report_supply_chain_anomaly_detection_38_57_62

by Nallamilli Naga Venkata REDDY

Submission date: 18-Nov-2023 09:36PM (UTC+0800)
Submission ID: 2225546357
File name: Supply_chain_anomaly_detection_final_report_38_57_62.docx (2.68M)
Word count: 4589
Character count: 26272

SUPPLY CHAIN ANOMALY DETECTION

Project report Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Computer and Communication Engineering

by

NALLAMILLI NAGA VENKATA REDDY-210953334

VARIKUTI MADHURIMA-210953314

VURA SAI NIKHIL-210953204

Under the guidance of

Dr. Chethan Sharma
Associate professor-senior scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

Dr. Raghavendra s
Associate professor-senior scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Unit of MAHE, Manipal

November 2023

Table of contents:

Chapter-1

Introduction:

In today's context, the supply chain plays a significant role in moving goods and services from manufacturers to end users. However, this supply chain is vulnerable to disruptions and there are unpredictable changes in demand. Identifying anomalies or unusual patterns in the supply chain is important for companies to maintain operational quality, reduce risk, and increase overall resilience Anomaly detection is the detection of data points or patterns that deviate significantly from the expected or normal behavior of a system or data set. In supply chain management, anomaly identification can be a powerful tool to anticipate and mitigate problems before they turn into serious issues.

Current Practices:

Looking at how things are done now in managing supply chains, we see a trend towards using more data to make decisions. The techniques we're using stand out for being good at finding anomalies in different situations. This project aims to use these techniques to solve the challenge of finding odd things in complex supply chain data.

Applications Related to the Project:

Finding anomalies is like discovering hidden problems. In the data about how things are priced for shipping, we want to make sure everything is fair and clear, with no overcharging or underpricing. In the logistics data, we're looking for signs of delays, disruptions in transportation, or strange patterns in how things are stored. If we find these problems early, we can fix them and make the whole system work better.

Motivation:

The motivation for the project "Supply Chain Anomaly Identification Using Outlier Analysis" is based on the critical need for businesses to adapt to the challenges posed by the dynamic and interconnected nature of today's supply chain thus helping to build a simpler, more cost-effective, adaptive supply chain design.

Chapter-2

Literature survey

Baumgartner et al. [1]

This paper gives an approach to simulate and analyze the different anomalies in Boxed Beef supply chain routes by using sensors, simulating different scenarios of supply chain anomalies, and distinguishing between different types of anomalies. Simulation is used to recreate the different scenarios to detect the anomalies by using temperature sensors, trip durations and cooling requirements. It uses logic-based language which can potentially observe the anomalies and determine the type of anomaly. The scenarios used in this paper are fit to be tested on to get accurate data and results from the supply chain. The Algorithm can be applied to other similar types of products to remove anomalies in their supply chain by making some changes to the algorithm and the scenarios by studying the model of the supply system. The potential advancements in the algorithms to make the algorithm useful for wide range of applications by increasing its functionalities. The algorithm increases the quality of Food and reducing the wastage from the problems faced in the supply chain.

Jiménez-Carmelo et al. [2]

This paper discusses about how to tackle food frauds using unique identification labels for the food as it helps in food traceability and recording of data about products and their respective batches. Unique ID and analysis using artificial intelligence of the data from the Unique ID reveals interesting details and patterns about the product to make sure the originality if the product is not tampered and Frauds are reduced. intrinsic approach in which markers on the products which change throughout the supply chain. extrinsic approach, the use of QR codes, barcodes which are digitally scanned and retrieve data ensuring data security and privacy and reducing the Counterfeit products. The use case of this algorithm in different scenarios in supply chain can give different results, some can give excellent results, and some cannot. This paper focuses attention on analysis on spatial and temporal data which is generated in traceability system can be an early sign to identify and mark the anomalies that are suspected in supply chain.

Sahin et al. [3]

This paper aims to explain about an algorithm which is an iterative optimized algorithm. It helps in transporting the container in a distribution system. Dynamic nature of maritime sector due to changing in weather, financial conditions and risks leads to uncertainties. This paper suggests a model that can be a perfect solution to the dynamic change in maritime sector. Shortest path algorithm like Dijkstra's algorithm is used in the optimized techniques to decrease the cost, risk and to achieve high level performance. Calculating weights for optimizing risks, cost, and performance in the networks. A fuzzy analytical hierarchy procedure is used as a multicriteria decision method. To achieve this result. But due to its modular nature it can adapt to new technologies, can be improved by embedding new algorithms to work faster. Maritime sector needs an optimized model which is dynamic and can withstand unforeseen challenges. The algorithm can be applied to a more complex

dynamic fuzzy environments like incorporating real-time big data and adapting to changing conditions in maritime supply chain.

Yeboah-Ofori et al. [4]

This paper reveals an innovative approach to detecting supply chain attacks in0 CPS domain. BBN It is a graphical model which uses Bayesian probability theory to understand dependencies among a set of variables. Collect data related to the supply chain operations and then clean and preprocess the data to remove inconsistencies. Determine the relationships between variables and design by specifying nodes for each variable and edges, Conditional probability tables. Probabilistic Modelling: Provides a comprehensive view of the attack scenario. Performance can vary depending on many factors like Complexity of the model, detection speed, data quality, and resource utilization. The performance will depend on how well these challenges are managed. BBN can enhance CPS security and it is the best approach to detect cyber-attacks. When developed wisely, BBNs serve as a valuable tool.

Chukkapalli et al. [5]

In this paper, we discuss the framework which is used to preserve smart farm owner's privacy. The leakage of data from the corporation will have a major impact on the farms which cause a loss in economics. Data perturbation with white Gaussian noise: It is a privacy-protecting technique, and this technique is widely used for data transformation while processing the signal for adding random noise to the original data or electronic devices. This framework consists of 4 modules which include data collection and exchange ontology, knowledge ledger The outcomes of the experiments demonstrated that it can be analysed through an anomaly detection model It can also be measured through data split and models used. Its performance on transformed data is slightly worse than that of non-transformed data. It is proved to be a promising approach and beneficial to and integrity of individual smart farm data.

Dimitrios Tychalas et al. [6]

The paper focuses on the growing threat of data theft in Industrial Control Systems (ICS) via the supply chain, using a firmware modification that unites the air gap and data is transferred from memory to analogue devices. By short-circuiting specific components in the aimed target device by making use of the Device Tree and lies exclusively in the devices connected externally, totally hidden from the main CPU. Implements the attack on an industrial Programmable Logic Controller, The paper fails to furnish a comprehensive evaluation of the attack's impact on the security and integrity. In terms of CPU performance and speed at which DMA is being transferred the introduced attack vector does not demonstrate any perceivable overhead. It emphasizes the crucial requirement for specialized strategies adopted to industrial control systems to quickly spot and rectify these menaces, while also stressing the significance of robust supply chain security measures.

Zunic et al. [7]

A progressive multi-stage anomaly detection set of rules with actual-global software within the context of supply chain control. In this process, it is very important to verify the accuracy of data which are used as data errors can result in significant additional costs it mainly applies to companies with similar services and processes. this paper gives individual processes that can be adapted for warehouse applications. This algorithm is implemented in some of the big and top distributed companies, showing its practical applicability and effectiveness, and this algorithm operates on the smart warehouse management system. Median-based improves the accuracy of anomaly detection. This algorithm combines online and offline parts and uses clusters and median. The importance of detection in information systems, its potential for cost savings, and an increase in the belief in data accuracy.

Vu et al. [8]

The main agenda of this article is to use the repositories, such as GitHub. This proceeds towards identifying threats and attacks through lightweight analysis. This will decrease the review effort by 97 percent and the processing time is speed. This procedure involves studying software artifacts from the repositories and analysing dispersed artifacts in different package repositories. This procedure includes various artifacts from different Python packages. Upcoming work includes making a classifier among benign and malicious administration. Processing speed will be extremely fast and speedy, which is one of the median times for separate artifacts. The algorithm will have speed processing techniques with a median time for processing and the upcoming work involves work that consists of making a classifier among benign and malicious administration. The lightweight analysis process will be because of distributed artifacts and repositories. This procedure will give fast processing in less than 33 sec. and finally, it is an efficient approach for on-the-fly detection of the software supply and finding suspicious activity.

Rejeb et al. [9]

This article focuses on the merging of IoT and blockchain technology. There are a few existing topics like traceability, efficiency, effectiveness and auditing, immutability, security, and scalability. The article has 6 research topics that trace the key features that are mentioned in the problem statement. Blockchain makes possible safe communication and honesty of exchange transactions, which will allow direct transactions with the machines through good contracts, it allows to call back the unsecured goods, and events. Product attribute. It refines the visibility of the product. And betters the traceability. live monitoring is one of the main advantages of this. technologies can improve final traceability. IoT solutions can improve operational costs. Both technologies have modern supply chains, and it will increase efficiency and effectiveness. Combination of IoT and blockchain will give better solutions to the problems. In both technologies, blockchain can face confrontations in supply chain management, and IoT enhances operational performance which includes traceability. It has a great capability for supply chain management.

Tsoukas et al [10]

The main theme of this article is to define the need for new approaches to control quality parameters. The blockchain and the TinyML the initial thing is to maintain data integrity it secures data tampering and gives clarity for all actors in the supply chain, and the next thing is to ease malicious behaviour. Using blockchain technology to pledge the purity of collected information. A technique called the self-sovereign identity approach is used for all actors to minimize single points of failures. The transparency levels are extremely elevated than related to the remaining and the quality parameters are very remarkable. anomaly detection assists in recognizing irregularities in the supply chain, and data integrity. It reviews performance through False Negative Rate [FNR] and False Positive Rate [FPR], and precision, and so forth.

The paper showcases a novel approach to enhancing food supply chain security by integrating blockchain technology and TinyML. The system encourages disclosure in the food industry. TinyML anomaly detection makes it easier to report misconduct to actors in the supply chain.

Chapter-3

Methodology:

The data we have chosen is supply chain shipment pricing data which contains twenty-four attributes with a huge amount of 10,325 tuples which can be used to analyze the dataset and find out the anomalies and outliers and make those predictions. The attributes are as follows:

- Id: it is a unique attribute which is different from everyone it is used for tracking and for referring purposes.
- Project code: it is a code for projects which is assigned for identification purpose.it is used for categorizing and organizing projects.
- Pq#: pq# it is prequalification number which is a unique number associated with prequalification process.
- Po/so#: po stands for purchase order, so stands for sales order. It is used for transactions between buyer and seller.
- Asn/dn#: asn stands for advance shipping notice and dn stands for delivery note number. These are used in the shipping process to notify about incoming shipments.
- Country: it is associated with the project, it indicates location.
- Managed by the person who is responsible for managing the project.
- Fulfill via method which is responsible for fulfilling project needs.
- Vendor inco term: it represents responsibilities of buyer and seller in international trade transactions.
- Shipment mode: mode of transportation used for shipping the products.
- pq first sent to client date: date when prequalification information was first sent to client.
- po sent to vendor date: the date when the purchased order was sent to vendor.
- scheduled delivery date: planned date for delivery of products.
- Delivered to client date: the actual date when product need to be delivered.
- Delivery recorded date: the delivery information when it was recorded.
- Product group: the category group which product belongs.
- Sub classification: subcategory of a product
- Vendor: the supplier who is providing the products
- Item description: a detailed description of project
- Molecular/test type: the specific molecule or type associated with project.
- Brand: brand name of product
- Dosage: amount of dose of product.
- Dosage form: form in which dosage is administered.
- Unit of measure (per pack): the unit used to measure quantity of product.
- Line-item quantity: quantity of the product ordered.
- line-item value: total values which are associated with line item.
- pack price: price per pack of product
- unit price: price per unit of product
- manufacturing site: location where product is manufactured.
- first line designation: designates the status of first line project.
- weight(kilograms): weight of products in kilograms.

- Freight cost (USD): cost associated with shipping or freight in u.s dollars.
- Line-item insurance (USD): insurance cost associated with line-item in us dollars.

Non continuous attributes: ["id", "project code", "pq#", " po/so #", "asn/dn #", "country", "managed by", "fulfill via", "vendor inco term", "shipment mode", "pq first sent to client date", "po sent to vendor date", "scheduled delivery date", "delivered to client date", "delivery recorded date", "product group", "sub classification", "vendor", "item description", "molecule/test type", "brand", "dosage", "dosage form", "unit of measure (per pack)", "manufacturing site", "first line designation", "freight cost (usd)"]

Continuous attributes: ["line_item_quantity," "unit_price", "pack_price" , "line_item_value", "line_item_insurance", "sub_classification", "dosage", "weight"]

Data cleaning:

We can find the null values in the dataset using **df. isnull().sum()** which is used to find no of null values in our data.
For the null values we found we are replacing them with a value by finding mean of a value
mean_value=df["line_item_insurance"].mean()
mean_value.
This function replaces all the null values with mean of line_item_insurance values:
df["line_item_insurance"].fillna(mean_value,inplace=True)
again, it checks if there are any null values:
df.isnull().sum()
a new data frame 'df1' is created by removing null values:
df1=df.dropna()
now it checks for null values in data frame:
df1.isnull().sum()
so, using **df1.info()** we can confirm that any null values are there are not.
so, we sent that data into new data file which has cleaned values and cleaned data.

1.Local outlier factor (LOF):

Lof is a density-based algorithm which identifies anomalies by comparing the local density of data points to their neighbours.
Lof is a unsupervised learning algorithm
It calculates the score for each data point based on ratio of local density of that point to local density of its neighbours.
Anomalies can be identified as the points with lower density compared to neighbours.
Local Outlier Factor(lof) focuses on identifying unusual or anomalies data points.
Step-1: import libraries

Step-2:data loading:
We have loaded dataset from a file named "cleaned_data_1.csv"
Step-3: selection of features:
: ["line_item_quantity," "unit price", "pack price", "line_item_value", "line_item_insurance", "weight"]

step 4: scaling:
for facilitating distance-based calculations for the selected features we are using "StandardScaler" function from scikit-learn
the scaled information has been fitted into Scaled.
step 5: LOF algorithm
model configuration:
lof algorithm was instantiated with following parameters they are n_neighbours, contamination.
model fitting:
the scaled data was used to fit the LOF model enabling the algorithm to learn that patterns within feature space.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import LocalOutlierFactor
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score, confusion_matrix

data = pd.read_csv("C:\\Users\\HP\\Desktop\\labs\\final\\cleaned_data_1.csv")

features = ['line_item_quantity', 'unit_price', 'pack_price', 'line_item_value', 'line_item_insurance', 'weight']

X = data[features]

X = X.apply(pd.to_numeric, errors='coerce')

X = X.dropna()

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

# Build and fitting the LOF model
lof = LocalOutlierFactor(n_neighbors=15, contamination=0.05)
y_pred = lof.fit_predict(X_scaled)
```

step 6: model evaluation:
lof predictions are used as true labels (-1 for anomalies, 1 for normal) and zero division is used to manage potential divisions by zero in precision, recall, f1 score.

```
true_labels = np.where(y_pred == -1, 1, 0)
predicted_labels = np.where(y_pred == -1, 1, 0)

# Confusion Matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Precision, Recall, F1 Score, and Accuracy
precision = precision_score(true_labels, predicted_labels, zero_division=1)
recall = recall_score(true_labels, predicted_labels, zero_division=1)
f1 = f1_score(true_labels, predicted_labels, zero_division=1)
accuracy = accuracy_score(true_labels, predicted_labels)

print("Confusion Matrix:")
print(cm)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"Accuracy: {accuracy:.2f}")
```

2.ISOLATION FOREST:

Isolation forest is an ensemble-based algorithm that isolates anomalies by constructiong a random forest of isolation trees. Isolation forest is an unsupervised learning algorithm. It will measure no. of points needed to isolate a data point as a measure of its anomalies. Anomalies are isolated more quickly than normal points.

Step-1: import required libraries.

Step-2:data loading

We have loaded dataset from a file named “cleaned_data_1.csv”

Step-3: selection of features:

: [“line_item_quantity,” “unit_price”, “pack_price, “line_item_value”, “line_item_insurance”, “weight”]

Step-4:data cleaning

Convert the selected columns to numeric format, managing any conversion errorsand dropping rows with missing values.

Step-4: feature scaling

Applied feature scaling using “standardscaler” to ensure uniformity in the given features.

Step-5: isolation forest model training:

Train the isolation forest model based on scaled data and set the model parameters that are n_jobs, random state, contamination rate, random state and fit the x_train_scaled.

Step-7: predict anomalies:

Predicting anomalies in training data using isolation forest model

```
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')

filepath = "C:\\Users\\HP\\Desktop\\labs\\final\\cleaned_data_1.csv"
df = pd.read_csv(filepath)

check_cols = ['line_item_quantity', 'unit_price', 'pack_price', 'line_item_value', 'line_item_insurance', 'weight']
x_train = df[check_cols]

x_train = x_train.apply(pd.to_numeric, errors='coerce')
x_train = x_train.dropna()

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)

# Isolation Forest model training
clf = IsolationForest(n_jobs=100, random_state=42, contamination=0.05)
clf.fit(x_train_scaled)

# Predicting anomalies
y_pred_train = clf.predict(x_train_scaled)

# Analyzing results
print(pd.value_counts(y_pred_train))
print(x_train.loc[y_pred_train == -1, :])
```

Step-8: model evalution:

lof predictions are used as true labels (-1 for anomalies, 1 for normal) and zero division is used to oversee potential divisions by zero in precision, recall, f1 score.

```python
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix,accuracy_score
true_labels = np.where(y_pred_train == -1, 1, 0)
predicted_labels = np.where(y_pred_train == -1, 1, 0)

# Confusion Matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Precision, Recall, F1 Score
precision = precision_score(true_labels, predicted_labels, zero_division=1)
recall = recall_score(true_labels, predicted_labels, zero_division=1)
f1 = f1_score(true_labels, predicted_labels, zero_division=1)
accuracy = accuracy_score(true_labels, predicted_labels)

print("Confusion Matrix:")
print(cm)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"Accuracy: {accuracy:.2f}")
```

3. Z-SCORE

The Z-score algorithm is a statistical technique for identifying outliers or anomalies in a data set. This algorithm is based on the concept of standard scores, also known as Z-scores, which define how many standard deviations a data point has from the mean Z-score of as given data point.

Z-score is calculated using the formula: $Z = (X - \mu) / \sigma$

The mean is the average value of a data set, and the standard deviation measures the spread or variability of the data. High absolute Z-scores indicate that the data points deviate significantly from the mean. In supply chain management, discrepancies can be associated with unusual or unexpected events, such as sudden increases or decreases in quantities, unit prices or incorrect pack prices.

Step 1: Import the required libraries, read the CSV file, define the list of features (features_for_anomaly_detection) to be used for anomaly detection. Include necessary features like 'line_item_quantity', 'unit_price', 'pack_price', 'line_item_value', 'line_item_insurance', and 'weight'. Finding central tendency and data spread using data.describe() method.

```
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
%matplotlib inline
from scipy.stats import zscore
matplotlib.rcParams['figure.figsize'] = (5,6)
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import train_test_split

data = pd.read_csv("C:\\Users\\madhu\\OneDrive\\Desktop\\DMPA_CCEB_38_57_62\\cleaned_data_1.csv")

features_for_anomaly_detection = ['line_item_quantity', 'unit_price', 'pack_price', 'line_item_value', 'line_item_insurance
data_subset = data[features_for_anomaly_detection]

data.describe()
```

✓ 0.0s Python

	id	line_item_quantity	unit_price	pack_price	line_item_value	line_item_insurance
count	7865.000000	7865.000000	7865.000000	7865.000000	7.865000e+03	7865.000000
mean	55428.856707	15411.948252	0.209888	12.664317	1.016927e+05	154.312588
std	31627.290002	26373.737049	0.288543	17.878098	1.637110e+05	238.537402
min	3.000000	1.000000	0.000000	0.000000	0.000000e+00	0.000000
25%	19250.000000	500.000000	0.060000	3.350000	3.822000e+03	5.990000
50%	70236.000000	3660.000000	0.130000	7.400000	2.493900e+04	40.870000
75%	84217.000000	19000.000000	0.250000	13.600000	1.277239e+05	202.980000
max	86823.000000	319150.000000	14.040000	306.880000	1.143013e+06	1175.020000

Step 2: Change the data_subset to numeric, constrain it to NaN and handle the error by discarding the NaN values. Divide the data into training and testing. Calculate the Z-score for the training and testing. Set the Z-score threshold for anomaly detection. Identify outliers in the training set based on the Z-score threshold and create a new column 'is_anomaly' in the training set to record the outliers

```
data_subset = data_subset.apply(pd.to_numeric, errors='coerce').dropna()

X_train, X_test = train_test_split(data_subset, test_size=0.05, random_state=42)

z_scores_train = np.abs(zscore(X_train))

threshold = 3

outliers_train = (z_scores_train > threshold).any(axis=1)

X_train['is_anomaly'] = outliers_train.astype(int)

z_scores_test = np.abs(zscore(X_test))

outliers_test = (z_scores_test > threshold).any(axis=1)

X_test['is_anomaly'] = outliers_test.astype(int)
```

Step-3

Label outliers in the test set and remove true labels (y_true) and predicted labels (y_pred) from the test set. Calculate and display confusion metrics and classification metrics such as precision, recall, F1 score, ROC-AUC

```
X_test['is_anomaly'] = outliers_test.astype(int)
y_true = X_test['is_anomaly']
y_pred = X_test['is_anomaly']
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
roc_auc = roc_auc_score(y_true, y_pred)

print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"ROC-AUC: {roc_auc:.2f}")

from sklearn.metrics import confusion_matrix, classification_report
y_true_train = X_train['is_anomaly']

y_pred_train = X_train['is_anomaly']
conf_matrix = confusion_matrix(y_true_train, y_pred_train)

print("Confusion Matrix:")
print(conf_matrix)
```

4)one class support vector machine:

A one class support vector machine is a statistical technique used to identify outliers or anomalies in the dataset. In one class svm the algorithm is trained on the normal or regular data instances or inlier instances. This algorithm trains on normal data and learns it, then it identifies instances that deviate from normal data and marks them as outliers or anomalies. In One class svm the 'nu' value, we can change the strictness of anomalies, higher the nu value higher strictness for detecting anomalies, so a greater number of anomalies and vice versa.

Step 1: Import the required libraries, read the CSV file, print sample data.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
data = pd.read_csv("C:/Users/nikhi/Downloads/cleaned_data_1_1.csv")
data.sample (5)
```

	id	line_item_quantity	unit_price	pack_price	line_item_value	line_item_insurance	sub_classification	dosage	weight
5731	84023	5900	0.05	3.28	19352.00	41.72	Adult	150/30mg	See DN-1948 (ID#:82906)
3454	57812	1144	0.33	10.00	11440.00	13.46	Adult	600/300/300mg	See ASN-32270 (ID#:35968)
7738	86596	63	0.04	2.26	142.38	0.18	Adult	200mg	Weight Captured Separately
5259	83408	434	0.05	2.73	1184.82	2.55	Adult	200mg	42
6703	85239	1530	0.10	8.87	13571.10	19.03	Pediatric	200mg	233

Step 2: A function to display the total price.

```
data['total price'] = data['pack_price'] * data['unit_price']
print(data.head(10))
```

[41]

```
...    id  line_item_quantity  unit_price  pack_price  line_item_value  \
0   3                1000        0.03        6.20          6200.00
1  15               31920        0.07        3.99        127360.80
2  16               38000        0.05        3.20        121600.00
3  23                 416        0.02        5.35          2225.60
4  44                 135        0.36       32.40          4374.00
5  45               16667        0.06        3.65         60834.55
6  46                 273        0.03        1.95           532.35
7  47                2800        0.34       41.10        115080.00
8  60                2800        0.34       41.10        115080.00
9  64               10000        0.17        9.98         99800.00

   line_item_insurance sub_classification     dosage                  weight  \
0           240.117626          Pediatric    10mg/ml                     358
1           240.117626              Adult      150mg                    1855
2           240.117626              Adult       30mg                    7590
3           240.117626          Pediatric    10mg/ml                     504
4           240.117626          Pediatric      200mg                     328
5           240.117626              Adult      200mg                    1478
6           240.117626              Adult       30mg  See ASN-93 (ID#:1281)
7           240.117626              Adult    200/50mg                     643
8           240.117626              Adult    200/50mg                     643
9           240.117626              Adult   150/300mg                    7416

   total price
...
6       0.0585
7      13.9740
8      13.9740
9       1.6966
```

Step 3: define the list of features (features_for_anomaly_detection) to be used for anomaly detection. Include necessary features like 'line_item_quantity', 'unit_price', 'pack_price', 'line_item_value', 'line_item_insurance', and 'weight'. Converting nonnumeric values to NaN values and dropping them and scaling the data.

```
numeric_features = ['line_item_quantity', 'unit_price', 'pack_price', 'line_item_value', 'line_item_insurance', 'weight', 'total price']
data[numeric_features] = data[numeric_features].apply(pd.to_numeric, errors='coerce')
data = data.dropna(subset=numeric_features)
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data[numeric_features])
[21]  ✓ 0.0s
```

Step 4: Train the OneClassSvm model using the scaled data and predict the anomalies, then convert predictions to anomalies 1 and normal as 0.

```
from sklearn.svm import OneClassSVM
model = OneClassSVM(nu=0.05)
model.fit(data_scaled)
predictions = model.predict(data_scaled)
predictions[predictions == 1] = 0
predictions[predictions == -1] = 1
data['predicted_labels'] = predictions
[22]  ✓ 0.3s
```

Step 5: Generation of synthetic labels as the dataset is unsupervised, then initialize all the instances to 0 considering then as normal data and 1 as predicted anomaly, and then evaluating the model by calculating accuracy, precision, recall and f1 score.

```
data['predicted_labels'] = predictions
data['true_labels'] = 0
data.loc[data['predicted_labels'] == 1, 'true_labels'] = 1
accuracy = accuracy_score(data['true_labels'], data['predicted_labels'])
precision = precision_score(data['true_labels'], data['predicted_labels'])
recall = recall_score(data['true_labels'], data['predicted_labels'])
f1 = f1_score(data['true_labels'], data['predicted_labels'])

[24]  ✓ 0.0s
```

Chapter-4
Results & discussions

Local Outlier Factor (LOF):

```
outlier_indices = np.where(y_pred == -1)[0]
print("Indices of outliers:", outlier_indices)

outliers = data.iloc[outlier_indices]
print("Details of outliers:")
print(outliers)
```

```
Indices of outliers: [   5   19   22   25   40   48   51   56   72   73   80   81   83   84
   85   87  100  108  109  128  129  133  134  145  149  151  155  159
  160  161  162  174  176  177  181  183  196  210  211  213  219  220
  230  243  246  260  273  284  353  403  408  413  430  451  453  463
  546  548  564  590  642  643  648  650  679  717  735  741  774  794
  801  819  849  863  869  903  971  972  982  994 1015 1017 1026 1029
 1030 1032 1035 1072 1075 1090 1145 1157 1178 1183 1196 1197 1198 1219
 1227 1236 1257 1258 1300 1332 1348 1369 1370 1379 1387 1390 1392 1411
 1468 1500 1508 1512 1524 1528 1559 1562 1566 1568 1624 1626 1642 1707
 1712 1733 1766 1813 1856 1904 1914 1935 1958 1965 1998 2006 2015 2059
 2095 2182 2196 2215 2346 2379 2399 2440 2475 2526 2533 2552 2561 2580
 2589 2616 2625 2650 2671 2672 2704 2705 2706 2713 2724 2726 2732 2766
 2825 2827 2833 2852 2869 2905 2966 2978 2989 3006 3012 3024 3026 3029
 3065 3079 3107 3253 3287 3300 3322 3374 3384 3414 3461 3524 3577 3601
 3643 3647 3649 3655 3662 3720 3742 3798 3802 3827 3844 3846 3864 3875
 3877 3944 4008 4028 4042 4053 4087 4092 4127 4230 4246 4296 4435 4457
 4466]
Details of outliers:
        id  line_item_quantity  unit_price  pack_price  line_item_value  \
5       45               16667        0.06        3.65         60834.55
19     420                8400        0.38       11.48         96432.00
22     438                 400        0.12        7.00          2800.00
25     532                 100        0.64       19.06          1906.00
40    2947                 400        0.02        1.50           600.00
...    ...                  ...         ...         ...              ...
4246  78340                1049        0.68       82.20         86227.80
4296  79841                7420        0.01        1.95         14469.00
4435  82307                 500        0.27        8.14          4070.00
4457  82332                3872        0.01        1.83          7085.76
4466  82343               23340        0.27       16.44        383709.60

      line_item_insurance sub_classification    dosage  \
5              240.117626              Adult     200mg
19             154.290000              Adult     600mg
22               4.480000          Pediatric      25mg
25               3.050000              Adult     250mg
40               0.960000              Adult      15mg
...                   ...                ...       ...
4246           142.280000              Adult  200/50mg
4296            20.290000          Pediatric   10mg/ml
4435             8.770000              Adult     300mg
4457            15.280000          Pediatric   10mg/ml
4466           827.280000              Adult     300mg

                          weight
5                           1478
19                           654
22                            81
25    Weight Captured Separately
40                            83
...                          ...
4246                         264
4296   See ASN-15328 (ID#:46005)
4435                          28
4457                        1438
4466                        2344

[225 rows x 9 columns]
```

This shows how many anomalies detected based on our lof model.

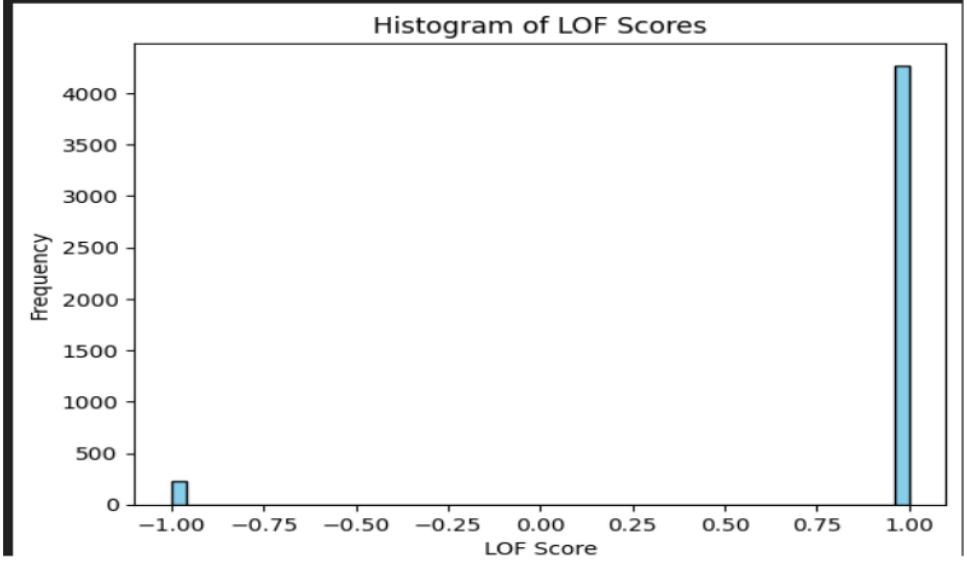
Evaluating results of lof:

```
Confusion Matrix:
[[4271    0]
 [   0  225]]
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
Accuracy: 1.00
```

It shows that our model is performing good based on confusion matrix.
This confusion matrix indicates that the dataset having 4271 true negative values which results in that they are correctly predicted as normal class (class 0)
There are 225 true positive values which means that they are detected as anomalies (class 1)
So, after analysing this we can say there are no false positive or false negative values and the precision, recall, f1 score is 1. Since there are no false negative and false positive values the accuracy is 1 which means 100%.

Visualization:

```
plt.hist(y_pred, bins=50, color='skyblue', edgecolor='black')
plt.xlabel('LOF Score')
plt.ylabel('Frequency')
plt.title('Histogram of LOF Scores')
plt.show()
```



Results of Isolation Forest:

Predicted anomalies:

```
 1    4271
-1     225
Name: count, dtype: int64
      line_item_quantity  unit_price  pack_price  line_item_value  \
97                   100        0.55      183.33         18333.00
112                61021        0.20       11.89        725539.69
169                60784        0.56       16.75       1018132.00
198                 5900        1.68       50.40        297360.00
243                  200        0.61      163.44         32688.00
...                  ...         ...         ...              ...
7646               24470        0.25       29.68        726269.60
7708              138974        0.16        4.70        653177.80
7724               69963        0.43       12.96        906720.48
7772               93700        0.14        8.26        773962.00
7773              169447        0.06        3.75        635426.25

      line_item_insurance   weight
97              29.330000    337.0
112           1160.860000   6768.0
169            240.117626  11179.0
198            475.780000    493.0
243            240.117626     54.0
...                   ...      ...
7646          1018.230000   6808.0
7708           915.760000  14845.0
7724          1121.610000   7819.0
7772           957.390000  12220.0
7773           786.020000   8039.0

[225 rows x 6 columns]
```

Evaluation of isolation forest model:

```
Confusion Matrix:
[[4271    0]
 [   0  225]]
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
Accuracy: 1.00
```

This confusion matrix shows that our code is working very well and finding the anomalies correctly.

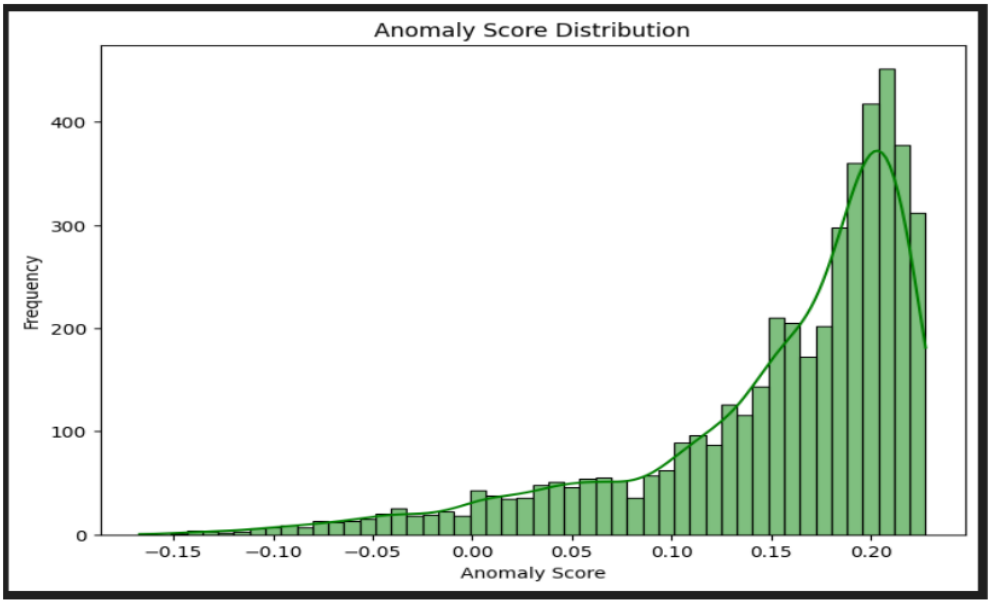
This confusion matrix indicates that the dataset having 4271 true negative values which means they are correctly predicted as normal class (class 0)

There are 225 true positive values which means they are detected as anomalies (class 1)

So, after analysing this we can say there are no false positive or false negative values and the precision, recall, f1 score is 1.

Visualization:

```python
import seaborn as sns
# Anomaly score distribution
plt.figure(figsize=(8, 6))
sns.histplot(clf.decision_function(x_train_scaled), bins=50, kde=True, color='green')
plt.title('Anomaly Score Distribution')
plt.xlabel('Anomaly Score')
plt.ylabel('Frequency')
plt.show()
```


Anomaly Score Distribution
Frequency
0
100
200
300
400
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
Anomaly Score

Z-Score:

Evaluation of z-score algorithm:

```
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
ROC-AUC: 1.00
Confusion Matrix:
[[3916    0]
 [   0  355]]
```

This confusion matrix shows that our code is working very well and finding the anomalies correctly.
This confusion matrix indicates that the dataset having 3916 true negative values which means they are correctly predicted as normal class (class 0)
There are 355 true positive values which means they are detected as anomalies (class 1)
So, after analysing this we can say there are no false positive or false negative values and the precision, recall, f1 score is 1.

Printing the detected outliers and anomalies:

```
print(pd.value_counts(y_pred_train))
print(X_train.loc[y_pred_train == 1, :])
```

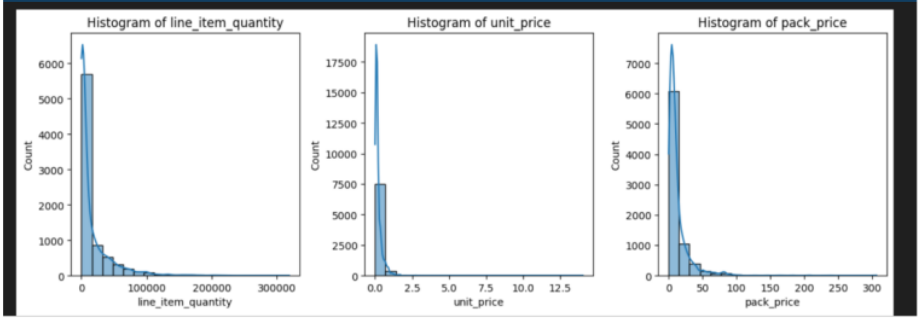
✓ 0.0s

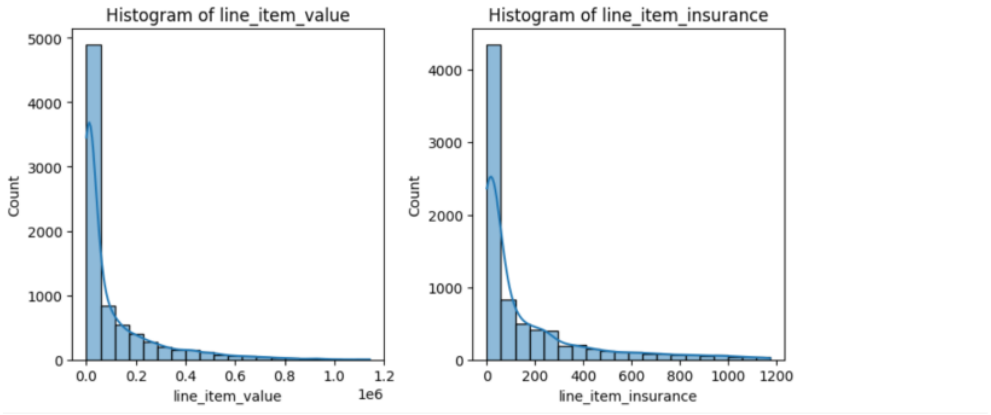
```
is_anomaly
0    3916
1     355
Name: count, dtype: int64
      line_item_quantity  unit_price  pack_price  line_item_val
2130                  40        0.47      125.94           5037
3608               65763        0.42       12.49         821379
2805                 120        0.75       90.00          10800
893                  995        1.17       70.00          69650
4803               50867        0.44       13.15         668901
...                  ...         ...         ...            ...
3781                 102        0.75       90.00           9180
619                 1847        1.17       70.00         129290
5382               85919        0.14        8.26         709690
3471              172428        0.05        2.87         494868
5113               51520        0.04        2.31         119011

      line_item_insurance   weight  is_anomaly
2130                10.86     11.0           1
3608              1016.05   8182.0           1
2805                21.17     18.0           1
893                111.44    219.0           1
4803               787.30   4749.0           1
...                   ...      ...         ...
3781                17.99     15.0           1
...
```

Data visualization of selected attributes.

```
import seaborn as sns
attributes = ['line_item_quantity', 'unit_price', 'pack_price', 'line_item_value', 'line_item_insu
data = pd.read_csv("C:\\Users\\madhu\\OneDrive\\Desktop\\DMPA_CCEB_38_57_62\\cleaned_data_1.csv")

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

for i, attribute in enumerate(attributes, 1):
    plt.subplot(3, 3, i)
    sns.histplot(data[attribute], bins=20, kde=True)
    plt.title(f'Histogram of {attribute}')

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

✓ 1.4s Python

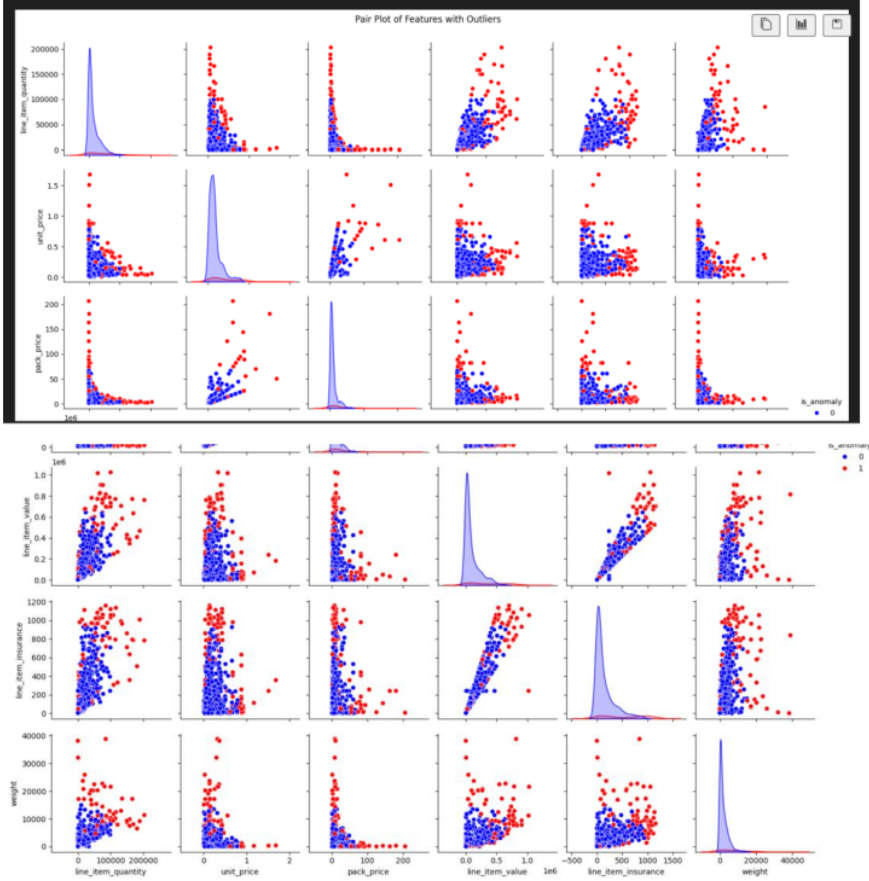




```python
X_test_combined = X_test.copy()
X_test_combined['is_anomaly'] = X_test['is_anomaly']
sns.pairplot(X_test_combined, hue='is_anomaly', palette={0: 'blue', 1: 'red'})
plt.suptitle('Pair Plot of Features with Outliers', y=1.02)
```

✓ 8.6s

```
Text(0.5, 1.02, 'Pair Plot of Features with Outliers')
```



One class svm:

print the accuracy, precision, recall and f1 score and print the count of anomalies in the dataset considering anomalies are labelled as 1.

```
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 Score: {f1:.2f}")
    anomaly_count = len(data[data['predicted_labels'] == 1])
    print(f"Number of anomalies: {anomaly_count}")
```

[15] ✓ 0.0s

```
Accuracy: 100.00%
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```

Code to count the number of anomalies in the dataset as per nu value and print them.

```
anomaly_count = len(data[data['predicted_labels'] == 1])
print(f"Number of anomalies: {anomaly_count}")
anomalies = data[data['true_labels'] == 1]
print("Anomalies:")
print(anomalies)
```

[26] ✓ 0.0s

```
Number of anomalies: 227
Anomalies:
         id  line_item_quantity  unit_price  pack_price  line_item_value  \
72     1179                   8        0.23       76.72           613.76
82     1283               95500        0.03        1.51        144205.00
84     1298               70000        0.12        7.50        525000.00
97     1519                 100        0.55      183.33         18333.00
169    2685               60784        0.56       16.75       1018132.00
...     ...                 ...         ...         ...              ...
7277  85985                  26        0.14        8.58           223.08
7383  86140               21497        0.16       19.66        422631.02
7646  86480               24470        0.25       29.68        726269.60
7724  86579               69963        0.43       12.96        906720.48
7773  86653              169447        0.06        3.75        635426.25

      line_item_insurance sub_classification         dosage   weight  \
72               1.200000              Adult          100mg     48.0
82             240.117626              Adult           30mg   9170.0
84             240.117626              Adult    150/200/30mg   6813.0
97              29.330000              Adult          100mg    337.0
169            240.117626              Adult          600mg  11179.0
...                   ...                ...            ...      ...
7277             0.280000              Adult  150/300/200mg  33051.0
7383           434.460000              Adult       200/50mg     21.0
7646          1018.230000              Adult       200/50mg   6808.0
...
7724       5.5728                1            1
7773       0.2250                1            1
```

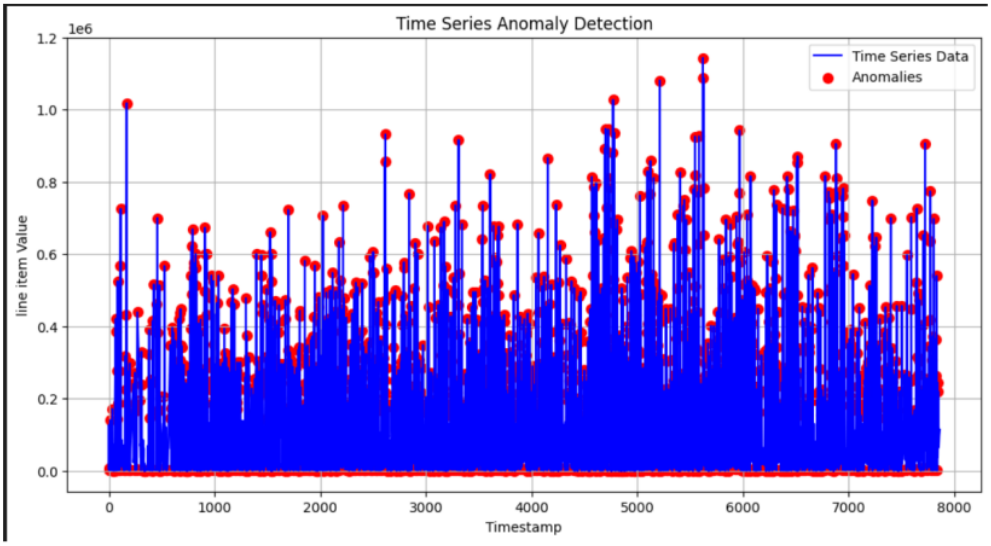
Visualization:

Data visualization of selected attributes, red are anomalies and blue are normal data. Scatter plot, time series plot and 3d scatter plot.

Timeseries

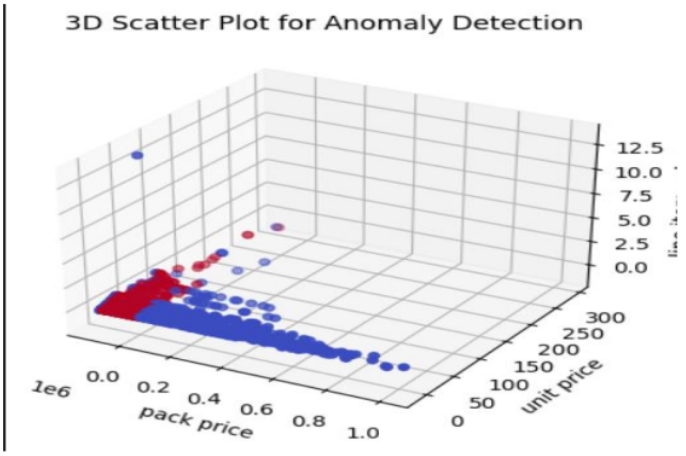
```
time_series = data['line_item_value']
svm_model = OneClassSVM()
svm_model.fit(time_series.values.reshape(-1, 1))
anomaly_predictions = svm_model.predict(time_series.values.reshape(-1, 1))
plt.figure(figsize=(12, 6))
plt.plot(time_series.index, time_series, label='Time Series Data', color='blue')
plt.scatter(time_series.index[anomaly_predictions == -1], time_series[anomaly_predictions == -1], c='red', label='Anomalies', s=50)
plt.title('Time Series Anomaly Detection')
plt.xlabel('Timestamp')
plt.ylabel('line item Value')
plt.legend()
plt.grid(True)
plt.show()
```

[19] ✓ 2.5s



3d scatter plot

```
from sklearn.decomposition import PCA
features = data[['pack_price', 'unit_price', 'line_item_value']]
svm_model = OneClassSVM()
svm_model.fit(features)
anomaly_predictions = svm_model.predict(features)
pca = PCA(n_components=3)
reduced_features = pca.fit_transform(features)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(reduced_features[:, 0], reduced_features[:, 1], reduced_features[:, 2], c=anomaly_predictions, cmap='coolwarm')
ax.set_title('3D Scatter Plot for Anomaly Detection')
ax.set_xlabel('pack price')
ax.set_ylabel('unit price')
ax.set_zlabel('line item value ')

plt.show()
```



Confusion matrices obtained:

Local outlier factor:

```
Confusion Matrix:
[[4271    0]
 [   0  225]]
```

Isolation Forest:

```
Confusion Matrix:
[[4271    0]
 [   0  225]]
```

z-score:

```
Confusion Matrix:
[[3916    0]
 [   0  355]]
```

one class svm:

```
Confusion Matrix:
[[4269    0]
 [   0  227]]
```

Anomalies detected:

Lof:

n_neighbours	Contamination	Anomalies detected
15	0.05	225
15	0.10	450
15	0.15	675
15	0.20	899

Isolation forest:

n_jobs	Random state	Contamination	Anomalies detected
15	42	0.05	225
15	42	0.10	449
15	42	0.15	675
15	42	0.20	898

z-score:

Threshold	Anomalies detected
2	669
2.5	520
3	355

one class svm:

Nu	Anomalies detected
0.05	227
0.10	448
0.15	674
0.20	899

Lof:

pros: LOF is effective in detecting local outliers and is generally robust. It provides a measure of the density of data points in the area.

Cons: The detected anomalies seem to increase linearly with the contamination level.

Isolated forest:

Pros: Isolation Forest is more efficient and scalable. Generally effective in detecting anomalies in high-dimensional data.

Cons: As with LOF, detected abnormalities increase with contamination.

Z-Score:

Pros: The Z-Score is simple and interpretable. It measures how many standard deviations a data point is from the mean.

Cons: As the threshold increases, the detected anomalies decrease, which may indicate that the method is sensitive to the choice of threshold and may also assume normality in data distribution.

One-class SVM:

pros: One-class SVM is effective in capturing the underlying structure of all available data.

Cons: Similar to LOF and Isolation Forest, the number of detected anomalies increases with the fraction of the parameter (nu) representing the upper limit of margin errors

Chapter 5

Conclusion:

Supply chain anomaly detection system provides a simple and valuable solution for identifying irregularities in supply chain data.

Through outlier analysis techniques discussed above the system classifies and analyses features in the data set, providing insight into potential anomalies. The anomaly detection system provides decision support for stakeholders, including supply chain managers. By updating the data used for training, the system can learn from new systems and adapt its anomaly detection capability to changing conditions.

One-Class SVM is a good candidate if we have aspirations for an algorithm that considers the underlying structure of all our data.

If performance and scalability are important, and we have high quality data, an isolated forest may be the right choice.

Finally, One class svm is detecting highest anomalies so it is the best algorithm among the four.

References:

Dataset from: data.gov.in

Supply Chain Shipment Pricing Data - Catalog

[1] Baumgartner, P., & Krumpholz, A. (2021). Anomaly Detection in a Boxed Beef Supply Chain. ACM International Conference Proceeding Series, 1–7. https://doi.org/10.1145/3474963.3474964

[2] Jiménez-Carmelo, A. M., Li, P., Erasmus, S. W., Wang, H., & van Ruth, S. M. (2023). Spatial-Temporal Event Analysis as a Prospective Approach for Signalling Emerging Food Fraud-Related Anomalies in Supply Chains. Foods, 12(1). https://doi.org/10.3390/foods12010061

[3] Sahin, B., & Soylu, A. (2020). Multi-Layer, Multi-Segment Iterative Optimization for Maritime Supply Chain Operations in a Dynamic Fuzzy Environment. IEEE Access, 8, 144993–145005. https://doi.org/10.1109/ACCESS.2020.3014968

[4] Yeboah-Ofori, A., Islam, S., & Brimicombe, A. (2019). Detecting cyber supply chain attacks on cyber physical systems using bayesian belief network. Proceedings - 2019 International Conference on Cyber Security and Internet of Things, ICSIoT 2019, 37–42. https://doi.org/10.1109/ICSIoT47925.2019.00014

[5] Chukkapalli, S. S. L., Ranade, P., Mittal, S., & Joshi, A. (2021). A Privacy Preserving Anomaly Detection Framework for Cooperative Smart Farming Ecosystem. Proceedings - 2021 3rd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2021, 340–347. https://doi.org/10.1109/TPSISA52974.2021.00037

[6] Dimitrios Tychalas, Anastasis Keliris, Michail Maniatakos, "Supply Chain threats for stealthy data exfiltration in industrial control systems", "IEEE", 2019. https://ieeexplore.ieee.org/abstract/document/8854451/

[7] Zunic, E., Tucakovic, Z., Delalic, S., Hasic, H., & Hodzic, K. (2020). Innovative Multi-Step Anomaly Detection Algorithm with Real-World Implementation: Case Study in Supply Chain Management. 2020 IEEE / ITU International Conference on Artificial Intelligence for Good, AI4G 2020, 9–16. https://doi.org/10.1109/AI4G50087.2020.9311045

[8] Vu, D. L., Pashchenko, I., Massacci, F., Plate, H., & Sabetta, A. (2020). Towards Using Source Code Repositories to Identify Software Supply Chain Attacks. Proceedings of the ACM Conference on Computer and Communications Security,

2093–2095. https://doi.org/10.1145/3372297.3420015

[9] Rejeb, A., Keogh, J. G., & Treiblmaier, H. (2019). Leveraging the Internet of Things and blockchain technology in Supply Chain Management. Future Internet, 11(7). https://doi.org/10.3390/fi11070161

[10] Tsoukas, V., Gkogkidis, A., Kampa, A., Spathoulas, G., & Kakarountas, A. (2022). Enhancing Food Supply Chain Security using Blockchain and TinyML. Information (Switzerland), 13(5). https://doi.org/10.3390/info13050213

report_supply_chain_anomaly_detection_38_57_62

ORIGINALITY REPORT

6%	5%	3%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.dtu.ac.in:8080 Internet Source	1%
2	Suzan Hajj, Joseph Azar, Jacques Bou Abdo, Jacques Demerjian, Christophe Guyeux, Abdallah Makhoul, Dominique Ginhac. "Cross-Layer Federated Learning for Lightweight IoT Intrusion Detection Systems", Sensors, 2023 Publication	1%
3	www.coursehero.com Internet Source	1%
4	doaj.org Internet Source	1%
5	eprints.utar.edu.my Internet Source	<1%
6	www.tutorialspoint.com Internet Source	<1%
7	Mariam Elnour, Nader Meskin, Khlaed M. Khan. "Hybrid Attack Detection Framework for Industrial Control Systems using 1D-Convolutional Neural Network and Isolation	<1%

Forest", 2020 IEEE Conference on Control Technology and Applications (CCTA), 2020
Publication

8	Clauss, Kersten, Huimin Yan, and Claudia Kuenzer. "Mapping Paddy Rice in China in 2002, 2005, 2010 and 2014 with MODIS Time Series", Remote Sensing, 2016. Publication	<1 %
9	core.ac.uk Internet Source	<1 %
10	discussionsbytopic.com Internet Source	<1 %
11	ntrs.nasa.gov Internet Source	<1 %
12	www.stuvia.com Internet Source	<1 %
13	deepai.org Internet Source	<1 %
14	ebin.pub Internet Source	<1 %
15	Yashvi Shah, Yashashvi Verma, Umang Sharma, Aum Sampat, Vikram Kulkarni. "Supply Chain for Safe & Timely Distribution of Medicines using Blockchain & Machine Learning", 2023 5th International Conference	<1 %

on Smart Systems and Inventive Technology (ICSSIT), 2023

Publication

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		