75.99
7	0	90.90	7	0	90.91
8	0	105.99	8	0	105.99
9	0	121.15	9	0	121.16
10	0	136.34	10	0	136.33

Table 7.5 Comparison of Total Cost Calculated with Excel and Python

v_j^1	q_j^1	<i>Total Cost (Excel)</i>	v_j^1	q_j^1	<i>Total Cost (Python)</i>
0	4	117.06	0	4	117.07
1	3	107.06	1	3	107.06
2	2	97.06	2	2	97.06
3	1	87.06	3	1	87.06
4	0	92.81	4	0	92.81
5	0	111.22	5	0	111.23

Validation was made by calculating the cost in both Excel and Python using the values in Table 7.3. The 2nd Period cost values are calculated, and results obtained as in Table 7.4, then, the 1st Period cost values are calculated, and results obtained as in Table 7.5. As a result of the calculations, the results obtained from Python and the results in Excel were the same, so the results were validated.

8 IMPLEMENTATION

Interfaces are critical touchpoints where users interact with a system, device, or application. Interfaces shape the user experience, determining the usability, accessibility, and user satisfaction of an application or device [19]. The user-friendly and ergonomically designed interface contributes to the improvement and efficiency of the process by meeting the user's demands more easily. In addition, it reflects the identity of the brand by ensuring employee satisfaction. Interfaces, which also play an important role in marketing processes, aim to facilitate the user's processes by increasing brand loyalty.

In our study, we designed an interface to enhance the usability of the mathematical model we built, specifically for LC Waikiki employees. This interface aims to visualize the outputs of the model more effectively and facilitate efficient use by users.

In this process, we used the PyQt5 binding set, which is implemented as a Python extension and is a link to the cross-platform GUI (Graphical User Interface) toolkit Qt. PyQt5 acts as a bridge between Python and Qt5, providing access to the rich features, user interface development tools and functions that Qt offers, from the Python language. PyQt5 can be used to develop GUI applications on various platforms such as desktop applications, mobile applications, and even embedded systems [20, 21].

We have developed a specially designed interface for the realization of our project. While prioritizing user experience, we took care to protect the brand image of LC Waikiki. For this reason, we stayed true to LC Waikiki's corporate color tones and based the coloring on blue tones and white. The color palette used in the interface was chosen in accordance with LC Waikiki's corporate identity and was enriched with different shades of blue. This approach ensures that users will associate the interface with LC Waikiki and will establish a connection between the reliability of the brand and the quality of our project.

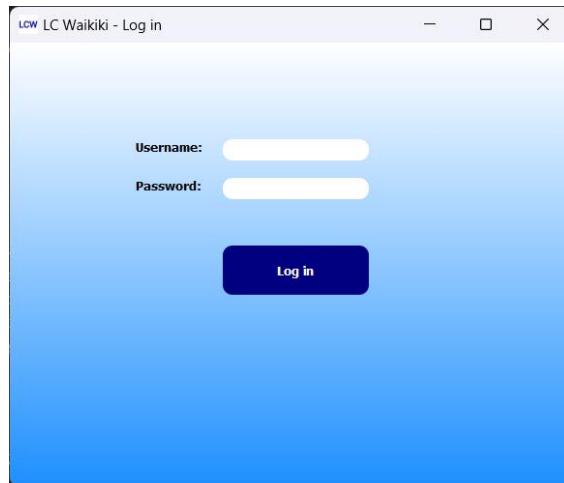


Figure 8.1 Login Page

On the start screen of our interface, there is a *Login Page* as shown in Figure 8.1. This page is of great importance for system security. It includes 2 labels, 2 text fields, and a combo box (button), designed to provide a secure entry point for users to access data safely. Once users pass through this secure entry point, they can easily navigate to the *Main Page*.

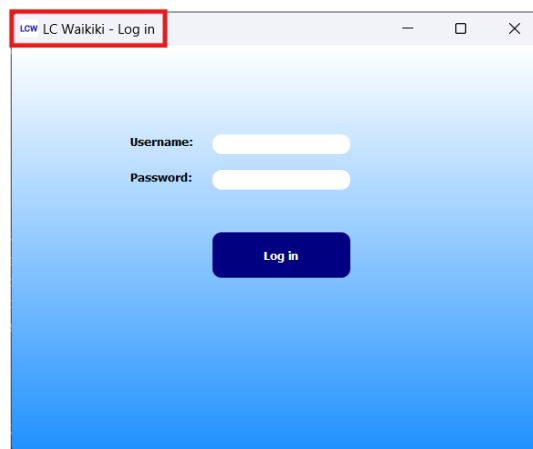


Figure 8.2 Logo

Additionally, as shown in Figure 8.2, the *Log-in Screen* features the company logo in the upper left corner and the phrase "Log-in" to indicate the current page.



Figure 8.3 Username and Password

As shown in Figure 8.3, there are *Username* and *Password* boxes required for the user to access the data. We also chose a structure with curved corners to give our interface a more professional appearance.

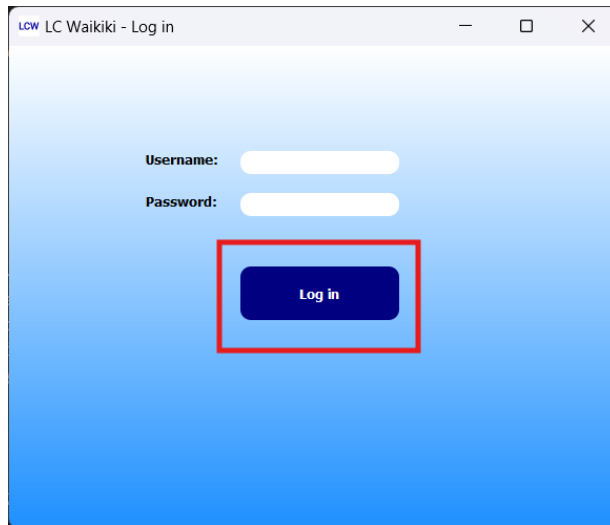


Figure 8.4 Log in Button

Figure 8.4 shows the *Log-in* button on the login screen of the LC Waikiki application. To access the system, users need to enter their username and password and then click on this *Log-in* button. This step is essential for authentication and ensuring that only authorized personnel can access the application's functionalities.

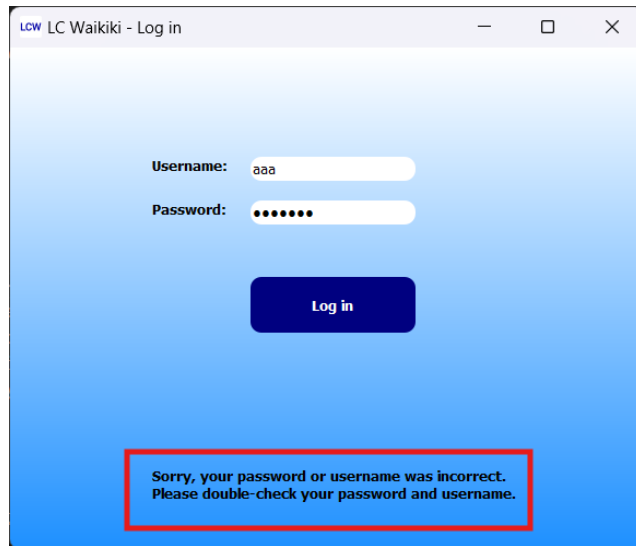


Figure 8.5 Error Message

As seen in Figure 8.5, if the username or password is entered incorrectly, an error message appears stating: "Sorry, your password or username was incorrect. Please double-check your password and username." At this stage, if we do not enter the username and password correctly, we cannot log in to the *Main Page* screen.

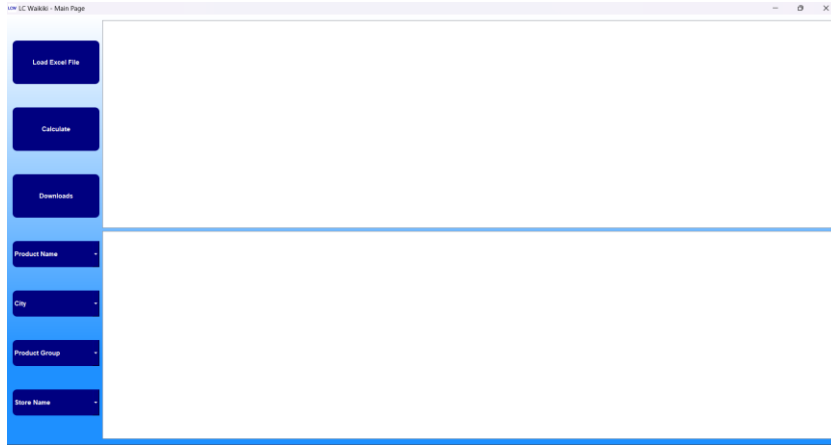


Figure 8.6 Main Page

After successfully passing the login screen, the Main Page appears. There are 3 buttons, 4 filters and 2 table widgets on the Main Page screen as can be seen in Figure 8.6. Button colors, logo and background color are consistent with those on the *Log-in* screen, as shown in Figure 8.4.

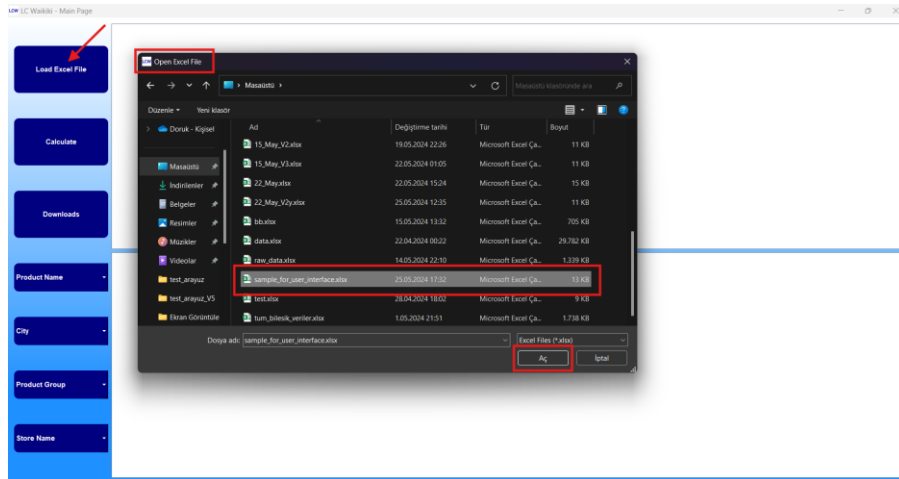
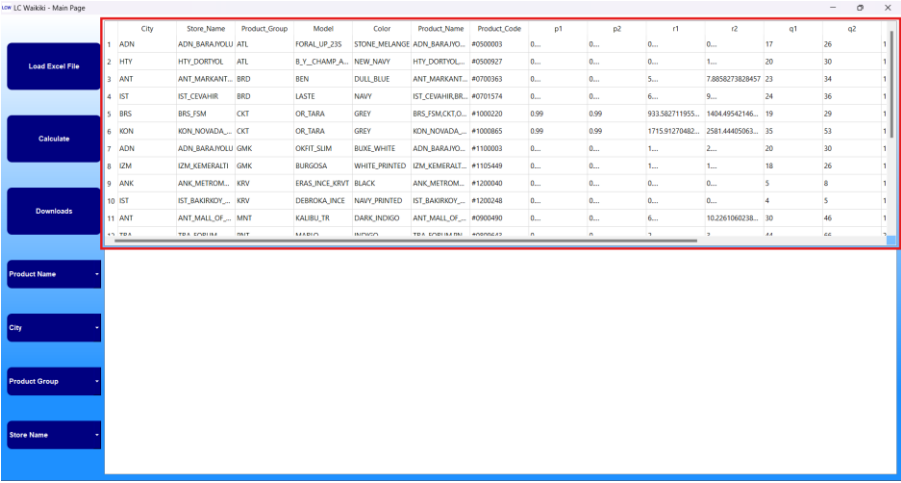


Figure 8.7 Main Page-Loading Excel File

In Figure 8.7, the process of extracting data is shown. First, to select the Excel file that feeds our main code and to display it in the interface, press the *Load Excel File* button. This opens the *Open Excel File* screen. After selecting the desired Excel file, click the *Open* button.



The screenshot shows a web application interface titled 'LC Waliki - Main Page'. On the left is a sidebar with three buttons: 'Load Excel File', 'Calculate', and 'Downloads'. Below these are four dropdown menus labeled 'Product Name', 'City', 'Product Group', and 'Store Name'. The main area displays a table with 15 columns: City, Store Name, Product Group, Model, Color, Product Name, Product Code, p1, p2, r1, r2, q1, and q2. The table contains 11 rows of data, with the first row highlighted in red. The data represents various products from different stores, including items like 'FORAL_UP_235', 'NEW_NAVY', 'DULL_BLUE', 'NAVY', 'BRS_FSM', 'KON_NOVADA', 'OKIT_SLM', 'BURGOSA', 'ERAS_INCE_KRYT', 'DEBROKA_INCE', and 'KALIBU_TR'.

	City	Store Name	Product Group	Model	Color	Product Name	Product Code	p1	p2	r1	r2	q1	q2
1	ADN	ADN_BARAPOU	ATL	FORAL_UP_235	STONE_MELANGE	ADN_BARAPOU...	A0500003	0...	0...	0...	0...	17	26
2	HTY	HTY_DORTYOL	ATL	B_Y_CHAMP_A...	NEW_NAVY	HTY_DORTYOL...	A0500927	0...	0...	0...	1...	20	30
3	ANT	ANT_MARKANT...	BRD	BEN	DULL_BLUE	ANT_MARKANT...	A0700363	0...	0...	5...	7.8658273828457	23	34
4	IST	IST_CEVAKIR	BRD	LASTE	NAVY	IST_CEVAKIR...	A0701574	0...	0...	6...	9...	24	36
5	BRS	BRS_FSM	CKT	OR_TAMA	GREY	BRS_FSM.CKT.O...	A1000220	0.99	0.99	933.582711955...	1404.49542146...	19	29
6	KON	KON_NOVADA...	CKT	OR_TAMA	GREY	KON_NOVADA...	A1000865	0.99	0.99	1715.91270482...	2581.4445063...	35	53
7	ADN	ADN_BARAPOU	GMK	OKIT_SLM	BLUE_WHITE	ADN_BARAPOU...	A1100003	0...	0...	1...	2...	20	30
8	IZM	IZM_KEMERALT...	GMK	BURGOSA	WHITE_PRINTED	IZM_KEMERALT...	A1105449	0...	0...	1...	1...	18	26
9	ANK	ANK_METROM...	KRV	ERAS_INCE_KRYT	BLACK	ANK_METROM...	A1200040	0...	0...	0...	0...	5	8
10	IST	IST_BAKIRKOV...	KRV	DEBROKA_INCE	NAVY_PRINTED	IST_BAKIRKOV...	A1200248	0...	0...	0...	0...	4	5
11	ANT	ANT_MALL_OF...	MNT	KALIBU_TR	DARK_INDIGO	ANT_MALL_OF...	A0900480	0...	0...	6...	10.2261080238...	30	46

Figure 8.8 Main Page with Excel File

For users to use our interface easily and to ensure the reliability of the DATA they upload, the Excel file you first uploaded is displayed in the upper window of our interface, as shown in Figure 8.8. This allows employees to review and verify the parameters used in the optimization calculation. Detailed explanations about the parameters are included in the relevant headings above. Understanding the parameters helps users perform necessary checks and ensure their accuracy, leading to reliable calculations.

	City	Store Name	Product Group	Model	Color	Product Name	Product Code	p1	p2	r1	r2	q1	q2
1	ADN	ADN_BARAHOLO	ATL	FORAL_UP_235	STONE_MELANGE	ADN_BARAHOLO...	#000000	0...	0...	0...	0...	17	26
2	HTY	HTY_DORTYOL	ATL	B_Y_CHAMP_A...	NEW_NAVY	HTY_DORTYOL...	#000027	0...	0...	0...	1...	20	30
3	ANT	ANT_MARKANT...	BRO	BEN	DULL_BLUE	ANT_MARKANT...	#000063	0...	0...	5...	7.8658273828457	23	34
4	IST	IST_CEVAKIR	BRO	LASTE	NAVY	IST_CEVAKIR...	#0001574	0...	0...	6...	9...	24	36
5	BRS	BRS_FSM	CKT	OR_TARA	GREY	BRS_FSM...	#1000220	0.99	0.99	933.582711955...	1404.49542146...	19	29
6	KDN	KDN_NOVADA...	CKT	OR_TARA	GREY	KDN_NOVADA...	#1000865	0.99	0.99	1715.91270462...	2581.44405963...	35	53
7	ADN	ADN_BARAHOLO	GMK	OKIT_SLM	BLUE_WHITE	ADN_BARAHOLO...	#1100003	0...	0...	1...	2...	20	30
8	IZM	IZM_KEMERALT...	GMK	BURGOSA	WHITE_PRINTED	IZM_KEMERALT...	#1100449	0...	0...	1...	1...	18	26
9	ANK	ANK_METROM...	KRV	ERAS_UNICE_KRVT	BLACK	ANK_METROM...	#1200040	0...	0...	0...	0...	5	8
10	IST	IST_BAKIRKOV...	KRV	DEBROKA_UNICE	NAVY_PRINTED	IST_BAKIRKOV...	#1200248	0...	0...	0...	0...	4	5
11	ANT	ANT_MALL_OF...	MNT	KAUBILTR	DARK_INDIGO	ANT_MALL_OF...	#0000490	0...	0...	6...	10.2261060238...	30	46
12	TEA	TEA_5700184	INT	MARSH...	BURGOSA	TEA_5700184...	#0000431	0...	0...	1...	7...	44	66

	Product Name	Product Code	City	Product Group	Store Name	first_period_beginning_amount	first_period_shipment	Target_First_Period	second_period_beginning_amount	second_period_shipment	Target_Second_Period
1	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	18	13	31	
2	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	17	14	31	
3	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	16	15	31	
4	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	15	16	31	
5	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	14	17	31	
6	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	13	18	31	
7	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	12	19	31	
8	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	11	20	31	
9	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	10	21	31	
10	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	9	22	31	
11	AFY_PARK_AUM...	#0100489	AFY	TSH	AFY_PARK_AUM_0	18	18	8	23	31	
12	KCV_PARK_AUM...	#0000000	KCV	TSH	KCV_PARK_AUM_0	18	18	7	24	31	

Figure 8.9 Main Page-Calculate Button

In Figure 8.9, the process of performing calculations with the selected parameters is illustrated. After loading the Excel file as previously described, the *Calculate* button is used to perform calculations using these parameters. The necessary code for the calculations is integrated behind this button. When clicked, the system automatically carries out the necessary computations according to the preset parameters. The results include the initial number of products in the first period, the amount sent in the first period, the targeted product amount in the first period, the initial number of products in the second period, the amount sent in the second period, and the targeted product amount in the second period. This leads to the achievement of the primary objective of our project, which is determining optimal values for inventory management. These calculations are crucial for making correct and efficient decisions in inventory management processes. Thus, it is ensured that stock levels are managed correctly, product supply processes are optimized, and operating costs are reduced.

The screenshot shows the 'LC Walki - Main Page' interface. On the left, there are four buttons: 'Load Excel File', 'Calculate', 'Downloads', and a red box around the filter options. The filter options are: 'Product Name', 'City', 'Product Group', and 'Store Name'. The main table displays data with columns: City, Store_Name, Product_Group, Model, Color, Product_Name, Product_Code, p1, p2, r1, r2, q1, q2. Below the table, there are filter dropdowns for Product Name, City, Product Group, and Store Name, each with a red arrow pointing to it. The table data includes rows for various products like ADN, HTY, ANT, IST, BRS, KON, ADM, IZM, ANK, IST, ANT, and TBA.

Figure 8.10 Main Page-Filter Options

In Figure 8.10, the filter options on the main page are illustrated. We added 4 buttons to interpret better the output we obtained. This way, a filter system is created based on the Product Name, City, Product Group, and Store Name columns of the desired product, making it easier for employees to reach the desired results.

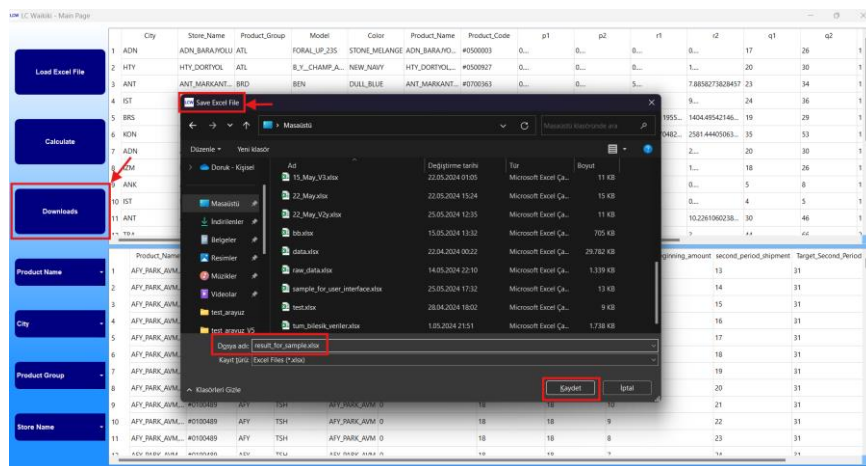


Figure 8.11 Main Page-Download Button

After pressing the *Downloads* button, the *Save Excel File* screen opens as can be seen in Figure 8.11, and the resulting output is downloaded to the desired area on the computer.

9 CONCLUSION

This thesis aims to show how important inventory allocation system is for apparel retailers, in line with this importance, to minimize the costs incurred due to inventory allocation decisions. For this purpose, we determined that the demand for the products in 2 sales periods was negative binomial distribution. Therefore, we found negative binomial parameters for each product for two different sales periods. Thanks to the mathematical model we established, we calculated costs for two different sales seasons using these parameters. As a result of cost calculations, the optimum product quantities to be allocated from the inventory were found, depending on the number of products at the beginning of the 1st and 2nd Periods. In addition to these results, thanks to the optimum shipment sizes determined depending on the number of products at the beginning of the sales periods, the quantities of Target products, which should be in stores in the 1st and 2nd periods for all product groups, were determined.

APPENDIX A: DATA PROCESSING

A.1 Separation of Product Groups by Using Raw Data

```
library(readxl)
library(openxlsx)
setwd("/Users/Salih/Desktop")
veri <- read_excel("Raw_Data.xlsx")
# Check sales values
veri$sales <- ifelse(veri$sales <= 0 | is.na(veri$sales) | veri$sales == "", 1, veri$sales)
# Split MODEL AD
ayrilmis <- strsplit(as.character(veri$Model_Ad), ",")
# Create new data
yeni_veri <- data.frame(
  Depo_Ad = veri$Depo_Ad,
  urun_grp = sapply(ayrilmis, function(x) x[1]),
  mdl_ad = sapply(ayrilmis, function(x) x[2]),
  Renk_Tanim = veri$Renk_Tanim,
  sales = veri$sales
)
# Separate each product group
unique_urun_gruplari <- unique(yeni_veri$urun_grp)
# Create a list for all groups
butun_gruplar <- list()

# Separate each product group and add to all_groups list
for (urun_grp in unique_urun_gruplari) {
  butun_gruplar[[urun_grp]] <- yeni_veri[yeni_veri$urun_grp == urun_grp, ]
}
# Select non-empty groups
```

```

butun_gruplar <- butun_gruplar[sapply(butun_gruplar, nrow) > 0]
# Create an empty composite data list
bilesik_veri_listesi <- list()
# Create composite data for each group
for (i in seq_along(butun_gruplar)) {
  bilesik_veri <- data.frame(
    depo_ad_urun_grup = paste(butun_gruplar[[i]]$Depo_Ad, butun_gruplar[[i]]$urun_grp, sep =
"_"),
    depo_ad_model_grup = paste(butun_gruplar[[i]]$Depo_Ad, butun_gruplar[[i]]$mdl_ad, sep =
"_"),
    depo_ad_renk = paste(butun_gruplar[[i]]$Depo_Ad, butun_gruplar[[i]]$Renk_Tanim, sep =
"_"),
    sales = butun_gruplar[[i]]$sales
  )
  bilesik_veri_listesi[[i]] <- bilesik_veri
}
# Specify the folder path
klasor_yolu <- file.path("/Users/Salih/Desktop")
# Create the folder if it doesn't exist
if (!dir.exists(klasor_yolu)) {
  dir.create(klasor_yolu)
}
# Create a single Excel file
wb <- createWorkbook()
# Write each composite data to the Excel file
for (i in seq_along(bilesik_veri_listesi)) {
  # Create a new sheet
  sheet_name <- paste("bilesik_veri_", i, sep = "_")
  addWorksheet(wb, sheet_name)
  # Add composite data to the Excel file
  writeData(wb, sheet_name, bilesik_veri_listesi[[i]])
}

```



```

}
# Save the Excel file
excel_yolu <- file.path(klasor_yolu, "tum_bilesik_veriler.xlsx")
saveWorkbook(wb, excel_yolu)

```

A.2 Regression Code

```

library(readxl)
library(caret)
# Read the Excel file
file_path <- "/Users/Salih/Desktop/ tum_bilesik_veriler.xlsx"
data <- read_excel(file_path)
# Creating dummy variables
a <- dummyVars(~ depo_ad_urun_grup + depo_ad_model_grup + depo_ad_renk, data = data)
dummy_data <- predict(a, newdata = data)
# Creating combined data frame with the generated dummy variables
combined_dummy <- cbind(data, dummy_data)
# Creating new data by deleting the first 3 columns
new_dummy <- combined_dummy[, -c(1:3)]
new_dummy$sales <- as.numeric(new_dummy$sales)
# Checking for NA values
sum(is.na(new_dummy$sales))
# Log transformation
y <- log(new_dummy$sales)
# Creating linear regression model
x <- lm(y ~ ., data = new_dummy)
summary(x)

```

A.3 Dummy Variable Code

```

library(openxlsx)
library(readxl)
library(caret)
# Specifying the excel file and path
file_path <- "C:/Users/Salih/Desktop/ tum_bilesik_veriler.xlsx "
# Creating excel file
output_file <- "C:/Users/Salih/Desktop/Dummy.xlsx"
wb <- createWorkbook()
# Loop for each sheet
for (i in 1:12) {
  # Reading data from the excel file
  sheet_name <- paste("bilesik_veri_", i, sep = "_")
  data <- read_excel(file_path, sheet = sheet_name)
  # Creating dummy variables
  a <- dummyVars(~ depo_ad_urun_grup + depo_ad_model_grup + depo_ad_renk, data = data)
  dummy_data <- predict(a, newdata = data)
  # Creating combined data frame
  combined_dummy <- cbind(data, dummy_data)
  # Converting the sales column to numeric
  combined_dummy$sales <- as.numeric(combined_dummy$sales)
  # Add the data to the excel file
  addWorksheet(wb, sheet_name)
  writeData(wb, sheet = sheet_name, x = combined_dummy)
}
# Save the excel file
tryCatch(saveWorkbook(wb, output_file),
  error = function(e) print(paste("Error:", e)))

```

A.4 Cross Validation Code

```
# Setting working directory
setwd("/Users/Salih/Desktop")

# Loading libraries
library(readxl)
library(writexl)
library(dplyr)

# Sheet names defining
my_sheet_names <- paste0("bilesik_veri__", 1:12)

# Creating an empty list to store all predictions and results
all_results <- list()

# Loop over each sheet
for (sheet_name in my_sheet_names) {
  # Read data from each sheet
  data <- read_excel("Dummy.xlsx", sheet = sheet_name)

  # Removing first three columns
  data <- data[, -c(1:3)]

  # Check if the data frame is empty
  if (nrow(data) == 0) {
    cat("No data found for Sheet", sheet_name, ".\n\n")
    next
  }

  # Logarithm of the 'sales' variable
  data$log_sales <- log(data$sales)

  # Determining the size of the dataset and the size of the slices
  num_rows <- nrow(data)
  num_slices <- 5

  # Calculate the size of each slice
  slice_sizes <- rep(floor(num_rows / num_slices), num_slices)
```

```

remainder <- num_rows %% num_slices
# Add the remaining data to the last slice
if (remainder > 0) {
  slice_sizes[num_slices] <- slice_sizes[num_slices] + remainder
}
index_starts <- c(1, cumsum(slice_sizes)[-num_slices])
# Create an empty data frame to store results for this sheet
sheet_results <- data.frame()
# Iterate over each slice
for (i in 1:num_slices) {
  start_index <- index_starts[i]
  end_index <- start_index + slice_sizes[i] - 1
  index.test <- start_index:end_index
  # Dividing data to train and test sets
  data.train <- data[-index.test, ]
  data.test <- data[index.test, ]
  # If there is not enough data, go to the next slice
  if (nrow(data.train) == 0 || nrow(data.test) == 0) {
    cat("Not enough data for Sheet", sheet_name, ".\n\n")
    next
  }
  # Fit a linear regression model with log(sales) as the response variable
  lm.fit <- lm(log_sales ~ ., data = data.train)
  # Predict log(sales)
  log_sales.predict <- predict(lm.fit, newdata = data.test)
  # Calculating predicted sales by taking exponential of predicted log(sales)
  sales.predict <- exp(log_sales.predict)
  # Creating data frame for the results of this slice
  slice_results <- data.frame(
    Actual_Sales = data.test$sales,

```

```
    Predicted = sales.predict
  )
  # Bind the results for this slice to the sheet_results data frame
  sheet_results <- bind_rows(sheet_results, slice_results)
}
# Store the results for this sheet in the all_results list
all_results[[sheet_name]] <- sheet_results
}
# Combine all results into one data frame
combined_results <- bind_rows(all_results, .id = "Sheet_Name")
# Write all predictions and results to an Excel file
write_xlsx(combined_results, "cross_validation_results.xlsx")
```

APPENDIX B: FIRST AND SECOND PERIOD COST CALCULATION

```

import pandas as pd
import numpy as np
from scipy.stats import nbinom
import openpyxl
import math

# used for each product group
file_path = 'Product_Data.xlsx'
df = pd.read_excel(file_path)

#second period cost calculation

data = {'Product_Name': [], 'Product_Code': [], 'City': [], 'Product_Group': [], 'Store_Name': [],
        'v2': [], 'q2': [], 'Toplam Maliyet': []}

index = 0
while index < len(df):
    row = df.iloc[index]
    product_name = row['Product_Name']
    product_code = row['Product_Code']
    city = row['City']
    product_group = row['Product_Group']
    store_name = row['Store_Name']
    p1, p2, r1, r2 = row[['p1', 'p2', 'r1', 'r2']]
    demand1, demand2, q1_values, q2_values = row[['D1', 'D2', 'q1', 'q2']]
    v2_values = row['v2']
    v2 = 0
    while v2 <= v2_values:
        q2 = 0
        while q2 <= q2_values:
            cost_for_sec_per = 0
            d2 = 0

```

```

while d2 <= demand2:
    D2 = nbinom.pmf(d2, r2, p2)
    cost_for_sec_per = (max((d2 - v2 - q2), 0) * D2) * 1000 + (max((v2 + q2 - d2), 0) *
D2) * 1 + cost_for_sec_per
    d2 += 1
    cost_for_sec_per += q2*10
    data['Product_Name'].append(product_name)
    data['Product_Code'].append(product_code)
    data['City'].append(city)
    data['Product_Group'].append(product_group)
    data['Store_Name'].append(store_name)
    data['v2'].append(v2)
    data['q2'].append(q2)
    data['Toplam Maliyet'].append(cost_for_sec_per)
    q2 += 1
    v2 += 1
    index += 1
result_rp1_output = pd.DataFrame(data)
# calculating the minimum cost in the second period
def cal_min_cost_for_sec_per(df):
    min_costs_sec_2 = df.loc[df.groupby('v2')['Toplam Maliyet'].idxmin(), ['v2', 'q2', 'Toplam
Maliyet', 'Product_Name', 'City', 'Product_Group', 'Store_Name' ]]
    min_costs_sec_2['Product_Code'] = df['Product_Code'].iloc[0]
    return min_costs_sec_2.reset_index(drop=True)
product_column = 'Product_Code'
unique_products = result_rp1_output[product_column].unique()
result_list = []
for product in unique_products:
    product_df = result_rp1_output[result_rp1_output[product_column] == product].copy()
    result = cal_min_cost_for_sec_per(product_df)

```

```

    result_list.append(result)
final_result = pd.concat(result_list)
final_result.reset_index(drop=True, inplace=True)
#first period cost calculation
data_2 = {
    'Product_Name': [],
    'Product_Code': [],
    'City': [],
    'Product_Group': [],
    'Store_Name': [],
    'v1': [],
    'q1': [],
    'Product Cost': []
}
for index, row in df.iterrows():
    city = row['City']
    product_group = row['Product_Group']
    store_name = row['Store_Name']
    product_name = row['Product_Name']
    product_code = row['Product_Code']
    p1, p2, r1, r2 = row[['p1', 'p2', 'r1', 'r2']]
    demand1, q1_values, v1_values = row[['D1', 'q1', 'v1']]
    v1_values = 0 if pd.isna(v1_values) else int(v1_values)
    q1_values = 0 if pd.isna(q1_values) else int(q1_values)
    demand1 = 0 if pd.isna(demand1) else int(demand1)
    for v1 in range(v1_values + 1):
        for q1 in range(q1_values + 1):
            costdp = 0
            for d1 in range(demand1 + 1):
                D1 = nbinom.pmf(d1, r1, p1)

```



```

v2 = max(v1 + q1 - d1, 0)
matching_rows = final_result.loc[(final_result['Product_Code'] == product_code) &
(final_result['v2'] == v2), 'Toplam Maliyet']
if not matching_rows.empty:
    secondcost = matching_rows.values[0]
else:
    secondcost = 0
costdp += (5 * max(v1 + q1 - d1, 0) + 100 * max(d1 - q1 - v1, 0) + secondcost) * D1
costdp += q1 * 10
data_2['Product_Name'].append(product_name)
data_2['Product_Code'].append(product_code)
data_2['City'].append(city)
data_2['Product_Group'].append(product_group)
data_2['Store_Name'].append(store_name)
data_2['v1'].append(v1)
data_2['q1'].append(q1)
data_2['Product Cost'].append(costdp)
output_df = pd.DataFrame(data_2)
# calculating the minimum cost in the first period
result = output_df.groupby(['Product_Code', 'v1']).apply(lambda x: x.loc[x['Product
Cost'].idxmin()])[['Product_Code', 'Product_Name', 'City', 'Product_Group', 'Store_Name', 'v1', 'q1',
'Product Cost']]
result = result.reset_index(drop=True)
result['d1_range'] = result['v1'] + result['q1']
# calculation of target values
expanded_rows = []
for index, row in result.iterrows():
    for d1 in range(row['d1_range'] + 1):
        new_row = row.copy()
        new_row['d1'] = d1
        new_row['v2'] = row['v1'] + row['q1'] - d1

```

```

new_row["Target_First_Period"] = row['v1'] + row['q1']
v2 = new_row['v2']
product_code = new_row['Product_Code']
matching_rows = final_result[(final_result['v2'] == v2) & (final_result['Product_Code'] ==
product_code)]
if not matching_rows.empty:
    q2 = matching_rows.iloc[0]['q2']
    new_row['q2'] = q2
    expanded_rows.append(new_row)
expanded_df = pd.DataFrame(expanded_rows)
expanded_df["Target_Second_Period"] = expanded_df['v2'] + expanded_df['q2']
expanded_df = expanded_df[['Product_Name', 'Product_Code', 'City',
'Product_Group', 'Store_Name', 'v1', 'q1', 'Target_First_Period', 'v2', 'q2', 'Target_Second_Period']]
expanded_df = expanded_df.copy()
expanded_df.rename(columns={
    'v1': 'first_period_beginning_amount',
    'q1': 'first_period_shipment',
    'v2': 'second_period_beginning_amount',
    'q2': 'second_period_shipment',
}, inplace=True)
expanded_df
expanded_df.to_excel('expanded_df_output_yelek.xlsx', index=False)
filtered_df = expanded_df[['Product_Code', 'Product_Name', 'City',
'Product_Group', 'Store_Name', 'Target_First_Period', 'Target_Second_Period']]
grouped_df = filtered_df.groupby('Product_Code').first().reset_index()
grouped_df
grouped_df.to_excel('grouped_df_output_yelek.xlsx', index=False)

```

APPENDIX C: IMPLEMENTATION

C.1 Login Page

```

from PyQt5 import QtCore, QtGui, QtWidgets
from main_page import ExcelProcessor
class Ui_Form(object):
    def setupUi(self, Form):
        self.Form = Form
        Form.setObjectName("Form")
        Form.resize(581, 459)
        gradient = QtGui.QLinearGradient(0, 0, 0, Form.height())
        gradient.setColorAt(0, QtGui.QColor(255, 255, 255))
        gradient.setColorAt(1, QtGui.QColor(30, 144, 255))
        palette = QtGui.QPalette()
        palette.setBrush(QtGui.QPalette.Window, QtGui.QBrush(gradient))
        Form.setPalette(palette)
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(130, 100, 71, 16))
        self.label.setObjectName("label")
        self.label.setStyleSheet("font-weight: bold; border-radius: 10px;")
        self.label_2 = QtWidgets.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(130, 140, 71, 16))
        self.label_2.setObjectName("label_2")
        self.label_2.setStyleSheet("font-weight: bold; border-radius: 10px;")
        self.lneUsername = QtWidgets.QLineEdit(Form)
        self.lneUsername.setGeometry(QtCore.QRect(220, 100, 151, 22))
        self.lneUsername.setObjectName("lneUsername")
        self.lneUsername.setStyleSheet("border-radius: 10px;")

```

```

self.lnePassword = QtWidgets.QLineEdit(Form)
self.lnePassword.setGeometry(QtCore.QRect(220, 140, 151, 22))
self.lnePassword.setEchoMode(QtWidgets.QLineEdit.Password)
self.lnePassword.setObjectName("lnePassword")
self.lnePassword.setStyleSheet("border-radius: 10px;")
self.btnLogin = QtWidgets.QPushButton(Form)
self.btnLogin.setGeometry(QtCore.QRect(220, 210, 151, 51))
self.btnLogin.setObjectName("btnLogin")
self.btnLogin.setStyleSheet("background-color: navy; color: white; font-weight: bold;
border-radius: 10px;")
self.btnLogin.clicked.connect(self.loginClicked)
self.lblError = QtWidgets.QLabel(Form)
self.lblError.setGeometry(QtCore.QRect(130, 370, 461, 60))
self.lblError.setObjectName("lblError")
self.lblError.setStyleSheet("color: black; font-weight: bold;")
self.lblError.setVisible(False)
icon_path = "/Users/DORUK/Desktop/test_arayuz_V5/baglanti4.png"
icon = QtGui.QIcon(icon_path)
Form.setWindowIcon(icon)
self.retranslateUi(Form)
QtCore.QMetaObject.connectSlotsByName(Form)
def retranslateUi(self, Form):
    _translate = QtCore.QCoreApplication.translate
    Form.setWindowTitle(_translate("Form", "LC Waikiki - Log in"))
    self.label.setText(_translate("Form", "Username:"))
    self.label_2.setText(_translate("Form", "Password:"))
    self.btnLogin.setText(_translate("Form", "Log in"))
    self.lblError.setText(_translate("Form", "Sorry, your password or username was incorrect.
\nPlease double-check your password and username.))
def loginClicked(self):

```

```

username = self.lneUsername.text()
password = self.lnePassword.text()
if username == "Group2.LCW" and password == "12345":
    self.window = QtWidgets.QWidget()
    self.ui = ExcelProcessor()
    self.ui.show()
    self.Form.close()
else:
    self.lblError.setVisible(True)
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())

```

C.2 Main Page

```

import sys
import pandas as pd
from PyQt5.QtWidgets import QApplication, QVBoxLayout, QHBoxLayout, QWidget,
QPushButton, QLabel, QFileDialog, QTableWidgetItem, QComboBox
from PyQt5.QtGui import QFont, QLinearGradient, QColor, QBrush, QIcon
from PyQt5 import QtCore
import math
from scipy.stats import nbinom
class ExcelProcessor(QWidget):
    def __init__(self):
        super().__init__()

```

```

self.initUI()
def initUI(self):
    main_layout = QHBoxLayout()
    left_layout = QVBoxLayout()
    self.uploadButton = QPushButton('Load Excel File')
    self.uploadButton.setStyleSheet("background-color: navy; color: white; font-weight: bold;
border-radius: 10px;")
    self.uploadButton.setFixedSize(200, 100)
    self.uploadButton.setFont(QFont("Arial", 9))
    left_layout.addWidget(self.uploadButton)
    self.calculateButton = QPushButton('Calculate')
    self.calculateButton.setStyleSheet("background-color: navy; color: white; font-weight: bold;
border-radius: 10px;")
    self.calculateButton.setFixedSize(200, 100)
    self.calculateButton.setFont(QFont("Arial", 9))
    left_layout.addWidget(self.calculateButton)
    self.downloadButton = QPushButton('Downloads')
    self.downloadButton.setStyleSheet("background-color: navy; color: white; font-weight:
bold; border-radius: 10px;")
    self.downloadButton.setFixedSize(200, 100)
    self.downloadButton.setFont(QFont("Arial", 9))
    left_layout.addWidget(self.downloadButton)
    self.filterComboBox1 = QComboBox()
    self.filterComboBox1.setStyleSheet("background-color: navy; color: white; font-weight:
bold; border-radius: 10px;")
    self.filterComboBox1.addItem("Product Name")
    self.filterComboBox1.setFixedSize(200, 60)
    self.filterComboBox1.setFont(QFont("Arial", 9))
    left_layout.addWidget(self.filterComboBox1)
    self.filterComboBox2 = QComboBox()
    self.filterComboBox2.setStyleSheet("background-color: navy; color: white; font-weight:
bold; border-radius: 10px;")

```

```

self.filterComboBox2.addItem("City")
self.filterComboBox2.setFixedSize(200, 60)
self.filterComboBox2.setFont(QFont("Arial", 9))
left_layout.addWidget(self.filterComboBox2)
self.filterComboBox3 = QComboBox()
self.filterComboBox3.setStyleSheet("background-color: navy; color: white; font-weight:
bold; border-radius: 10px;")
self.filterComboBox3.addItem("Product Group")
self.filterComboBox3.setFixedSize(200, 60)
self.filterComboBox3.setFont(QFont("Arial", 9))
left_layout.addWidget(self.filterComboBox3)
self.filterComboBox4 = QComboBox()
self.filterComboBox4.setStyleSheet("background-color: navy; color: white; font-weight:
bold; border-radius: 10px;")
self.filterComboBox4.addItem("Store Name")
self.filterComboBox4.setFixedSize(200, 60)
self.filterComboBox4.setFont(QFont("Arial", 9))
left_layout.addWidget(self.filterComboBox4)
main_layout.addLayout(left_layout)
right_layout = QVBoxLayout()
self.excelTable = QTableWidget()
right_layout.addWidget(self.excelTable)
self.resultTable = QTableWidget()
right_layout.addWidget(self.resultTable)
main_layout.addLayout(right_layout)
self.setLayout(main_layout)
self.setWindowTitle('LC Waikiki - Main Page')
self.setWindowIcon(QIcon('/Users/DORUK/Desktop/test_arayuz_V5/baglanti4.png'))
self.uploadButton.clicked.connect(self.loadFile)
self.calculateButton.clicked.connect(self.calculate)

```

```

self.downloadButton.clicked.connect(self.downloadExcel)
self.filterComboBox1.currentIndexChanged.connect(self.filterResults)
self.filterComboBox1.currentIndexChanged.connect(self.filterResults)
self.filterComboBox2.currentIndexChanged.connect(self.filterResults)
self.filterComboBox3.currentIndexChanged.connect(self.filterResults)
self.filterComboBox4.currentIndexChanged.connect(self.filterResults)
self.setGeometry(200, 150, 1500, 800)
gradient = QLinearGradient(0, 0, 0, self.height())
gradient.setColorAt(0, QColor(255, 255, 255))
gradient.setColorAt(1, QColor(30, 144, 255))
palette = self.palette()
palette.setBrush(self.backgroundRole(), QBrush(gradient))
self.setPalette(palette)
def loadFile(self):
    options = QFileDialog.Options()
    fileName, _ = QFileDialog.getOpenFileName(self, "Open Excel File", "", "Excel Files
(*.xlsx);;All Files (*)", options=options)
    if fileName:
        self.df = pd.read_excel(fileName)
        self.displayExcelData(self.df)
def displayExcelData(self, df):
    df.columns = df.columns.astype(str)
    self.excelTable.setRowCount(df.shape[0])
    self.excelTable.setColumnCount(df.shape[1])
    self.excelTable.setHorizontalHeaderLabels(df.columns)
    for i in range(df.shape[0]):
        for j in range(df.shape[1]):
            self.excelTable.setItem(i, j, QTableWidgetItem(str(df.iat[i, j])))
def calculate(self):

```



```

data = {'Product_Name': [], 'Product_Code': [], 'City': [], 'Product_Group': [], 'Store_Name':
[], 'v2': [], 'q2': [], 'Toplam Maliyet': []}

for index, row in self.df.iterrows():
    product_name = row['Product_Name']
    product_code = row['Product_Code']
    city = row['City']
    product_group = row['Product_Group']
    store_name = row['Store_Name']
    p1, p2, r1, r2 = row[['p1', 'p2', 'r1', 'r2']]
    demand1, demand2, q1_values, q2_values = row[['D1', 'D2', 'q1', 'q2']]
    v2_values = row['v2']
    v2_values = 0 if pd.isna(v2_values) else int(v2_values)
    for v2 in range(v2_values + 1):
        q2_values = 0 if pd.isna(q2_values) else int(q2_values)
        for q2 in range(q2_values + 1):
            cost_for_sec_per = 0
            demand2 = 0 if pd.isna(demand2) else int(demand2)
            for d2 in range(demand2 + 1):
                D2 = nbinom.pmf(d2, r2, p2)
                cost_for_sec_per += (max((d2 - v2 - q2), 0) * D2) * 1000 + (max((v2 + q2 - d2), 0)
*D2) * 1
            cost_for_sec_per += q2 * 10
            data['Product_Name'].append(product_name)
            data['Product_Code'].append(product_code)
            data['City'].append(city)
            data['Product_Group'].append(product_group)
            data['Store_Name'].append(store_name)
            data['v2'].append(v2)
            data['q2'].append(q2)
            data['Toplam Maliyet'].append(cost_for_sec_per)

```

```

result_rp1_output = pd.DataFrame(data)
def cal_min_cost_for_sec_per(df):
    if not df.empty:
        min_costs_sec_2 = df.loc[df.groupby('v2')['Toplam Maliyet'].idxmin(), ['v2', 'q2',
'Toplam Maliyet', 'Product_Name', 'City', 'Product_Group', 'Store_Name' ]]
        min_costs_sec_2['Product_Code'] = df['Product_Code'].iloc[0]
        return min_costs_sec_2.reset_index(drop=True)
    else:
        return pd.DataFrame()
product_column = 'Product_Code'
unique_products = result_rp1_output[product_column].unique()
result_list = []
for product in unique_products:
    product_df = result_rp1_output[result_rp1_output[product_column] == product].copy()
    result = cal_min_cost_for_sec_per(product_df)
    result_list.append(result)
final_result = pd.concat(result_list)
final_result.reset_index(drop=True, inplace=True)
data_2 = {
    'Product_Name': [],
    'Product_Code': [],
    'City': [],
    'Product_Group': [],
    'Store_Name': [],
    'v1': [],
    'q1': [],
    'Product Cost': []
}
for index, row in self.df.iterrows():
    product_name = row['Product_Name']

```

```

product_code = row['Product_Code']
city = row['City']
product_group = row['Product_Group']
store_name = row['Store_Name']
p1, p2, r1, r2 = row[['p1', 'p2', 'r1', 'r2']]
demand1, q1_values, v1_values = row[['D1', 'q1', 'v1']]
v1_values = 0 if pd.isna(v1_values) else int(v1_values)
q1_values = 0 if pd.isna(q1_values) else int(q1_values)
demand1 = 0 if pd.isna(demand1) else int(demand1)
for v1 in range(v1_values + 1):
    for q1 in range(q1_values + 1):
        costdp = 0
        for d1 in range(demand1 + 1):
            D1 = nbinom.pmf(d1, r1, p1)
            v2 = max(v1 + q1 - d1, 0)
            matching_rows = final_result.loc[(final_result['Product_Code'] == product_code)
& (final_result['v2'] == v2), 'Toplam Maliyet']
            if not matching_rows.empty:
                secondcost = matching_rows.values[0]
            else:
                secondcost = 0
            costdp += (5 * max(v1 + q1 - d1, 0) + 100 * max(d1 - q1 - v1, 0) + secondcost) *
D1

costdp += q1 * 10
data_2['Product_Name'].append(product_name)
data_2['Product_Code'].append(product_code)
data_2['City'].append(city)
data_2['Product_Group'].append(product_group)
data_2['Store_Name'].append(store_name)
data_2['v1'].append(v1)

```

```

        data_2['q1'].append(q1)
        data_2['Product Cost'].append(costdp)
    output_df = pd.DataFrame(data_2)

    result = output_df.groupby(['Product_Code', 'v1']).apply(lambda x: x.loc[x['Product
Cost'].idxmin()])[['Product_Code', 'Product_Name', 'City', 'Product_Group', 'Store_Name', 'v1', 'q1',
'Product Cost']]

    result = result.reset_index(drop=True)
    result['d1_range'] = result['v1'] + result['q1']
    expanded_rows = []
    for index, row in result.iterrows():
        for d1 in range(row['d1_range'] + 1):
            new_row = row.copy()
            new_row['d1'] = d1
            new_row['v2'] = row['v1'] + row['q1'] - d1
            new_row['Target_First_Period'] = row['v1'] + row['q1']
            v2 = new_row['v2']
            product_code = new_row['Product_Code']
            matching_rows = final_result[(final_result['v2'] == v2) & (final_result['Product_Code']
== product_code)]
            if not matching_rows.empty:
                q2 = matching_rows.iloc[0]['q2']
                new_row['q2'] = q2
            expanded_rows.append(new_row)
    expanded_df = pd.DataFrame(expanded_rows)
    expanded_df['Target_Second_Period'] = expanded_df['v2'] + expanded_df['q2']
    expanded_df = expanded_df[['Product_Name', 'Product_Code', 'City',
'Product_Group', 'Store_Name', 'v1', 'q1', 'Target_First_Period', 'v2', 'q2', 'Target_Second_Period']]
    expanded_df = expanded_df.copy()
    expanded_df.rename(columns={
        'v1': 'first_period_beginning_amount',
        'q1': 'first_period_shipment',

```

```

        'v2': 'second_period_beginning_amount',
        'q2': 'second_period_shipment',
    }, inplace=True)
    self.result_df = expanded_df
    self.displayResultData(expanded_df)
    self.filterComboBox1.clear()
    self.filterComboBox1.addItem("Product Name")
    self.filterComboBox1.addItems(expanded_df['Product_Name'].unique())
    self.filterComboBox2.clear()
    self.filterComboBox2.addItem("City")
    self.filterComboBox2.addItems(expanded_df['City'].unique())
    self.filterComboBox3.clear()
    self.filterComboBox3.addItem("Product Group")
    self.filterComboBox3.addItems(expanded_df['Product_Group'].unique())
    self.filterComboBox4.clear()
    self.filterComboBox4.addItem("Store Name")
    self.filterComboBox4.addItems(expanded_df['Store_Name'].unique())
def displayResultData(self, df):
    self.resultTable.setRowCount(df.shape[0])
    self.resultTable.setColumnCount(df.shape[1])
    self.resultTable.setHorizontalHeaderLabels(df.columns)
    for i in range(df.shape[0]):
        for j in range(df.shape[1]):
            self.resultTable.setItem(i, j, QTableWidgetItem(str(df.iat[i, j])))
def downloadExcel(self):
    if hasattr(self, 'result_df'):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getSaveFileName(self, "Save Excel File", "", "Excel Files (*.xlsx);;All Files (*)", options=options)
        if fileName:

```

```

        self.result_df.to_excel(fileName, index=False)
    else:
        self.resultTable.setRowCount(0)
        self.resultTable.setColumnCount(0)
def filterResults(self):
    if not hasattr(self, 'result_df'):
        return
    filtered_df = self.result_df.copy()
    filter1 = self.filterComboBox1.currentText()
    if filter1 != "Product Name":
        filtered_df = filtered_df[filtered_df['Product_Name'] == filter1]
    filter2 = self.filterComboBox2.currentText()
    if filter2 != "City":
        filtered_df = filtered_df[filtered_df['City'] == filter2]
    filter3 = self.filterComboBox3.currentText()
    if filter3 != "Product Group":
        filtered_df = filtered_df[filtered_df['Product_Group'] == filter3]
    filter4 = self.filterComboBox4.currentText()
    if filter4 != "Store Name":
        filtered_df = filtered_df[filtered_df['Store_Name'] == filter4]
    self.displayResultData(filtered_df)
def populateComboBoxes(self, df):
    product_names = ["Product Name"] + df['Product_Name'].unique().tolist()
    self.filterComboBox1.clear()
    self.filterComboBox1.addItem(product_names)
    cities = ["City"] + df['City'].unique().tolist()
    self.filterComboBox2.clear()
    self.filterComboBox2.addItem(cities)
    product_groups = ["Product Group"] + df['Product_Group'].unique().tolist()
    self.filterComboBox3.clear()

```

```
self.filterComboBox3.addItem(product_groups)
store_names = ["Store Name"] + df['Store_Name'].unique().tolist()
self.filterComboBox4.clear()
self.filterComboBox4.addItem(store_names)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = ExcelProcessor()
    ex.show()
    sys.exit(app.exec_())
```

REFERENCES

- [1] Giri, C., & Chen, Y. (2022). Deep learning for demand forecasting in the fashion and apparel retail industry. *Forecasting*, 4(2), 565-581. <https://doi.org/10.3390/forecast4020031>
- [2] Gallien, J., Mersereau, A. J., Garro, A., Mora, A. D., & Vidal, M. N. (2015). Initial shipment decisions for new products at Zara. *Operations Research*, 63(2), 269–286. <https://doi.org/10.1287/opre.2014.1343>
- [3] Jackson, P. L., & Muckstadt, J. A. (1989, February). Risk pooling in a two-period, two-echelon inventory stocking and allocation problem. *Naval Research Logistics*, 36(1), 1–26. [http://dx.doi.org/10.1002/1520-6750\(198902\)36:1<1::aid-nav3220360102>3.0.co;2-s](http://dx.doi.org/10.1002/1520-6750(198902)36:1<1::aid-nav3220360102>3.0.co;2-s)
- [4] Özer, Z. (2003, March). Replenishment Strategies for Distribution Systems Under Advance Demand Information. *Management Science*, 49(3), 255–272. <https://doi.org/10.1287/mnsc.49.3.255.12738>
- [5] Agrawal, N., & Smith, S. A. (2013, March). Optimal inventory management for a retail chain with diverse store demands. *European Journal of Operational Research*, 225(3), 393–403. <https://doi.org/10.1016/j.ejor.2012.10.006>
- [6] Fisher, M., & Rajaram, K. (2000, August). Accurate Retail Testing of Fashion Merchandise: Methodology and Application. *Marketing Science*, 19(3), 266–278. <https://doi.org/10.1287/mksc.19.3.266.11800>
- [7] Nambiar, M., Simchi-Levi, D., & Wang, H. (2020, December 14). Dynamic Inventory Allocation with Demand Learning for Seasonal Goods. *Production and Operations Management*, 30(3), 750–765. <https://doi.org/10.1111/poms.13315>

[8] Huang, H., Li, S., & Yu, Y. (2018, November 10). Evaluation of the allocation performance in a fashion retail chain using data envelopment analysis. *The Journal of the Textile Institute*, 110(6), 901–910. <https://doi.org/10.1080/00405000.2018.1532376>

[9] Naderi, S., Kilic, K., & Dasci, A. (2020, September). A deterministic model for the transshipment problem of a fast fashion retailer under capacity constraints. *International Journal of Production Economics*, 227, 107687. <https://doi.org/10.1016/j.ijpe.2020.107687>

[10] Skrivanek, S. (2009). *The Use of Dummy Variables in Regression Analysis* <https://www.moresteam.com/whitepapers/download/dummy-variables.pdf>

[11] Sykes, A., O. (1993), *An Introduction to Regression Analysis* https://chicagounbound.uchicago.edu/law_and_economics/51/

[12] Thieme, C., (2021), *Understanding Linear Regression Output in R* <https://towardsdatascience.com/understanding-linear-regression-output-in-r-7a9cbda948b3>

[13] Zach, (2020) *How to Interpret Regression Output in R* <https://www.statology.org/interpret-regression-output-in-r/>

[14] Bellman, R., & Kalaba, R. E. (1957). On the role of dynamic programming in statistical communication theory. *IEEE Transactions on Information Theory*, 3(3), 197–203. <https://doi.org/10.1109/tit.1957.1057416>

[15] Topaloglu, H., & Kunnumkal, S. (2006). Approximate dynamic programming methods for an inventory allocation problem under uncertainty. *Naval Research Logistics (NRL)*, 53(8), 822–841. <https://doi.org/10.1002/nav.20164>

[16] Oxford Academic. (n.d.). Analysis effect of K values used in K-fold cross validation. Oxford Academic. Retrieved from <https://academic.oup.com/forestry/article/87/5/654/2756029>

[17] Axsäter, S. (2000). Inventory control. In International series in management science/operations research/International series in operations research & management science. <https://doi.org/10.1007/978-1-4757-5606-7>

[18] BioMed Central. (n.d.). A Bayesian approach to negative binomial parameter estimation. BioMed Central. Retrieved from <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3044-0>

[19] Omar, R., Sharudin, S. A., & Ghazali, M. (2023). Investigating the user interface design frameworks of current mobile learning applications: A systematic review. Education Sciences, 13(1), 94. <https://doi.org/10.3390/educsci13010094>

[20] <https://realpython.com/python-pyqt-gui-calculator/>

[21] <https://www.tutorialspoint.com/pyqt5/index.htm>