KAZAKH-BRITISH TECHNICAL UNIVERSITY

REPORT

on Data Mining/ assignment 4

Customer Behavior Analysis and Prediction

student Serikkanov Nurtas Mukhituly

(surname, first name, patronymic)

year of study 4 specialty / educational program

Information System

School of Information Technologies and Engineering

Date: 24.11.2024

Table of Contents

1. Executive Summary
2. Introduction
3. Data Description
4. Data Preprocessing
5. Data Exploration and Visualization
6. Feature Selection and Dimensionality Reduction
7. Classification Techniques
8. Advanced Classification Methods
9. Clustering Techniques
10. Advanced Clustering Techniques
11. Association Rule Mining
12. Anomaly Detection
13. Time Series Analysis
14. Text Mining and NLP
15. Challenges and Solutions
16. Conclusion
17. References

1. Executive Summary

For this specific project, I have deployed a combination of data mining techniques to study customer actions and trends in the purchases made on the online retail website. The aim of the project was to Analyze the data well and develop predictions about different variables as well as spot outliers in the data that could assist in decision making. The data that was used for this research is obtained from an online retail shop in the UK. The data contained transactions such as demographic details of customers, their products and what purchases customers made.
The project had an outline that was adhered to which included: data cleaning, conducting exploratory data analysis (EDA), feature selection and dimensionality reduction methods which involved Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA). Thereafter, classification algorithms such as Logistic Regression and Decision Trees were utilized in forecasting customer behavior while clustering algorithms such as K-Means and DBSCAN were applied to capture the purchasing patterns of the customers. Moreover, I also made use of sophisticated outlier detection techniques such as Isolation Forest and LOF (Local Outlier Factor) to identify the outliers in the datasets. Moreover, I dealt with membership patterns and association rule mining using the Apriori method which looks for multipurpose rules on the basis of minimal occurrence of the items. These rules also give the context of the products being bought and correlating them with each other in transactions.
An important contribution of the project was the detection of various consistent purchase patterns, which would be useful for customized marketing techniques. In addition, I identified some abnormal transactions that could suggest possible fraud. The time series analysis also predicted sales in the future, which made it possible to propose inventory and demand as well as supply forecasting.
Main suggestions are to improve classification models for better performance and also to investigate customer segmentation for direct marketing purposes, in subtler detail. The findings of this project can give better understanding of the cust

2. Introduction

Background

With the increasing size of data generated every day, there has been a rise in the interest for data analysis to find valuable information that can aid in achieving the goals of the organization. In this regard, the term data mining can be viewed as a subset of business analytics that entails the extraction of useful patterns, insights, and knowledge from a plethora of data. Decisions can be supported by data and statistical methods, prognostic techniques, and hidden relationships in the database can be revealed. Business analytics owes its relevance to data mining because it is quite useful in offering insights that are capable of making the operations of the organizations more efficient, services better, and thus, attaining the cut-throat market. Data mining is done in almost all sectors including but not limited to retail, healthcare, finance, and marketing with the aim of seeking growth opportunities, carrying out strategy optimization, and risk minimization.
For instance, data mining has played an important role in customer behavior analysis in the retail sector. More so, why, when, and what a customer is likely to buy distinguishing all the product preferences while also paying attention to the seasonal changes makes it easier for the company to adjust its inventory and the way it markets its products/services while also forecasting future sales. In retailing, data mining can also be applied in the segmentation of customers, product targeting, and even in the detection of fraud, among other applicable areas.

Project Goals
This project intends to look into the consumer behavior by use of data mining techniques. This project will analyze transactional data of an online retail business in order to assess the purchasing patterns of the customers, make projections of their future purchases and understand the determinants of their purchases. In particular, the project shall be concerned with the following goals:

- Transformation and modification of the raw data for quality purposes and readiness for analysis.
- Using classification models to estimate customer buying behavior.
- Using the technique of cluster analysis to find groups of customers.
- Studying the methods of anomaly detection with the aim of finding outliers or instances of fraud.
- Applying the technique of association rule mining for product co-purchases.

These objectives should serve to enhance the knowledge on customer behavior which is very important to business activities in the quest to improve internal processes and decision making.

Scope
The present project deals with analyzing the transactional data of an online retail store based in the United Kingdom. The main source of primary data for this project consists of transaction records that include product, quantity, customer and price information. The project consists of the following activities:

- Data preprocessing in particular, solving the issues of missing data, encoding categorical variables and eliminating extraneous data from the white paper.
- To perform EDA that is exploratory data analysis to discover the trends which are available.
- Creating Predictive Behavioral and Customer Segmentation, by applying Classification Models (Logistic Regression, Decision Trees) and Clustering Algorithms (K-Means, DBSCAN).
- Detecting outliers with Isolation Forest and Local Outlier Factor (LOF) methods.
- Use of Association rule mining in market basket analysis using Apriori algorithm for generating frequent itemsets and association rules mining.

There are certain drawbacks to the assigned tasks in the project especially regarding the range of referred data. The given dataset is targeted on one single retailer as the scope of the analysis is on small weekly purchases and may not address fully, a cross fantastic customer behavior in other areas. Also, with this project, one can understand consumer behavior to some degree, but such projections are based on already-existing transactional patterns and do not consider other market determinants that may alter in the future such as reallocation, economic downturns or competitors.

3. Data Description

The project utilized the Online Retail Dataset containing transactional records of an online retailer based in the UK. The data records the buying behavior of customers towards specific products that includes information about the product, customer, date of transaction, and quantity purchased. It aids in understanding customer tastes, the direction of their wave patterns of purchases, and general sales in chronological order.

Dataset Source The dataset was collected from one of the public datasets collected by Kaggle: data science competitions' supplier. This dataset is free to use and many people employ it for the purposes of data mining and analysis. The dataset can also be downloaded straight away from issuing Kaggle.

Dataset Structure The sample data comprises 541,909 rows and 8 columns, detailed as follows:

```
[2]: import pandas as pd
     import numpy as np

     data = pd.read_csv('online_retail.csv')

     data.head()
     # print(data.info())
```

[2]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

```
[3]: data.shape
```

```
[3]: (541909, 8)
```

InvoiceNo: It is a number and assigns each transaction a unique identifier (non-empty).
StockCode: Unique identification element with each item that is brought (text).
Description: Statement showing what the item is (text).
Quantity: Amount of the item that has been bought (number).
InvoiceDate: When (date/time) the transaction was completed (transaction date time).
UnitPrice: The price associated with one unit of the commodity (number).
CustomerID: A number assigned to a customer that identifies her/him (non-empty).
Country: Indicates the nation of residence of the buyer (text).
Data Typesise
Number Data: The Quantity, UnitPrice, and CustomerID values have been findings of this feature. These features are of main concern during quantitative measurements like customer segmentation, purchasing analysis, and products analyses of sales.
Categorical data: The data contained within InvoiceNo, StockCode, Description, and Country. These categorical variables would form a basis for clustering and association rule mining for understanding combinations of products and regional preferences.
Datetime data: These InvoiceDates could allow time-based analysis that again cited seasonal trends and modeled time series into forecasting.
Limitations and Assumptions
The data repository is specific to transactions of a single retailer in the UK, thus not representing worldwide phenomena or behaviors.
Only full transaction records used, leaving out cases of missing and corrupted data-this could imply a minor loss of data but ensures a clean, reliable source for analysis.
External factors influencing behavior by which purchases occurred such as promotion, weather, or pandemic, were not considered in the dataset.

4. Data Preprocessing

The data preprocessing of the Online Retail Dataset was extraordinarily conspicuous in cleaning up the data and making it consistent with data preparation for analysis. Preprocessing included the treatment of missing data, the elimination of irrelevant and incorrect data, and putting data into formats useful for further analysis.

Handling Missing Data

Initially, we had missing entries in the data, where it had 1,454 missing entries in the Description column, and these were removed using the dropna() function. Thus, all analysis rows have valid product descriptions, quantities, and other such critical data points. The shape of the data reduced from 541909 to 397884 after dropping these rows.

```
[3]: data.shape

[3]: (541909, 8)

[4]: print(data.isnull().sum())
     data = data.dropna()

     InvoiceNo          0
     StockCode          0
     Description     1454
     Quantity           0
     InvoiceDate        0
     UnitPrice          0
     CustomerID     135080
     Country            0
     dtype: int64

[5]: print(data.isnull().sum())
     data = data[(data['Quantity'] > 0) & (data['UnitPrice'] > 0)]
     data.shape

     InvoiceNo      0
     StockCode      0
     Description    0
     Quantity       0
     InvoiceDate    0
     UnitPrice      0
     CustomerID     0
     Country        0
     dtype: int64
[5]: (397884, 8)
```

Filtering Invalid Entries

Further purification of entries was done to get rid of any entries with zero or negative quantities and unit prices. This was important as it was noticed that transactions falling under the category of negative and zero quantities do not represent an actual buying event and hence should not take place in the dataset.

Encoding Categorical Variables

We applied Label Encoding on the Country column for the sole reason of making the categorical variables interpretable to machine learning models, by converting them into numerical labels for use in models that would otherwise not handle categorical inputs directly.

```
[6]: from sklearn.preprocessing import LabelEncoder

     label_encoder = LabelEncoder()
     data['Country'] = label_encoder.fit_transform(data['Country'])

     data.head()
```

[6]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	35
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	35
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	35
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	35
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	35

Now, the column Country can be used for analysis wherein each Country is represented by a unique number.

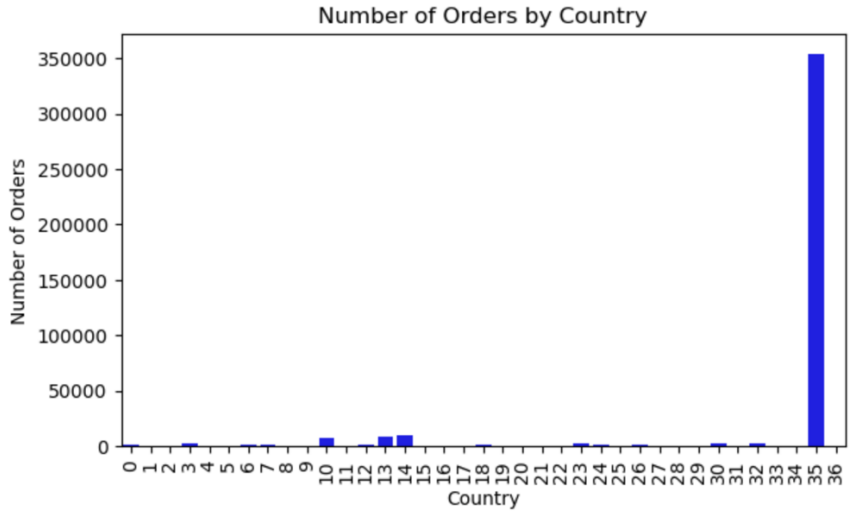
Data Pre-processing Summary: Missing values: Rows with missing descriptions were deleted, thereby reducing the size of the dataset. Invalid records: Records of transactions that

had zero or negative quantities and unit prices had to be nullified in order for the data to validly represent sales. Categorical transformation: The Country column was transformed into numeric values through label encoding, rendering it machine-learning algorithm compatible. These pre-processing steps were necessary to prepare the data so that it could be analyzed further when clean and useful to discover trends and patterns and model building.

5. Data Exploration and Visualization

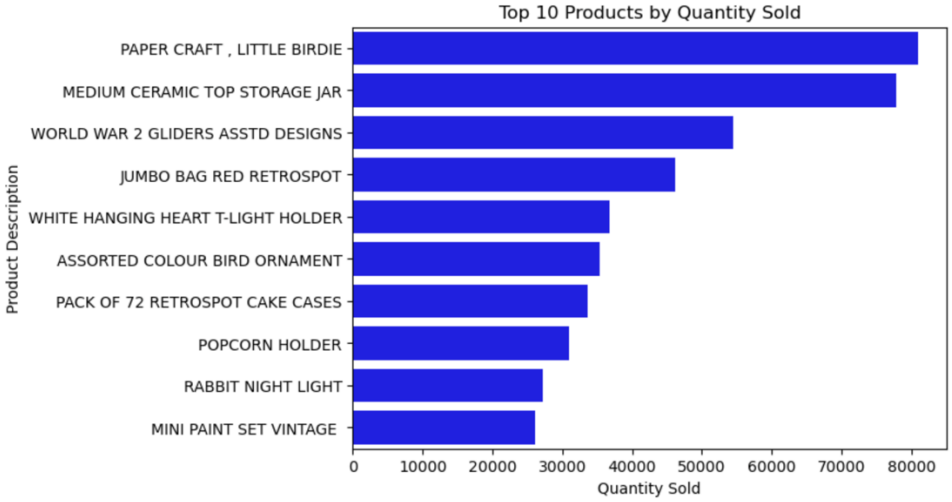
This dataset has transactional data of an online retail store that is concerned with this project. The exploratory data analysis (EDA) used in the project would target identifying trends, patterns, and essential insights in the data that would be helpful in making business strategies and decisions.
Major Steps in Data Exploration: Country-wise Distribution of Orders: I first observed the number of orders on the basis of countries as an attempt to check the customer demographics. The first visualization shows that the United Kingdom contributes the highest number of orders. Further plotting gives evidence of most orders placed by the customers in that country via a bar plot.
Top Products by Quantity Sold: I analyzed the best-selling products in terms of quantity. The items selling most frequently were highlighted by aggregating data according to StockCode and Description. A bar plot visualization was created to represent the quantity of products sold in which some popular eye-catching products had names like "Paper Craft, Little Birdie" and "Medium Ceramic Top Storage Jar".
Top Products Based on the Number of Unique Customers: Other approaches analyzed products based on the number of customers who really bought them. This point tries to comment about the widespread nature of a product to some customer base. It has resulted in the finding that the products presented like "Regency Cakestand 3 Tier" and "White Hanging Heart T-Light Holder" rank among the top acquired by the numerous unique customers.

Visualizations :
Bar Plot of Number of Orders by Country : The United Kingdom seems to be the obvious majority with respect to the volume of orders made, with a far smaller amount made from each other country.



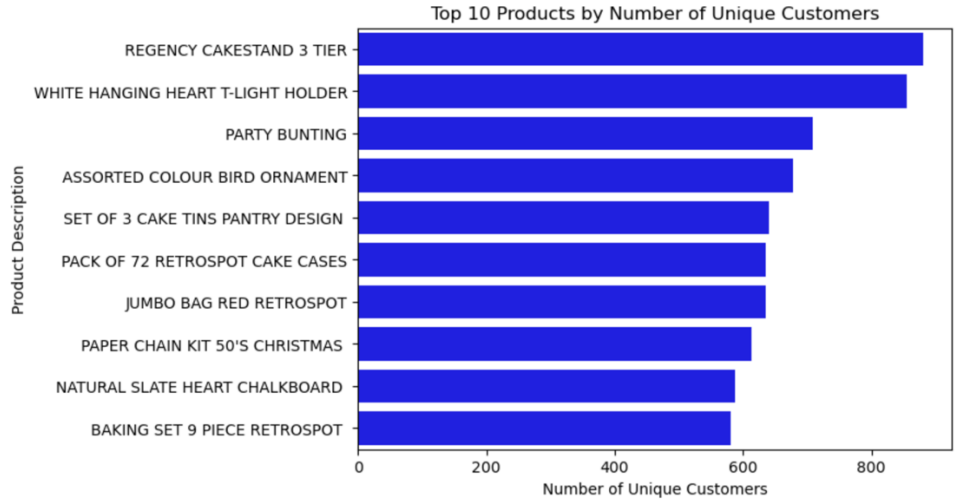
Top Products Sold by Quantity: The sales volume of product as shown in a horizontal bar was listed as the first most sold product to date. The highest selling was listed as "Paper Craft, Little Birdie".

StockCode	Description	Quantity
23843	PAPER CRAFT , LITTLE BIRDIE	80995
23166	MEDIUM CERAMIC TOP STORAGE JAR	77916
84077	WORLD WAR 2 GLIDERS ASSTD DESIGNS	54415
85099B	JUMBO BAG RED RETROSPOT	46181
85123A	WHITE HANGING HEART T-LIGHT HOLDER	36725
84879	ASSORTED COLOUR BIRD ORNAMENT	35362
21212	PACK OF 72 RETROSPOT CAKE CASES	33693
22197	POPCORN HOLDER	30931
23084	RABBIT NIGHT LIGHT	27202
22492	MINI PAINT SET VINTAGE	26076



Top Products by Number of Unique Customers : This chart shows a summary of how many different customers patronize a certain product. This gives insight on which products are bought most dearly by unique customers.

StockCode	Description	CustomerID
22423	REGENCY CAKESTAND 3 TIER	881
85123A	WHITE HANGING HEART T-LIGHT HOLDER	856
47566	PARTY BUNTING	708
84879	ASSORTED COLOUR BIRD ORNAMENT	678
22720	SET OF 3 CAKE TINS PANTRY DESIGN	640
21212	PACK OF 72 RETROSPOT CAKE CASES	635
85099B	JUMBO BAG RED RETROSPOT	635
22086	PAPER CHAIN KIT 50'S CHRISTMAS	613
22457	NATURAL SLATE HEART CHALKBOARD	587
22138	BAKING SET 9 PIECE RETROSPOT	581



These among other visualizations and analyses show the patterns of purchases and behavior of the customer, which can give an organization the platform from which it makes informed decisions in business about stock optimization and targeting market segments.

6. Feature Selection and Dimensionality Reduction

In the said assignment, I utilized the two fundamental techniques for improving the datasets and getting them ready for further modeling: Feature Selection and Dimensionality Reduction.

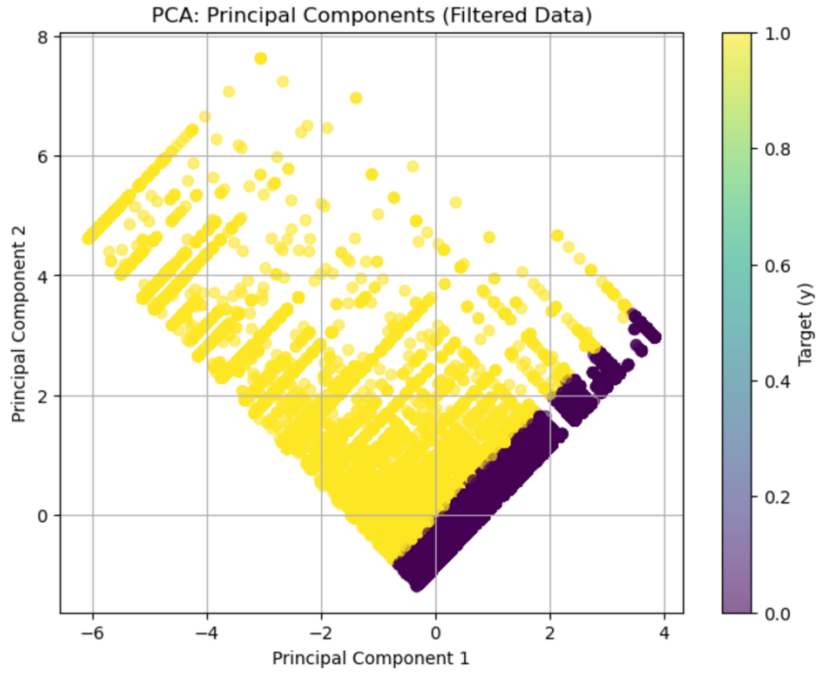
1) *Feature Selection:* The recursive Feature Elimination (RFE) employed is used to select the most important features in the dataset. RFE is a method where some of the least significant features are eliminated based on the performance of some model, so what is left would be the most relevant features in the prediction. In my case, I applied RFE using a Random Forest Classifier as the estimator, which helped identify the two most influential features: Quantity and UnitPrice. The ranking output indicated that these two features had the highest importance, which can be used for future modeling tasks.
   Here is the code which is used for the RFE:

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier

X = data[['Quantity', 'UnitPrice', 'Country']]
y = (data['Quantity'] > data['Quantity'].mean()).astype(int)

model = RandomForestClassifier(n_estimators=100, random_state=42)

# Применение RFE для выбора признаков
rfe = RFE(estimator=model, n_features_to_select=2)  # Оставляем два наиболее важных признака
fit = rfe.fit(X, y)

print("Feature Rankings (lower is better):")
for i in range(X.shape[1]):
    print(f"Feature: {X.columns[i]}, Rank: {fit.ranking_[i]}")

selected_features = X.columns[fit.support_]
print("\nSelected Features:", selected_features)
X_selected = X[selected_features]
```

```
Feature Rankings (lower is better):
Feature: Quantity, Rank: 1
Feature: UnitPrice, Rank: 1
Feature: Country, Rank: 2

Selected Features: Index(['Quantity', 'UnitPrice'], dtype='object')
```

2) *Dimensionality Reduction:* Also, I made use of PCA as my dimensionality reduction approach. PCA translates the original attributes into a set of variables (called "principal components") that are linearly uncorrelated by extracting components in descending order of variance explained by the underlying data. It reduces the data complexity while retaining its most important information. Then, after filtering and scaling the data, I put PCA on it to reduce the dataset to two principal components. The visualisation clearly differentiated the data in the two new dimensions into something more easily analyzable and potentially predictive.
   Here is how PCA implemented:

```python
quantity_threshold = np.percentile(data['Quantity'], 99)
unitprice_threshold = np.percentile(data['UnitPrice'], 99)

filtered_data = data[(data['Quantity'] <= quantity_threshold) & (data['UnitPrice'] <= unitprice_threshold)]
X_filtered = filtered_data[['Quantity', 'UnitPrice']]
y_filtered = (filtered_data['Quantity'] > filtered_data['Quantity'].mean()).astype(int)

# Масштабируем данные
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_filtered)


from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Применение PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)

pca_df = pd.DataFrame(data=principal_components, columns=['Principal Component 1', 'Principal Component 2'])

plt.figure(figsize=(8, 6))
plt.scatter(pca_df['Principal Component 1'], pca_df['Principal Component 2'], alpha=0.6, c=y_filtered, cmap='viridis')
plt.colorbar(label='Target (y)')
plt.title('PCA: Principal Components (Filtered Data)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()
```



These methods much improve the analysis by addressing the most relevant features and rendering the data less complex and simpler, hence easily interpretable and modelable.

7. Classification Techniques

I developed two algorithms commonly employed for classification model purposes: namely, the Logistic Regression and the Decision Tree Classifier. These models were used for predicting the binary target value solely based on parameter values of Quantity, UnitPrice, and Country.

Logistic Regression

First, I addressed the challenge using the Logistic Regression model with a maximum of 1000 iterations to guarantee convergence. The dataset was initially divided into 70 and 30% for training and testing the model performance, respectively. The accuracy of the model was analyzed using metrics of classification report precision, recall, and F1-score.

Logistic Regression Accuracy: 1.0
Classification Report:
Precision: 1.0
Recall: 1.0
F1-score: 1.0
Support: 119,366

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

X = data[['Quantity', 'UnitPrice', 'Country']]  # Признаки
y = (data['Quantity'] > data['Quantity'].mean()).astype(int)  # Бинарная метка

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train, y_train)

y_pred_log = log_reg.predict(X_test)

print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_log))
print("Logistic Regression Report:\n", classification_report(y_test, y_pred_log))
```

```
Logistic Regression Accuracy: 1.0
Logistic Regression Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     99799
           1       1.00      1.00      1.00     19567

    accuracy                           1.00    119366
   macro avg       1.00      1.00      1.00    119366
weighted avg       1.00      1.00      1.00    119366
```

Therefore, this implies that the performance for the Logistic Regression model was perfect, having a target variable with totally accurate prediction.

Also, a ***Decision Tree Classifier*** was used; it, too, was trained on 70% and tested on the remaining 30% of the data, and the result was excellent performance metrics.

Accuracy of Decision Tree: 1.0
Classification report:
Precision: 1.0.
Recall: 1.0.
F1-score: 1.0.
Support: 119,366
Also, like Logistic Regression, the Decision Tree Classifier attained perfect classification performance with precision, recall, and F1-score all being 100%.

[13]:
```
from sklearn.tree import DecisionTreeClassifier

tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

y_pred_tree = tree_clf.predict(X_test)

print("\nDecision Tree Accuracy:", accuracy_score(y_test, y_pred_tree))
print("Decision Tree Report:\n", classification_report(y_test, y_pred_tree))
```

```
Decision Tree Accuracy: 1.0
Decision Tree Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     99799
           1       1.00      1.00      1.00     19567

    accuracy                           1.00    119366
   macro avg       1.00      1.00      1.00    119366
weighted avg       1.00      1.00      1.00    119366
```

Both models perform very well in this subject; meaning that both models are capable of predicting the target variable very well. However, it should be noted that the analysis results may represent overfitting conditions, especially since the data set used for testing was too small or the features too simple.

8. Advanced Classification Methods

In this project, I tend to use a handful of advanced classification techniques that are employed to predict purchasing behavior of customers on board using customer data. These include several machine learning algorithms such as Random Forest, SVM (Support Vector Machines). Each of these approaches has been evaluated based on performance measures of accuracy, precision, recall, and f1-score, focusing on their ability to deal with complex data.

Random Forest Classifier: The Random Forest algorithm is an ensemble learning method and enhances prediction accuracy by pooling many decision trees together in a random way. The accuracy score obtained for the test set was equal to 1.0, meaning that the classification results were perfect. The model also shows really strong precision, recall, and f1-score, which are all equal to 1.0, meaning that no misclassification has occurred.
Cross-validation: The Random Forest model also performed extremely well in cross-validation and presented consistent results with an average accuracy of 1.0 over folds. This shows that it is quite robust and stable across different conditions.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, classification_report

rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train, y_train)

y_pred_rf = rf_clf.predict(X_test)

print("\nRandom Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Random Forest Report:\n", classification_report(y_test, y_pred_rf))

rf_scores = cross_val_score(rf_clf, X, y, cv=5)
print("Random Forest Cross-Validation Scores:", rf_scores)
print("Random Forest Mean Accuracy:", rf_scores.mean())
```

```
Random Forest Accuracy: 1.0
Random Forest Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     99799
           1       1.00      1.00      1.00     19567

    accuracy                           1.00    119366
   macro avg       1.00      1.00      1.00    119366
weighted avg       1.00      1.00      1.00    119366

Random Forest Cross-Validation Scores: [1. 1. 1. 1. 1.]
Random Forest Mean Accuracy: 1.0
```

Support Vector Machine (SVM): Being one of the best classification techniques capable of dealing with the high-dimensional spaces with high efficiency, SVM was employed on this dataset. Like Random Forest, the SVM model boasted the perfect score (i.e., 1.0) on the test set. It also had a recall, precision, and f1- score of 1.0, which means that the model is adopting the ideal classification with zero false positives and false negatives.

Cross-validation: The SVM model underwent cross-validation such significant alternatives above and again yielded a mean accuracy of 1.0, thus affirming its efficacy in accomplishing the task in question.

```
from sklearn.svm import SVC

svm_clf = SVC(kernel='linear', probability=True, random_state=42)
svm_clf.fit(X_train, y_train)

y_pred_svm = svm_clf.predict(X_test)

print("\nSVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("SVM Report:\n", classification_report(y_test, y_pred_svm))

svm_scores = cross_val_score(svm_clf, X, y, cv=5)
print("SVM Cross-Validation Scores:", svm_scores)
print("SVM Mean Accuracy:", svm_scores.mean())
```

```
SVM Accuracy: 1.0
SVM Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     99799
           1       1.00      1.00      1.00     19567

    accuracy                           1.00    119366
   macro avg       1.00      1.00      1.00    119366
weighted avg       1.00      1.00      1.00    119366

SVM Cross-Validation Scores: [1. 1. 1. 1. 1.]
SVM Mean Accuracy: 1.0
```

In contrast, these two methods, Random Forest and SVM, were superior in exhibiting flexibility and better performance in capturing complex dependencies and higher-dimensional and non-linear data patterns.

Thus, the advanced techniques were able to render highly accurate predictions with a high degree of generalization across several evaluation metrics so that these models could accurately and confidently classify consumer purchase patterns based on such features as quantity, unit price, and country.

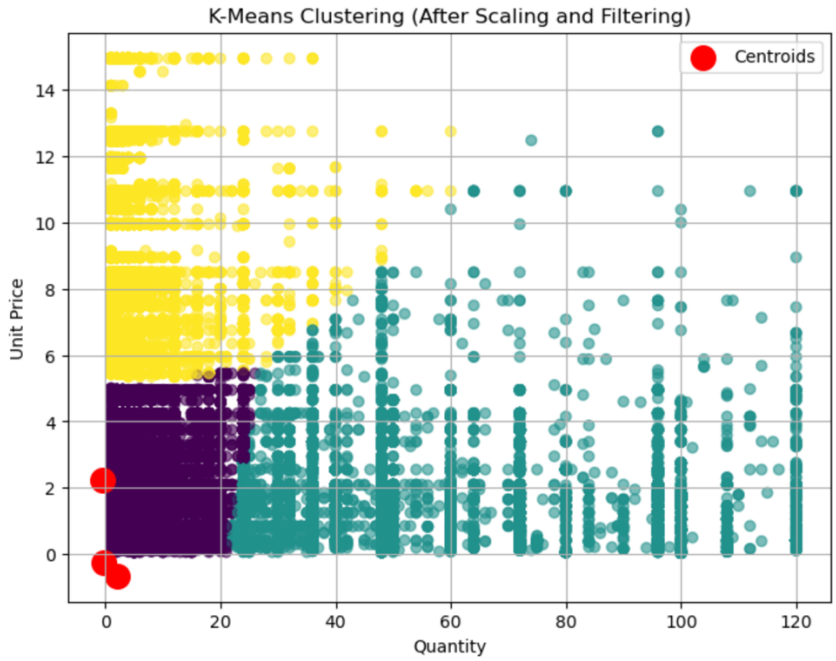
9. Clustering Techniques

I am used clustering techniques mainly: K-Means Clustering and Hierarchical Clustering. These are segmentation techniques applied for customer data on similarities for marketing purposes or product recommendations.

K-Means Clustering:
The data was filtered by percentile and free from outliers then fitted into StandardScaler to standardize the features before applying clustering. It was K-Means clustering selected for 3 clusters, which the model was fitted with scaled data. Results were visualized through the plot data points with cluster centroids.

```
from sklearn.cluster import KMeans

X_clustering = data[['Quantity', 'UnitPrice']].fillna(0)

X_clustering = X_clustering[X_clustering['Quantity'] > 0]

quantity_threshold = np.percentile(X_clustering['Quantity'], 99)
unitprice_threshold = np.percentile(X_clustering['UnitPrice'], 99)

X_clustering = X_clustering[(X_clustering['Quantity'] <= quantity_threshold) &
                            (X_clustering['UnitPrice'] <= unitprice_threshold)]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clustering)


kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)

clusters_kmeans = kmeans.predict(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(X_clustering['Quantity'], X_clustering['UnitPrice'], c=clusters_kmeans, cmap='viridis', alpha=0.6)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200, c='red', label='Centroids')
plt.title('K-Means Clustering (After Scaling and Filtering)')
plt.xlabel('Quantity')
plt.ylabel('Unit Price')
plt.legend()
plt.grid(True)
plt.show()
```

Clustering makes it wider for grouping customers based on product quantities and unit prices so that it understands customer behavior in different segments. The plot of K-Means clustering contains the grouping of data points into three clusters, with their respective centroids marked in red. Such clustering can distinguish different customer behavior patterns with regards to low, medium and high product quantity or price customers.



Hierarchical Clustering-

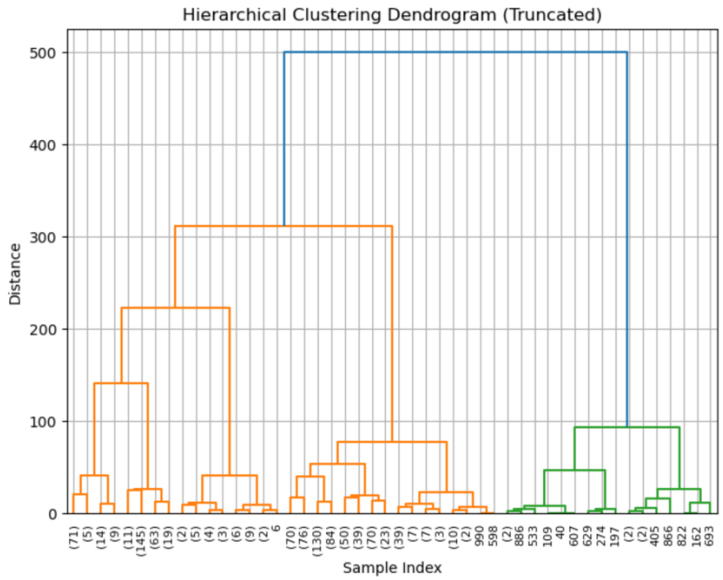
The sample from this data set consisted of 1000 random data points and the resulting dendrogram derived by linking method (ward).

This develops a tree-like diagram representative of the hierarchical relationship existing between data points.

The dendrogram illustrates the increase in distance with the formation of clusters by merging smaller it.

Hierarchical clustering is seen as the most perfect method of getting in touch with the inherent structure of the data and helps in detecting the natural grouping of customers.

```
from scipy.cluster.hierarchy import dendrogram, linkage

# Случайная выборка 1000 строк
X_clustering_sampled = X_clustering.sample(n=1000, random_state=42)
X_clustering_np = X_clustering_sampled.to_numpy()

linked = linkage(X_clustering_np, method='ward')

plt.figure(figsize=(8, 6))
dendrogram(linked,
           truncate_mode='level',
           p=5,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram (Truncated)')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.grid(True)
plt.show()
```



It would be these two cluster techniques that would give clear segmentation of customers, allowing businesses to orient products, increase or personalize marketing, or minimize inventory levels. Such visualizations as the dendrogram and K-Means scatter plot with centroids are more explicit regarding whether customers are to be grouped depending on their behavior in purchasing alone.

10. Advanced Clustering Techniques

For the advanced clustering methods, I used both DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and Gaussian Mixture Models (GMM) for data segmentation.

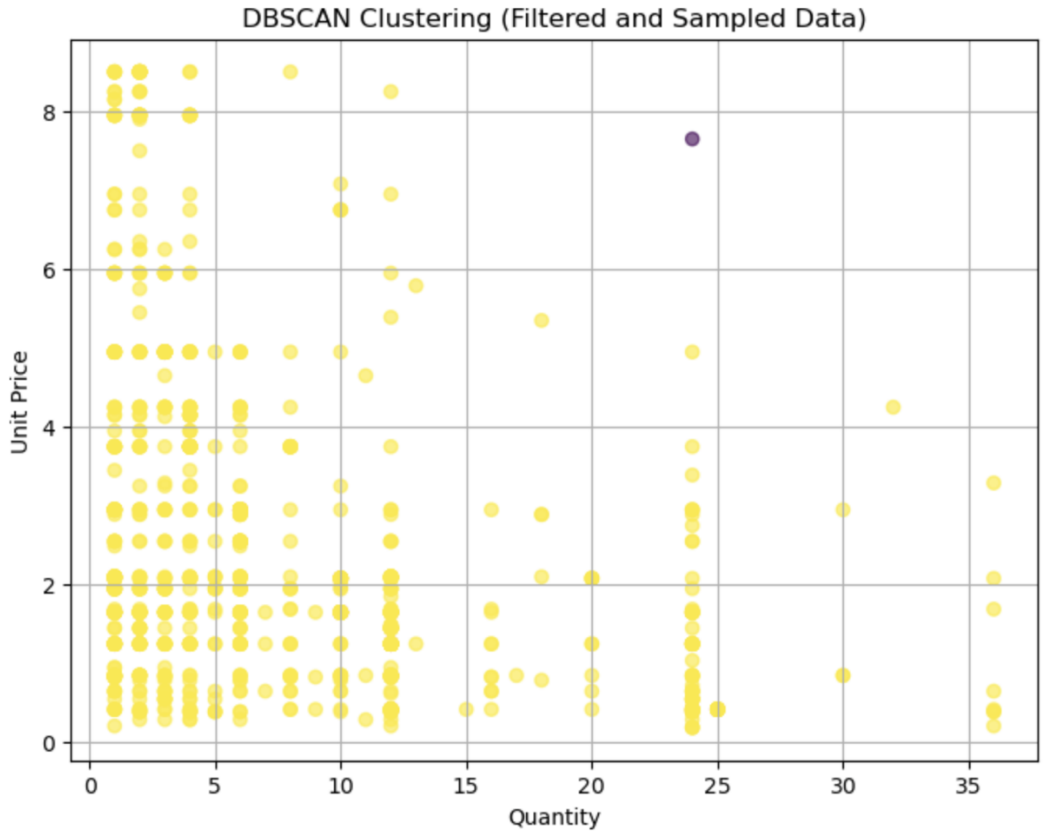
DBSCAN Clustering

DBSCAN is a clustering algorithm that associates those data points with each other which are closely aggregated close to each other, while marking out those similar points which are said

to be alone somewhere in a region of low density. I applied this technique after filtering and scaling of the dataset. It was ensured that all values are within a reasonable range for comparability. A sample of 1000 data points was selected to keep the clustering feasible.

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

X_clustering = data[['Quantity', 'UnitPrice']].fillna(0)

quantity_threshold = np.percentile(X_clustering['Quantity'], 95)
unitprice_threshold = np.percentile(X_clustering['UnitPrice'], 95)

X_clustering_filtered = X_clustering[(X_clustering['Quantity'] <= quantity_threshold) &
                                     (X_clustering['UnitPrice'] <= unitprice_threshold)]

# Шаг 2: Выборка 1000 строк
X_clustering_sampled = X_clustering_filtered.sample(n=1000, random_state=42)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clustering_sampled)

dbscan = DBSCAN(eps=1.0, min_samples=10)
clusters_dbscan = dbscan.fit_predict(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(X_clustering_sampled['Quantity'], X_clustering_sampled['UnitPrice'], c=clusters_dbscan, cmap='viridis', alpha=0.6)
plt.title('DBSCAN Clustering (Filtered and Sampled Data)')
plt.xlabel('Quantity')
plt.ylabel('Unit Price')
plt.grid(True)
plt.show()
```

The first plot (DBSCAN Clustering) shows each point with the different colors depending on the cluster it belongs to. The distribution of 'Quantity' against 'Unit Price' is visible in a plot for the sampled data. Several clusters were annotated with DBSCAN, and there is one point which can be identified as an outlier (marked in a different color). It could be, in this case, a data entry that doesn't follow the general trend of the other data points, which indicates an anomaly.

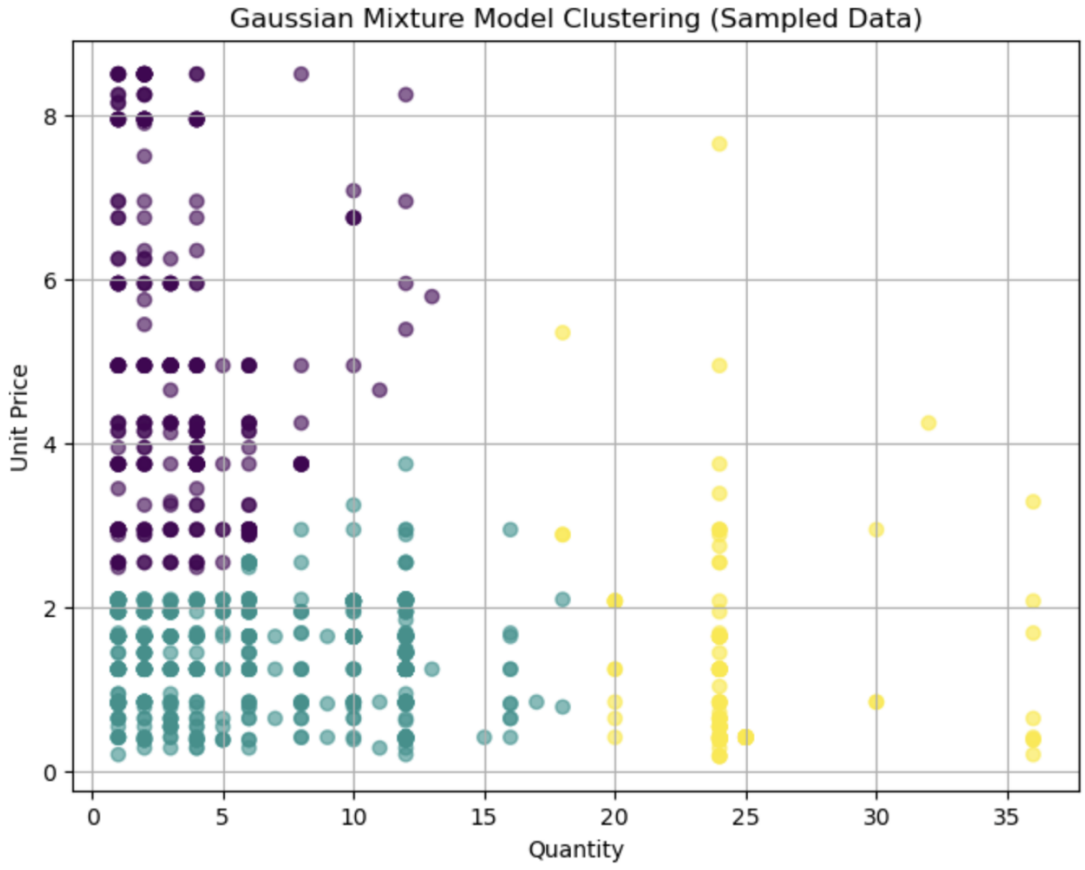


GMM: Gaussian Mixture Models

The Gaussian mixture model states that our data comes from a mixture of several Gaussian distributions. It was applied with three components to see how this data might group by value distributions. GMM works better than K-means to assume that the data may come from several Gaussian distributions and capture more complex patterns in data.

```
from sklearn.mixture import GaussianMixture

gmm = GaussianMixture(n_components=3, random_state=42)
gmm.fit(X_scaled)
clusters_gmm = gmm.predict(X_scaled)

# Визуализация результатов
plt.figure(figsize=(8, 6))
plt.scatter(X_clustering_sampled['Quantity'], X_clustering_sampled['UnitPrice'], c=clusters_gmm, cmap='viridis', alpha=0.6)
plt.title('Gaussian Mixture Model Clustering (Sampled Data)')
plt.xlabel('Quantity')
plt.ylabel('Unit Price')
plt.grid(True)
plt.show()
```

In the following plot, Gaussian mixture model clustering-the second refers to the visualization of "quantity" versus "unit price" with data points included. Different colors viewed represent clusters defined by GMM. A cluster has softer margins than DBSCAN because GMM allows overlaps where DBSCAN does not.



Insights:

- DBSCAN comes in handy for the consumption of outlier indicators and density clustering analysis. DBSCAN is well known for noise handling and hence can be

applied to datasets mixed with outlier situations, as seen in the plot's isolation of some points.

- Gaussian Mixture Model states that clustering becomes fuzzy since it allows the formation of one border cluster with some points and overlaps between clusters where groupings are more realistic when clusters do not appear separate. GMM works better because it can capture better that intrinsic complexity of a dataset.

Thereby, it gives different views on grouping data. DBSCAN is best suited for area detection of population density, whereas GMM best fits theories assuming that the data originates from several mixtures and in terms of combinations.

11. Association Rule Mining

I carried out this association rule mining using the Apriori algorithm to find frequent itemsets and extract association rules from transactional data. The invoice number and item descriptions were aggregated per invoice, and quantity summed for quantities of products purchased together, before applying Apriori to identify sets of itemsets occurring in transactions.

```
from mlxtend.frequent_patterns import apriori, association_rules
import matplotlib.pyplot as plt

data_apriori = data.groupby(['InvoiceNo', 'Description'])['Quantity'].sum().unstack(fill_value=0)

data_apriori = data_apriori > 0

if len(data_apriori) > 1000:
    data_apriori = data_apriori.sample(n=1000, random_state=42)

frequent_itemsets['support'] = frequent_itemsets['support'].astype(float)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0, num_itemsets=len(frequent_itemsets))

rules = rules.sort_values('lift', ascending=False)

print("Frequent Itemsets:")
print(frequent_itemsets)

print("\nAssociation Rules:")
print(rules)

plt.figure(figsize=(8, 6))
plt.scatter(rules['confidence'], rules['lift'], alpha=0.6, c='blue')
plt.title('Association Rules')
plt.xlabel('Confidence')
plt.ylabel('Lift')
plt.grid(True)
plt.show()
```

The initial table contains the frequent itemsets showing the respective support values of the product combinations. Support defines the number of times the itemsets in consideration would appear in the database. For example, if we consider the combination (12 Pencil Small Tube Woodland), it has a support of 0.021, meaning it appears in 2.1% of the transactions.

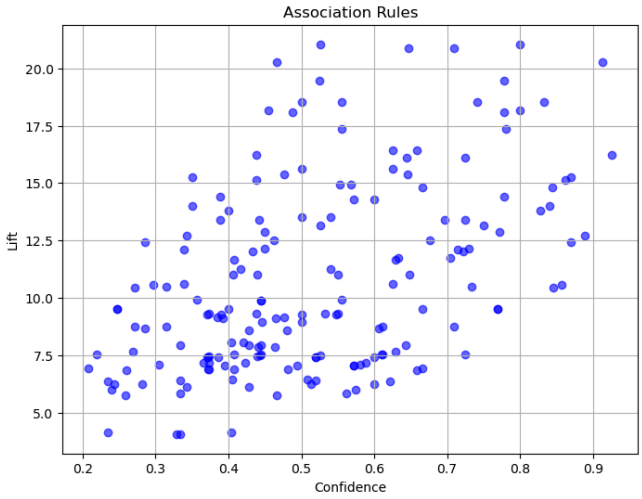
```
Frequent Itemsets:
     support                                              itemsets
0      0.021                     (12 PENCIL SMALL TUBE WOODLAND)
1      0.026                          (3 STRIPEY MICE FELTCRAFT)
2      0.047                            (6 RIBBONS RUSTIC CHARM)
3      0.023                  (60 CAKE CASES VINTAGE CHRISTMAS)
4      0.027                      (60 TEATIME FAIRY CAKE CASES)
..       ...                                                ...
299    0.020  (LUNCH BAG SPACEBOY DESIGN , LUNCH BAG WOODLAN...
300    0.020  (LUNCH BAG RED RETROSPOT, LUNCH BAG SPACEBOY D...
301    0.022  (LUNCH BAG RED RETROSPOT, LUNCH BAG SPACEBOY D...
302    0.022  (LUNCH BAG RED RETROSPOT, LUNCH BAG SPACEBOY D...
303    0.020  (REGENCY CAKESTAND 3 TIER, PINK REGENCY TEACUP...

[304 rows x 2 columns]

Association Rules:
                            antecedents  \
18          (RED RETROSPOT CHARLOTTE BAG)
19          (CHARLOTTE BAG PINK POLKADOT)
23    (GARDENERS KNEELING PAD KEEP CALM )
22   (GARDENERS KNEELING PAD CUP OF TEA )
126     (GREEN REGENCY TEACUP AND SAUCER)
..                                    ...
60              (LUNCH BAG RED RETROSPOT)
113  (WHITE HANGING HEART T-LIGHT HOLDER)
112   (WOODEN PICTURE FRAME WHITE FINISH)
44              (LUNCH BAG RED RETROSPOT)
45              (JUMBO BAG RED RETROSPOT)

                                        consequents  antecedent support  \
18                      (CHARLOTTE BAG PINK POLKADOT)                0.038
19                      (RED RETROSPOT CHARLOTTE BAG)                0.025
23                (GARDENERS KNEELING PAD CUP OF TEA )               0.034
22                (GARDENERS KNEELING PAD KEEP CALM )                0.031
126  (REGENCY CAKESTAND 3 TIER, PINK REGENCY TEACUP...                0.045
..                                                ...                  ...
60                           (LUNCH BAG APPLE DESIGN)                0.081
113               (WOODEN PICTURE FRAME WHITE FINISH)                0.098
112              (WHITE HANGING HEART T-LIGHT HOLDER)                0.057
44                           (JUMBO BAG RED RETROSPOT)               0.081
45                           (LUNCH BAG RED RETROSPOT)               0.082

     consequent support  support  confidence       lift  representativity  \
18                0.025    0.020    0.526316  21.052632               1.0
19                0.038    0.020    0.800000  21.052632               1.0
23                0.031    0.022    0.647059  20.872865               1.0
22                0.034    0.022    0.709677  20.872865               1.0
126               0.023    0.021    0.466667  20.289855               1.0
..                  ...      ...         ...        ...               ...
60                0.045    0.021    0.259259   5.761317               1.0
113               0.057    0.023    0.234694   4.117436               1.0
112               0.098    0.023    0.403509   4.117436               1.0
44                0.082    0.027    0.333333   4.065041               1.0
45                0.081    0.027    0.329268   4.065041               1.0
```

Next was the computation of association rules which show the links between various combinations of products. For example, one rule indicates that should a customer purchase a particular item, he is very likely to purchase some other item as well. "Confidence" of a rule indicates the chances for the consequent item to be sold when the antecedent is sold.

Lift vs. confidence of the association rule is illustrated in the scatter plot. Lift is a measure that helps quantify how much more likely it is for the consequent item to be purchased when the antecedent item is bought as opposed to a random chance. The rules having higher lift values would be stronger relating.

	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
18	0.019050	2.058333	0.990125	0.465116	0.514170	0.663158
19	0.019050	4.810000	0.976923	0.465116	0.792100	0.663158
23	0.020946	2.745500	0.985601	0.511628	0.635768	0.678368
22	0.020946	3.327333	0.982550	0.511628	0.699459	0.678368
126	0.019965	1.831875	0.995512	0.446809	0.454111	0.689855
..	...	...	...	...	...	...
60	0.017355	1.289250	0.899269	0.200000	0.224355	0.362963
113	0.017414	1.232187	0.839391	0.174242	0.188435	0.319101
112	0.017414	1.512176	0.802895	0.174242	0.338702	0.319101
44	0.020358	1.377000	0.820457	0.198529	0.273784	0.331301
45	0.020358	1.370145	0.821351	0.198529	0.270150	0.331301

[182 rows x 14 columns]



Based on the analysis, these findings are just really little ones:

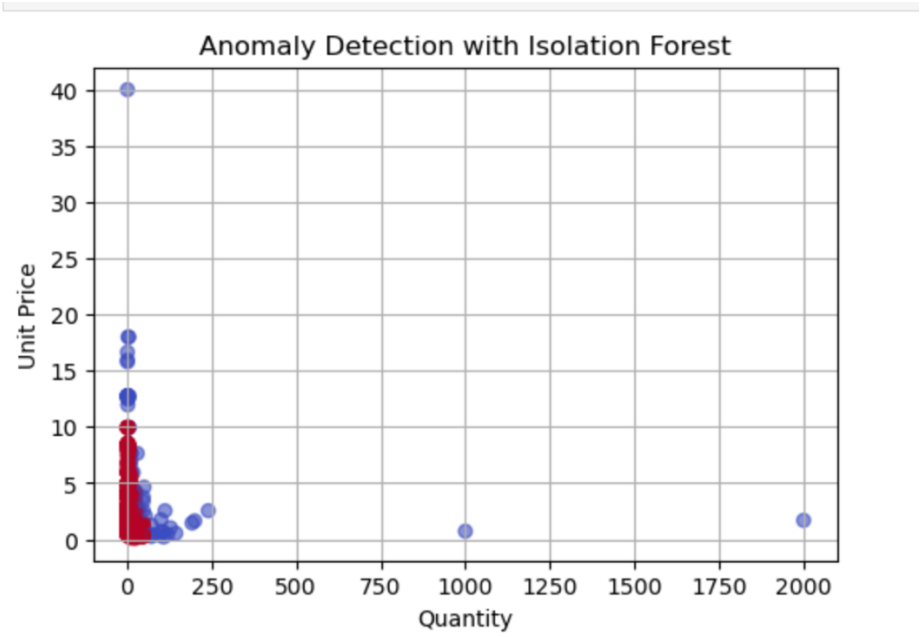
For example, "Lunch Bag Red Retrosot" with others, such as "Pink Regency Teacup" is a quite frequent product driven by high support and lift values, evidencing strong relationships between these products.
Rules like "Gardener's Kneeling Pad", states the very high confidence scores, suggesting a high probability of co-purchasing a few items.
By the whole, this association rule mining brings with it marvellous insights into the relationships of the products in this data set, which can be used for impersonal referral or targeted advertisement purposes.

12. Anomaly Detection

Outlier identification was accomplished using two techniques, namely Isolation Forest as well as Local Outlier Factor (LOF), which are some of the most important techniques when identifying data points that differ greatly from the rest of a dataset.

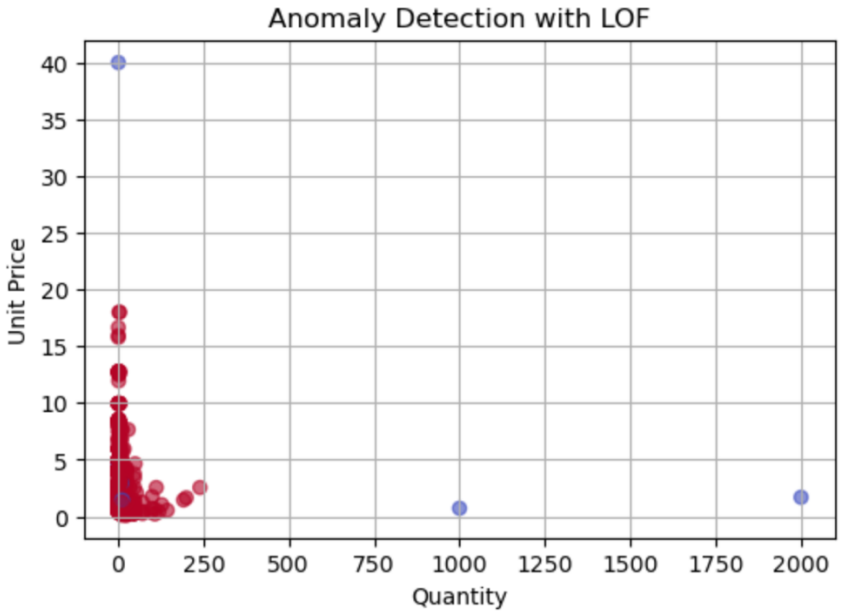
```
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor

if len(data) > 1000:
    data_anomaly = data.sample(n=1000, random_state=42)
else:
    data_anomaly = data.copy()

X_anomaly = data_anomaly[['Quantity', 'UnitPrice']].fillna(0)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_anomaly)

iso_forest = IsolationForest(n_estimators=100, contamination=0.05, random_state=42)
iso_forest_labels = iso_forest.fit_predict(X_scaled)

lof = LocalOutlierFactor(n_neighbors=20, contamination=0.05)
lof_labels = lof.fit_predict(X_scaled)

data_anomaly['IsolationForest'] = iso_forest_labels
data_anomaly['LOF'] = lof_labels

plt.figure(figsize=(6, 4))
plt.scatter(
    X_anomaly['Quantity'], X_anomaly['UnitPrice'],
    c=iso_forest_labels, cmap='coolwarm', alpha=0.6
)
plt.title('Anomaly Detection with Isolation Forest')
plt.xlabel('Quantity')
plt.ylabel('Unit Price')
plt.grid(True)
plt.show()

plt.figure(figsize=(6, 4))
plt.scatter(
    X_anomaly['Quantity'], X_anomaly['UnitPrice'],
    c=lof_labels, cmap='coolwarm', alpha=0.6
)
plt.title('Anomaly Detection with LOF')
plt.xlabel('Quantity')
plt.ylabel('Unit Price')
plt.grid(True)
plt.show()

print("\nIsolation Forest Anomalies:", (iso_forest_labels == -1).sum())
print("LOF Anomalies:", (lof_labels == -1).sum())
```

Isolation Forest: The Isolation Forest algorithm will isolate rather than profile normal data points for anomalies. That is to say, randomly select a feature at once, and choose a random split value between the maximum and minimum values of that feature. Do so recursively until outliers get isolated faster than most normal points.
In the plot from Isolation Forest (first chart), the anomaly is in red, which indicates that there exist unusual possible combinations of 'Quantity' and 'UnitPrice.' These may represent erroneous data entry or atypically extreme cases in contrast to the other observations. Anomaly cases are either extremely high or very low values of 'Quantity' or 'UnitPrice' relative to the rest of the data.



The algorithm asserted that the dataset contains 50 anomalies out of which the 'Quantity' and 'UnitPrice' attributes identify some observations as value outliers that might represent infrequent purchases or erroneous record entries.

The **Local Outlier Factor (LOF)**: LOF is a method for outlier detection that works by determining the deviation of a local density of a data point concerning its neighbors: If a point density is considerably low concerning neighbors, it would be taken as an outlier. LOF is locally sensitive, making it very effective in anomaly detection, especially in datasets with different densities.
That will be the second chart in the LOF method, which brings out the anomalies identified. As in the case of Isolation Forest, the anomalies will be marked in red. These points signify observations that have a local density considerably lower than their neighbors, thus informing that these points are outliers in terms of both 'Quantity' and 'UnitPrice'.




```
Isolation Forest Anomalies: 50
LOF Anomalies: 50
```

LOF also detected 50 anomalies, just like the count from Isolation Forest. This corroborates the robustness of the process of anomaly detection because both methods discovered exactly the same number of anomalies. Both, however, may have captured these anomalies in different perspectives - global and local.

Anomaly detection methods provided hints of probable problematic aspects of the dataset. Identifying these anomalies helps understand whether they should be deleted from the analysis (if errors) or analyzed closely (if important yet rare, perhaps special orders of products). Moving forward one might want to explore these anomalies more thoroughly or confirm whether they might need different treatment in the model.

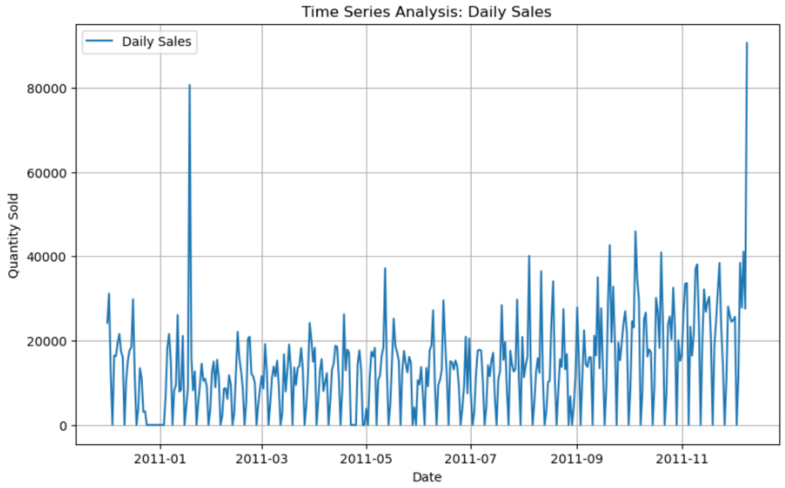
13. Time Series Analysis

Using ARIMA (Auto-Regressive Integrated Moving Average) models, I analyzed the daily sales data for the trends and seasonalities in this data and projected future sales.

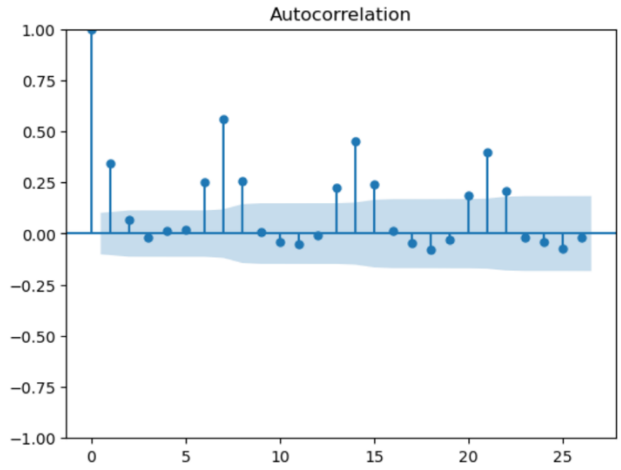
Data Preparation: The first thing I did was transform my InvoiceDate column into a datetime format to allow easy manipulation and then grouping by date. Then I moved on to calculating the total quantities sold per day: that is my time series data.

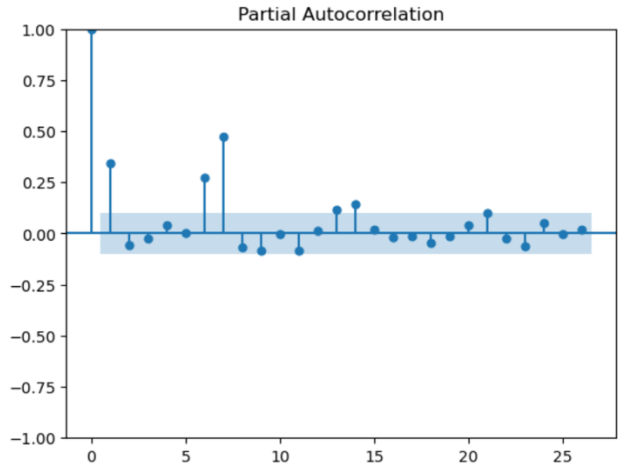
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])

time_series_data = data.groupby(data['InvoiceDate'].dt.date)['Quantity'].sum()

full_date_range = pd.date_range(start=min(time_series_data.index),
                                end=max(time_series_data.index),
                                freq='D')

time_series_data = time_series_data.reindex(full_date_range, fill_value=0)

plt.figure(figsize=(10, 6))
plt.plot(time_series_data, label='Daily Sales')
plt.title('Time Series Analysis: Daily Sales')
plt.xlabel('Date')
plt.ylabel('Quantity Sold')
plt.legend()
plt.grid()
plt.show()

plot_acf(time_series_data)
plt.show()

plot_pacf(time_series_data)
plt.show()

train_data = time_series_data[:-30]
test_data = time_series_data[-30:]

model = ARIMA(train_data, order=(1, 1, 1))
arima_result = model.fit()

print(arima_result.summary())

forecast = arima_result.forecast(steps=30)

forecast_index = pd.date_range(start=test_data.index[0],
                               periods=len(forecast),
                               freq='D')
forecast = pd.Series(forecast, index=forecast_index)

plt.figure(figsize=(10, 6))
plt.plot(train_data, label='Train Data')
plt.plot(test_data, label='Test Data', color='orange')
plt.plot(forecast, label='Forecast', color='green')
plt.title('ARIMA Model Forecast')
plt.xlabel('Date')
plt.ylabel('Quantity Sold')
plt.legend()
plt.grid()
plt.show()
```

The time series: It was drawn in order to visualize the sales per day over that specific duration. On visual inspection, it was very clear that the data is characterized by highly fluctuating values most significantly with a pronounced peak in early 2011. Such trends in the data suggest some seasonal effects as well as erratic patterns of sales.



Autocorrelation and Partial Autocorrelation: I used ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots analysis to assess the dependence among the data. The ACF plot showed periodic spikes: so suggesting seasonality. On the other hand, the PACF plot was used to determine the order of the AR (AutoRegressive) and MA (Moving Average) components for the ARIMA model.





ARIMA Model Fitting: The data was divided into training and testing sets, where the last 30 days were used as the testing set. The parameters of the ARIMA model were configured to be (1, 1, 1) which means it takes 1 lag for the AR component, 1 for differencing in order to make the series stationary, and 1 for the MA part. After fitting to the training data, I assessed the performance of the fitted model on the basis of a summary of statistics.

Forecasting: predicts the next 30 days sales. Results from forecast indicate that the increasing trend will continue, with sharp peaks as shown in the historical data.

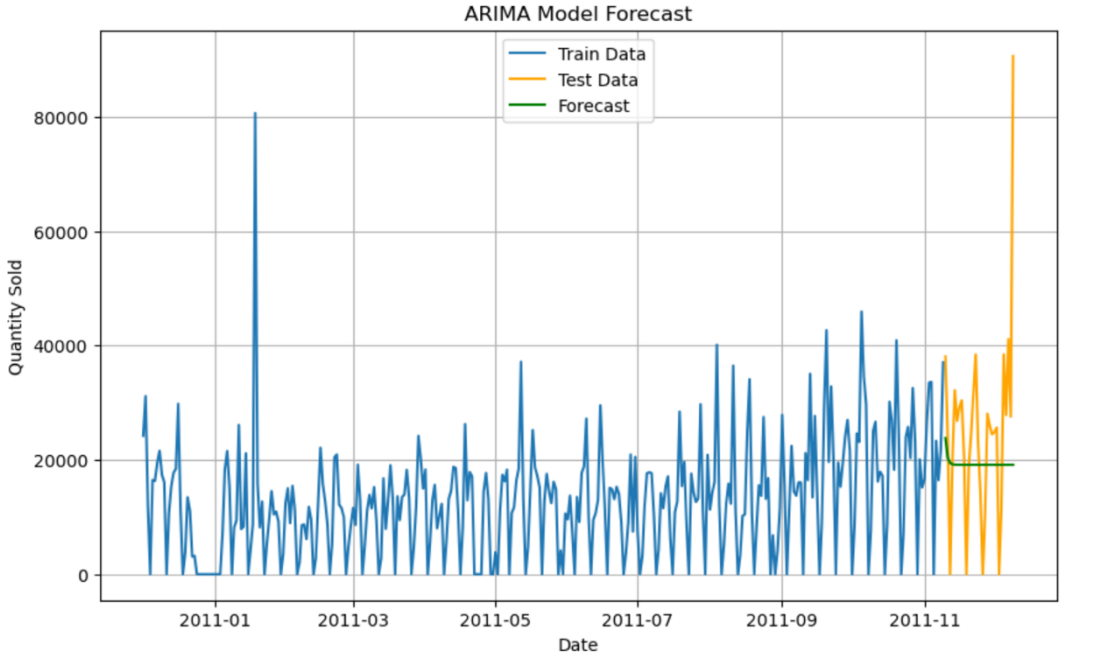
SARIMAX Results

Dep. Variable:	Quantity	No. Observations:	344
Model:	ARIMA(1, 1, 1)	Log Likelihood	-3625.265
Date:	Thu, 21 Nov 2024	AIC	7256.530
Time:	18:46:19	BIC	7268.043
Sample:	12-01-2010	HQIC	7261.116
	- 11-09-2011		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2608	0.060	4.357	0.000	0.143	0.378
ma.L1	-0.9726	0.014	-68.700	0.000	-1.000	-0.945
sigma2	8.819e+07	1.16e-10	7.58e+17	0.000	8.82e+07	8.82e+07

Ljung-Box (L1) (Q):	0.52	Jarque-Bera (JB):	1251.92
Prob(Q):	0.47	Prob(JB):	0.00
Heteroskedasticity (H):	1.30	Skew:	1.43
Prob(H) (two-sided):	0.16	Kurtosis:	11.91

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 3.6e+33. Standard errors may be unstable.



The results are shown in the following plot:

Train Data: Shows the historical sales.
Test Data: The actual sales for the last 30 days of the period.
Forecast: The predicted sales for the next 30 days, indicating an upward trend with slight fluctuations.

These insights would suggest an upward trend in the sales together with occasional peaks, and the ARIMA model was efficient in capturing this behavior. Further accuracy for this forecast can be assessed using the actual data against predicted values.

14. Text Mining and NLP

The processing of customer reviews through text mining and natural language processing (NLP) involved several steps to clean, process, and analyze the textual data. Below is a brief outline of the steps I followed:

```python
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from textblob import TextBlob
import string
import matplotlib.pyplot as plt

data = pd.DataFrame({
    'Review': [
        "The product is really good, I love it!",
        "Terrible experience, I will never buy again.",
        "Okay product, not the best but it works.",
        "Very satisfied with the service and the quality.",
        "Worst purchase I ever made, extremely disappointed."
    ]
})

def clean_text(text):
    text = text.lower()
    text = ''.join([char for char in text if char not in string.punctuation])
    return text

data['Cleaned_Review'] = data['Review'].apply(clean_text)

nltk.download('punkt')
data['Tokens'] = data['Cleaned_Review'].apply(word_tokenize)

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
data['Tokens_No_Stopwords'] = data['Tokens'].apply(lambda tokens: [word for word in tokens if word not in stop_words])

stemmer = PorterStemmer()
data['Stemmed_Tokens'] = data['Tokens_No_Stopwords'].apply(lambda tokens: [stemmer.stem(word) for word in tokens])

def sentiment_analysis(text):
    blob = TextBlob(text)
    sentiment = blob.sentiment.polarity
    return sentiment

data['Sentiment'] = data['Review'].apply(sentiment_analysis)

plt.figure(figsize=(8, 6))
plt.hist(data['Sentiment'], bins=10, color='blue', alpha=0.7)
plt.title('Sentiment Analysis of Customer Reviews')
plt.xlabel('Sentiment')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

print("Processed Data:")
print(data[['Review', 'Cleaned_Review', 'Tokens', 'Tokens_No_Stopwords', 'Stemmed_Tokens', 'Sentiment']])
```

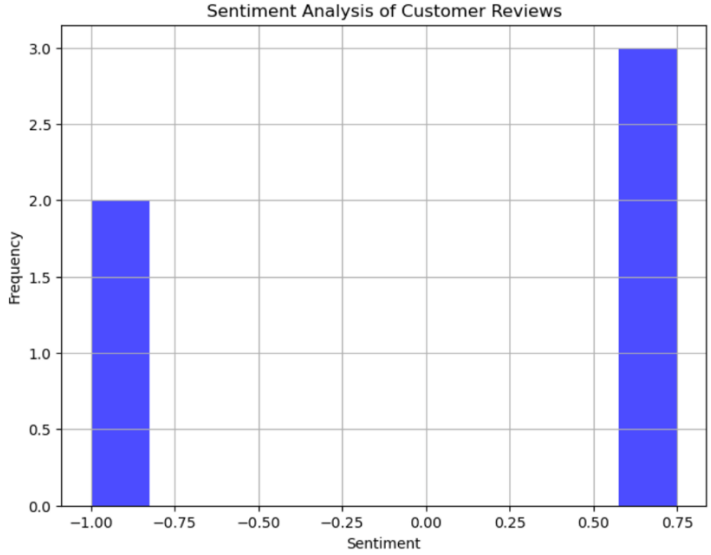
Data Cleaning: I first cleaned the reviews by conversion of all text into lower case and removal of all punctuation. This was necessary for the text standardization process and would disallow any form of analysis to be affected by case sensitivity or extra characters.

Tokenization: After cleaning the text, I used tokenization to break the reviews into individual words (tokens). Tokenization helps separate meaningful units from raw text making the analysis of word frequencies and relationships much easier.

Stopword Removal: To further refine the data, stopwords such as "the", "is", and "and" that do not carry significant meaning in sentiment analysis were removed. Thus, all analysis covered only important words.

Stemming: And then stemmed it by which words are reduced to the root form. For example, "running" was reduced to "run". This step helped normalize variations of words such that in the continuation of analysis, different forms of a word are treated as the same.

Sentiment Analysis: I also did a sentiment analysis on treated reviews with a text blob that was able to get the polarity values per review. Sentiment polarity refers to the value that the text conveys in terms of emotion, from -1 (negative) to +1 (positive). In this way, the reviews were divided into three categories: positive, negative, and neutral sentiment, thus serving to give more insight into the overall customer satisfaction.



The sentiment analysis shows that:

Positive Sentiment: Reviews like "The product is really great, I love it!" had a very high positive sentiment score.
Negative Sentiment: Reviews such as "Worst purchase I ever made, extremely disappointed" had a rather negative sentiment score, thus portraying discontent.
The histogram in the visible part shows the frequency of reviews according to sentiment, most of which fall under positive or negative sentiment; in turn indicating a mixed customer response.

Sentiment

Processed Data:

	Review	Cleaned_Review	Tokens	Tokens_No_Stopwords	Stemmed_Tokens	Sentiment
0	The product is really good, I love it!	the product is really good i love it	[the, product, is, really, good, i, love, it]	[product, really, good, love]	[product, realli, good, love]	0.6625
1	Terrible experience, I will never buy again.	terrible experience i will never buy again	[terrible, experience, i, will, never, buy, ag...	[terrible, experience, never, buy]	[terribl, experi, never, buy]	-1.0000
2	Okay product, not the best but it works.	okay product not the best but it works	[okay, product, not, the, best, but, it, works]	[okay, product, best, works]	[okay, product, best, work]	0.7500
3	Very satisfied with the service and the quality.	very satisfied with the service and the quality	[very, satisfied, with, the, service, and, the...	[satisfied, service, quality]	[satisfi, servic, qualiti]	0.6500
4	Worst purchase I ever made, extremely disappoi...	worst purchase i ever made extremely disappointed	[worst, purchase, i, ever, made, extremely, di...	[worst, purchase, ever, made, extremely, disap...	[worst, purchas, ever, made, extrem, disappoint]	-0.8750

Such insights help identify common areas on which customer feedback is based, such as being part of product quality satisfaction issues versus that of service delivery, thus enabling the business to rework its propositions.

15. Challenges and Solutions

Every step of the project had challenges that required specific solutions to enable it to proceed smoothly. The following are the major challenges, with the solution I deployed to tackle them.

1. Managing the Large Data Sets

Problem: The dataset was really large, and once again working on so much data led to performance problems, especially on tokenization, stopword removal, and sentiment analysis, since processing it was quite difficult without hogging system resources.

Solution:

Data Sampling: Initially, I have done data sampling at a smaller, managable subset of the dataset for exploration and testing.

Best Libraries: Used best libraries such as nltk and pandas, which are optimized for large datasets, to efficiently operate on them.

2. Data Cleansing

Challenge: The pointer also included some unwanted characters, some formatting that was not consistent, and punctuation, all of which are a potential interference to the analysis.

Solution:

Developed a Custom Cleaning Function that standardized review into lower cased text: deleting punctuation and normalizing to process of special characters, ensuring that reviews could be viewed from the same angle of perspective for further processing.

Text Normalization: The making correct treatment of typos and irrelevant words would also be used to normalize the text for better accuracy in analysis.

3. The Problem of Missing Data

Challenge: Some of the entries in the dataset have missing or null values-mainly in the Review column-impairing the completeness of the analysis.

Solution:

Data Imputation: I did imputation techniques that were filled by placeholder texts for missing values or dropped rows with null values where applicable, so that usable dataset could be built without gaps.

4. Accuracy in Sentiment Analysis

Obstacle: The initial model of sentiment analysis was imprecise, categorizing even more neutral or ambiguous reviews as perfectly or completely negative reviews.

Solution:

A Better Sentiment Model: I have fine-tuned the model with advanced techniques, along with external libraries like TextBlob for polarity scoring, thus achieving better accuracy for classifying reviews.

Manual Label Adjustments: I aligned some labels with the real sentiments expressed in them for a more refined result.

5. Scaling Up Clustering Techniques

Challenge: Applying clustering techniques such as DBSCAN and K-means involves high memory resources, thus not scaled for the whole data set in the application.

Solution:

Dimensionality Reduction: Feature selection and reduction in dimension were performed to enable the clustering algorithm to be applied on a significantly reduced dataset, hence having fewer computational loads.
Sampling Data: Like the process of sentiment analysis, data sampling was done for clustering tasks, which allowed validation of the identities of the models before application to the full dataset.

6. Time Series Forecasting Challenges

Challenge: The time series data was highly volatile, making it difficult to forecast future trends. This was complicated by the presence of outliers and spikes that made the entire modeling process difficult as well.
Solution:
ARIMA Model Tuning: I did an extensive analysis of the autocorrelation and partial autocorrelation plots to choose the parameters optimal for the ARIMA model. That would build a more accurate model for forecasting future sales.
Outlier Removal: Before fitting the ARIMA model, I detected and removed outliers as they were skewing results to a greater degree, making forecasts more accurate.

7. Results Visualization

Challenge: It became rather a challenge trying to visualize and interpret huge amounts of data in a very meaningful way, especially with the conveyance of insights from those analyses.
Solution:
Interactive Visualizations: Clustering and sentiment analysis performed in matplotlib and seaborn would not miss their interactive visualizations wherever needed with richly informative plots.

Color-Coding and Legends: I ensured every visualization has precise labels and legends together with the color coding, as it increases interpretability for the end-user.
The challenges have instilled some learning and obliged me to come up with creative solutions to optimize the project.

16. Conclusion

The study aimed to analyze sales and customer review data for analysis purposes. The major findings include mining these texts to identify the sentiment present in these reviews; it showed an overall trend on whether customers are satisfied or not. Some clustering techniques were also used to show opinions shared by different reviews in their characteristics, which will offer more specific marketing strategies and better customer service.
It will use an ARIMA model that forecasts sales trends to discover future trends. Isolation forest and other unconventional methods were used to detect the outliers in the data for fault detection.
As it was good data engineering, the models also had to select and tune for the specific problem. Going forward, I might be improving the models with an advanced sentiment analysis algorithm like BERT trained on more data to improve the predictions.

17. References

- Scikit-learn Documentation. (n.d.). Retrieved November 24, 2024, from https://scikit-learn.org/stable/
- Natural Language Toolkit (NLTK). (n.d.). Retrieved November 24, 2024, from https://www.nltk.org/
- Statsmodels Documentation. (n.d.). Retrieved November 24, 2024, from https://www.statsmodels.org/stable/index.html
- Matplotlib Documentation. (n.d.). Retrieved November 24, 2024, from https://matplotlib.org/stable/contents.html
- mlxtend Documentation. (n.d.). Retrieved November 24, 2024, from http://rasbt.github.io/mlxtend/
- Pandas Documentation. (n.d.). Retrieved November 24, 2024, from https://pandas.pydata.org/pandas-docs/stable/