TUM Data Innovation Lab
Munich Data Science Institute (MDSI)
Technical University of Munich
&
BMW GROUP

Final report of project:

Project: Prediction and Clustering Critical Suppliers

Authors	Robin Falter, Shen Hu, Dr. Jesus Ocariz, Hongfei Yan and Runyao Yu
Mentor(s)	B.Sc Sergio Correa, MSc.(TUM) Katarina Golubovic, MSc.(Doktorand) Ishansh Gupta, MSc.(TUM) Mykhailo Kulakov and MSc. Ashok Rupesh Kumar
Co-Mentor	MSc. Jun Wu (TUM Chair of Mathematical Statistics)
Project Lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

Aug 2022

Abstract

Machine learning and big data analytics optimize the intricate relationships in supply chain management, supporting end-to-end transparency management, rapid decision making, and maximizing enterprise value.
The focus of this project is to categorize and rate existing suppliers. Currently, a supplier's ranking is defined as a simple average of different features. Features are captured on a daily/weekly basis and include information about wrong deliveries, backlogs, and missing parts. Our goal is to use machine learning methods to optimize supplier escalation and support and improve the process of decision-making.
The project is organized in the following steps. First, compare and identify the key features that define the suppliers. Second, a machine learning classification model is trained with the selected essential features. Finally, the suppliers are classified through the model, and the manual ratings are compared with the machine learning recommended ratings to improve decision-making. The project is divided into two main parts. Initially, an end-to-end pipeline is built on Jupiter Notebook to train and optimize the model more efficiently and faster. The best model is then selected, and the entire end-to-end process is deployed to Palantir foundry, BMW's in-house big data platform.

Contents

1 Introduction

1.1 Mentors' affiliation: Think Tank BMW Group

Think Tank is part of BMW's Supply Chain Digitization. Their goal is to deliver assurance by making sure that their plants have enough required parts to produce cars on time by analyzing and reporting critical suppliers daily within the BMW Group.

1.2 Problem definition

Exacerbated by the COVID-19 pandemic, agility and flexibility are keys. However, predicting and managing the supply chain has become more challenging due to the flow complexity and the increased market volatility. For that reason, supply chain management solutions based on artificial intelligence (AI) have the potential to tackle current challenges in a very agile and flexible way, thanks to AI's ability to analyze vast volumes of data.

1.3 Goals of the project

During the ideation phase, three main objectives were defined for the project.

1. Research supplier classification and do exploratory data analysis (EDA)
2. Develop a model that predicts critical suppliers from the available information.
3. Implement a model in BMW's digital structure via Palantir Foundry.

1.4 Structure of the project

In Chapter 2 we will introduce the raw data and the data processing process. Later, in Chapter 3 we will describe the different models and approaches. In Chapter 4 we will show the experiments and results of evaluation and their deployment in Palantir Foundry. Finally, in Chapter 5 we state the final remarks and conclusions derived from the work of this project.

2 Description of the Datasets

2.1 Data Acquisition

The following is a specific description of the known datasets:

- **Backlog**: happens when supplier doesn't deliver parts on time.
- **Missing Parts**: parts that were unavailable during the car's production. The vehicle is incomplete and can therefore not be shipped. Critical KPI that should be reduced as much as possible.
- **Special transport** contains information about special transport that are caused by backlog or urgent need for certain parts. Special transport usually represents delivery of parts by plane.
- **Wrong Deliveries**: happens when suppliers send material parts but the parts are wrong, or have a wrong label. Or when label of box is right but part label is not, etc.
- **Escalation**: History about suppliers escalations.
- **LPKM**: Grade of the supplier for different kind of categories - backlog, wrong delivery etc. Grade is updated monthly.
- **Supplier**: Dataset includes cities and countries.
- **Events**: Dataset with different weather hazards that had a protentional impact on the supplier.

2.2 Preprocessing

Next, we describe the main steps we follow in cleaning the data and preparing them to be helpful in training machine learning models.

Initial merging of files

The backlog came in different files since some of them were very long or came from different time periods. Therefore, we have merged them.

Missing values cleaning

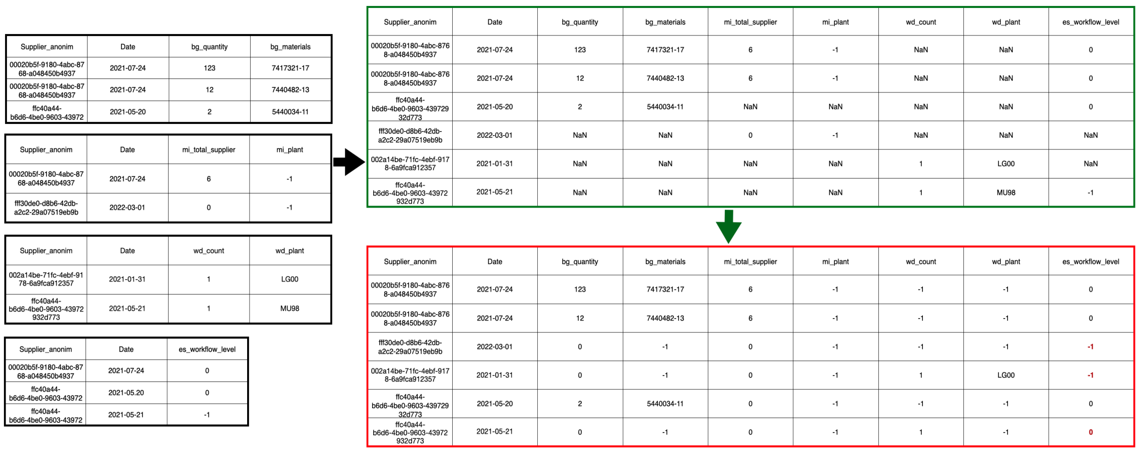
We have analyzed the proportion of missing values in each feature. Then, we removed the less relevant variables according to their importance and the percentage of missing values. We have deleted the entries where the supplier id is missing. If the supplier is non-critical, we set its escalation level to -1. If, for someday, there was no particular delivery problem, we set 0 and "-1" for the numerical and categorical variables, respectively.

Format of the content of the variables

There were different formats for the dates and different labels for the factories among the different files, and we solved these issues by changing the dates to a standard format (year-month-day) and the labels of the factories to belong to the same dictionary. Moreover, we also ensured that numerical features had type int and categorical ones had type object.

Final merging

We have four different DataFrames, one for each type of delivery problem (backlog, missing parts, wrong deliveries, and special transports). Using the date and the supplier id, we merged them into only one. Note that we can have more than one entry for the same supplier and date (usually in backlog), and we solve this issue by keeping as many entries as needed (see Figure 1). Furthermore, we also ensure that the escalation level is the same as the previous day unless a workflow of escalation appears.



Supplier_anonim	Date	bg_quantity	bg_materials
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	123	7417321-17
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	12	7440482-13
ffc40a44-b6d6-4be0-9603-43972	2021-05-20	2	5440034-11

Supplier_anonim	Date	mi_total_supplier	mi_plant
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	6	-1
fff30de0-d8b6-42db-a2c2-29a07519eb9b	2022-03-01	0	-1

Supplier_anonim	Date	wd_count	wd_plant
002a14be-71fc-4ebf-9178-6a9fca912357	2021-01-31	1	LG00
ffc40a44-b6d6-4be0-9603-43972932d773	2021-05-21	1	MU98

Supplier_anonim	Date	es_workflow_level
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	0
ffc40a44-b6d6-4be0-9603-43972	2021-05.20	0
ffc40a44-b6d6-4be0-9603-43972	2021-05-21	-1

Supplier_anonim	Date	bg_quantity	bg_materials	mi_total_supplier	mi_plant	wd_count	wd_plant	es_workflow_level
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	123	7417321-17	6	-1	NaN	NaN	0
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	12	7440482-13	6	-1	NaN	NaN	0
ffc40a44-b6d6-4be0-9603-43972932d773	2021-05-20	2	5440034-11	NaN	NaN	NaN	NaN	0
fff30de0-d8b6-42db-a2c2-29a07519eb9b	2022-03-01	NaN	NaN	0	-1	NaN	NaN	NaN
002a14be-71fc-4ebf-9178-6a9fca912357	2021-01-31	NaN	NaN	NaN	NaN	1	LG00	NaN
ffc40a44-b6d6-4be0-9603-43972932d773	2021-05-21	NaN	NaN	NaN	NaN	1	MU98	-1

Supplier_anonim	Date	bg_quantity	bg_materials	mi_total_supplier	mi_plant	wd_count	wd_plant	es_workflow_level
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	123	7417321-17	6	-1	-1	-1	0
00020b5f-9180-4abc-8768-a048450b4937	2021-07-24	12	7440482-13	6	-1	-1	-1	0
fff30de0-d8b6-42db-a2c2-29a07519eb9b	2022-03-01	0	-1	0	-1	-1	-1	-1
002a14be-71fc-4ebf-9178-6a9fca912357	2021-01-31	0	-1	0	-1	1	LG00	-1
ffc40a44-b6d6-4be0-9603-43972932d773	2021-05-20	2	5440034-11	0	-1	-1	-1	0
ffc40a44-b6d6-4be0-9603-43972932d773	2021-05-21	0	-1	0	-1	1	-1	0

Figure 1: The process of merging the different DataFrames

2.3 Anomaly Detection

The method we chose is isolation forest introduced in 2008 [14] because it has linear time complexity, and thus it will help to deal fast with the large dataset. Moreover, the algorithm can be computed in parallel using clusters.

Anomalies are supposed to be very few and pretty different from the other data. Consequently, the idea is that they could appear in the first splittings of a binary tree. (see Figure 2).
The isolation forest consists of two steps: training and computing the anomaly score.

Training Isolation Forest

In the training phase, several Isolation Trees (iTrees) are constructed. The construction is pretty similar to the decision trees where the splitting is random. More precisely, we select a feature as the starting node from a batch of samples and an intermediate value

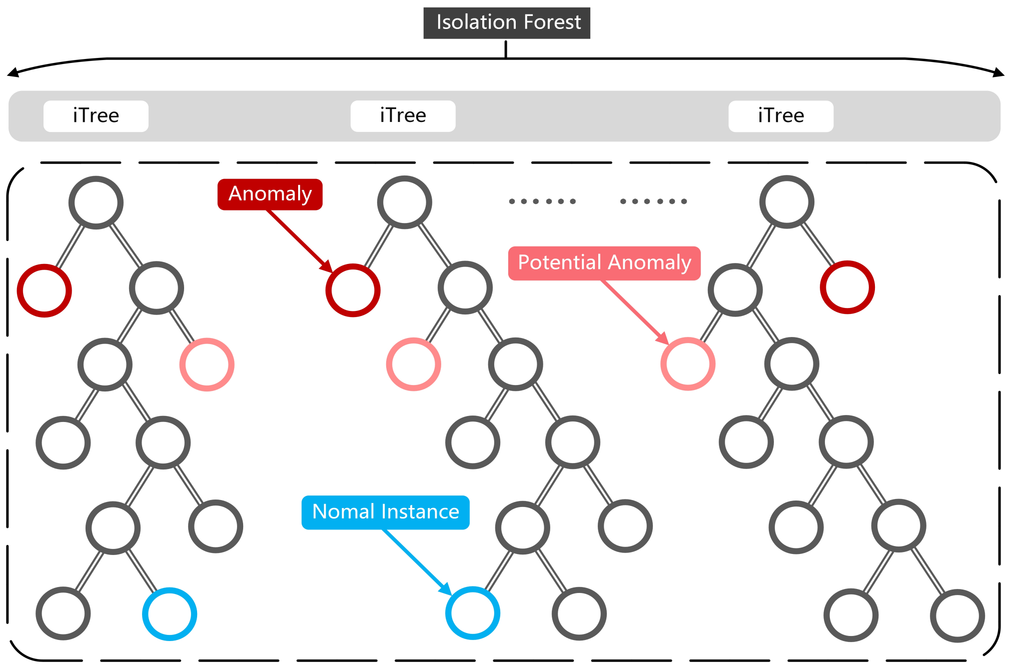


Figure 2: Isolation Forest

between the maximum and minimum values of the feature will be randomly selected. The data in the sample that is smaller than this value will be drawn to the left branch, and the data that is greater than or equal to this value will be drawn to the right branch. Then, the above steps will be repeated for data from both branches until the data cannot be subdivided or the iTree reaches the maximum depth.

Computing the anomaly score

Since the structure of iTrees is equivalent to Binary Search Trees (BST), the anomaly score is the same as the unsuccessful searches in BST. The anomaly score of the data x in a sample of size m is defined by:

$$S(x) := 2^{-\frac{E(h(x))}{c(m)}},$$

where $E(h(x))$ is the average path length to x across all tress and $c(m)$ is the average path length to every terminal code, which is further computed as:

$$c(m) := 2\left(\log(m-1) + \gamma\right) - \frac{2(m-1)}{m},$$

where γ is the Euler-Mascheroni constant (the limiting difference between the harmonic series and the natural logarithm).

The definition clearly shows that the score is between 0 and 1. The anomalies are the ones where their expected path length is going to be small and, consequently, their score is going to be above 0.5, and the closer it gets to 1, the more likely it is an anomaly. On the other hand, data with a score between 0 and 0.5 is considered normal data.

2.4 NLP Feature Extraction

In order to utilize the event data, it first needs to be represented as categorical features. A popular method for NLP Embedding called TF-IDF was used, which assigns scores to texts based on the frequency of appearance of words. After applying TF-IDF embedding, the texts were embedded into vectors.
Afterward, the embedded texts, i.e. vectors, were put into the clustering model: Mini-Batch K-Means. Next, we used the elbow's method (look for a flat part) for the sum of squared distances and the silhouette score (a metric that ranges from 0 to 1, and 1 denotes the most precise pattern) to find a good compromise about the number of clusters. Figure 3 shows the silhouette score and the sum of squared distances dependent on the number of clusters. The blue curve plateaus between 10 and 20 clusters. Every choice in this region would be valid. The sum of squared distances peaks at 18 clusters within that range, so we decided to go with this value.

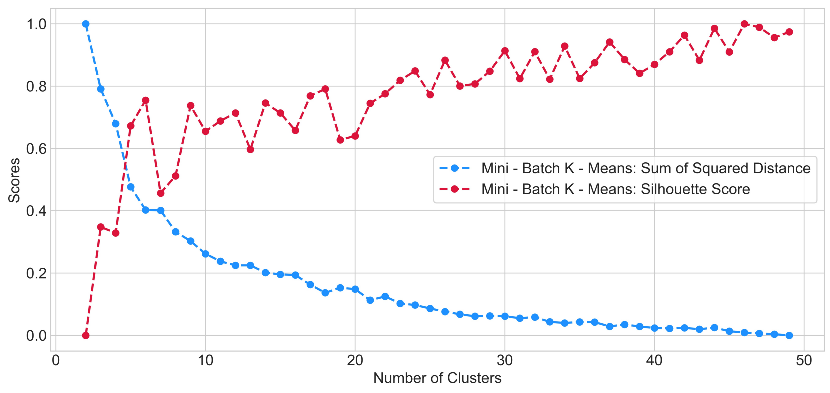


Figure 3: Isolation Forest

After selecting k as 18, Figure 4 shows a clear separation after using PCA or TSNE to visualize different high-dimensional vectors, which indicates that this is a good clustering. Next, each vector was assigned a label based on the clustering. We used the supplier-ID as the index to add each supplier's cluster label (from 0 to 17). In other words, we created a new column so-called NLP feature, which is categorically varied from 0 to 17.

2.5 Encoding

Categorical variables are a big challenge for Machine Learning algorithms. Since most of them accept only numerical values as inputs. In particular, most features in this project are categorical, so we need to transform the categories into numbers to use them in the model.
One-Hot Encoding: This is a common encoding strategy, which expands every feature (typically a column) into several columns, depending on the number of categories of the feature. For each data entry, only one of those columns would be one, and the rest would be zero (thus the name).
Label Encoding: This approach is the most straightforward and most intuitive. It merely involves converting each category of a feature into a number. As long as the same

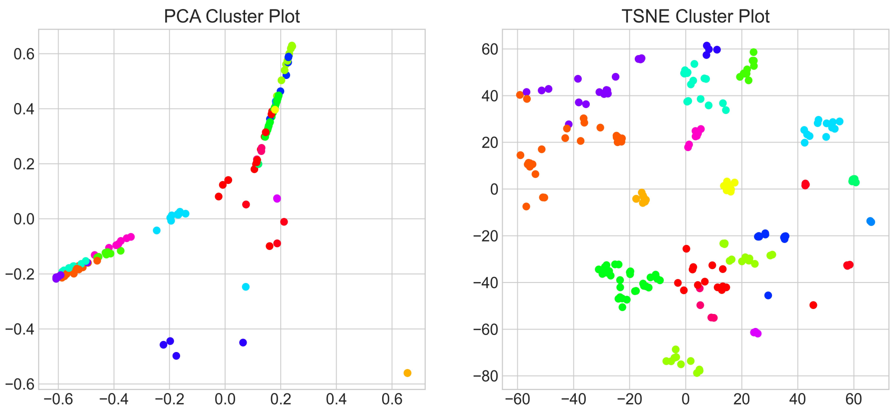


Figure 4: Dimension Reduction

category of a feature is turned into the same number, these numbers can be arbitrary. This could be potentially problematic as it introduces ordinal meanings to every categorical feature.

Target Encoding: This is an excellent alternative to one-hot encoding. It also ensures that the same category of a feature would be turned into the same number but removes the arbitrariness of the values: the number depicts (to some extent) the frequency of this feature category leading to a positive label. Nevertheless, if a category appears very seldom in the data, this would quickly lead to over-fitting. Therefore a smoothing strategy would usually be preferred for this encoding strategy.

We may try all the aforementioned encoding strategies during experiments, but after a brief analysis, one-hot encoding quickly turns out to be clearly infeasible:

- With one-hot encoding, a sparse matrix will be created, inflating the dataset's dimensions, and the model quickly falls victim to the infamous Curse of Dimensionality. This is amplified when the feature has too many categories, and most of them will be useless for the prediction, which is our case.
- Not only does one-hot encoding bring classification quality problems, but it also brings performance issues. The oversized data frame would easily overload any reasonably-sized memory, let alone the model's run time.

Therefore, only label encoding and target encoding will be discussed during later experiments.

2.6 Data Visualization

Features

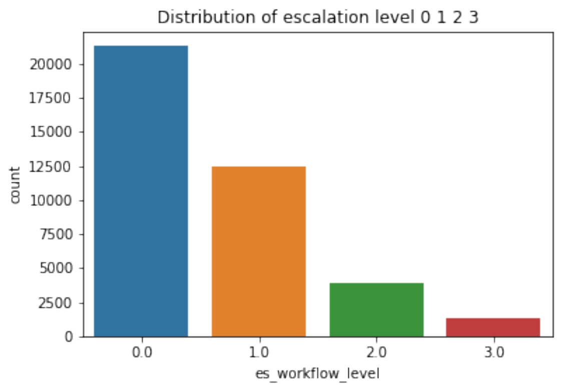
Table 1 shows the important features and data types.

Features	Types
supplier id anonim	object
bl material	object
bl plant	object
bl quantity	int64
date	object
mi plant	object
mi part number	object
mi product group	object
mi reason code	object
mi total supplier	int64
wd count	int64
wd plant	object
sp cause	object
sp plant	object
sp grund	object
nlp featrue	int64
es workflow level	int64

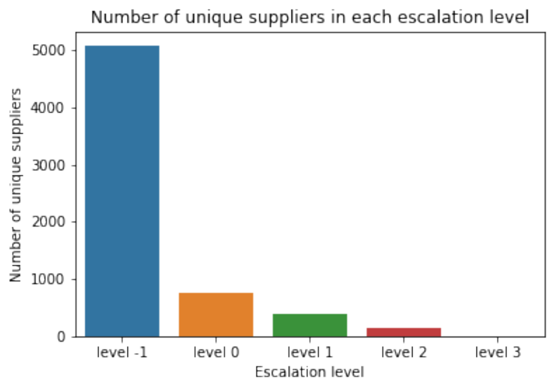
Table 1: Features of the final dataset

Imbalance problem

Most suppliers are not escalated, and suppliers in escalation level 3 are particularly few. Furthermore, the distribution of escalation levels in the entire dataset is very unbalanced, as can be seen in figure 5



(a) Data with different levels of escalation



(b) Number of unique suppliers in each escalation level

Figure 5: Distribution of escalation level

No correlation between numerical features

Figure 6 shows that the correlation of numerical features is very weak. In addition to autocorrelation, the highest correlation is only 0.066 between mi total supplier and bl quantity.

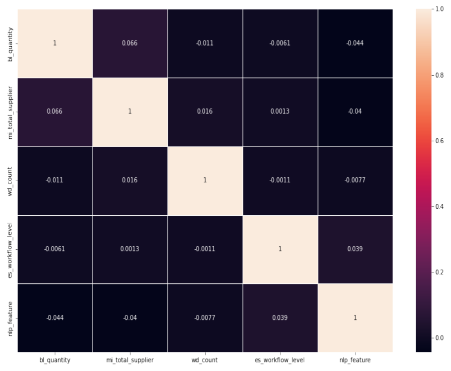
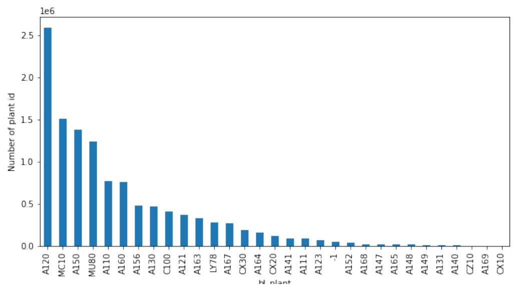


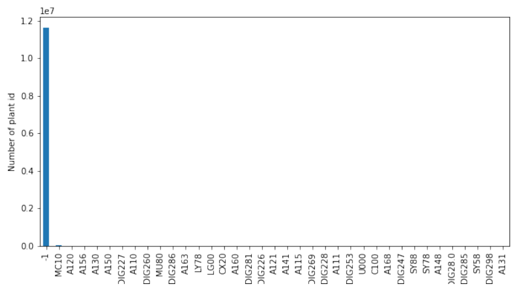
Figure 6: Numerical feature to numerical feature correlation

Analysis based on categorical features:

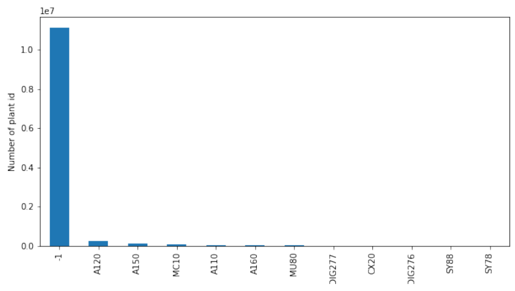
We show the frequency of the appearance of plants among the different types of delivery problems. Recall that - 1 indicates missing information.



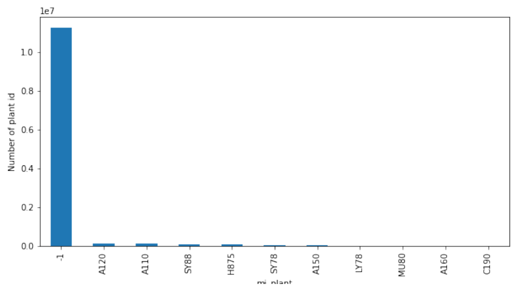
(a) Frequency of occurrence in backlog



(b) Frequency of occurrence in wrong delivery



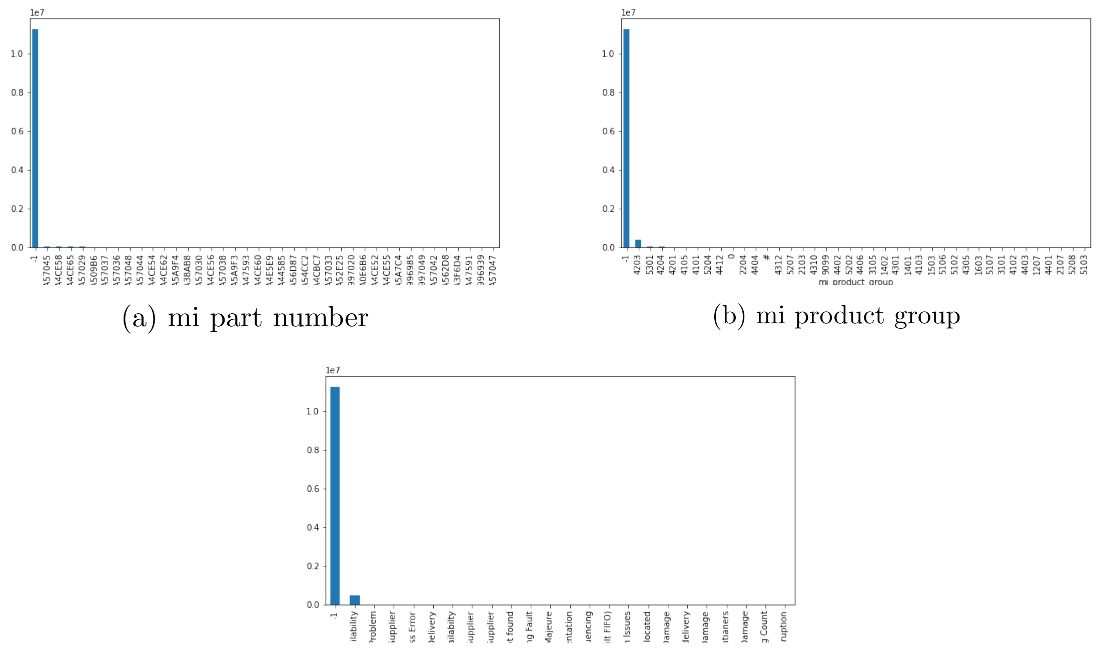
(c) Frequency of occurrence in special transport



(d) Frequency of occurrence in missing parts

Figure 7: Distribution of plant id

Figure 8 and 9 are a visualization of the other features in the missing parts and special transports respectively. It is obvious that they are very sparse.



(a) mi part number

(b) mi product group

(c) mi reason code

Figure 8: Additional information in missing parts

3 Methodologies and Approaches

We choose the following four models to conduct experiments such as Random Forest, Logistic Regression, SVM and XGBoost. in this section, we introduce these classifiers.

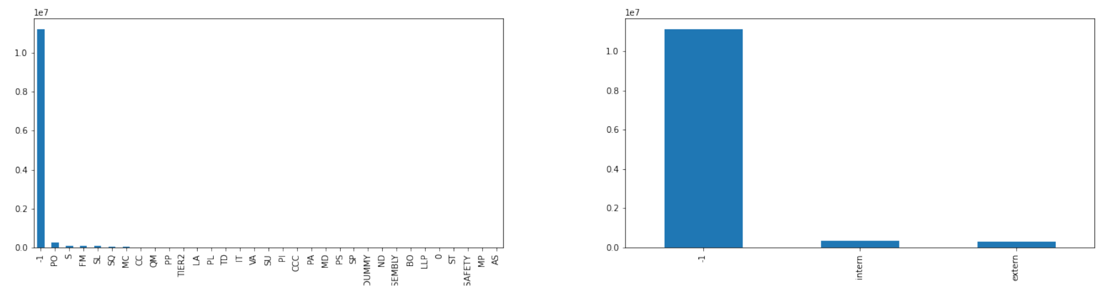
3.1 Random Forest

Random Forest Classifier is one of the most revolutionary and essential classification algorithms. It consists of a set of decision trees with various node-splitting rules constructed from random samples. Random Forest employs Bootstrap Aggregating (also known as Bagging) and builds decision trees based on a majority vote in classification, as can be seen in Figure 10 .

Bagging is a machine learning ensemble meta-algorithm to enhance the accuracy and robustness of statistical machine learning algorithms. It lessens variance and aids in preventing overfitting. Precisely, Bagging creates a random sample with replacement from the original sample with sample size N and replicates B times such that there will be B Bootstrap samples in total. Next, it evaluates the mean-variance with sum variance in B set Bootstrap as following:

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{i=1}^{B} \hat{f}^{i}(x)\,.$$

In the classification setting, an entropy-like measurement is usually taken to reduce the amount of entropy at each branch such that the best branch split can be obtained. The Gini Index [9] is a commonly used metric.



(a) sp grund (b) sp cause

Figure 9: Additional information in special transport

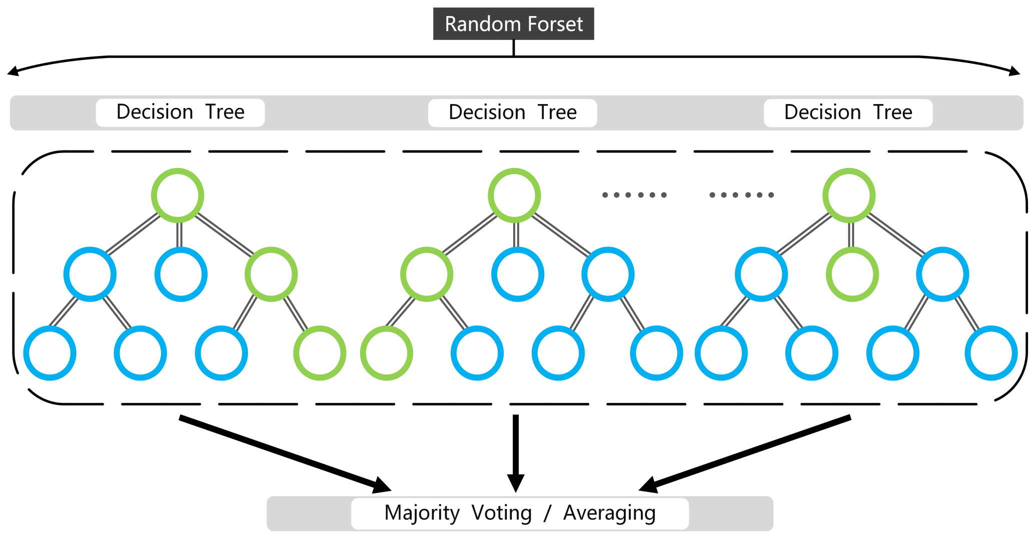


Figure 10: Random Forest

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right) ,$$

where the percentage of observations from the k-th class that were made in the m-th region is shown by $\hat{p}_{mk}$. The Gini Index can be thought of as an variance indicator. In other words, the model will receive greater misclassification results, if it obtains a bigger variance. Therefore, the Gini Index with lower values produces the better classification.

Some advantages of Random Forest are the following:

- It can handle very high-dimensional data without dimensionality reduction and feature selection.
- It is highly interpretable and can determine the importance of features and the interaction between different features.
- It is not prone to overfitting due to the usage of the Bagging technique.

- The convergence speed is relatively fast with the time complexity of $O(B^*(N^*D^*logN))$, where B describes the number of trees, N the number of samples and D the number of features considered at every split. It is possible to run the algorithm in parallel, improving convergence speed.

However, Random Forest has been proven to be easily overfitting with the noisy dataset. Thus, the **outlier removal** is **critical** for this algorithm.

3.2 Logistic Regression

Logistic Regression is a prevalent machine learning approach used for classification tasks because its output can be interpreted as a probability, i.e., a continuous number between 0 and 1. Then, we obtain a classification algorithm by taking a threshold (usually 0.5).

In this section, we aim to apply an isolation forest to remove the outliers such that we could have a classifier that can make more objective decisions.

$$\hat{y} = \frac{1}{1+e^{-z}},$$

where z can be replaced by

$$z = \sum_{i=1}^{N} \omega_i x_i + b\,,$$

in which ω_i stands for the weights of features x_i, and b represent the bias or intercept, which can be treated as an error calibration.

Since the sigmoid function is not linear, squaring it would let the solver become non-convex, which results in finding an improper local minimum. Thus, instead of using Mean Square Error (MSE), Logistic Regression utilizes the Cross-Entropy Loss, which is defined for the binary problem as:

$$J(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})\,.$$

Next, Gradient Descent is employed to minimize the loss function, equal to maximizing the probability we desire to have. After applying the chain rule, the derivative can be obtained as

$$\frac{\partial J}{\partial \omega_1} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \omega_1}\,.$$

Additionally, we have

$$\frac{\partial J}{\partial \hat{y}} = \frac{\partial(-y\log(\hat{y}) - (1-y)\log(1-\hat{y}))}{\partial \hat{y}} = \frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}}\,,$$

$$\frac{\partial \hat{y}}{\partial z} = \frac{\partial \frac{1}{1-e^{-z}}}{\partial z} = \frac{e^{-z}}{(1-e^{-z})^2} = \hat{y}(1-\hat{y})\,,$$

and

$$\frac{\partial z}{\partial \omega_1} = x_1 \, .$$

Therefore, the final derivative is

$$\frac{\partial J}{\partial \omega_1} = \left(\frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}} \right) \hat{y}(1-\hat{y})x_1 = (\hat{y} - y)x_1 \, .$$

The algorithm will repeat it for every ω_i such that optimal direction of the gradient will be informed:

$$\omega_i = \omega_i - \frac{\partial C}{\partial \omega_i} \, .$$

The advantage of Logistic Regression is that it computes probability scores for observations, and the structure is relatively simple and has excellent interpretability. The time complexity of Logistic Regression is $O(N{*}D)$. Therefore, it is suitable to be utilized for the low-dimensional dataset. The drawback is that the transformation of non-numerical features is necessary. Thus, it does not work well with many categorical features.

3.3 Support Vector Machines (SVM)

Support Vector Machines is a supervised machine learning algorithm that can be used for regression and classification. The algorithm aims to find a hyperplane that maximizes the margin between two classes. The margin m is defined as

$$m = \frac{2}{||\mathbf{w}||}$$

where w is the weight vector of the hyperplane.

$$y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b) \geq 1 - \xi_i.$$

ξ can be seen as a slack variable that allows samples to violate the margin and relaxes the constraint of the optimization problem. The optimal hyperplane can be found by maximizing the margin, which is equivalent to minimizing the norm of the weight vector.

$$\min \; f_0(\boldsymbol{w}, b, \xi) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N} \xi_i$$
$$\text{subject to } y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_i} + b) - 1 + \xi_i \geq 0$$

The solution to this quadratic programming problem is determined by the maximum of the lagrange dual function, which gives a lower bound estimate of the optimal value of the original optimization problem. The dual function can be written as

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x_i}, \boldsymbol{x_j}).$$

The function k is a kernel function that maps the input to a nonlinear feature space where the data is linearly separable. The kernel function is easier to evaluate than the dot product of the individual basis functions, which reduces the computational complexity of support vector machines. This is also well known as kernel trick. Examples for popular kernel functions are Polynomial, Gaussian and RBF kernels. [17] The runtime complexity of support vector machines depends on the used kernel and is higher for functions with nonlinear transformations. The training time scales at least quadratically with the number of samples. [18]

3.4 XGBoost

XGBoost is a recent machine learning tool [6], which gained popularity after being used in the winning solution of the Higgs Machine Learning Challenge in Kaggle.

Given a training set $\{(\mathbf{x}_i, y_i)\}_{1\leq i\leq N}$ where $\mathbf{x}_i \in \mathbf{R}^m$ (m-features) and $y_i \in \mathbf{R}$ a tree ensemble model uses K additive functions to predict the output:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=i}^{K} f_k(\mathbf{x}_i),$$

where $f_k \in \mathcal{F}$ is the space of regression trees (also known as CART).

The learning process comes from the minimization of the following function:

$$\mathcal{L}(\phi) = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where l is a differentiable convex loss function and Ω is a regularization term that penalizes the complexity of the model to avoid overfitting.

The idea is to train the model in an additive way, i.e, use the previous prediction for the next one. Let $\hat{y}_i^(t)$ be the prediction for the i-instance at the t-th iteration. Then, the idea is to greedly add f_t that most improves the model:

$$\mathcal{L}(\phi) = \sum_{i=1}^{N} l\left(y_i, \hat{y}_i^{(t)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t)$$

Moreover, we use the second Taylor expansion to approximate the value of l. Then, we only need to sum up the gradient and second-order gradient statistics from one iteration to the next.

The main advantage is that it is a simple and agile method that has performed very well in several machine learning competitions.

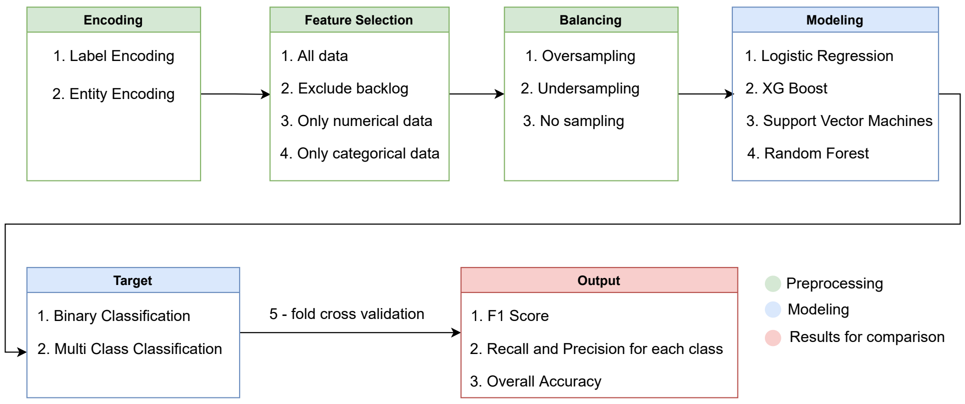


Figure 11: Overall pipeline structure for both the binary and the multiclass classification stage

4 Experiments and Results

4.1 Structure of the Pipeline

As discussed in Section 2.6, our dataset demonstrates a huge imbalance between the escalation class (level 0 - 3) and the non-escalation class (level -1). On the other hand, the distribution among different levels of escalation shows a relatively balanced pattern. Considering this, we have decided to structure our workflow as a two-stage process so that the different distribution patterns can be addressed separately. In the first stage, a classifier would focus only on distinguishing escalated cases from the rest, regardless of their escalation level, which makes it a binary classifier. In the second stage, a multiclass classifier would classify the escalated cases into different escalation levels (level 0 - 3).
Even though the two stages above address different problems and can be implemented with entirely different types of classifiers and under different settings, we may still structure the same pipeline for both stages, as they share the same input-data structure.
This pipeline structure is shown in Figure 11, which consists of four different steps (the first four boxes in the figure are the steps where the data flows through, whilst the last two boxes are both tunable options which can be specified to fit the aforementioned binary and multiclass classification stages). During the encoding step, categorical features were encoded into numerical features using assigned strategies. This step enables the usage of many machine learning models that cannot deal with categorical features, and the encoding strategy also plays an essential role in the final performance of the pipeline. During the feature selection step, a subset of the features can be selected to be experimented on. This step can be used to experiment on the importance of different features or discard certain unimportant ones to speed up the pipeline. The balancing step is where different balancing strategies are applied to deal with the prominent data imbalance issue in this project. The modeling step is where the classifier is chosen and trained, and the grid search for every classifier is also applied here. It is worth mentioning that further encoding, feature selection, balancing, or modeling strategies can be easily added to the pipeline in the corresponding steps without modifying the pipeline's result, making it much easier for further experimenting and training.

4.2 Election of Metrics

With the pipeline above at hand, it is critical to choose a matching metric so that different settings for the steps in the pipeline can be quantitatively compared and the best ones can be chosen for the final implementation in the pipeline.
Our project is a typical classification problem, to which there are already several quantitative metrics at our disposal. The most common ones are precision, recall, and their combination - f1 score. Their mathematical definitions are as follows:

$$Precision = \frac{TP}{TP + FP}$$

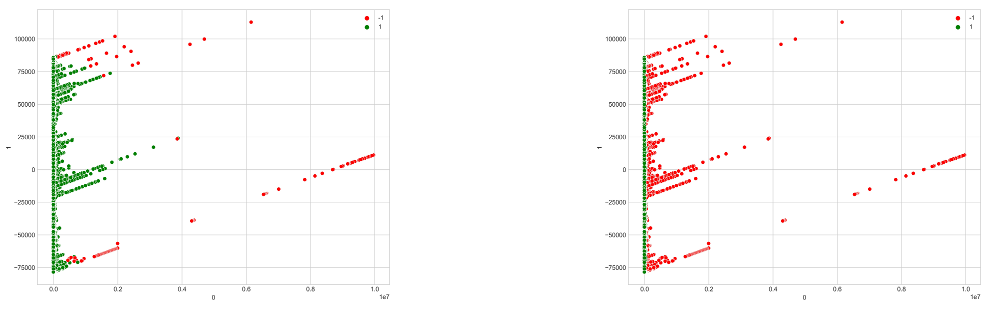
$$Recall = \frac{TP}{TP + FN}$$

$$f1 = \frac{1}{Precision^{-1} + Recall^{-1}}$$

In the above formula, TP, FP, TN, and FN are short for the number of data entries that are correctly classified as a positive case, that are wrongly classified as a positive case, that is correctly classified as a negative case, and that is wrongly classified as positive case, respectively. Precision measures how accurate the optimistic predictions of the model are. Recall measures the percentage of positive data the model can specify from the whole testing dataset. F1 score takes care of both aspects and is specified as the harmonic mean of precision and recall.
For general cases, the f1 score is an excellent first choice because 100% precision can be easily reached by always making negative predictions, and it is the same with recall by always making positive predictions. Nevertheless, in our case, when performing the first stage classification, the dataset demonstrates a massive imbalance between the escalation class and the non-escalation class, as shown in Section 2.6. In this case, it would be challenging to construct a model that performs well on both precision and recall, which means the f1 score would always be very low and fails to distinguish the excellent model from the rest.
Considering the real-world meaning of this classification problem, recall would be the better choice over precision. In this case, high recall for the escalation class means the model is able to capture most of the suppliers that should be escalated, which is crucial since this enables the company to specify an escalation in an early stage and plan ahead so that the loss of money can be accordingly reduced. In the meanwhile, some amounts of false positives are considered as acceptable, as there would always be a human expert to decide whether or not to perform an escalation at the end of the pipeline. In this regard, as long as the recall for the non-escalation class is kept at a reasonable level, a model with good recall for the escalation class can be considered as a good one and put into deployment.
As for the second stage classification, the non-escalation class is not considered anymore, thus leading the target classes to be roughly balanced in terms of the order of magnitude of their sizes. In this regard, we use f1 score and the general accuracy over all classes to compare different models.

4.3 Anomaly detection

In this section, we aim to apply an isolation forest to remove the outliers such that we could have a classifier that can make more objective decisions.
As illustrated in Figure 12, the outliers are marked as red, and the standard data with green. By increasing the anomaly rate, more data (in red) will be considered anomalies. We repeat this experiment till the mainstream does not change and ensure the optimal anomaly rate.
Figure 12 shows the experiment of removing the outliers for class -1, which denotes for non-escalation level. If we choose the anomaly rate as 0.1%, the data points with red and green colors are somehow stacked together. After increasing the anomaly rate from 0.1% to 0.5%, most data points with the green color are accumulated on the y-axis. Keeping increasing the anomaly rate will lead to no change in the color distribution, which indicates that the optimal anomaly rate for this class should be chosen as 0.5%. Similarly, we repeated this procedure for other classes and obtained the optimal anomaly rate for each class.



(a) Anomaly Rate= 0.1%   (b) Anomaly Rate= 0.5%

Figure 12: Isolation Forest with different anomaly rates

4.4 Comparison of the Results

4.4.1 Binary Classification Stage

Regarding different steps of the pipeline, we have multiple choices for each. Firstly, for encoding, label encoding and target encoding are viable considering our dataset size. Secondly, for sampling strategies, random oversampling and random undersampling will be experimented with, and the case where no sampling is applied will also be considered for comparison. Thirdly, for the classifiers, random forest, XGBoost, SVM are considered, and even though logistic regression tends to be unsuitable for a classification problem with so many categorical features, we still experimented on it to use it as a baseline comparison. Lastly, for the essential features to use during training, it is ideal to use them as all the features after the preprocessing share a very low covariance. However, it is still interesting to experiment on a different subset of features so that we can have a rough idea of how different features contribute to the final classification decision. In this regard, four subsets of features are chosen for the feature selection step in the experiments:

Table 2: Binary stage pipeline performance comparison under different encoding, feature selection, balancing and classifier settings

Encoding Strategy	Feature Selection	Balancing Strategy	Target Label	Random Forest		Logistic Regression		SVM		XGBoost	
				Recall	F1 Score	Recall	F1 Score	Recall	F1 Score	Recall	F1 Score
label encoding	all features	random over-sampling	-1	1.00	1.00	0.77	0.87	0.74	0.81	0.96	0.98
			0	0.10	0.08	0.30	0.01	0.39	0.24	0.27	0.03
label encoding	all features	no sampling	-1	1.00	1.00	1.00	1.00	0.99	0.93	1.0	1.0
			0	0.08	0.11	0.00	0.00	0.02	0.04	0.1	0.18
label encoding	all features	random under-sampling	-1	0.62	0.77	0.96	0.98	0.86	0.87	0.67	0.8
			0	0.69	0.01	0.06	0.01	0.16	0.15	0.71	0.01
label encoding	exclude backlog	random over-sampling	-1	0.99	0.99	0.98	0.99	0.99	0.93	0.98	0.99
			0	0.13	0.04	0.11	0.03	0.05	0.08	0.13	0.04
label encoding	exclude backlog	no sampling	-1	1.00	1.00	1.00	1.00	1.00	0.93	1.00	1.00
			0	0.05	0.09	0.00	0.00	0.01	0.02	0.06	0.12
label encoding	exclude backlog	random under-sampling	-1	0.98	0.99	0.97	0.98	0.92	0.90	0.98	0.99
			0	0.13	0.13	0.06	0.01	0.09	0.11	0.13	0.03
label encoding	only numerical features	random over-sampling	-1	0.50	0.67	0.83	0.91	0.49	0.63	0.52	0.69
			0	0.63	0.01	0.21	0.01	0.59	0.23	0.58	0.01
label encoding	only numerical features	no sampling	-1	1.00	1.00	1.00	1.00	1.00	0.93	1.0	1.0
			0	0.02	0.04	0.00	0.00	0.00	0.00	0.03	0.06
label encoding	only numerical features	random under-sampling	-1	0.51	0.67	0.83	0.91	0.44	0.59	0.55	0.71
			0	0.59	0.01	0.21	0.01	0.61	0.22	0.57	0.01
label encoding	only categirical features	random over-sampling	-1	0.99	0.99	0.92	0.96	0.69	0.78	0.93	0.96
			0	0.09	0.03	0.98	0.06	0.43	0.24	0.29	0.02
label encoding	only categirical features	no sampling	-1	1.00	1.00	1.00	1.00	0.99	0.93	1.0	1.0
			0	0.06	0.09	0.00	0.00	0.01	0.01	0.03	0.06
label encoding	only categirical features	random under-sampling	-1	0.60	0.75	0.92	0.96	0.91	0.90	0.67	0.8
			0	0.63	0.01	0.96	0.06	0.12	0.14	0.71	0.01
target encoding	all features	random over-sampling	-1	1.00	1.00	0.77	0.87	0.76	0.83	0.98	0.99
			0	0.40	0.27	0.30	0.01	0.56	0.34	0.5	0.12
target encoding	all features	no sampling	-1	1.00	1.00	1.00	1.00	0.99	0.93	1.0	1.0
			0	0.28	0.34	0.00	0.00	0.09	0.16	0.15	0.24
target encoding	all features	random under-sampling	-1	0.97	0.99	0.96	0.98	0.66	0.77	0.93	0.96
			0	1.00	0.17	0.06	0.01	0.58	0.30	0.96	0.07
target encoding	exclude backlog	random over-sampling	-1	0.99	0.99	0.98	0.99	0.99	0.93	0.98	0.99
			0	0.13	0.04	0.11	0.03	0.05	0.08	0.13	0.04
target encoding	exclude backlog	no sampling	-1	1.00	1.00	1.00	1.00	1.00	0.93	1.0	1.0
			0	0.05	0.09	0.00	0.00	0.01	0.02	0.06	0.12
target encoding	exclude backlog	random under-sampling	-1	0.98	0.99	0.97	0.98	0.92	0.90	0.98	0.99
			0	0.13	0.03	0.06	0.01	0.09	0.11	0.13	0.03
target encoding	only numerical features	random over-sampling	-1	0.50	0.67	0.83	0.91	0.68	0.80	0.52	0.69
			0	0.63	0.01	0.21	0.01	0.86	0.42	0.58	0.01
target encoding	only numerical features	no sampling	-1	1.00	1.00	1.00	1.00	1.00	0.93	1.0	1.0
			0	0.02	0.04	0.00	0.00	0.04	0.08	0.03	0.06
target encoding	only numerical features	random under-sampling	-1	0.51	0.67	0.83	0.91	0.56	0.70	0.55	0.71
			0	0.59	0.01	0.21	0.01	0.81	0.33	0.57	0.01
target encoding	only categirical features	random over-sampling	-1	0.98	0.99	0.92	0.96	0.69	0.78	0.94	0.97
			0	0.77	0.21	0.98	0.06	0.43	0.24	0.74	0.07
target encoding	only categirical features	no sampling	-1	1.00	1.00	1.00	1.00	0.99	0.93	1.0	1.0
			0	0.25	0.35	0.00	0.00	0.01	0.01	0.08	0.14
target encoding	only categirical features	random under-sampling	-1	0.97	0.98	0.92	0.96	0.91	0.90	0.93	0.96
			0	1.00	0.17	0.96	0.06	0.12	0.14	0.96	0.07

All but backlog-related features, all categorical features, all numerical features, and all features (no exclusion).

With the above-mentioned different combinations, there are, in total, 96 different experiments to be conducted. Within each experiment, it is also necessary to do a grid search to find the best hyperparameters for each classifier under different settings. Considering the size of our dataset, it is not viable to perform every experiment with the whole dataset, which will not finish in days and will also easily result in an out-of-memory issue, especially for classifiers like SVM whose time complexity grows quadratically with the dataset size. Therefore, we performed the experiments on only stratified one-tenth of the whole dataset (for SVM, the size is further reduced to one-thousandth, as even one-tenth is not viable) and set the test-set size as 25%. The experiments' results will be used to analyze and specify the best pipeline combination, and the final test about the best setting will be performed afterward with the whole dataset.

The overall experiment results showed in Table 2.

As discussed in the previous subsection, in this binary classification stage, we will focus on recalls, especially the recall of class 0 (escalated classes, regardless of the escalation level). From Table 2, we can draw the following initial conclusions and inferences:

1. Regarding encoding, target encoding depicts much better performance compared to label encoding. This might be the result of label encoding's drawback by nature:

Table 3: Test result of the binary stage pipeline under the best setting discovered in previous experiments

Target Label	Recall	F1-score
-1	0.84	0.91
0	0.90	0.03

it successfully encoded all categorical values at the cost of introducing groundless ordinal meanings for every value of each feature. On the contrary, the ordinal meanings in target encoding are related to the frequencies of each feature value that leads to a positive label, which proved to beneficial in our experiments.

2. Regarding using a subset of features, the classification result is better by using all features as expected. In addition, if backlog-related features were excluded, the classification scores dropped dramatically, providing all other settings stayed the same. This means backlog features are crucial, which coincides with the observation that backlog features are our dataset's most frequent non-empty features. Furthermore, experiments with only categorical features perform better than the ones with only numerical features. This proves that not only are there more categorical features than numerical features (10 vs. 3), but the categorical features are also closely related to supplier escalation.

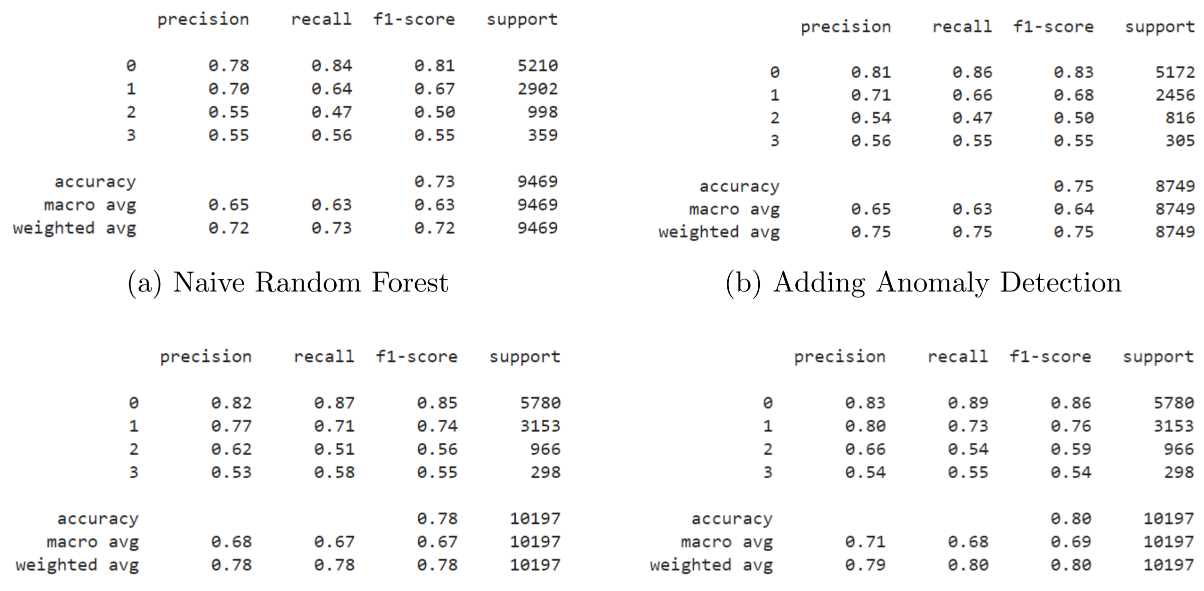
3. Regarding sampling strategy, it is clear that without sampling, the pipeline generally performs the worst due to the severe imbalance issue within the dataset. We initially expected random over-sampling and under-sampling results to be roughly similar. Nevertheless, random under-sampling turns out to have much better results. This again might be the result of the severe imbalance issue: if the pipeline over-samples the minor class (escalated suppliers) by brute force, the same data will be used for training about 300 times on average, which can lead to over-fitting.

4. Regarding the classifier, both random forest and XGBoost perform similarly as tree-based classifiers, with Random Forest being slightly better in this case. Logistic regression generally performs not as well as the former ones, probably due to its simple and not-fit-with-categorical-feature natures, and SVM performs poorly simply because of its high time complexity and therefore cannot utilize as much data as possible like the other three classifiers.

Following the discussions above, we implement the binary classification stage pipeline: encoding categorical features using target encoding, using all features, performing random under-sampling during training, and using the random forest as the classifier. After the deployment, the following test result in Table 3 are obtained by training on 75% and testing on the rest 25% of the whole dataset (stratified).
As can be seen, the test result drops slightly with the whole dataset but remains within a reasonable and acceptable region, as the misclassification rate for the non-escalation class remains reasonable (f1 score 0.91), and the pipeline can capture 90 percent of suppliers that should have been escalated.

4.4.2 Multiclass Classification Stage

We designed a two-stage classification process. Namely, the first stage aims to classify if the supplier should be escalated or not (Binary Classification). The second stage's goal is to classify which level the supplier should be escalated with (Multiclass Classification). Therefore, the data with class -1, i.e., the non-escalation class, were removed in this subsection.
The results of the multiclass classification problem are summarized in Figure 13. As can be seen, the test accuracy by using naive random forest was only 73%. After filtering the outliers, the recall for class 3 was decreased with 1%. However, the overall model performance was slightly improved to 75%. Afterwards, the NLP features were then added to each row by retrieving the supplier-ID. It can be noticed that the recall were all improved. The overall test accuracy increased to 78%, which indicates that the event data could be a big potential for improving the model's performance. Based on the anomaly detection and NLP features, we applied XGBoost with grid search. In the end, the model's overall performance was further improved to 80%. However, it should be noted that the recall for class 3 decreased from 58% to 55% after changing the reference model from Random Forest to XGBoost, even if the recall for other classes were improved.



```
              precision    recall  f1-score   support

           0       0.78      0.84      0.81      5210
           1       0.70      0.64      0.67      2902
           2       0.55      0.47      0.50       998
           3       0.55      0.56      0.55       359

    accuracy                           0.73      9469
   macro avg       0.65      0.63      0.63      9469
weighted avg       0.72      0.73      0.72      9469
```

(a) Naive Random Forest

```
              precision    recall  f1-score   support

           0       0.81      0.86      0.83      5172
           1       0.71      0.66      0.68      2456
           2       0.54      0.47      0.50       816
           3       0.56      0.55      0.55       305

    accuracy                           0.75      8749
   macro avg       0.65      0.63      0.64      8749
weighted avg       0.75      0.75      0.75      8749
```

(b) Adding Anomaly Detection

```
              precision    recall  f1-score   support

           0       0.82      0.87      0.85      5780
           1       0.77      0.71      0.74      3153
           2       0.62      0.51      0.56       966
           3       0.53      0.58      0.55       298

    accuracy                           0.78     10197
   macro avg       0.68      0.67      0.67     10197
weighted avg       0.78      0.78      0.78     10197
```

(c) Adding NLP feature

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      5780
           1       0.80      0.73      0.76      3153
           2       0.66      0.54      0.59       966
           3       0.54      0.55      0.54       298

    accuracy                           0.80     10197
   macro avg       0.71      0.68      0.69     10197
weighted avg       0.79      0.80      0.80     10197
```

(d) Grid search with XGBoost

Figure 13: Summary of results of the multiclass classification problem

4.5 Deployment in Palantir Foundry

Palantir Foundry is a large-scale data integration platform that BMW uses to manage its supply chain information. The platform provides functionality for storing, visualization, and transforming data with a focus on scaling and collaboration. Foundry keeps track of the relationship between the data with a functionality called data lineage, which gives access to intermediate processing steps. Files in foundry are organized into datasets. These files can be loaded and modified by transformation pipelines. Individual processing

steps of the transformation pipelines have to be defined inside a code repository, which also supports branching and versioning. Scripts can be written in Java, Python, SQL and Mesa. Foundry uses an internal API for loading and saving datafiles. [20] For transformations in Python, it is recommended to use the PySpark library, a Python API for Apache Spark. Apache Spark is optimized for large-scale distributed data processing and machine learning. [19]
The goal of the project is to deploy a working classification model inside Foundry. Figure 14 shows the structure of the used training pipeline. In the first step backlog, special transports, missing parts and wrong deliveries datasets are loaded. These datasets are combined with the escalation dataset to provide the model's training features. The label is the escalation level for each supplier. Two models are used for predicting escalated suppliers, whereas the first model is responsible for finding suppliers that should be escalated. The second model determines the exact level of escalation for the selected suppliers. The combined feature dataset is prepared to match both models. All escalation levels for the binary model are summarized into one class, resulting in a dataset that only contains escalated and non-escalated suppliers. For the multiclass classification model, all non-escalated suppliers are removed. The resulting dataset consists of 4 classes with escalation level from 0 to 3. The dataset is split into a train and test part with a ratio of 75/25. Both models use identical preprocessing pipelines. In the first step, all categorical variables are transformed into numerical values with label encoding. The encoded features are then passed to a vector assembler that combines the features into one single column. This processing step is required since the PySpark machine learning API only accepts one single column as feature input. For both binary and multiclass classification, a random forest classifier was selected. The hyperparamteres where choosen according to the results of section 4.4. Palantir enables the connection of different processing steps and thus enables the time-accurate retraining of the model with the use of schedules. Schedules execute the training pipeline at a certain timestep or after the occurrence of a specific event. With modeling objectives, Palantir provides internal functionality for training, storing, and deploying machine learning models. So the pipeline can be retrained with different parameters. All models are stored within one modeling objective and can be compared using metrics. For deployment the model with the best results can easily be selected and used for production. Table 4 depicts the performance of the binary and multiclass model on the test set. The metrics are calculated as the unweighted mean over all classes.
Inference requires a new feature dataset without escalation information. This dataset is created and then passed through the binary classification model, which decides whether a supplier should get escalated or not. The escalated suppliers are passed through the multiclass model in the second step to determine the exact escalation level. The inference is triggered daily or when the information in backlog, missing parts, wrong deliveries, or special transports changes. A parameter controls the number of recent days that are considered for creating the feature dataset. By default, a value of 30 is chosen. The dataset can contain multiple rows for one specific supplier. This is the case when the supplier has, for example, backlogs for multiple materials or backlogs at different timesteps within the chosen timeframe, which results in multiple predictions for that specific supplier. In order to make the model more robust against outliers, we calculate the mean over the output for each supplier. The results are then visualized and monitored in a dashboard, which provides additional functions for analyzing and filtering the predictions.

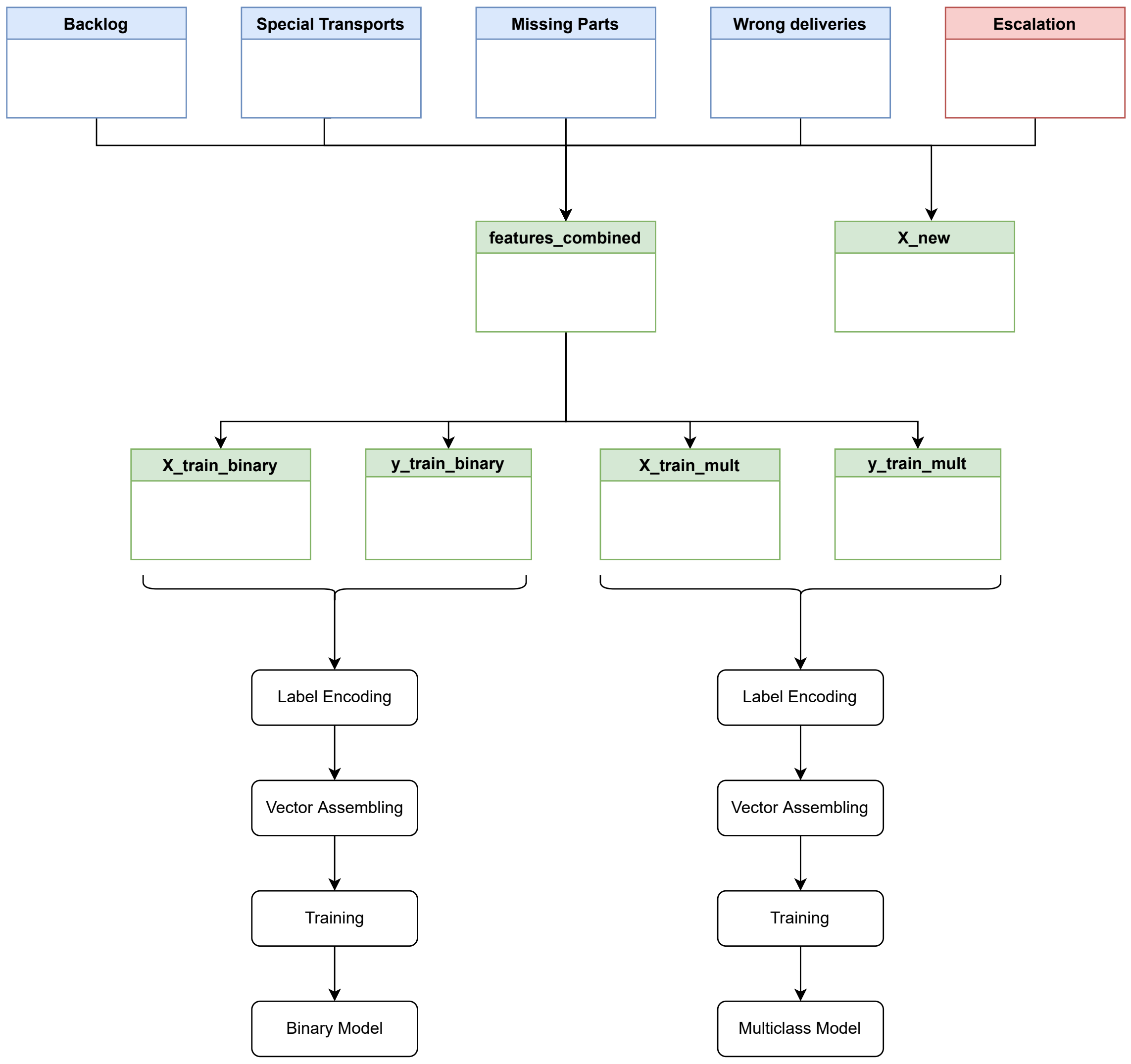


Figure 14: Training pipeline for the binary and multiclass classification models in Palantir foundry

Table 4: Test results of the binary and multiclass classification model trained within Palantir foundry

Model	Accuracy	Average Recall	Average Precision	Average F1-score
Binary	0.59	0.59	0.62	0.60
Multiclass	0.79	0.79	0.82	0.75

5 Conclusions

5.1 Business benefits

We remind that this project aims to understand and predict the critical suppliers of BMW. Whether a supplier is critical or not is currently based on the opinion of some experts. The model uses the dataset's information to predict the suppliers' escalation status. The predictions are not supposed to automatize the process but rather support the experts' decision-making. The benefit of the model is to ease the complex and challenging process of evaluating suppliers. This will lead to many implications in favor of the company: better choice of suppliers is linked with a reduction of factory downtime, which will increase the benefits.

Right now, the binary model has an accuracy of roughly 60%, leading to misclassifications for some suppliers. As the model has been optimized for recall, the misclassifications correspond to false positive suppliers. So right now, the model escalates to more suppliers than necessary. The additionally escalated suppliers can be removed from the dashboard. The random forest classifier outputs the probability for each class. A high probability for a class means that the model is more confident. For escalation, it is recommended only to escalate suppliers with a high probability. A threshold could be used to make the escalation more objective and reproducible. A supplier then gets escalated if the probability for the escalation class is larger than the threshold.

5.2 Evaluation of the deployment in Palantir

For the department, it was the first project that used Palantir foundry ml to create an end-to-end training and inference pipeline. Therefore we would like to briefly draw attention to the drawbacks and benefits of using foundry ml to develop machine learning algorithms. A summary of this evaluation can be found in 4. The biggest strength of Palantir is the seamless integration of data processing, model training, and inference. Predictions can be scheduled daily on new data and directly visualized in a dashboard to create value for the company. The biggest drawback is the slow development process. The foundry ml package is not fully matured yet and has several bugs that complicate the development process. Debugging is only possible on a small subset of the data, on which it is not possible to find errors that occur due to resource limitations. Model predictions are also not possible in preview mode, so for debugging and checking your code, you always have to start a whole building process. The building process checks the dependencies of your repository first and then has to wait for the cluster resources before it can start to run the script. Usually, it takes ten between minutes and two hours until a build throws an error message or finishes successfully. The long build process and limited debug possibilities considerably decrease development productivity. The lower productivity also leads to less exploration of different hyperparameters and preprocessing steps and, therefore, lower overall model quality. For future projects, it is recommended to use Palantir foundry for data preprocessing, inference and visualization. The model should be trained and tuned outside the platform to enable faster experimentation and higher quality.

Table 5: Major benefits and drawbacks of using foundry ml for creating end-to end machine learning pipelines

Category	**Benefits**	**Drawbacks**
Development	- Support for all conda packages	- Code usually only runs in Pyspark →useful third party ml-packages can not be used - Debugging only for a preview of data supported (10k rows) - Foundry ml not fully matured - Palantir support only partially helpful - Model can not predict in preview mode →debugging only over build process possible →each build takes c.a 20 minutes →no experimentation or hyperparameter tuning possible
Training	- Distributed system →training can be executed on multiple nodes	- Training in cluster much slower than on local device - Very large memory and computation requirements →GridSearch not possible due to out of memory issues
Deployment	- Fast deployment - Easy comparison of different models - Fast rollback (old model versions can be reused) - Integration of data processing, inference and visualisation	- Hard to customize dashboard

References

[1] A. Aamodt, and E. Plaza. *Case-based reasoning: Foundational issues, methodological variations, and system approaches.* AI communications 7.1 (1994): 39-59.

[2] R. C. Baker, and S. Talluri. *A closer look at the use of data envelopment analysis for technology selection.* Computers & Industrial Engineering 32.1 (1997): 101-108.

[3] S. Y. Balaman. *Decision-Making for Biomass-Based Production Chains: The Basic Concepts and Methodologies.* Academic Press, 2018.

[4] M. Braglia, and A. Petroni. *A quality assurance-oriented methodology for handling trade-offs in supplier selection.* International Journal of Physical Distribution & Logistics Management (2000).

[5] I. M. Cavalcante, et al. *A supervised machine learning approach to data-driven simulation of resilient supplier selection in digital manufacturing.* International Journal of Information Management 49 (2019): 86-97.

[6] T. Chen, and C. Guestrin. *Xgboost: A scalable tree boosting system.* Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.

[7] Z. Degraeve, E. Labro, and F. Roodhooft. *An evaluation of vendor selection models from a total cost of ownership perspective.* European journal of operational research 125.1 (2000): 34-58.

[8] L. De Boer, E. Labro P. and Morlacchi. *A review of methods supporting supplier selection.* European journal of purchasing & supply management 7.2 (2001): 75-89.

[9] J. L. Gastwirth. *The estimation of the Lorenz curve and Gini index.* The review of economics and statistics (1972): 306-316.

[10] K. Govindan, et al. *Multi criteria decision making approaches for green supplier evaluation and selection: a literature review.* Journal of cleaner production 98 (2015): 66-83.

[11] S. H. Ha, and , R. Krishnan. *A hybrid approach to supplier selection for the maintenance of a competitive supply chain.* Expert systems with applications 34.2 (2008): 1303-1311.

[12] N. Harale, S. Thomassey, and X. Zeng. *Supplier prediction in fashion industry using data mining technology.* 2019 International Conference on Industrial Engineering and Systems Management (IESM). IEEE, 2019.

[13] W. Ho, X. Xu, and K. D. Prasanta. *Multi-criteria decision making approaches for supplier evaluation and selection: A literature review.* European Journal of operational research 202.1 (2010): 16-24.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou. *Isolation forest.* 2008 eighth ieee international conference on data mining. IEEE, 2008. p. 413-422.

[15] G. Wang, et al. *Big data analytics in logistics and supply chain management: Certain investigations for research and applications.* International journal of production economics 176 (2016): 98-110.

[16] C. A. Weber, J.R. Current, and W. C. Benton . *Vendor selection criteria and methods.* European journal of operational research 50.1 (1991): 2-18.

[17] C. M. Bishop. *Pattern Recogniton and Machine Learning.* Springer (2006): p. 325-345.

[18] scikit-learn developers. *sklearn.svm.SVC* https://scikit-learn.org/stable/modules/generated/ sklearn.svm.SVC.html (accessed 21.07.2022)

[19] Sparkbyexamples. *Spark By Examples — Learn Spark Tutorial with Examples* https://sparkbyexamples.com/ (accessed 21.07.2022)

[20] Palantir Technologies Inc. *Code Repositories* https://impact.bmw.cloud/workspace/ documentation/product/transforms/overview (accessed 21.07.2022)

A Appendix

A.1 Distribution of the tasks

For evaluation's purposes, we specify the main roles of the different components of the team:

- Hongfei:

  - Worked on EDA at the beginning of the project (wrong delivery, special transport and supplier anonim).

  - Created time series and non-time series datasets and added nlp features(from Runyao) in non-time series dataset

  - Did multi-label classification (KNN, Random Forest, Decision Tree) experiments using datasets of different sizes and with/without PCA for the 2. Milestone meeting.

  - Did experiments of logistic regression based on Shen's training pipeline.

  - Did data preprocessing (data combination) and inference (data combination) in Palantir (Code conversion) and supported fixed bugs in these two parts

  - Supported Robin to fix Java Jar Packages bug

  - Did data analysis and visualization on combined dataset

  - Wrote the report (abstract, 2.3, 2.6 and 2.7).

- Jesus

  - Research the state of the art of supplier evaluation. Explore another alternatives of the problem using AHP.(Analytic Hierarchy Process).

  - EDA focusing on feature engineering: new variables such as the monthly factories affected and the type of delivery (national or overseas).

  - First steps towards a monthly time series that summarizes the delivery problems of a supplier and takes into account the tendency.

  - Implementing a K-Nearest Neighbourhood model to evaluate the right splitting of training and evaluation sets even without the access of a virtual machine.

  - Evaluation of the processes. Detection of the appearance of duplicates before the splitting in the training and evaluation datasets.

  - Writing and modifying parts of the documentation.

- Robin

  - Project Organisation (setting up GitLab, Notion and Miro)

  - Exploratory Data Analysis (EDA) for Backlog, Missing Parts and Wrong deliveries

  - Visualisation of high dimensional data with PCA, UMAP and T-SNE

  - Time-series based feature extraction with TS-FRESH (failed)

  - Escalation level classifier training utilizing support vector machines for binary and multiclass classification

- Deployment of the project in Palantir

- Converting sklearn pipeline to PySpark for distributed and fast code execution

- Inference, postprocessing and visualisation of the results in a dashboard

- Documentation support (Support Vector Machines, deployment in Palantir + creating figures)

- Daily meetings with BMW mentors on project updates

- Runyao

  - Exploratory Data Analysis (EDA) for Event Data Understanding

  - NLP Embedding Training, Clustering and Knowledge Management applying TF-IDF, Mini-Batch K-Means and Statistical Methods for Event Data

  - Anomaly Detection and Validation for Binary and Multiclass classification using Isolation Forest

  - Data Balancing applying CTGAN (Failed) and Random Over-/Undersampling

  - Escalation Level Classifier Training utilizing XGBoost and Random Forest with Grid Search for Binary and Multiclass Classification

  - Advanced Visualization for High Dimensional Data utilizing PCA, UMAP, and T-SNE for Validating Latent Pattern for Classification and NLP Clustering

  - Documentation Support (PowerPoint Creation, Report Writing: Methodologies and Plots for Random Forest, Logistic Regression, Anomaly Detection, and Multiclass classification, Visualization for NLP Clustering and Dimension Reduction)

- Shen

  - Exploratory Data Analysis for escalation and LPKM;

  - Preliminary time series model exploration (failed due to the sparsity of features over time)

  - Research and implementation of different encoding strategies (integrated in the pipeline later)

  - Design and implementation of a modularized training pipeline library and skeleton, which enables the team to perform experiments in a coordinated manner

  - Experimenting on different encoding, feature selection and balancing strategies using various classifiers; proposing the final modeling strategy (Focusing on the binary classification stage)

  - Joining in composing the documentation (encoding part of Chapter 2, majority of Chapter 4 (until 4.4.1))

A.2 Future research approaches

Supplier evaluation and prediction is a very active research field whose importance was highlighted due to the uncertain times we are living in nowadays. You can see some examples in the fashion industry [12], digital manufacturing [5], among others. The most relevant articles showing the state of the art until 2000 are: [16], [8] and [7]. This work was followed by [13] and [10] where 78 and 33 journals from this millennia are considered respectively. Finally, we would like to also cite this article that explains the complexity of this issue with the new Big Data techniques [15].

Most popular criteria considered by the recent articles for evaluating and selecting the most appropriate suppliers are: price, quality, delivery and others (see [11]). In this project, we have focused on the ones regarding the provided data, i.e. mostly delivery with a slight information about quality.

In [13], it is further detailed the different approaches to evaluate and choose suppliers and we are going to explain briefly because they can be used in the future.

DEA: Data Envelopment Analysis

DEA consists mainly in an optimization problem where we try to maximize the efficiency. Applied DEA based on the work of [2] and [4] is used to measure the efficiencies of alternative suppliers.

Mathematical Programming

This includes different mathematical algorithms to optimize a function (linear or non-linear) defined for continuum or integer values under certain type of constraints.

AHP: Analytical Hierarchy Process

It is widely used for analyzing complex decisions. Based on mathematics and psychology, it consists on the description of some key factors regarding the decision and their weights are computed from the subjective opinion of some experts.

For multi-criteria decision analysis, it is also used a more general form AHP denoted by ANP (Analytic Network Process).

CBR: Case-Based Reasoning

It is the process of solving new problems based on the solutions of similar past problems in philosophy and artificial intelligence [1].

Fuzzy Set Theory

A fuzzy set in mathematics is one way to formalize uncertain sets. This can be used in psychology and decision-making (see [3]).

Machine Learning

There are recent articles using machine learning techniques like the one we obtain fig 15. There, using python packages, they can preprocess their data (supplier evaluation forms) to use supervised machine learning to give some supplier selection.

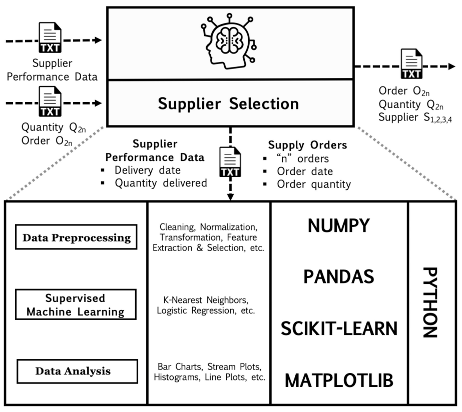


Figure 15: Example of ML in Supplier Selection. Source: [5]