MÄLARDALEN UNIVERSITY
SWEDEN

PROJECT ASSIGNMENT 2.1

Module 2

THOMAS SELIN

School of Business, Society and Engineering
Course: Multivariate Data Analysis in Engineering
Course code: MTK337
Credits: 7.5 ECTS

Supervisor: Prof. Amare Desalegn Fentaye
Examiner: Prof. Amare Desalegn Fentaye

Date: 2024-10-09
Email:
thomasselin.studies@gmail.com

ABSTRACT

This project assignment is the second assignment in the course “Multivariate Data Analysis in Engineering”. The problem statement remains the same as in the first assignment, but as the dataset has been modified significantly after learning from the first assignment, a new exploratory data analysis and a new principal component analysis was done. After that, a multivariate classification model was created using the Isolation Forest algorithm. The chosen goal for the entirety of the course projects is to improve the alerting for incidents in the IT (Information Technology) system. Accomplishing this would improve the operations of the system and can potentially also give insights to how the system can be improved.

The dataset that was used in this assignment contains values from metrics and logs from the system which is made up of many small components, called microservices. The dataset contains observations from one of those microservices.

The final model was successfully able to identify anomalies, but further evaluation is needed, and ways of doing that are described in the report.

Keywords: anomaly detection, multivariate data analysis, machine learning, microservices, information technology

Content

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Description
IT	Information technology
EDA	Exploratory Data Analysis - methods for getting to know and finding patterns in data
PCA	Principal Component Analysis - method used in multivariate analysis to model data and reduce dimensions
CPU	Central processing unit - computer hardware component that executes programs that for example processes data
RAM memory	Random Access Memory - computer hardware component that enables quick storing and reading of data

DEFINITIONS

Definition	Description
Microservice	A small component of an information technology system that handles one or a few tasks. Modern information technology systems tend to be partly or fully made up of microservices, and often a large number of microservices exist within such an IT system.

1 INTRODUCTION

This report is on the topic of using multivariate data analysis and machine learning to improve alerting when anomalies occur in an IT system. The report is structured as follows: introduction, materials, methods, results and discussion, and conclusion. The code that was used to perform the analysis and model development can be found in Appendix 2.

1.1 BACKGROUND

The goal for the entirety of the course projects is to improve the alerting for incidents in an IT system that is made up of many microservices. Microservices most oftenly handles business logic, for example the calculation of the price of a product. The microservices often communicate and cooperate with one another to fulfill tasks.

There are many aspects of the alerting that would be valuable to improve, like better root cause analysis for incidents and more efficient anomaly detection (preferably as early as possible). This second project assignment of the course has included 3 parts:

- Exploratory Data Analysis (EDA), which are methods for getting to know and finding patterns in data.
- Principal Component Analysis, a method used in multivariate analysis to model data and reduce dimensions.
- Creating a suitable machine learning model to solve the problems defined below.

1.1.1 PROBLEMS

A few challenges with alerting in IT systems were identified as relevant to this project:

- Quickly becoming aware of serious anomalies.
- Quickly identifying where in the IT system a serious anomaly originates from.
- Quickly judge how severe the anomaly is.
- Minimizing the risk that more people than necessary get involved and spend time on troubleshooting an anomaly. This is more likely to happen if alerts for the IT system are imprecise so that alerts are set off in other parts of the system than where the problem originates from.

An important factor to be taken into account when creating a model is to which extent it is possible to explain the results that the model generates. Being able to understand the results well can give better general insights regarding what causes anomalies in the microservice and appropriate actions can be taken to minimize those causes.

1.2 Purpose and aim

With the logs and metrics from the system, be able to significantly improve the alerting for one microservice in the IT system. The same method could then be used to set up alerts for other microservices as well.

2 MATERIALS

A dataset was extracted from the logs and metrics of a microservice containing variables and values of the type shown in Table 1 below. See *Appendix 1* for a further representation of the dataset. The data was selected by using domain knowledge of what are important variables for how well a microservice is performing.

The data is somewhat sensitive and therefore not described in detail in this report. I consider the data trustworthy, and I didn't find any significant problems with the data throughout the entire process of this project assignment.

The data was collected from the beginning of 2024-10-01 to, and including, 2024-10-07. That is a total of 7 days and was the maximum amount of data available at the time of collection. As it is most interesting to have good alerting and a performant model for the main usage hours of the system, which is 8.00-22.00 each day of the week, data from other times of the day were excluded. During 8.00-22.00 the usage is quite even which I believe is beneficial as timestamp was not used in the training of the Isolation Forest model and therefore the model doesn't model changes over time. All variable values are of numerical type. The total number of observations in the dataset after preprocessing was 5880. See *Table 1* below for a structured description of the dataset.

Variable name	Variable description	Example value
timestamp	Which minute the other values below belongs to	2024-10-03 13:00:00
warning_proportion	The proportion of all log events labeled with a log level that had the WARN log level	0.0022
error_proportion	The proportion of all log events labeled with a log level that had the ERROR log level	0.0013
avg_cpu_usage_percent	Average percentage of available CPU that was used	34
avg_memory_usage_percent	Average percentage of available RAM memory that was used	23
avg_disk_usage_percent	Average percentage of available harddrive disk space that was used	33
avg_latency_milliseconds	Average nr of milliseconds it took to process a request to the microservice	144

Table 1: *Variables in the dataset*

As the dataset does not contain labels of which data should have warranted raising an alert, an unsupervised machine learning approach was implemented. Target classes will therefore be defined as the desired output classes from inference of the later developed machine learning model. Intended target classes when inferencing the final model is whether or not an alert should be triggered.

3 METHODS

3.1 PREPROCESSING

The dataset had no missing values or values that indicated potential measurement problem. In preparation for PCA, the data was scaled using the function **StandardScaler().fit_transform()** from the Scikit-learn Python library [1].

3.2 EXPLORATORY DATA ANALYSIS

The relationships between the variables were visualized with pair plots. Correlation values between all variables were generated and visualized in a correlation-plot.

3.3 PRINCIPAL COMPONENT ANALYSIS

[3]. The following section about the background of Principal Component Analysis was sourced from www.claude.ai on 2024-09-21 and fact checked against this course's lecture materials.

Principal Component Analysis (PCA) is a fundamental method in multivariate analysis and machine learning, used mainly for better understanding data as well as simplification of data while keeping important patterns within it.

Key aspects of PCA include:

1. **Dimensionality Reduction**: PCA transforms high-dimensional data into a lower-dimensional space while preserving as much variance as possible.
2. **Orthogonal Transformation**: It creates new variables (principal components) that are linear combinations of the original features and are orthogonal to each other.
3. **Variance Maximization**: The first principal component accounts for the maximum variance in the data, with each subsequent component capturing the highest remaining variance.
4. **Data Visualization**: PCA facilitates the visualization of high-dimensional data by projecting it onto a lower-dimensional space, often two or three dimensions.
5. **Noise Reduction**: By focusing on the components with the highest variance, PCA can help filter out noise in the data.

PCA's ability to simplify complex datasets while retaining essential information makes it a valuable preprocessing step in many machine learning pipelines, enhancing model performance and interpretability. One limitation of the method is its inability to account for non-linear relationships between variables.

One technique to be considered in the context of PCA is Kernel Principal Component Analysis (KPCA). If you see in the EDA or have domain knowledge that indicate non-linear patterns in your data set, then KPCA could be a better method because it doesn't disregard non-linearity which PCA does.

3.4 ISOLATION FOREST

[2]. The following section about the background of Principal Component Analysis was sourced from www.claude.ai on 2024-10-09.

Isolation Forest is an unsupervised machine learning algorithm designed for anomaly detection. The method is based on the principle that anomalies are rare and different from normal data points, making them easier to separate or "isolate" from the rest of the data.

The algorithm works as follows:

1. It creates multiple decision trees, called isolation trees (iTrees), using random subsets of the data.
2. For each tree, it randomly selects a feature and a split point to divide the data.
3. This process continues until each data point is isolated or a predefined tree depth is reached.

Anomalies typically require fewer splits to be isolated, resulting in shorter paths from the root to the leaf in these trees. The algorithm calculates an anomaly score for each data point based on the average path length across all trees. Data points with consistently shorter paths are more likely to be anomalies.

Isolation Forest is computationally efficient and particularly effective for high-dimensional datasets. Unlike many other anomaly detection methods, it doesn't rely on calculating distances between points or estimating density, which makes it well-suited for large and complex datasets.

4 Results and Discussion

The results from running the analysis are illustrated in plots below and commented on.

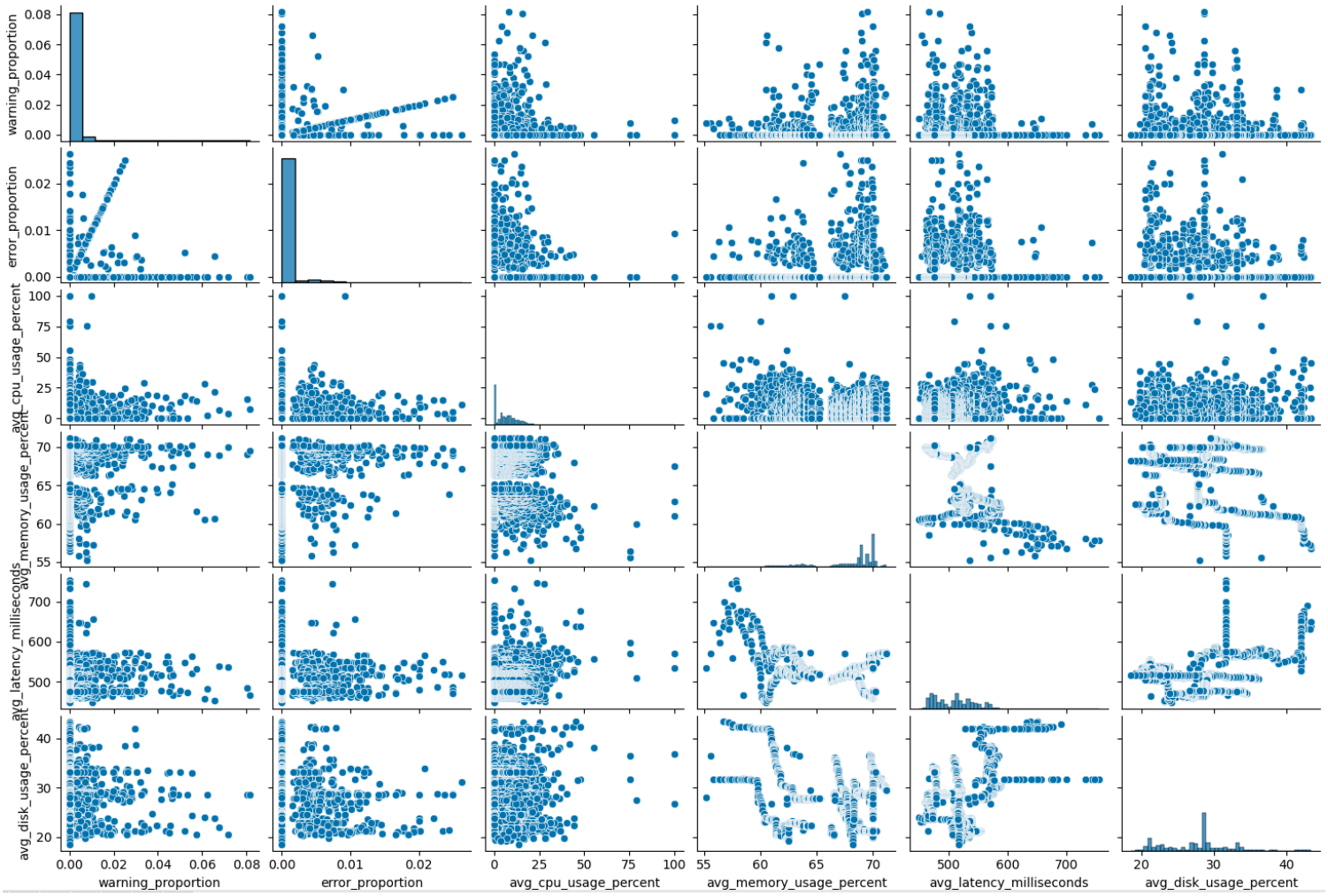

Figure 1. Pair plot illustrating the relationships between variables

I did not see, in the above *Figure 1*, a clear indication of clear non-linear patterns in the data, and therefore KPCA was not conducted.

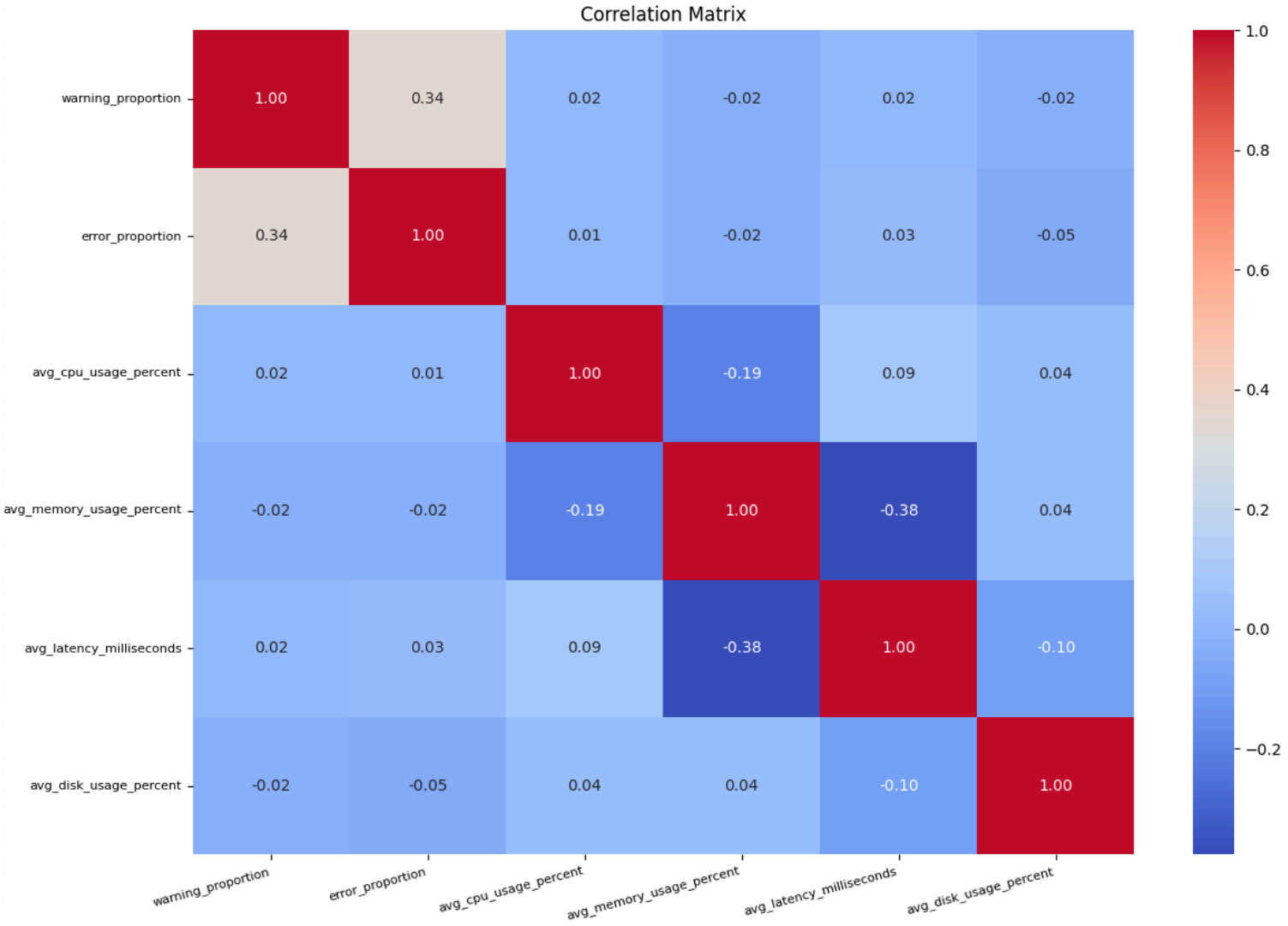

Figure 2. Plot of correlations between the variables

As seen above in *Figure 2*, there are in general low correlations between the variables, with the correlation between latency and memory usage as well as the correlation between error and warning proportions being the most significant.

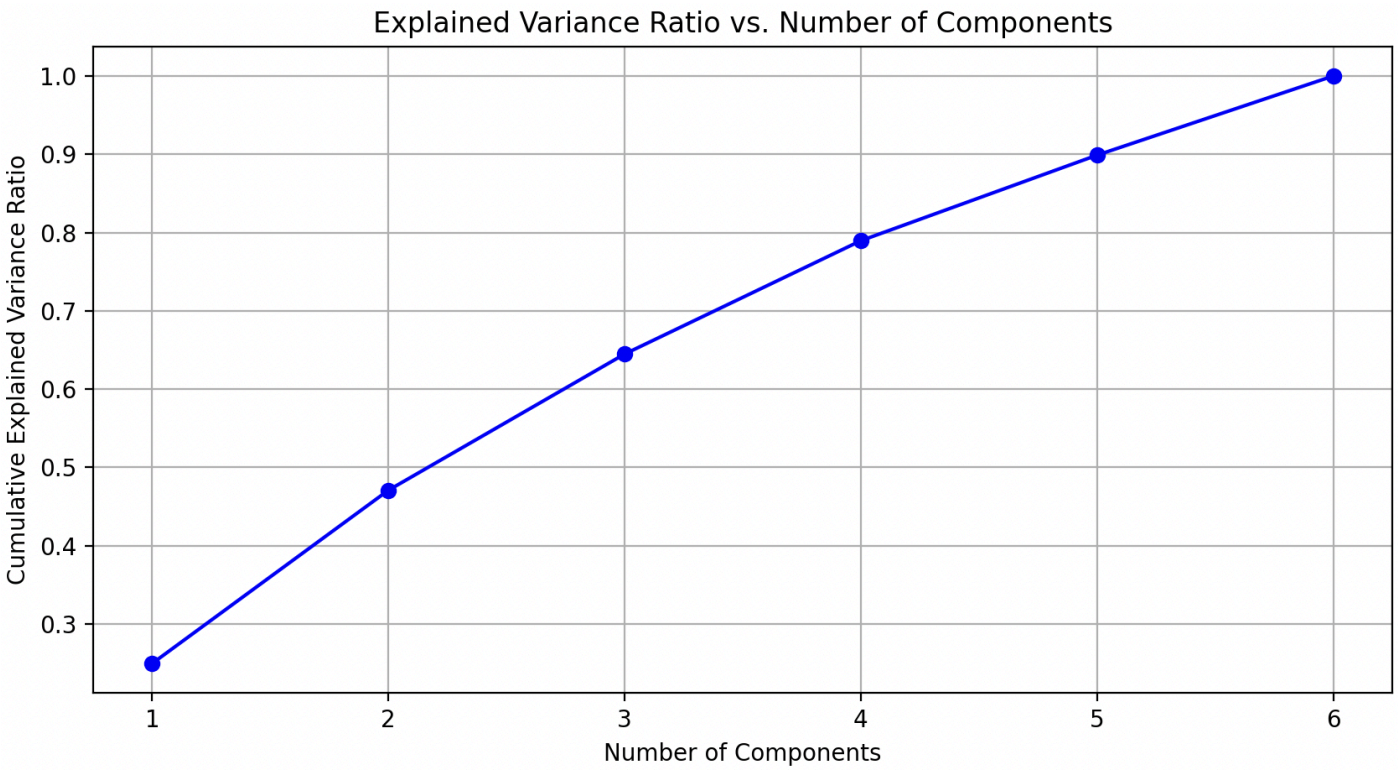

Figure 3. PCA: Plot of the percentage of variance explained by number of principal components

A PCA was conducted with the goal of reducing dimensions. Plots to visualize the results of the analysis can be seen above in *Figure 3*. A decision of the number of principal components to keep was made while taking in consideration the potential later use of the PCA model as input for the Isolation Forest model and the risk of overfitting the model to the data.

As seen in *Figure 3* above, for PCA, 5 principal components can explain around 90% of the variance. This means that the PCA was not very efficient at reducing dimensions in this case, as 5 principal components is only one less then the dimension of the data set.

There are some benefits and disadvantages to using the data set with principal components that was created as a result of the PCA as input when creating the Isolation Forest model. It could potentially lessen the risk of having an overfitted model, and could also reduce inference time.

On the other hand, to instead use the original data set as input when creating the Isolation Forest model, would mean better explainability of the model, and less steps to creating and updating the model.

Isolation Forest is already quite efficient with high-dimensional data. The speed improvement from combining with PCA might be less dramatic compared to other algorithms. PCA might also remove some information that could be useful for anomaly detection. [4]

Therefore the Isolation Forest model was developed using the original dataset.

After the model was created, I used the original dataset again to detect anomalies in that data. That is not optimal as it is better to test out the model on data that was not used during training of the mode, but that will be done in the actual real world application of the model when monitoring the system. Therefore that was excluded at this stage.

When creating the Isolation Forest model with the tools/libraries that were used, one input is a variable called *contamination*. This sets the threshold for what to consider an anomaly, and can be set based on your expected anomaly rate.

With the *contamination* variable set to 0.005, 30 anomalies were detected in the original data set. See *Figure 4* below for an example of 2 of those 30. The first anomaly has significantly higher cpu usage than what's normal for the microservice and also high latency and warning proportion.
The second anomaly has higher latency and higher error proportion than what's normal.

avg_cpu_usage_percent	avg_disk_usage_percent	avg_latency_milliseconds	avg_memory_usage_percent	warning_proportion	error_proportion
75.6	36.465	571	55.6	0.007813	0.000000
44.0	42.320	559	57.5	0.004673	0.004673

Figure 4. Example of 2 detected anomalies

From inspecting the values for those 30 detected anomalies, my conclusion was that most of them seemed like potential anomalies but a few of them didn't show any indication of errors but instead that there was no customer using the microservice at that particular time. This seems in fact to be an anomaly, but it is likely that not all users of the model will want to alert in those cases. In that case, ignoring those kinds of anomalies must be enabled by further development of the model or implementing logical filters.

Further evaluation could be done by collecting more data on earlier anomalies that have been deemed as clear cases where an alert should have been triggered. Then that data could be used in evaluating the model with more certainty.

Most of the evaluation will need to be done by the users, most often IT engineers, who will, after investigating the anomaly, give feedback if they concluded that a specific alert was valuable or not. The *contamination* could then be adjusted to set the alerting threshold as wanted. In general though, it can be said that if customers were significantly impacted by the anomaly then an alert would be considered valuable.

The method and code used in this assignment can with no or little adjustment be used for alerting in other microservices as well. New data for a new microservice needs to be collected and the Isolation Forest model should be trained on that data.

5 Conclusions

The EDA showed a low correlation between most of the variables.

PCA showed that 5 principal components would explain a little more than 90% of the variance. Using the PCA output as input when creating the Isolation Forest model was judged to not be advantageous. Instead the original data was used.

The Isolation Forest model that was developed was successfully used to find anomalies in the original dataset. Plans for how to further evaluate, develop and use the model were described.

The use of multivariate analysis and modeling using machine learning, in this case with the Isolation Forest algorithm, seems highly applicable for the use case of alerting for IT systems. The developed model seems promising for solving the defined problems and reaching the aim of the project assignment.

References

1. scikit-learn documentation. (n.d.). sklearn.preprocessing.StandardScaler. scikit-learn. Retrieved October 5, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html

2. www.claude.ai, 2024-09-21

3. www.claude.ai, 2024-10-09

APPENDIX 1: Representation of the dataset

This appendix gives a more complete representation of the dataset. As the actual used data contains sensitive information, the below data is made up data but has the same format as the actual used data and similar values.

timestamp	warning_proportion	error_proportion	avg_cpu_usage_percent	avg_memory_usage_percent	avg_disk_usage_percent	avg_latency_milliseconds
2024-10-03 13:00:00	0.0110	0.0060	32	32	44	129
2024-10-03 13:01:00	0.0010	0.0020	20	35	47	250
2024-10-03 13:02:00	0.0000	0.0001	4	11	35	70

APPENDIX 2: CODE USED FOR DATA ANALYSIS AND MODEL DEVELOPMENT

This appendix contains the code that was used to perform the data analysis. The code was written in the Python programming language.

```
import time

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.ensemble import IsolationForest

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA


################ DATA IMPORT SECTION ##########################################

dataset = pd.read_csv('FINAL_DATASET.csv')


############# DATA PREPROCESSING SECTION ######################################

dataset = dataset.dropna()


# Debug: Print the first few rows of the merged DataFrame

print("Merged Data:")

print(dataset.head())


csv_file = 'dataset.csv'

dataset.to_csv(csv_file, index=False)


# Check if the merged DataFrame is empty

if dataset.empty:

    raise ValueError("Merged DataFrame is empty. Check the 'timestamp' values in the CSV files.")
```

```
# Prepare the features

features = ['warning_proportion', 'error_proportion', 'avg_cpu_usage_percent', 'avg_memory_usage_percent',
'avg_latency_milliseconds', 'avg_disk_usage_percent']

X = dataset[features]


# Check if the feature DataFrame is empty

if X.empty:

   raise ValueError("Feature DataFrame is empty. Ensure the features exist in the merged DataFrame.")


############ EXPLORATORY DATA ANALYSIS SECTION ###################################


# Calculate the correlation matrix

correlation_matrix = X.corr()


plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Matrix')

plt.xticks(rotation=15, ha='right', fontsize=8)  # Adjust the fontsize as needed

plt.yticks(fontsize=8)  # Adjust the fontsize as needed

plt.show()


# Plot pair plot to show relationships between variables

sns.pairplot(X)

plt.suptitle('Pair Plot of Features', y=1.02)

plt.show()


################ PCA SECTION ######################################################


# Standardize the features

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

```
# Perform PCA
pca = PCA()
pca.fit(X_scaled)

# Calculate cumulative explained variance ratio
cumulative_variance_ratio = np.cumsum(pca.explained_variance_ratio_)

# Plot explained variance ratio
plt.figure(figsize=(10, 5))
plt.plot(range(1, len(cumulative_variance_ratio) + 1), cumulative_variance_ratio, 'bo-')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('Explained Variance Ratio vs. Number of Components')
plt.grid(True)
plt.show()

# Select number of components (e.g., 90% explained variance)
n_components = np.argmax(cumulative_variance_ratio >= 0.9) + 1
print(f"Number of components explaining 90% variance: {n_components}")

# Transform data using selected number of components
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X_scaled)

print(f"X_pca top 5 rows: {X_pca[:5]}")
print(f"X top 5 rows: {X[:5]}")
```

```
############ ANOMALY DETECTION SECTION ####################################

# Create and fit the Isolation Forest model
contamination = 0.005  # Adjust this based on the expected anomaly rate
model = IsolationForest(contamination=contamination, random_state=42)
model.fit(X)

# Predict anomalies

# Start the timer
start_time = time.time()
# Perform the prediction
anomaly_labels = model.predict(X)
# End the timer
end_time = time.time()
# Calculate the elapsed time
elapsed_time = end_time - start_time

# Print the elapsed time
print(f"Time taken for model.predict(X): {elapsed_time:.4f} seconds")
dataset['is_anomaly'] = anomaly_labels

# Function to detect anomalies in new data
def detect_anomalies(new_data):
    new_data_scaled = scaler.transform(new_data[features])
    # new_data_pca = pca.transform(new_data_scaled)
    predictions = model.predict(new_data_scaled)
    return predictions

print("            RESULTS")
print("-----------------------------------")
print("Anomalies detected:")
```

```
# Drop the 'timestamp' column and print the anomalies
anomalies = dataset[dataset['is_anomaly'] == -1].drop(columns=['timestamp'])
print(anomalies)


# Example usage: To use the model for new data:
# new_data = pd.DataFrame(...)  # New data with the same features
# new_anomalies = detect_anomalies(new_data)
```

MÄLARDALEN UNIVERSITY
SWEDEN

P.O. Box 883, SE-721 23 Västerås, Sweden **Phone: +46 21 101 300**
P.O. Box 325, SE-631 05 Eskilstuna, Sweden **Phone: +46 16 153 600**
E-mail: info@mdh.se **Webb:** www.mdh.se