Contents

[Introduction 1](#_Toc170551771)

[Data Exploration and Preprocessing 2](#_Toc170551772)

[Time Series Forecasting for Software Demand 7](#_Toc170551773)

[Classification Model 9](#_Toc170551774)

[Model Evaluation Results 9](#_Toc170551775)

[Confusion Matrix (Random Forest) 10](#_Toc170551776)

[Best Hyperparameters (Random Forest) 10](#_Toc170551777)

[Confusion Matrix (Logistic Regression) 10](#_Toc170551778)

[Best Hyperparameters (Logistic Regression) 10](#_Toc170551779)

[Confusion Matrix (Support Vector Machine) 11](#_Toc170551780)

[Best Hyperparameters (Support Vector Machine) 11](#_Toc170551781)

[Confusion Matrix (K-Nearest Neighbors) 11](#_Toc170551782)

[Best Hyperparameters (K-Nearest Neighbors) 11](#_Toc170551783)

[Anomaly Detection 12](#_Toc170551784)

[Recommendation system 13](#_Toc170551785)

[Most Popular Items 14](#_Toc170551786)

[Summary Statistics 15](#_Toc170551787)

[Conclusion 16](#_Toc170551788)

[References 16](#_Toc170551789)

[Appendix 16](#_Toc170551790)

[Part 1 EDA 16](#_Toc170551791)

[PART 1 Forecasting 19](#_Toc170551792)

[PART 2 21](#_Toc170551793)

[PART 2 Anomaly Detection 24](#_Toc170551794)

[Part 3 Recommendation 25](#_Toc170551795)

# Introduction

Recommendations, insights, analysis, and anomaly detection in multiple business sectors are vital areas where machine learning (ML) is effective. ML assists in proposing a unique solution to customers and needs, such as collaborative and content-related filtering. It also enables the process of customer clustering for marketing strategies and customer behaviour analysis with the help of algorithms such as K-means and Deep Cluster Networks (DCN). Furthermore, ML helps determine the likelihood of customer attrition by Analyzing the customers’ behaviour and flagging customers with the tendency to churn, which can be prevented by targeted marketing strategies. Furthermore, it contributes to identifying outliers, fraud, and other operation anomalies in customer data to improve business intelligence and decision-making factors. Applying ML methods can be highly beneficial for companies in helping them derive insights, enhance their clients’ satisfaction levels, and remain relevant within the business environment. Machine learning allows firms to apply customized client strategies, analyze data, conduct predictions and address issues with exception analysis in business analytics to improve business decisions and consumer experiences. Many papers have also discussed customer recommendations, insights, analysis, and alarms based on machine learning. Looking at the anomaly detection techniques based on Machine Learning (ML) models which have been analyzed, the use of customer reviews for evaluating similarity and recommendation, and the use of ML algorithms to classify product and shop information from the experience of the customers, the literature is filled with Methodologies.

This assignment covers classification of customers, anomaly detection, time series models and recommendation for an online retail store. Classification assists in sorting out the issues that a system has into more urgent ones, thus enhancing the functioning of operations. Anomaly detection is the process of detecting what is irregular in the logs that are important for security maintenance. Forecasting using time series focuses on the trends of the sales data to be helpful in the management of stocks. Last but not the least, through the recommendation system, it alerts the customers about other items to be purchased, making the customer engage and buy more.

# Data Exploration and Preprocessing

The EDA of the given time series data involved looking for trends, seasonality, and any irregularity in the data obtained from the following dataset. The dataset has fifteen columns, offering information on fifteen software products, with a summary of sixty rows for every product. Descriptive statistics determine the means and the spread of each software category. For instance, the mean value for Software\_1 is approximately 99. Likewise, other software products have also depicted dissimilar distributions, where the mean value of Software\_9 (128. 18) has a higher S. D. (11. 99) and indicates more variation regarding the sales and uses in the different periods.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Software | Mean | Std Dev | Min | 25% | 50% | 75% | Max |
| Software\_1 | 99.97 | 9.60 | 81 | 92 | 101 | 107 | 118 |
| Software\_2 | 102.28 | 10.55 | 79 | 94.75 | 103 | 107.75 | 128 |
| Software\_3 | 64.12 | 8.57 | 44 | 58 | 65 | 69 | 87 |
| Software\_4 | 88.12 | 10.76 | 66 | 82.5 | 88 | 94.25 | 114 |
| Software\_5 | 51.88 | 7.31 | 35 | 47 | 51 | 55.5 | 73 |
| Software\_6 | 89.90 | 8.48 | 71 | 84 | 90 | 96 | 106 |
| Software\_7 | 109.77 | 8.80 | 93 | 102.75 | 111.5 | 116.25 | 126 |
| Software\_8 | 67.80 | 9.14 | 48 | 62.75 | 68.5 | 74 | 90 |
| Software\_9 | 128.18 | 11.99 | 89 | 121.75 | 129 | 136 | 153 |
| Software\_10 | 94.22 | 8.59 | 70 | 89.75 | 93 | 98 | 117 |
| Software\_11 | 71.10 | 8.27 | 57 | 65 | 71 | 77 | 94 |
| Software\_12 | 122.75 | 11.19 | 100 | 115 | 121 | 129 | 161 |
| Software\_13 | 116.17 | 10.37 | 95 | 108.75 | 116.5 | 122.25 | 149 |
| Software\_14 | 78.73 | 7.80 | 60 | 72.75 | 79 | 84 | 95 |
| Software\_15 | 104.33 | 12.74 | 72 | 96 | 105 | 116 | 131 |

Fortunately, there are no missing values in columns, and the data analysis done here will be based on clean data. This completeness is essential for the proper trend and seasonality analysis.

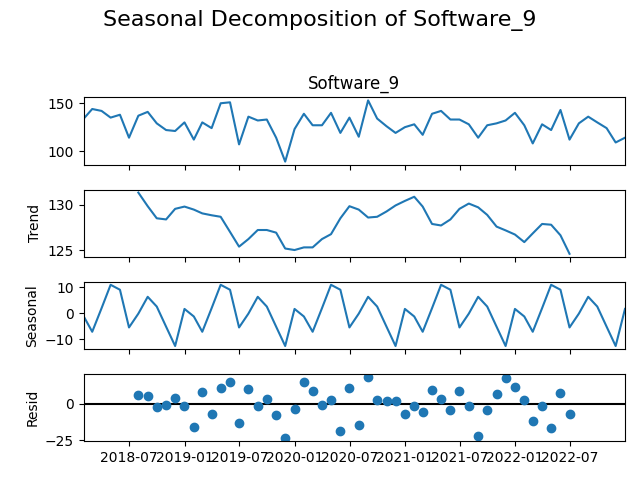
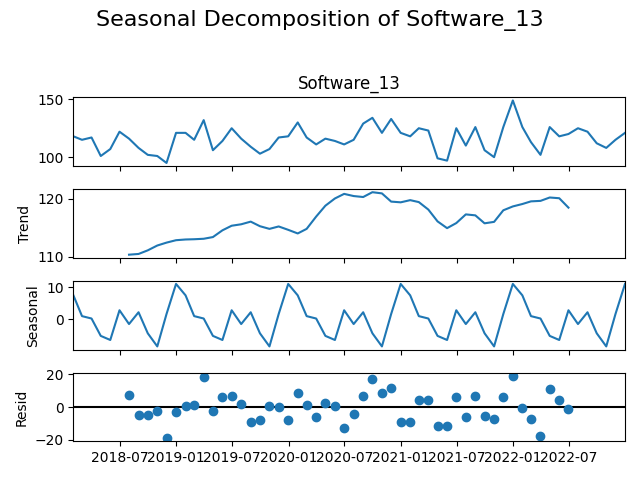
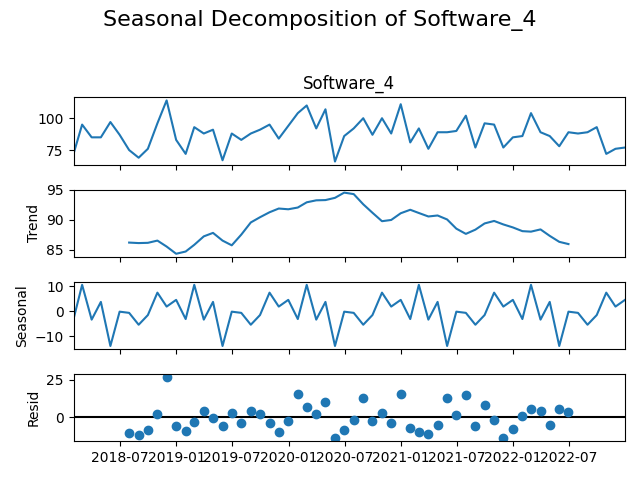
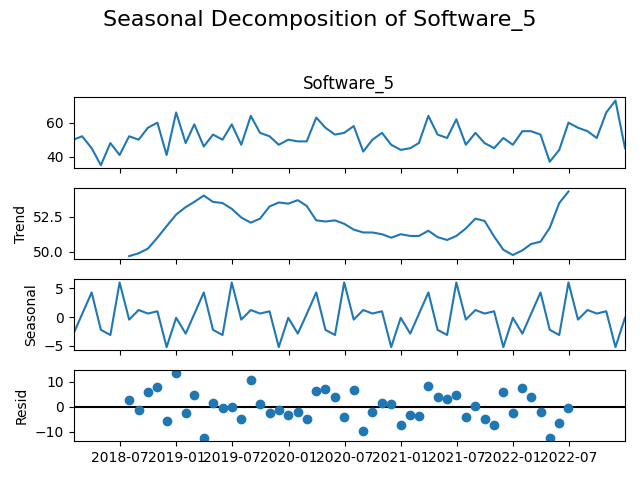
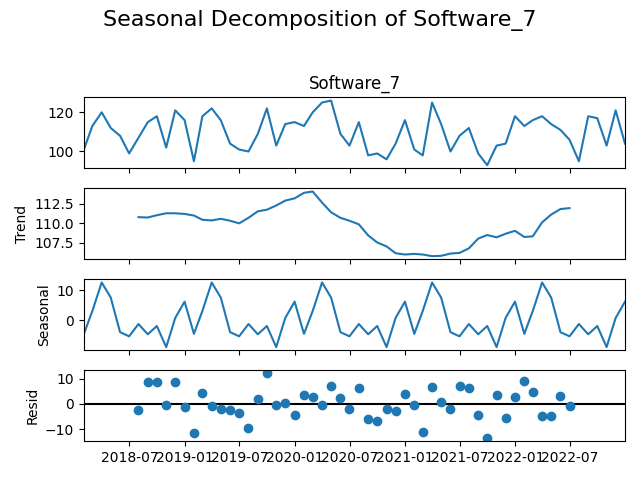
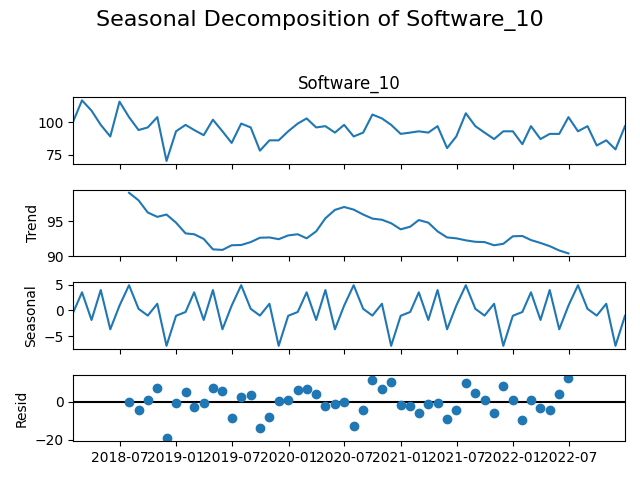
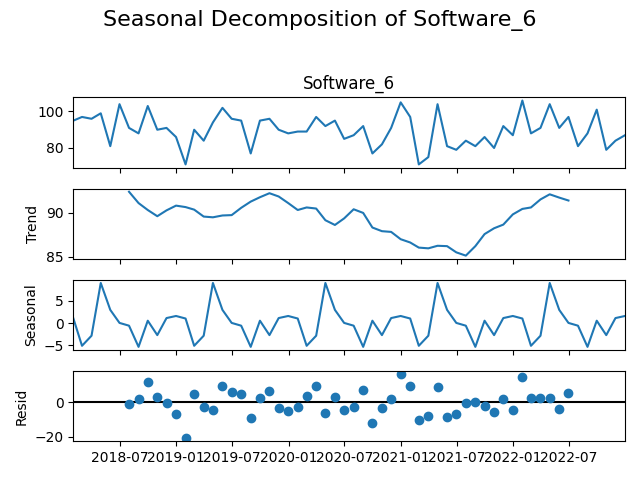
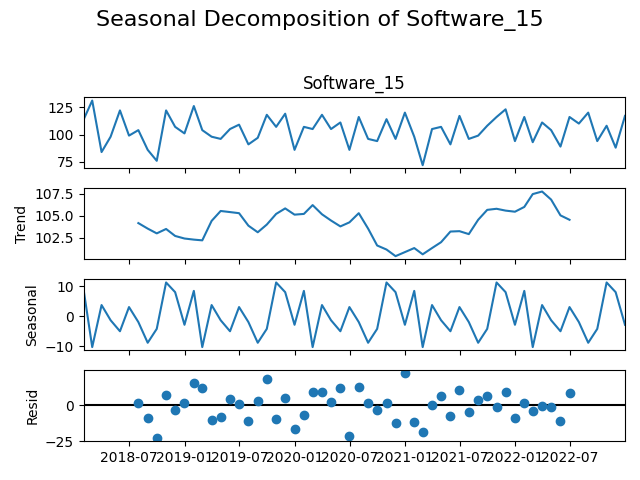
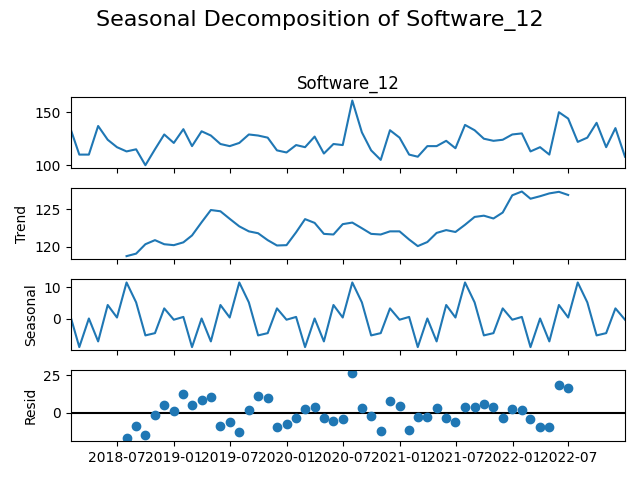
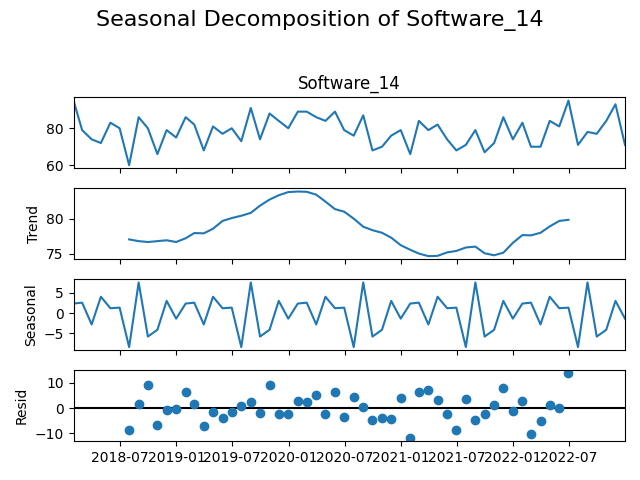
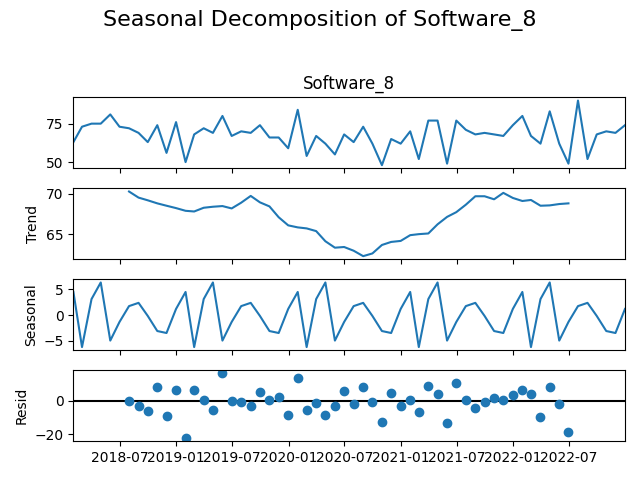
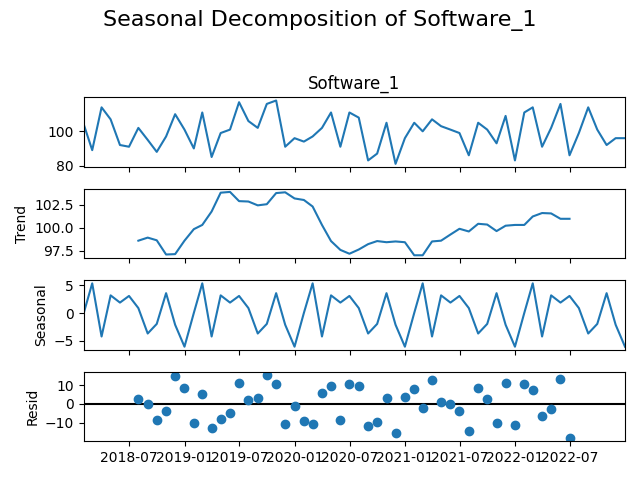
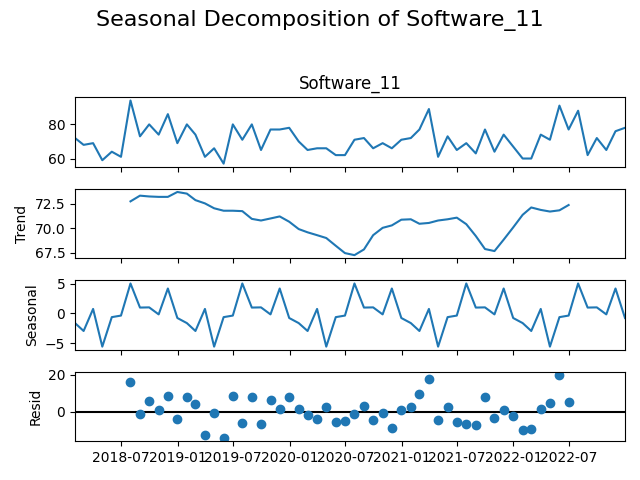
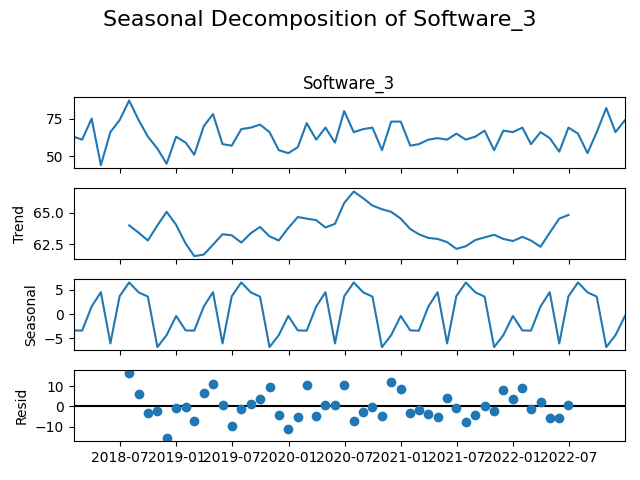
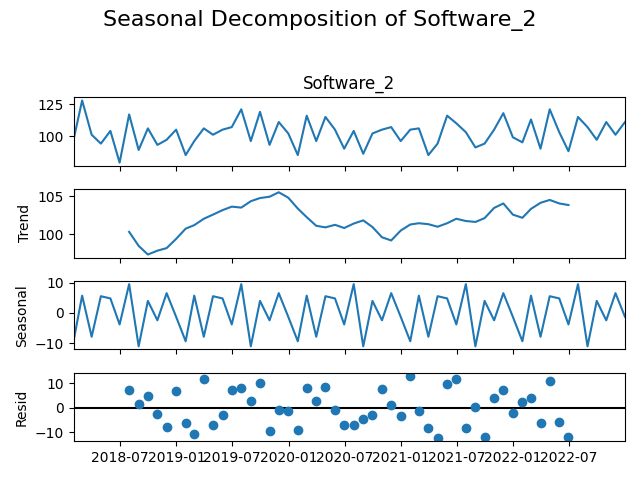
|  |  |
| --- | --- |
| Software | Missing Values |
| Software\_1 | 0 |
| Software\_2 | 0 |
| Software\_3 | 0 |
| Software\_4 | 0 |
| Software\_5 | 0 |
| Software\_6 | 0 |
| Software\_7 | 0 |
| Software\_8 | 0 |
| Software\_9 | 0 |
| Software\_10 | 0 |
| Software\_11 | 0 |
| Software\_12 | 0 |
| Software\_13 | 0 |
| Software\_14 | 0 |
| Software\_15 | 0 |

The time series usage is illustrated in the Figure below.

A screenshot of a computer

Description automatically generated

Further, seasonal decomposition was conducted to break down the time series data into its constituent components: trend, seasonality and the remainder. This decomposition assists in identifying the fundamental trends and cyclical patterns that may affect the software's usage or sale. The decomposition plots helped them understand each software's sales or usage trend in one dimension and revealed seasonal patterns and irregularities.

A graph of different types of data

Description automatically generated with medium confidence

Developing the EDA of the given data is beneficial, providing the primary information for cleaning this dataset and identifying the tendencies and seasonality for subsequent analyses. It becomes essential for creating sound forecast models and deciding on the strategic direction of businesses regarding software services.

# Time Series Forecasting for Software Demand

Specifically, in this task, we will use a Seasonal Autoregressive Integrated Moving Average (SARIMA) model to predict the demand for several software products. The dataset has the type of software shown alongside it and the demands per time on different software recorded. We first import the data, convert it to pandas, and convert the 'Date' column to DateTime, creating the basis of time-related analysis. Here, this particular column is set as the index of the DataFrame for more accessible analysis along a time series. Whenever there are blank/corresponding slots for the parameter in question, all chemical-solute rows are omitted to ensure the data is clean. Subsequently, we define the columns holding the software products and create a function called apply\_sarima\_and\_forecast to use SARIMA on each product. The given data is separated into the training set that includes all the data except 12 months at the end of the dataset, and the test set consists of the last 12 months of the dataset. This division allows the model's performance to be checked for making forecasts based on the data not included in the training stage. This is done with ARIMA components equal to (1, 1, 1) and seasonal components equal to (1, 1, 1, 12) where numbers indicate the order of integration, order of difference and thus, the number of identifying parameters in the models. The parameters of such a model are estimated using the least squares on the training data for the considered software product, and based on the built model, a forecast is produced, equal to 12 steps, corresponding to the test period. Thus, the forecasted values are stored in the test set to compare them with actual demand. For each software product, a plot of training data, test data, and forecasted value is produced and saved as a PNG file for further use.

A graph with blue lines and orange and green lines

Description automatically generatedA graph of blue and orange lines

Description automatically generatedA graph with blue and green lines

Description automatically generatedA graph with blue and orange lines

Description automatically generatedA graph of blue and orange lines

Description automatically generatedA graph of a graph

Description automatically generated with medium confidenceA graph of a graph

Description automatically generated with medium confidenceA graph of blue green and orange lines

Description automatically generatedA graph of a graph

Description automatically generated with medium confidenceA graph of a graph

Description automatically generated with medium confidenceA graph with blue and orange lines

Description automatically generatedA graph with blue lines and orange dots

Description automatically generatedA graph of blue and orange lines

Description automatically generated

Therefore, the Mean Squared Error (MSE) is computed to measure the closeness of the forecasted values to the actual test values. The above approach also has the advantage of revealing the model's performance and the demand for each software product in the distribution process to ensure strategic planning and stock control.

|  |  |  |  |
| --- | --- | --- | --- |
| Software | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Mean Absolute Percentage Error (MAPE) |
| Software\_1 | 223.68 | 12.33 | 11.92% |
| Software\_2 | 106.19 | 8.39 | 7.87% |
| Software\_3 | 101.09 | 6.84 | 10.29% |
| Software\_4 | 116.01 | 8.29 | 10.41% |
| Software\_5 | 135.67 | 9.93 | 18.30% |
| Software\_6 | 127.07 | 9.25 | 9.63% |
| Software\_7 | 166.80 | 11.16 | 9.94% |
| Software\_8 | 161.33 | 9.04 | 14.87% |
| Software\_9 | 116.22 | 7.80 | 6.67% |
| Software\_10 | 102.64 | 8.40 | 9.71% |
| Software\_11 | 130.75 | 9.36 | 12.53% |
| Software\_12 | 214.70 | 12.15 | 9.42% |
| Software\_13 | 142.83 | 10.07 | 8.79% |
| Software\_14 | 186.08 | 10.82 | 13.17% |
| Software\_15 | 181.10 | 10.80 | 10.26% |

# Classification Model

This task created a classification model to sort out system issues, such as issue type, component impacted, and the customer. To analyze, the data was imported using pandas, and the data type of the categorical variable, Issue\_Type, System\_Component, and a nominal variable, Customer\_Impact, was one hot encoded. The first preprocessing step was vital in transforming categorical data into a format easily understandable by the machine learning models. The Priority column was assigned to the target variable; to convert this variable into numerical labels, a LabelEncoder was used. Next, the feature variables (X) were standardized – this is carried out using StandardScaler to bring the values of each feature to a standard scale; it is helpful in normally distributed models, such as Support Vector Machines (SVM) and K Nearest Neighbors (KNN). The data was then randomly divided into training and testing data sets with a ratio of 80:20. This split is beneficial in assessing model performance using new data. The evaluated classifiers are Random Forests, Logistic Regression, SVM, and K NN. The evaluation of each model involved accuracy, confusion matrix, and classification report evaluations. I selected the Random Forest model for its high accuracy; it is even better when you use GridSearchCV to help determine the best hyperparameters. This involved creating different architectures of the parameters, such as several estimators, max depth, and others, to improve the model. After the tuning, a more accurate model was again tested, giving more information on the classification and performance of the model. The confusion matrix and classification report provided each priority level's precision, recall, and F1 division. They pointed out potential directions for further enhancement of the model's performance. This way of generating the selections offers considerable support in categorizing arriving system issues, aiding in prioritization and response.

Model Evaluation Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision (Avg) | Recall (Avg) | F1-Score (Avg) |
| Random Forest | 0.255 | 0.24 | 0.26 | 0.24 |
| Logistic Regression | 0.255 | 0.26 | 0.26 | 0.24 |
| Support Vector Machine | 0.265 | 0.26 | 0.27 | 0.25 |
| K-Nearest Neighbors | 0.250 | 0.24 | 0.25 | 0.23 |

Confusion Matrix (Random Forest)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Priority | Urgent | High | Medium | Low |
| Urgent | 16 | 12 | 10 | 5 |
| High | 18 | 14 | 11 | 11 |
| Medium | 13 | 9 | 16 | 7 |
| Low | 21 | 20 | 12 | 5 |

Best Hyperparameters (Random Forest)

|  |  |
| --- | --- |
| Parameter | Best Value |
| Bootstrap | False |
| Max Depth | None |
| Min Samples Split | 5 |
| Min Samples Leaf | 1 |
| Number of Estimators | 100 |

Confusion Matrix (Logistic Regression)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Priority | Urgent | High | Medium | Low |
| Urgent | 19 | 11 | 10 | 3 |
| High | 18 | 15 | 14 | 7 |
| Medium | 14 | 13 | 11 | 7 |
| Low | 17 | 15 | 20 | 6 |

Best Hyperparameters (Logistic Regression)

|  |  |
| --- | --- |
| Parameter | Best Value |
| C | 0.1 |
| Solver | blogs |

Confusion Matrix (Support Vector Machine)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Priority | Urgent | High | Medium | Low |
| Urgent | 17 | 9 | 12 | 5 |
| High | 17 | 16 | 14 | 7 |
| Medium | 15 | 11 | 16 | 3 |
| Low | 19 | 19 | 16 | 4 |

Best Hyperparameters (Support Vector Machine)

|  |  |
| --- | --- |
| Parameter | Best Value |
| C | 100 |
| Gamma | 0.01 |
| Kernel | RBF |

Confusion Matrix (K-Nearest Neighbors)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Priority | Urgent | High | Medium | Low |
| Urgent | 21 | 8 | 7 | 7 |
| High | 24 | 14 | 8 | 8 |
| Medium | 16 | 16 | 10 | 3 |
| Low | 22 | 13 | 18 | 5 |

Best Hyperparameters (K-Nearest Neighbors)

|  |  |
| --- | --- |
| Parameter | Best Value |
| Metric | manhattan |
| Neighbours | 9 |
| Weights | uniform |

# Anomaly Detection

An anomaly detection process involves the Isolation Forest algorithm to look for deviant patterns in system logs, which could mean security threats or system failures. First, preprocessing is performed, including dropping irrelevant columns and converting categorical columns to numerical ones by applying one hot encoding. The features for analysis exclude the 'Resolved' column. They are given as the feature matrix X: The model choice is the Isolation Forest with the contamination parameter = 0. 01, which is used in the model output to add items flagged as 'Anomaly' in a new column created in the dataset. Finally, the anomalies represented by -1 are plotted on a scatter plot against 'Time\_to\_Resolve\_hrs. ' The deviations are in red. The detected anomalies are escalated with details like the kind of issue the system component is involved in a customer contribution, helping identify problems within the system. This method proves helpful in discovering suspicious patterns as they can be further examined in the logs.

Detected Anomalies

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issue\_ID | Issue\_Type | System\_Component | Customer\_Impact | Time\_to\_Resolve\_hrs | Reported\_By | Priority | Previous\_Occurrences | Issue\_Reported\_Month | Resolved | Anomaly |
| 1040 | Server | Component\_D | High | 68.81 | Automated\_System | High | 9 | Sep | True | -1 |
| 1340 | Server | Component\_A | Low | 50.27 | Customer | High | 9 | Jul | True | -1 |
| 9365 | Server | Component\_A | High | 53.52 | Customer | Low | 1 | Sep | True | -1 |
| 1115 | Network | Component\_C | Medium | 62.35 | Customer | Medium | 0 | Jul | True | -1 |
| 3130 | Software | Component\_D | High | 2.00 | Automated\_System | Medium | 0 | Jun | False | -1 |
| 2948 | Network | Component\_D | Low | 69.62 | Customer | Medium | 9 | Apr | False | -1 |
| 3016 | Hardware | Component\_A | Low | 4.09 | Customer | Medium | 1 | Jul | False | -1 |
| 5444 | Software | Component\_D | Medium | 5.36 | Customer | Medium | 9 | Dec | False | -1 |
| 8263 | Hardware | Component\_A | Medium | 4.44 | Automated\_System | Medium | 0 | Dec | True | -1 |

A screenshot of a computer

Description automatically generated

# Recommendation system

The company served an online retail store and presented it with the tools to create a list of the fifty items most purchased daily. Furthermore, a recommendation system was integrated to improve the purchasing experience and present items that fit a customer's buying pattern. Returning to the present study, the market basket analysis and the association rules depict the various items customers will likely purchase together.

A screenshot of a computer

Description automatically generated

Selecting three of the clients at random, the algorithm creates the client-specific recommendations. This involves the following steps: taking the customers' past purchase records as a basis for determining sets of products which other customers usually purchase together but the customer still needs to purchase. The result is a specific list of guidelines, which will improve the shop's atmosphere and offer targeted products for a customer depending on preferences and purchases made earlier.

Most Popular Items

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| InvoiceNo | ItemCode | Most Popular Item | Quantity | Date | UnitPrice | CustomerID |
| 1.0 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6.0 | 01/12/2018 | 2.55 | 43009.0 |
| 2.0 | 71053 | WHITE METAL LANTERN | 6.0 | 02/12/2018 | 3.39 | 79874.0 |
| 3.0 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8.0 | 03/12/2018 | 2.75 | 45061.0 |
| 4.0 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6.0 | 04/12/2018 | 3.39 | 47110.0 |
| 5.0 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6.0 | 05/12/2018 | 3.39 | 77834.0 |

Summary Statistics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | InvoiceNo | Quantity | UnitPrice | CustomerID |
| Count | 1000 | 1000 | 1000 | 1000 |
| Mean | 500.5 | 12.785 | 3.037 | 55584.374 |
| Std Dev | 288.819 | 38.424 | 5.897 | 25435.887 |
| Min | 1 | -24 | 0 | 10563 |
| 25th Percent | 250.75 | 2 | 1.25 | 33045 |
| 50th Percent | 500.5 | 4 | 2.1 | 54561.5 |
| 75th Percent | 750.25 | 12 | 3.75 | 77877.25 |
| Max | 1000 | 600 | 165 | 99866 |

# Conclusion

Indeed, with the help of classification, anomaly detection, time series, and recommendation systems, the retail store can benefit dramatically and augment its service level with customers. This all-encompassing solution can improve employees’ operational efficiency while providing insights into the order inventory and customer preferences. In summary, this strategic approach promotes better efficiency, security, and customer satisfaction, resulting in business development.

# References

Ali, Bou, Nassif., Manar, Abu, Talib., Qassim, Nasir., Fatima, Mohamad, Dakalbab. (2021). Machine Learning for Anomaly Detection: A Systematic Review. IEEE Access, Available from: 10.1109/ACCESS.2021.3083060

N, Srinu., J., R., Paramesh., M., Sriram. (2023). Comprehensive Analysis of Deep Learning-based Customer Feedback Classification Methodologies for Recommender Systems. Available from: 10.1109/ICACCS57279.2023.10112661

Shanshan, Yi., Xiaofang, Liu. (2020). Machine learning based customer sentiment analysis for recommending shoppers, shops based on customers’ review. Complex & Intelligent Systems, Available from: 10.1007/S40747-020-00155-2

Ahmed, Alsayat. (2022). Customer decision-making analysis based on big social data using machine learning: a case study of hotels in Mecca. Neural Computing and Applications, Available from: 10.1007/s00521-022-07992-x

Qazi, Mudassar, Ilyas., Abid, Mehmood., Ashfaq, Ahmad., Muneer, Ahmad. (2022). A Systematic Study on a Customer’s Next-Items Recommendation Techniques. Sustainability, Available from: 10.3390/su14127175

# Appendix

# Part 1 EDA

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

df = pd.read\_csv('/kaggle/input/machine-learning-ca7/IT\_Company\_Time\_Series.csv')

# Convert 'Date' to datetime

df['Date'] = pd.to\_datetime(df['Date'])

# Set 'Date' as the index

df.set\_index('Date', inplace=True)

# Display basic information about the dataset

print(df.info())

print(df.describe())

# Check for missing values

missing\_values = df.isnull().sum()

print(f"Missing values:\n{missing\_values}")

# Impute or drop missing values

df = df.fillna(method='ffill') # Forward fill as an example

# Plot the time series for each software

plt.figure(figsize=(15, 10))

for column in df.columns:

plt.plot(df.index, df[column], label=column)

plt.title('Time Series of Software Usage')

plt.xlabel('Date')

plt.ylabel('Usage')

plt.legend()

fig.savefig(f'Time Series of Software Usage.png')

plt.show()

# Check for trends and seasonality using decomposition

from statsmodels.tsa.seasonal import seasonal\_decompose

for column in df.columns:

decomposition = seasonal\_decompose(df[column], model='additive')

fig = decomposition.plot()

fig.suptitle(f'Seasonal Decomposition of {column}', fontsize=16)

fig.tight\_layout(rect=[0, 0, 1, 0.95]) # Adjust layout to fit the title

# Save the figure

fig.savefig(f'{column}\_decomposition.png')

plt.show()

# Check for data inconsistencies

inconsistent\_data = df[(df < 0) | (df > df.quantile(0.99))] # Example threshold for inconsistency

print(f"Inconsistent data:\n{inconsistent\_data}")

# Remove or handle inconsistencies

df = df.clip(lower=0) # Example of handling negative values

# PART 1 Forecasting

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

from sklearn.metrics import mean\_squared\_error

from statsmodels.tsa.seasonal import seasonal\_decompose

# Load the dataset

time\_series\_df = pd.read\_csv('/kaggle/input/machine-learning-ca7/IT\_Company\_Time\_Series.csv')

# Convert 'Date' to datetime

time\_series\_df['Date'] = pd.to\_datetime(time\_series\_df['Date'])

# Set 'Date' as the index

time\_series\_df.set\_index('Date', inplace=True)

# Check for null values

print(time\_series\_df.isnull().sum())

# Drop rows with null values

time\_series\_df.dropna(inplace=True)

# List of software columns

software\_columns = time\_series\_df.columns

# Function to apply SARIMA model and plot the results

def apply\_sarima\_and\_forecast(data, column):

# Split data into training and test sets

train = data.iloc[:-12]

test = data.iloc[-12:]

# Apply SARIMA

sarima\_model = SARIMAX(train[column], order=(1, 1, 1), seasonal\_order=(1, 1, 1, 12))

sarima\_fit = sarima\_model.fit(disp=False)

# Forecast

forecast = sarima\_fit.get\_forecast(steps=12)

forecast\_index = test.index

forecast\_values = forecast.predicted\_mean

test['Forecast'] = forecast\_values.values

# Plot the forecast

plt.figure(figsize=(12, 6))

plt.plot(train[column], label='Train')

plt.plot(test[column], label='Test')

plt.plot(test['Forecast'], label='Forecast', linestyle='--')

plt.title(f'{column} Demand Forecast')

plt.xlabel('Date')

plt.ylabel('Demand')

plt.legend()

plt.savefig(f'{column}\_demand\_forecast.png') # Save the plot as a PNG file

plt.close() # Close the plot to avoid display

# Calculate the mean squared error

mse = mean\_squared\_error(test[column], test['Forecast'])

print(f'{column} Mean Squared Error: {mse}')

# Apply SARIMA model to each software column

for column in software\_columns:

apply\_sarima\_and\_forecast(time\_series\_df, column)

# PART 2

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

from sklearn.metrics import mean\_squared\_error

from statsmodels.tsa.seasonal import seasonal\_decompose

# Load the dataset

time\_series\_df = pd.read\_csv('/kaggle/input/machine-learning-ca7/IT\_Company\_Time\_Series.csv')

# Convert 'Date' to datetime

time\_series\_df['Date'] = pd.to\_datetime(time\_series\_df['Date'])

# Set 'Date' as the index

time\_series\_df.set\_index('Date', inplace=True)

# Check for null values

print(time\_series\_df.isnull().sum())

# Drop rows with null values

time\_series\_df.dropna(inplace=True)

# List of software columns

software\_columns = time\_series\_df.columns

# Function to apply SARIMA model and plot the results

def apply\_sarima\_and\_forecast(data, column):

# Split data into training and test sets

train = data.iloc[:-12]

test = data.iloc[-12:]

# Apply SARIMA

sarima\_model = SARIMAX(train[column], order=(1, 1, 1), seasonal\_order=(1, 1, 1, 12))

sarima\_fit = sarima\_model.fit(disp=False)

# Forecast

forecast = sarima\_fit.get\_forecast(steps=12)

forecast\_index = test.index

forecast\_values = forecast.predicted\_mean

test['Forecast'] = forecast\_values.values

# Plot the forecast

plt.figure(figsize=(12, 6))

plt.plot(train[column], label='Train')

plt.plot(test[column], label='Test')

plt.plot(test['Forecast'], label='Forecast', linestyle='--')

plt.title(f'{column} Demand Forecast')

plt.xlabel('Date')

plt.ylabel('Demand')

plt.legend()

plt.savefig(f'{column}\_demand\_forecast.png') # Save the plot as a PNG file

plt.close() # Close the plot to avoid display

# Calculate the mean squared error

mse = mean\_squared\_error(test[column], test['Forecast'])

print(f'{column} Mean Squared Error: {mse}')

# Apply SARIMA model to each software column

for column in software\_columns:

apply\_sarima\_and\_forecast(time\_series\_df, column)

# PART 2 Anomaly Detection

import pandas as pd

from sklearn.ensemble import IsolationForest

import matplotlib.pyplot as plt

# Load the dataset

issues\_df = pd.read\_csv('/kaggle/input/machine-learning-ca7/IT\_Company\_System\_Issues\_Classification.csv')

# Drop the index column if present

if 'Unnamed: 0' in issues\_df.columns:

issues\_df = issues\_df.drop(columns=['Unnamed: 0'])

# Encode the categorical variables using one-hot encoding

issues\_df\_encoded = pd.get\_dummies(issues\_df, columns=['Issue\_Type', 'System\_Component', 'Customer\_Impact', 'Reported\_By', 'Priority', 'Issue\_Reported\_Month'])

# Define the features 'X'

X = issues\_df\_encoded.drop('Resolved', axis=1)

# Fit the Isolation Forest model

iso\_forest = IsolationForest(contamination=0.01, random\_state=42)

issues\_df['Anomaly'] = iso\_forest.fit\_predict(X)

# Plot anomalies

anomalies = issues\_df[issues\_df['Anomaly'] == -1]

plt.figure(figsize=(12, 6))

plt.plot(issues\_df.index, issues\_df['Time\_to\_Resolve\_hrs'], label='Time to Resolve (hrs)')

plt.scatter(anomalies.index, anomalies['Time\_to\_Resolve\_hrs'], color='red', label='Anomalies')

plt.legend()

plt.show()

# Display detected anomalies

print("Detected anomalies:")

print(anomalies)

# Part 3 Recommendation

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

retail\_df = pd.read\_csv('/kaggle/input/machine-learning-ca7/IT\_Retail\_List.csv')

# Display the first few rows

print(retail\_df.head())

# Summary statistics

print(retail\_df.describe())

# Check for missing values

print(retail\_df.isnull().sum())

# Convert 'Date' to datetime, handling different date formats

retail\_df['Date'] = pd.to\_datetime(retail\_df['Date'], dayfirst=True, errors='coerce')

# Check for any parsing issues

print(retail\_df[retail\_df['Date'].isnull()])

# Handle missing values if any (removing rows with null dates or other null values)

retail\_df.dropna(inplace=True)

# Plotting the quantity of items sold over time

plt.figure(figsize=(12, 6))

plt.plot(retail\_df['Date'], retail\_df['Quantity'], marker='o', linestyle='-', color='b')

plt.xlabel('Date')

plt.ylabel('Quantity')

plt.title('Quantity of Items Sold Over Time')

plt.xticks(rotation=45)

plt.tight\_layout()

fig.savefig(f'Retail Store.png')

plt.show()

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

import random

# Load the dataset

retail\_df = pd.read\_csv('/kaggle/input/machine-learning-ca7/IT\_Retail\_List.csv')

# Data preprocessing: One-hot encoding

basket = (retail\_df

.groupby(['CustomerID', 'ItemCode'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('CustomerID'))

# Convert quantities to 1s and 0s

basket = basket.applymap(lambda x: 1 if x > 0 else 0)

# Frequent itemsets using Apriori

frequent\_itemsets = apriori(basket, min\_support=0.0005, use\_colnames=True)

# Generate association rules

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=0.05)

# Select 3 random customers

random\_customers = random.sample(list(basket.index), 3)

print(f"Selected customers: {random\_customers}")

# Generate recommendations for each customer

for customer in random\_customers:

purchased\_items = set(basket.columns[basket.loc[customer] > 0])

recommendations = set()

for \_, row in rules.iterrows():

if purchased\_items.intersection(row['antecedents']) and not purchased\_items.intersection(row['consequents']):

recommendations.update(row['consequents'])

print(f"Recommendations for Customer {customer}: {list(recommendations)[:3]}")