Contents

[Problem identification 2](#_Toc64038464)

[Problem Statement 2](#_Toc64038465)

[Context 2](#_Toc64038466)

[Criteria for Success 2](#_Toc64038467)

[Scope of Solution Space 2](#_Toc64038468)

[Constraints 2](#_Toc64038469)

[Stakeholders 3](#_Toc64038470)

[Dataset Description 3](#_Toc64038471)

[Customer Transaction Data 3](#_Toc64038472)

[Customer Demographics 3](#_Toc64038473)

[Item Data 4](#_Toc64038474)

[Coupon Item Mapping 4](#_Toc64038475)

[Campaign Data 5](#_Toc64038476)

[Coupon Data 5](#_Toc64038477)

[Data Wrangling 5](#_Toc64038478)

[EDA 6](#_Toc64038479)

[Univariate Analysis 6](#_Toc64038480)

[Bivariate Analysis 10](#_Toc64038481)

[Feature Engineering 12](#_Toc64038482)

[Modeling 12](#_Toc64038483)

[Hyperparameter Tuning 12](#_Toc64038484)

[Feature Importance 13](#_Toc64038485)

# Problem identification

## Problem Statement

Given an IT department’s operational data for multiple software machines, is it possible to predict run time anomalies to investigate potential failures and prevent downtimes before they happen

## Context

The dataset consists of metrics measured from the operating system and from WebLogic Server monitoring beans for 200 operational machines. We want to apply machine learning to detect anomalies and prevent failures which can then be used by software companies for valuable cost savings.

## Criteria for Success

* Using machine learning, we are successfully able to categorize data as regular and anomalous
* Set reliable thresholds to predict anomalies

## Scope of Solution Space

Analysis is restricted to data from part 67.

## Constraints

* Data only available for 2.5-month span
* Subject matter expertise required to understand features
* Potentially many highly correlated features

## Stakeholders

* VP Information Technology
* Engineering Director
* VP Finance

# Dataset Description

The data is taken from Kaggle and consists of 200 datasets representing separate software machines. We will be looking at machine 67, with the following columns:

## Customer Transaction Data

Transaction data for all customers for duration of campaigns in the train data. It consists of 235 columns with information like:

* Active connections
* DB connection activity
* Connection delay
* Daemon thread count
* Failed wait for connections
* Heap usage activity
* Heap committed activity
* Memory space activity
* Physical mem activity
* Rel. unavailable connections
* Reserve request
* Stuck threads
* Successful wait for connections
* Swap activity
* System CPU
* Thread CPU time
* Thread User time
* Total thread count

# Data Wrangling

1. We start by dropping all columns that do not contain any unique value
2. We change column names to a more readable format.
3. We change the data type of ‘timestamp’ column from string to datetime.
4. Lastly, we drop columns with correlation of greater than 0.5 or less than -0.5.

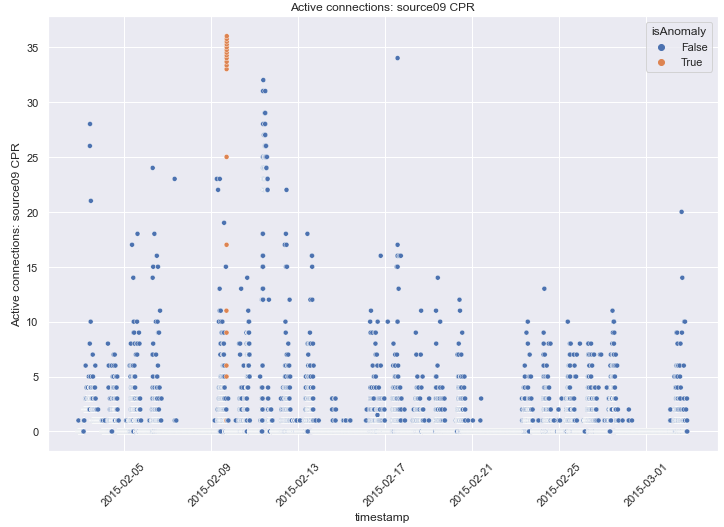
We end up with 43 columns/variables df1and save this dataset in our folder.

# EDA

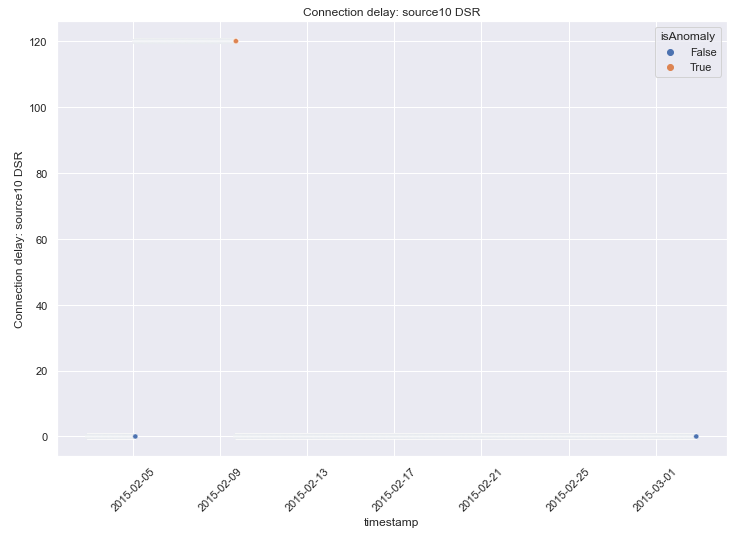
## Scatter Plots

We looked at scatter plots of features against timestamp and below are some interesting insights:

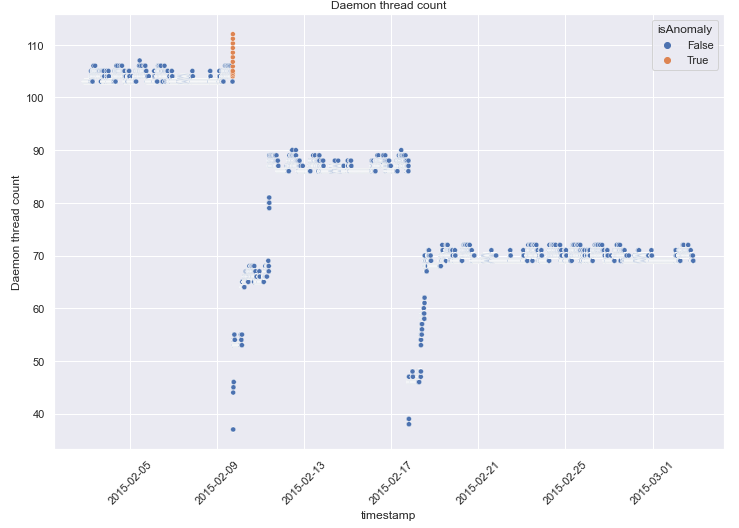
### Active connections: source09 CPR:



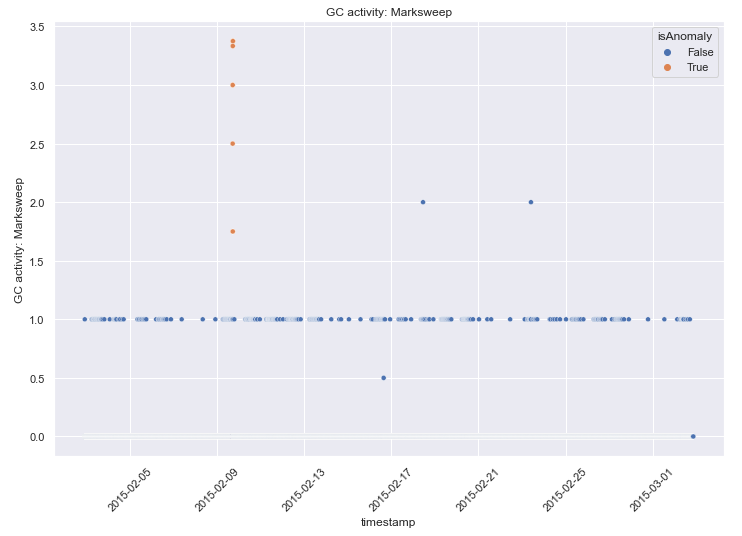
### Connection delay: source10 DSR:



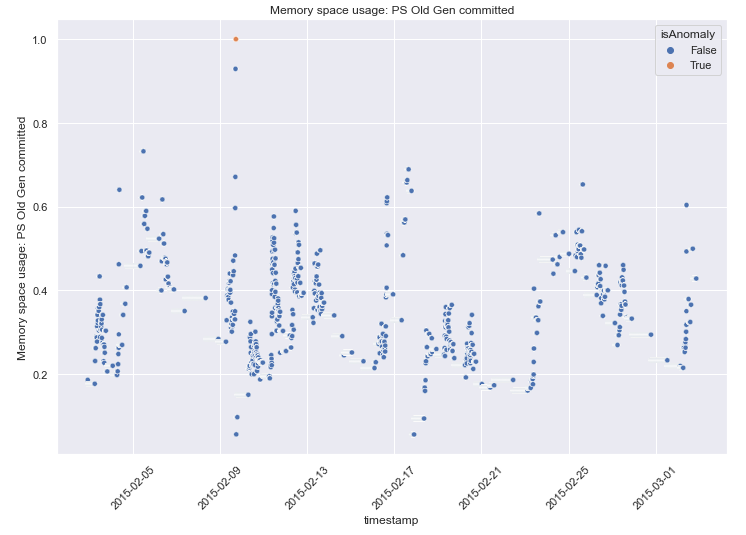
### Daemon thread count:



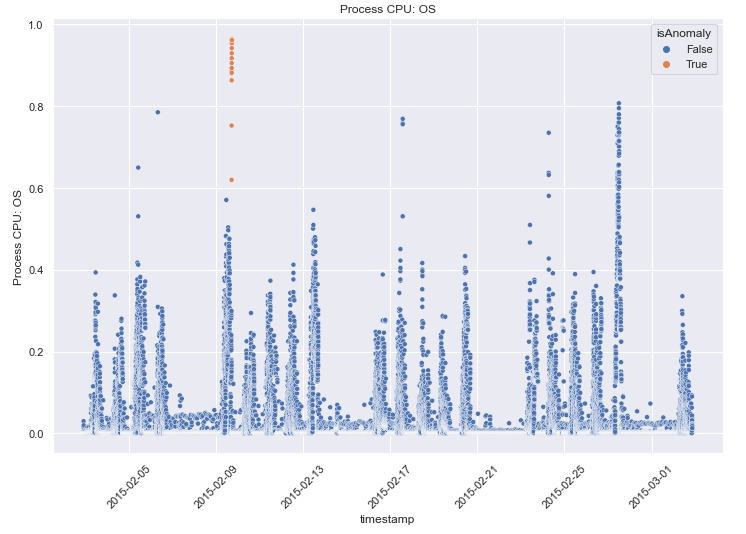
### GC activity: Marksweep:



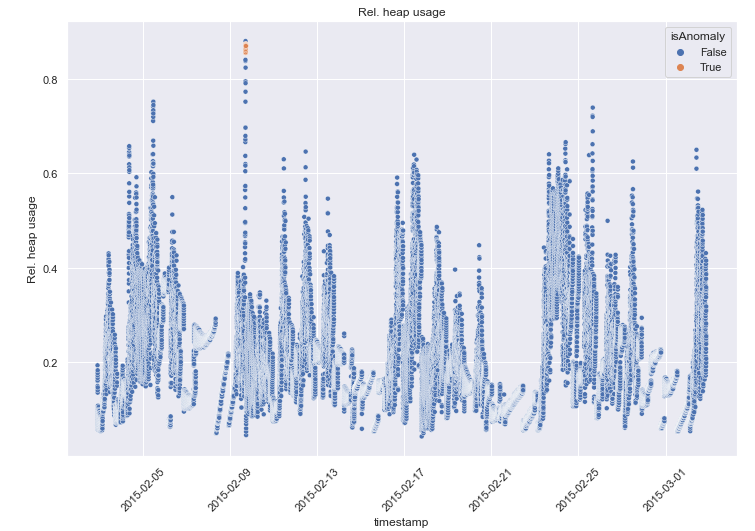
### Memory space usage: PS Old Gen committed:



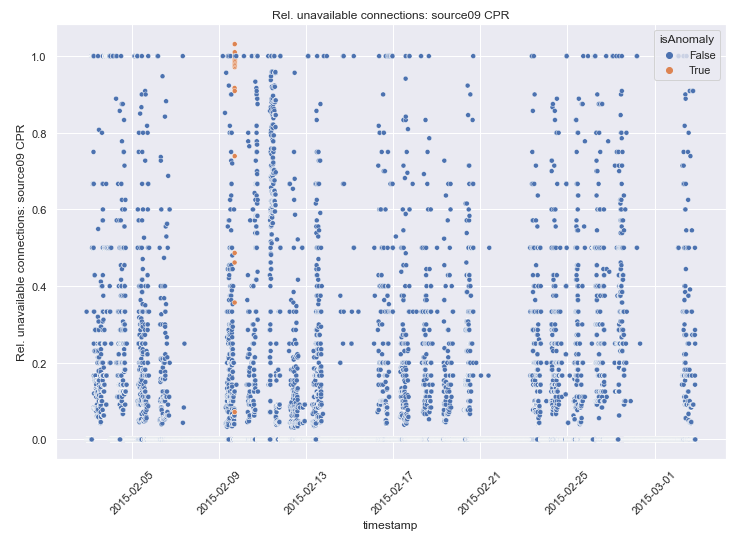
### Process CPU: OS:



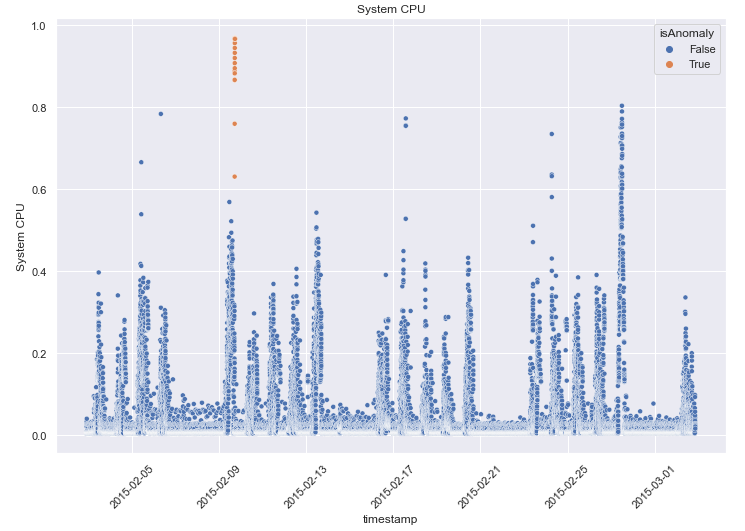
### Rel. heap usage:



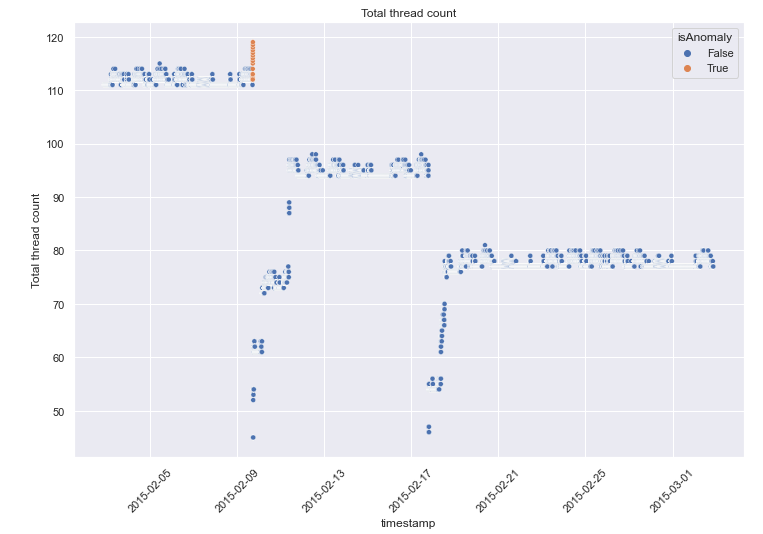
### Rel. unavailable connections: source09 CPR:



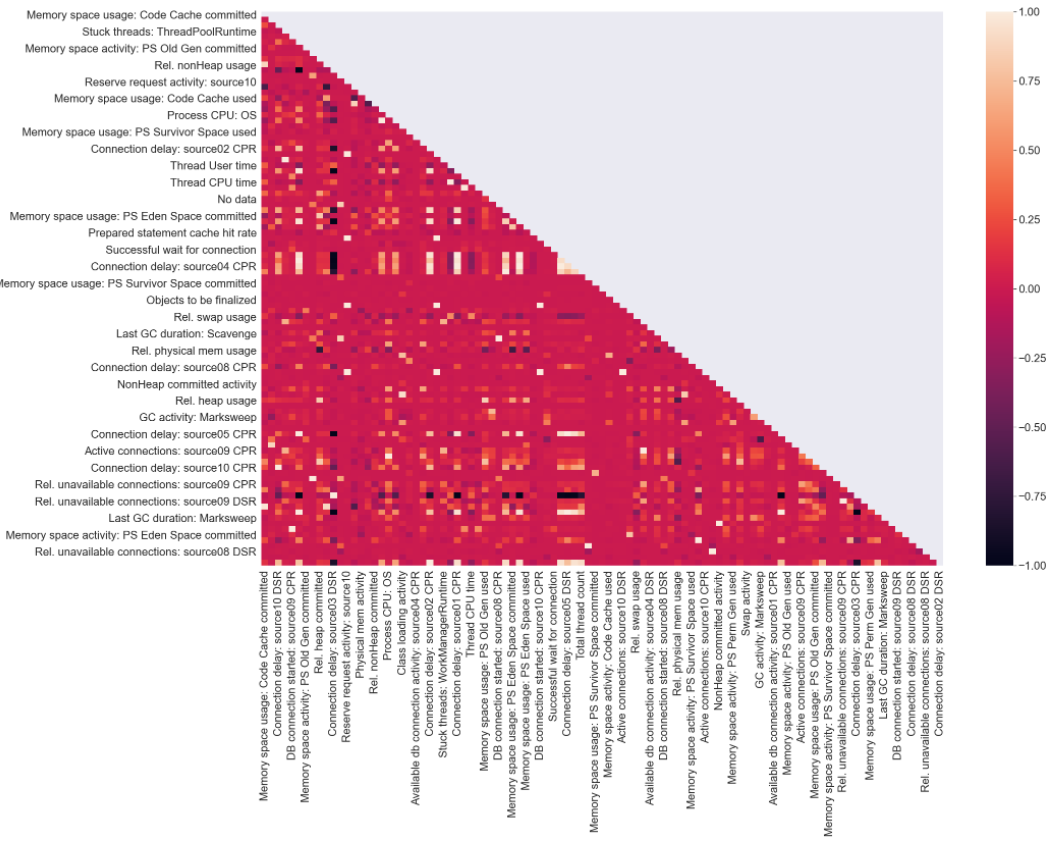
### System CPU:



### Total thread count:



## Correlation Matrix



From the correlation matrix it is clear that there are multiple features with a moderate to very high degree of correlation with each other.

# Modeling

We started by separating the dataset into train and test sets. We checked multicollinearity between all features and removed redundant columns.

We then fitted the test data to three models and compared their performances. Our metric of choice was AUC-ROC curve. Here are our calculations:

|  |  |
| --- | --- |
| Model | Area under ROC curve |
| Random Forest | 1 |
| Logistic Regression | 0.57 |
| XGB Classifier | 0.94 |

We decide to go with the XG boost model and proceed with tuning the model hyperparameters.

# Hyperparameter Tuning

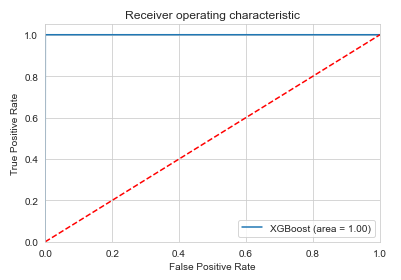
We selected the following parameter grid to use with GridSearchCV:

|  |  |
| --- | --- |
| eta | 0.01, 0.05, 0.1 |
| max\_depth | 9, 10, 11 |
| scale\_pos\_weight | 49, 50, 51 |
| max\_delta\_step | 4, 6, 8 |
| grow\_policy | depthwise, losswise |

This gave us a best score of 0.86 for the following parameters:

|  |  |
| --- | --- |
| eta | 0.1 |
| max\_depth | 10 |
| scale\_pos\_weight | 51 |
| max\_delta\_step | 4 |
| grow\_policy | depthwise |

We used these hyperparameters in our XGB Classifier model, fit it to our training data and predict the test data labels. We get a roc\_auc score of 0.77:

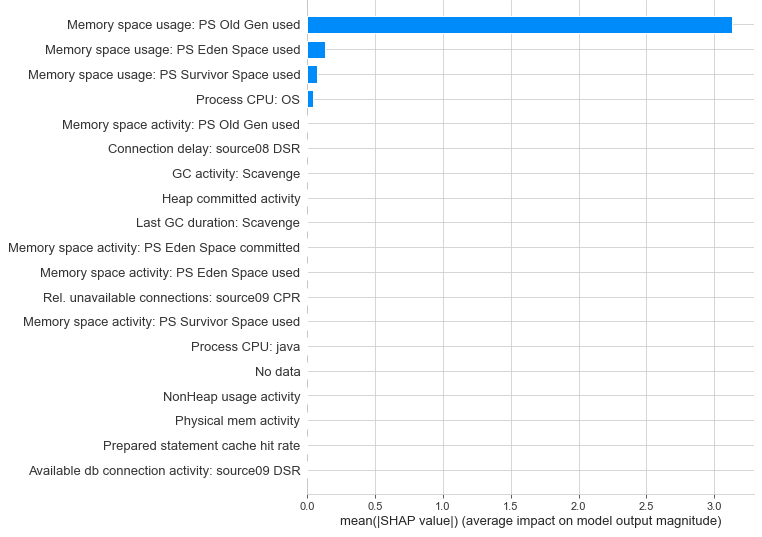


This is the confusion matrix we obtained:



# Feature Importance

At the end, we performed feature importance analysis using the SHAP module:



The graph above show us that the most important features that contribute to anomaly detection are:

1. Memory space usage: PS Old Gen used
2. Memory space usage: PS Eden Space used
3. Memory space usage: PS Survivor Space used
4. Process CPU: OS

# Feature Aggregation

Finally, we aggregated the four most important features to determine the cutoff points. For doing this, we first calculated the mean of ach of the four features from the training set, and then multiplied them with 0.75 to calculate the threshold. We obtained the following cutoff limits:

|  |  |
| --- | --- |
| Memory space usage: PS Old Gen used | 0.165 |
| Memory space usage: PS Eden Space used | 0.101 |
| Memory space usage: PS Survivor Space used | 0.43 |
| Process CPU: OS | 0.028 |

# Conclusion

The biggest indicators of anomalous behavior in software operational data are ‘Memory space usage: PS Old Gen used’, ‘Memory space usage: PS Eden Space used’, ‘Memory space usage: PS Survivor Space used’, ‘Process CPU: OS’. The software will most likely fail soon after the system approaches these threshold values.

## Recommendation

To minimize future failures, we recommend implementation of an alert mechanism to be implemented that notifies stakeholders as soon as the cutoff limit of any of the 4 most important features is reached. The stakeholders can then troubleshoot to determine the underlying cause and prevent operational failures.

## Future Scope

We recommend continuing to record this data to refine the threshold limits to get more reliable values. We also recommend applying this analysis to remaining software machines for additional cost savings.