# Automated Identification of Incorrectly Labelled Shipments using Machine Learning Models

MASTER THESIS

by

Vijaya Madasu

submitted to obtain the degree of

MASTER OF SCIENCE (M.SC.)

at

COLOGNE UNIVERSITY OF APPLIED SCIENCES
INSTITUTE OF COMPUTER AND COMMUNICATION TECHNOLOGY

Course of Studies

COMMUNICATION SYSTEMS AND NETWORKS

| | |
|---|---|
| First supervisor: | Prof. Dr. Uwe DETTMAR<br>Cologne University of Applied Sciences, Köln. |
| Second supervisor: | Mr. Bernhard KÖNSGEN<br>Data Scientist, DHL Group, Bonn. |

Cologne, September 2024

# Declaration of Authorship

I hereby declare that this master thesis was independently composed and authored by myself.

All content and ideas drawn directly or indirectly from external sources are indicated as such. All sources and materials that have been used are referred to in this thesis.

The thesis has not been submitted to any other examining body and has not been published.

Place, date

Vijaya Madasu

# Abstract

Currently, heavy packages weighing more than 10 or 20 kg are not separately labeled. Furthermore, labeling of heavy packages greater than 20 kg is not consistently implemented across all DHL products and systems. Additionally, there is currently no separate operational treatment of heavy packages during delivery. As part of the revision of the Postal Act, requirements for the labeling of heavy packages from business customers and private customers, as well as the appropriate handling of shipments over 20 kilograms, are formulated. In § 73, the requirements for labeling packages with increased weight between 10 and 20 kilograms, as well as labeling packages with high weight over 20 kilograms, are specified that shipments weighing more than 20kg require special handling during delivery (e.g. two-man handling or the use of a suitable means of transport such as a hand truck). The labeling must be done no later than the final processing step at a fixed location of DHL, so that it is present during delivery. This project aims to address this issue by developing a machine learning model to identify incorrectly labeled or unlabeled shipments based on different weights of the shipments both for all national shipments and international import shipments from private customers because DHL only uses scales in some parcel centers and for this reason the use of the algorithm is of great importance for the company in complying with the new postal law.

The project involves extensive data processing to prepare the shipment data for classification of shipments weighing more than 20kg and also prediction of shipment weight by regression model. This includes data collection, feature creation, cleaning, normalization, and transformation to ensure the data is in a suitable format for machine learning model training. Feature engineering techniques are used to extract relevant features and create new features from the shipment data that can effectively help machine learning model to differentiate between correctly labeled and incorrectly labeled shipments.

The selected machine learning models such as Random Forest and Support Vector Machine are trained using a labeled dataset, carefully chosen to represent a diverse range of correctly and incorrectly labeled shipments. The model is then evaluated using the appropriate metrics to assess its accuracy, precision, recall, or F1 score. The model selection process involves evaluating these two algorithms, Random Forest and support vector machines to determine the most suitable approach for the identification task.

Chosen supervised machine learning model is optimized for improving the performance of the model. Hyper-parameter tuning is performed to find the optimal configuration for the selected model, ensuring the best possible performance. The tuned model analyzes shipment data to accurately classify shipments as either correctly labeled or requiring labeling correction. Using a labeled data set and employing feature engineering techniques, the machine learning model is trained to effectively identify shipments that do not conform to labeling standards.

Upon successful evaluation of cross validation results of selected model, the machine learning model will be integrated into existing logistics systems. This enables automated identification of incorrectly labeled or unlabeled shipments, enabling timely corrective measures to be taken. By reducing the occurrence of incorrectly labeled or unlabeled shipments, this project aims to enhance the efficiency and accuracy of logistics operations, ultimately meeting business requirements.

***Keywords:*** *Feature Engineering, Random Forest, Support Vector Machines, Hyper-parameter tuning, Accuracy, Precision, Recall, F1 score,cross-validation.*

# Acknowledgements

I am grateful to the individuals who have played a significant role in enabling me to complete this project and successfully obtain my master's degree. Without their valuable support, guidance, and encouragement, this achievement would not have been possible.

I would like to start by giving a big thanks to ***Prof. Dr.Uwe Dettmar*** for his acceptance to this Master thesis and being the first examiner. He was very patient and guided in the right direction towards the completion of the thesis. He formulated and regulated me in achieving the goal of the thesis within the time frame.

Second, I would like to thank my supervisor ***Mr. Benhard Könsgen***, Data scientist, DHL Group, Bonn, for his willingness to accompany this master's thesis with his expertise and as external examiner. He structured this thesis in a more scientific approach from the initial idea and guided through the project and made me overcome in every problem I encountered. His guidance throughout the journey was very useful and made this thesis possible. His technical expertise in data science helped me handle all the technical aspects of the thesis. Special thanks to his help rendered in developing this machine learning model. Despite his busy schedule, he helped me understand data preprocessing, data analysis, feature engineering, Machine learning, model optimization, evaluation, and cross validation.

I am also very thankful to my colleagues for their invaluable assistance during periods when my supervisor was unavailable. Their constructive feedback, encouragement, and collaborative spirit have not only filled the void left by the absence of my supervisor, but have also enriched the quality of this work. Lastly, I extend my sincere gratitude to ***Mr. Markus Bergmeier***, Manager, DHL Group, who gave me this opportunity to undertake my master's thesis.

I extend my heartfelt gratitude to DHL Group, Bonn for granting me the opportunity to conduct my master's thesis research. The exposure to DHL Group world-class facilities and resources has been instrumental in shaping my understanding of my thesis topic and in preparing me for a successful career.

Finally, in profound gratitude, I offer my sincere praise to the Almighty for providing me with the strength, wisdom, and resilience throughout the journey of completing this master's thesis. I am profoundly thankful to my family and friends, who supported me and motivated me with infinite patience and pushed me forward.

# Contents

# List of Figures

# List of Tables

## List of Equations

# Glossary

**CV** Cross Validation.

**DT** Decision Tree.

**MAE** Mean Absolute Error.

**ML** Machine Learning.

**MSE** Mean Squared Error.

**RF** Random Forest.

**SVM** Support Vector Machine.

# Chapter 1

# Introduction

## 1.1 Motivation

DHL Group is the world's leading logistic company. The Group facilitates international trade by bringing people and markets together. It wants to be the preferred choice for clients, staff, and investors globally. In order to do this, DHL Group is intensifying digital transformation across all business divisions while concentrating on expanding its lucrative core logistics activities. A component of the DHL Group, Post and Parcel Germany employs about 187,000 people. Transporting, sorting, and delivering letters and packages is its primary operation, also known as the national mail and parcel business. The division, which has two brands in Germany, is a leader in its sector when it comes to social and environmental sustainability.

As part of the revision of the Postal Act, specific guidelines have been formulated in § 73 to delineate the labeling requirements for packages with increased weight of 20 kilograms. According to these regulations, labeling must be completed no later than the final processing step at designated DHL facilities, ensuring that labels are present and accurate during the delivery process. Moreover, packages exceeding 20 kilograms are mandated to be delivered using suitable technical aids to ensure safe handling and compliance with occupational health and safety standards.

There is currently no requirement for DHL to label shipments weighing more than 20 kilograms separately. Only with the introduction of the new postal law at the beginning of 2025 will DHL be obliged to label such shipments in a special way. This is because the labelling is meant to provide the deliverers with greater protection. In the future, 20kg shipments should either only be delivered by two people or with the help of appropriate transport aids (exoskeleton or hand truck).

## 1.2 Problem Statement

The problem addressed in this project arises from the prevalence of incorrectly labeled shipments, which can occur due to human error, system glitches, or deliberate tampering. These inaccuracies can manifest in various forms, including incorrect product descriptions, misreported weights, inaccurate origin or destination information, and mismatched shipping labels. Such discrepancies not only disrupt the flow of goods but can also lead to a cascade of downstream issues, including delayed deliveries, inventory discrepancies, customer dissatisfaction, and potential legal liabilities. In addition, many parcel centers do not have the option of installing additional automatic scales because there is either not enough space or they would reduce the throughput speed of the shipments too much.

Traditional methods for identifying mislabeled shipments often rely on manual inspection processes, which are time-consuming, labor-intensive and prone to errors. Moreover, manual inspection becomes increasingly challenging as shipment volumes scale up, making it impractical for large-scale distribution centers and logistics hubs.

To address this challenge, this project aims to develop a machine learning-based solution for the automated identification of incorrectly labeled shipments. By harnessing the power of artificial intelligence and data analytics, we seek to create a system capable of autonomously detecting and flagging mislabeled items in near real-time, i.e., we identify parcels during PZA time and forecast them before reaching PZE event so that we can eject them in PZE time, thereby mitigating the risks associated with inaccurate labeling and enhancing the efficiency and reliability of shipment handling processes.

Specifically, the proposed machine learning model will be trained on a dataset comprising comprehensive information about shipped items, including but not limited to product descriptions, weights, dimensions, origins, destinations, and other relevant attributes. Leveraging supervised learning techniques, the model will learn to distinguish between parcels weighing more than 20kg and less than 20kg. Additionally, a supervised prediction machine learning model may be employed to forecast the real weight of the parcel based on the data provided by the customer and the history data of the customers.

By addressing the problem of incorrectly labeled shipments through automated machine learning-based detection, this project seeks to contribute to the changes in new postal law.

## 1.3 Thesis Objective

Against this backdrop, the objective of this project is to develop and implement a machine learning model capable of accurately identifying shipments that are incorrectly labeled, particularly focusing on heavy packages exceeding 20 kilograms for both national shipments and international import shipments from private customers. Using machine learning techniques, the project aims to improve efficiency and precision in identifying the parcel whose real weight is more than 20kg, thus ensuring compliance with regulatory standards and improving overall service quality.

In pursuit of this objective, the project will perform a comprehensive analysis of shipment data to identify patterns and trends associated with labeling errors, particularly focusing on heavy packages. Subsequently, a machine learning model will be developed and integrated into existing parcel processing systems to enable near real-time detection and flagging of mislabeled shipments during sorting and delivery operations. The performance of the implemented model will be rigorously evaluated using representative datasets, with a focus on precision, recall, F1 score and operational impact.

## 1.4 Thesis Outline

The following five chapters make up the thesis:

1. In chapter 2, I define the theoretical foundations of data types, collection, feature engineering, data cleaning, basics of machine learning algorithms. Moreover, I mentioned the model tuning methods. Furthermore, the evaluation techniques to assess the classifier and regressor performance are discussed in detail.
2. In chapter 3, I define the business process, data sources along with description of data features, data collection, analysis and pre-processing methodologies such as feature engineering, data cleaning, splitting of data into train and test data and training the chosen models for all shipments within Germany (National).
3. In chapter 4, I define the business process, data sources along with description of data features, data collection, analysis and pre-processing methodologies such as feature engineering, data cleaning, splitting of data into train and test data and training the chosen models for all international import shipments for private customers (International).
4. In Chapter 5, I focus on the evaluation of machine learning models to identify incorrectly labeled shipments. Specifically, I compare Support Vector Machine (SVM) and Random Forest models. I evaluate both models using metrics such as accuracy, precision, recall, F1 score. After presenting the initial results, I select the better-performing model based on a combination of these metrics. The selected model is tuned using manual hyperparameter tuning, where parameters like the number of trees, maximum depth, and minimum samples split are adjusted for the Random Forest model, or kernel type and regularization parameters for the SVM. The results of the tuning process are presented, showcasing improvements in the model's performance. This chapter concludes with selection of machine learning model with better evaluation results and tuned hyper-parameters. I present the results of each finalized classifier and regression models in terms of the evaluation techniques for both national and international shipments.
5. In chapter 6, I conclude the thesis by summarizing the key findings and contributions of the study, while also acknowledging its limitations and areas for future work.

# Chapter 2

# Theoretical Background

In this chapter, we dive into the theoretical concepts fundamental to automated identification of incorrectly labeled shipments using machine learning models. We will explore SQL, particularly Teradata, data collection and preprocessing techniques, supervised machine learning classification models and regression models, optimization of the machine learning models, evaluation metrics and methods to validate results. These theoretical foundations provide the essential basics for understanding and implementing the practical aspects of our study.

## 2.1 Understanding Data Types

The data utilised to train machine learning algorithms is what gives them their power. To build effective models, we must understand the data we use. Data refers to a set of information or records in a format that can be used to train a machine learning model. From the standpoint of machine learning, the majority of data can be divided into four main categories that we can utilise to train any model: numerical data, categorical data, time-series data, and text data. Let's briefly understand each of these data types.

1. **Numeric data:** This type of data consists of numbers and can include continuous values (such as prices or temperatures) or discrete values (such as counts or rankings). Numeric data can be used as input or output in machine learning algorithms.

2. **Categorical data:** This type of data consists of categories or labels, such as names, types, or categories. Categorical data can be used as input or output in machine learning algorithms, but it may need to be converted into a numerical form in order to be used by certain algorithms[23].

3. **Time series data:** This type of data consists of measurements taken at regular intervals over a period of time. Time series data is often used in machine learning algorithms for tasks such as forecasting or trend analysis.

4. **Text data:** This type of data consists of written or spoken words and can include things like emails, social media posts, or customer reviews. Text data is often used in ma-

chine learning algorithms for tasks such as natural language processing or sentiment analysis[23].

It is important to note that these are general categories of data and that different machine learning algorithms may be suited to different types of data or may be able to handle multiple types of data.

Data is stored in a database in organized structures known as tables, which consist of rows and columns. Each row represents a unique shipment and each column represents a specific piece of information about that shipment, such as weight, volume, or date. These tables are stored in databases. A database is an organized collection of data, generally stored and accessed electronically from a computer system. Databases can be classified according to their organizational approach into several types of Database Management Systems (DBMS). The main types of DBMS include:

1. Relational Database Management Systems (RDBMS)
2. NoSQL Database Management Systems
3. Object-oriented Database Management Systems (OODBMS)
4. In-Memory Database Management Systems
5. NewSQL Database Management Systems

As we are using RDBMS to store the data, SQL is used to manage the data. RDBMS is a type of database management system that keeps information organised into relevant tables. The relationships among the data are also stored in the form of tables.

## 2.2 SQL and Teradata

### 2.2.1 Introduction to SQL

Structured Query Language (SQL) is a standardized language used to manage and manipulate relational databases. Users can do CRUD (create, read, update, and delete) tasks using SQL within a database. SQL is fundamental for querying and operating database systems, enabling efficient data management and retrieval.

### 2.2.2 Overview of Teradata

Teradata is a highly scalable and robust relational database management system (RDBMS) known for handling large volumes of data. It supports SQL and is widely used for data warehousing and analytics. Teradata's architecture is designed to optimize parallel processing, making it appropriate for complex queries and large datasets.

## 2.3 Machine Learning Overview

The machine learning process is a systematic workflow that starts with problem definition, where the objective and scope are clearly identified, followed by data collection to gather relevant and quality data. The data then undergoes pre-processing to clean, normalize, and transform it, making it suitable for analysis. Next, exploratory data analysis (EDA) is conducted to understand patterns and relationships within the data. Afterward, an appropriate model is selected and trained using the dataset, followed by model evaluation to assess its performance using relevant metrics. Once the model is validated, it is deployed into a production environment for real-time or batch predictions. Continuous monitoring and maintenance ensure the model remains accurate over time, adapting to new data or changes in the environment, with comprehensive documentation throughout to ensure transparency and reproducibility.

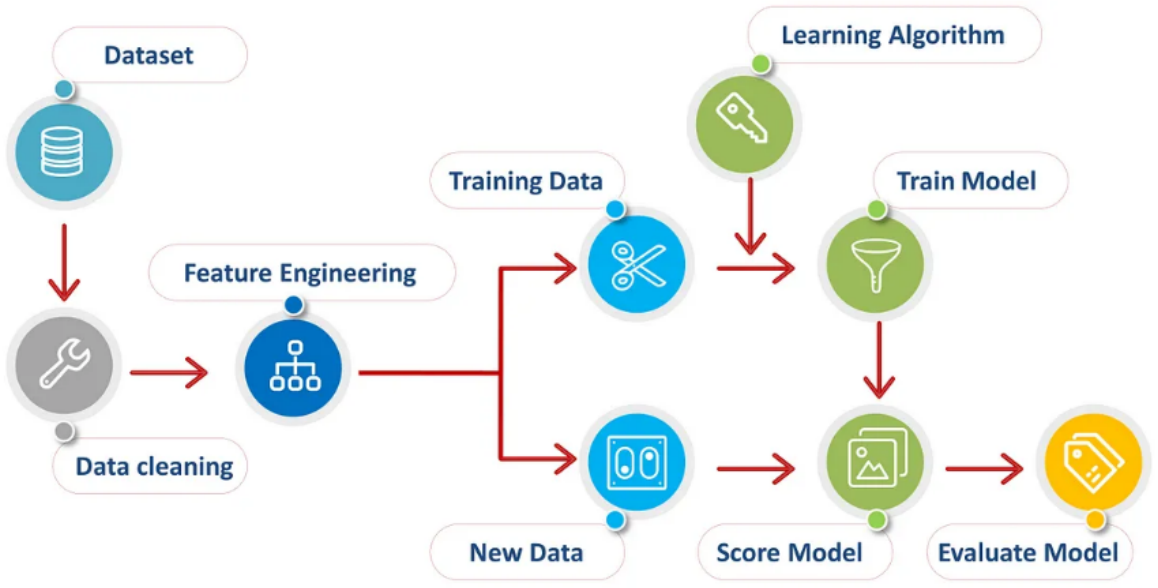


Figure 2.1: Machine Learning Process[7].

## 2.4 Data Collection and Preprocessing

Data collecting has grown into one of the main bottlenecks among the many difficulties in machine learning. It is well recognised that the bulk of the time required to complete an end-to-end machine learning project is devoted to data preparation, which includes feature engineering, data collection, cleaning, analysis, and visualisation.

### 2.4.1 Data Collection

The initial stage of every project involving data analysis or machine learning is data collection. It involves gathering the information needed to answer specific research questions or to develop a predictive model. This step is crucial because the quality, quantity, and relevance of the data collected directly impact the outcomes of subsequent analysis.The process of collecting relevant information for training a machine learning model is known as Data Collection. Data collection consists mainly of data acquisition, data labeling, and improvement of existing data.

Data collection for this project involves the acquisition of a comprehensive data set that contains information about shipped items. This data set is sourced from internal company databases. The data collected will undergo pre-processing to ensure consistency, accuracy, and compatibility with the subsequent stages of the analysis. There are several methods in data collection which are listed below:

1. Surveys and Questionnaires: Structured forms that gather quantitative or qualitative data from a specific audience.
2. Observational Data: Data collected through observing and recording behaviors or events.
3. Web Scraping: Extracting data from websites, often using automated tools or scripts.
4. Sensor Data: Information gathered from sensors, such as IoT devices, measuring environmental or operational parameters.
5. Database Extraction: Retrieving data from existing databases or data warehouses.
6. API Calls: Using Application Programming Interfaces (APIs) to collect data from online services.

### 2.4.2 Data Preprocessing

It is necessary to preprocess the data after it has been collected to prepare them for a machine learning task. Data preprocessing encompasses many different tasks, from data formatting to feature creation. Data preprocessing is a critical step in preparing shipment data ready to train machine learning models. This involves cleaning the data, handling missing values, transforming the data, detecting outliers, and normalizing or scaling the features, which is essential to improve the accuracy and efficiency of machine learning models. The quality of the data has a major impact on how well a prediction model works. Handling missing numbers and addressing data errors reduces noise in the data. This is known as data cleaning. The encoding of features into the appropriate data format and data creation are included in the data transformation. An outcome from the data cleaning stage is a dataset was utilised in the modelling stage that follows. A variety of approaches and methods are used to prepare data. The approaches of feature selection, outlier removal, encoding of text variables like one hot encoding method and null value imputation have all been applied in this thesis.

## 2.5 Machine Learning

Machine Learning is the science (and art) of programming computers so they can learn from data[10]. Applying ML techniques to dig into large amounts of data can help discover patterns that were not immediately apparent and this is called data mining. Machine Learning is great for problems, for which existing solutions require a lot of hand-tuning or long lists of rules or for complex problems, for which there is no good solution at all using a traditional approach or for fluctuating environments where a system requires to adapt to new data or for getting insights about complex problems and large amounts of data. Machine learning is generally a training system to learn from past experiences and improve performance over time[9]. Machine learning helps to predict massive amounts of data. Machine learning has emerged as a powerful tool in the logistics industry, offering the potential to automate processes, improve operational efficiency, and optimize decision-making. By leveraging historical shipment data, machine learning models can be trained to identify patterns, make predictions, and classify shipments based on various attributes.

Machine learning (ML) encompasses various types, each suited to different kinds of tasks and data. The major types of machine learning are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning

### 2.5.1 Supervised Machine Learning

In supervised learning, the training data used to feed to the algorithm includes the desired solutions, called labels[10].Machine learning models for shipment labeling typically fall under the category of supervised learning, where the algorithm learns from labeled examples to make predictions on unseen data. Labeled data consist of historical shipments with known correct labels, along with their corresponding attributes and features. There are two main categories of supervised learning that are mentioned below:

#### 2.5.1.1 Classification

Classification deals with predicting categorical target variables, which represent discrete classes or labels. For instance, classifying shipments as heavier than 20kg or not, or predicting weight of parcel is more than 20kg or not. Classification algorithms learn to map the input features to one of the predefined classes. Here are some classification algorithms:

1. Logistic Regression
2. Support Vector Machine
3. Random Forest

4. Decision Tree
5. K-Nearest Neighbors (KNN)
6. Naive Bayes

#### 2.5.1.2 Regression

It is a supervised machine learning technique, used to predict a target numeric value, such as the weight of the shipment, given a set of input features (length, breadth, height, volume, etc.) called predictors. It models the relationship between the input features and the target variable, allowing for the estimation or prediction of numerical values. Here are some regression algorithms:

1. Linear Regression
2. Polynomial Regression
3. Ridge Regression
4. Lasso Regression
5. Decision tree
6. Random Forest

Commonly used supervised learning algorithms in logistics include decision trees, random forests, support vector machines (SVM), and neural networks. Decision trees provide a simple and intuitive way to classify shipments based on a series of binary decisions. Random forests combine multiple decision trees to improve accuracy and handle more complex patterns in the data.

### 2.5.2 Unsupervised Machine Learning

Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabeled data[9]. That means it deals with datasets that have not been classified or categorized, and the algorithm itself must find patterns and relationships within the data. While unsupervised learning can be incredibly powerful, it also has its challenges and limitations. Because the data is unlabeled, it can be difficult to verify the accuracy of the algorithm's findings. Additionally, unsupervised learning can be more computationally intensive than other types of machine learning, and it may require more advanced expertise to implement effectively. Unsupervised learning is a significant area of machine learning that has the potential to extract valuable insights from unlabeled data. As technology advances and more data becomes available, the capabilities and applications of unsupervised learning will likely continue to grow.

### 2.5.3 Semi-Supervised Machine Learning

Semi-supervised machine learning is a combination of supervised and unsupervised machine learning methods[21]. It makes use of both labeled and unlabeled data for the training process. SSL is used when there is a large amount of data available, but only a small portion is labeled. The purpose of SSL is to use the unlabeled data to improve the learning accuracy of the model. SSL is common in many real-world scenarios where labeling data can be expensive or time-consuming. SSL uses a combination of supervised learning methods (like decision trees, k-nearest neighbors, and support vector machines) and unsupervised learning methods (like clustering and dimensionality reduction)[31]. The most common SSL methods include self-training, multi-view training, and semi-supervised support vector machines (SVMs).

### 2.5.4 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to behave in an environment, by performing certain actions and observing the rewards/results of those actions. The purpose of RL is to enable an agent to learn and make informed decisions autonomously, based on its observations and interactions with its environment.Reinforcement Learning methods can be divided into two main types: Value-Based and Policy-Based. Reinforcement Learning has a wide range of applications, including Robotics, Game Playing, Resource Management, Recommendation Systems. While reinforcement learning has shown great promise, it also comes with several challenges, such as the trade-off between exploration and exploitation, the difficulty of handling environments with large state and action spaces, and the instability and slow convergence of learning. Future directions in RL research might focus on addressing these challenges and extending RL to handle more complex and real-world environments.

## 2.6 Machine Learning Models

### 2.6.1 Support Vector Machines (SVMs)

Support Vector Machines are supervised machine learning algorithms that are primarily used for classification tasks. They aim to find an optimal hyperplane that effectively separates data points belonging to different classes. The core principle of SVMs is to maximize the margin, which is the distance between the hyperplane and the closest data points of each class. SVM separates data into different classes by a hyperplane or hyperplanes since it has the ability to handle multidimensional data[19].

SVMs can be categorized based on the type of data they handle and the complexity of the decision boundary:

Based on Data Separability:

1. Linear SVM: Handles linearly separable data, meaning data points can be divided into

two classes by a straight line or hyperplane.

2. Non-linear SVM: Employs the kernel trick to handle non-linearly separable data by mapping it into a higher-dimensional space where it becomes linearly separable.

Based on Problem Type:

1. Classification SVM: The most common type, used to categorize data into different classes.
2. Regression SVM: Used to predict continuous numerical values.
3. One-Class SVM: Used for anomaly detection, identifying outliers in data.

Based on Margin:

1. Hard Margin SVM: Assumes perfectly separable data and aims to find the hyperplane with the maximum margin.
2. Soft Margin SVM: Allows for some misclassification by introducing slack variables, making it more robust to noise and outliers.

#### 2.6.1.1 Key terms in SVM

1. Hyperplanes: A hyperplane is a decision boundary separating data points of different classes.
2. Margin: The distance between the hyperplane and the closest data points of each class is determined as margin. Maximizing the margin is a key principle in SVM to improve generalization performance.
3. Support vectors: Support vectors are the data points that lie closest to the decision boundary (hyperplane) of an SVM. They are important in determining the orientation and position of the hyperplane.

#### 2.6.1.2 Hyperparameters in SVM

1. Regularization Parameter (C): Controls the trade-off between achieving a low training error and a low testing error. A small ’C’ makes the margin larger and less strict, leading to a simpler model that might generalize better. A large ’C’ attempts to classify all training points correctly.
2. Kernel Parameters: Degree (d) for Polynomial Kernel,determines the degree of polynomial.Gamma for RBF Kernel, defines how far the influence of a single training example reaches. Low values mean ’far’ and high values mean ’close’.

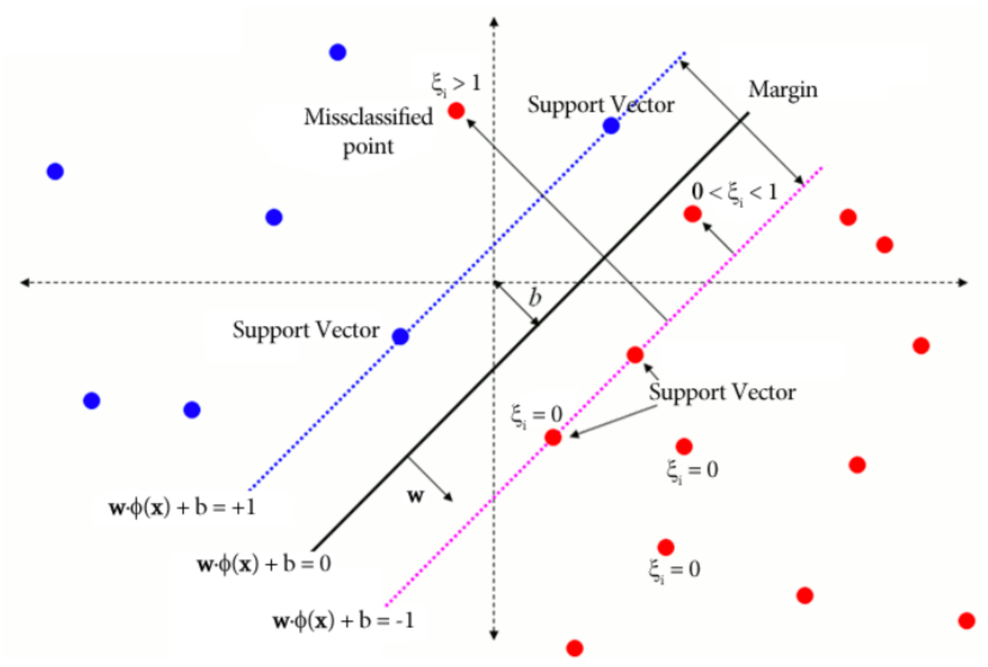


Figure 2.2: SVM[13].

### 2.6.2 Decision Tree

A decision tree is a versatile and widely used machine learning model that mimics human decision-making processes through a tree-like structure comprising nodes, branches, and leaves. The root node represents the entire dataset, internal nodes denote features, and leaf nodes represent the outcomes or predictions. Decision trees can be broadly categorized into two types: classification trees, which are used for categorizing data into predefined classes (e.g., spam or not spam), and regression trees, which predict continuous values (e.g., house prices). The tree-building process involves recursively splitting the data based on feature values to reduce impurity, using criteria such as Gini impurity for classification or mean squared error for regression. Hyperparameters like max_depth (maximum depth of the tree), min_samples_split (minimum samples required to split a node), and min_samples_leaf (minimum samples required at a leaf node) are crucial in controlling the complexity and preventing overfitting. Despite their simplicity and interpretability, decision trees can suffer from overfitting and instability, making them sensitive to small changes in the data. However, they are capable of handling both numerical and categorical data without extensive preprocessing. Decision trees find applications in various fields such as medical diagnosis, financial analysis, and marketing due to their intuitive nature and ease of visualization. They are often used as the foundational elements in more advanced ensemble methods like Random Forests to enhance performance and robustness.

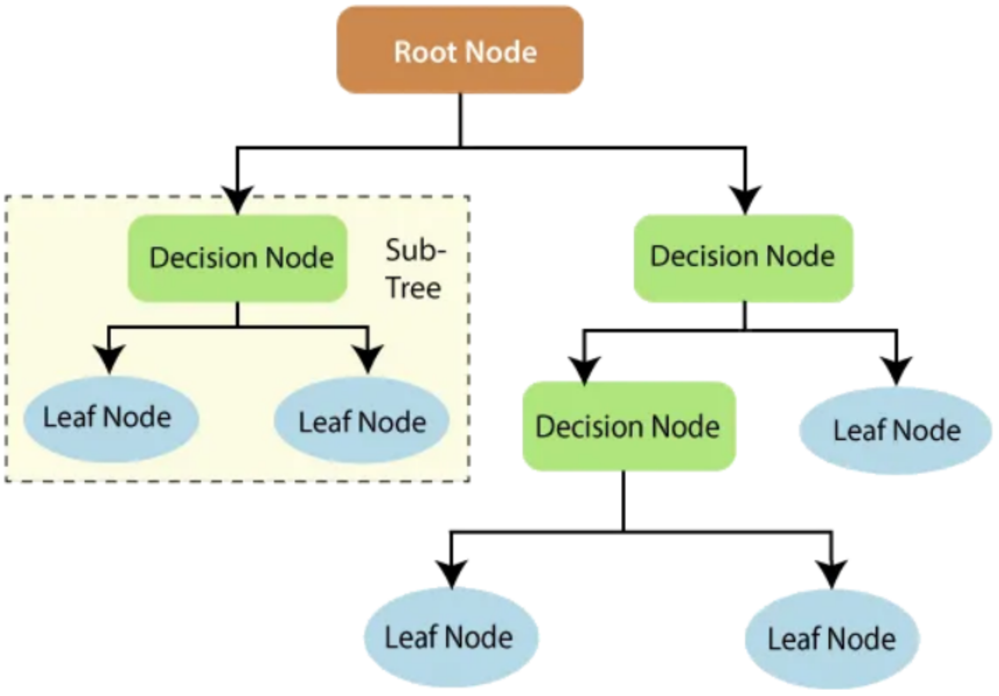


Figure 2.3: Decision Tree structure[1].

Decision trees are mainly classified into two types:

#### 2.6.2.1 Classification Trees

If the target variable is categorical then Classification trees are used. In this type of tree, the outcome variable is a class label, and the tree is constructed by identifying splits that result in the most significant information gain to classify the data points into their respective categories.

#### 2.6.2.2 Regression Trees

If the target variable is continuous then regression tress are used. In this type of tree, the outcome is a real number, and the tree is constructed by selecting splits that minimize the variance within each node.

The important terms that are fundamental in understanding how decision trees function and are crucial for interpreting and constructing these models are explained below:

1. Root Node: The topmost node in a decision tree is called root node. It represents the entire dataset and the first point at which the dataset is split based on an attribute. The root node is chosen based on the attribute that provides the maximum information gain or the best split according to a particular criterion (e.g., Gini index, entropy).
2. Decision Nodes: Decision nodes, also known as internal nodes, are the points within the tree where the data set is further split based on different attributes. Each decision

node represents a test or decision on a specific attribute, leading to more specific subgroups of data.

3. Leaf Nodes: Leaf nodes, also known as terminal nodes, are the end points of a decision tree. These nodes do not split any further and represent the final decision or outcome. Each leaf node corresponds to a class label (in classification trees) or a predicted value (in regression trees).
4. Splitting: Splitting refers to the process of dividing a node into two or more sub-nodes based on a condition or a set of conditions on an attribute. The objective of splitting is to create nodes that are as pure as possible, which means that they contain data points that are similar to each other with respect to the target variable.
5. Tree Depth: Tree depth refers to the number of levels in the decision tree from the root node to the farthest leaf node. The depth of the tree affects its complexity; deeper trees are more complex but may overfit the data, while shallower trees are simpler but may underfit.
6. Pruning: Pruning is the process of removing branches from the decision tree that have little importance or contribute minimally to the predictive power of the model. Pruning helps in reducing the complexity of the tree, preventing over-fitting, and improving generalization to new data.
7. Impurity Measures: Impurity measures are used to evaluate the quality of a split at each node. Common impurity measures are Gini Index, Entropy (Information Gain), Mean Squared Error (MSE).

A decision tree employs a variety of algorithms to determine the optimal way to split a node into two or more sub-nodes, with the goal of increasing the homogeneity of the resulting sub-nodes. The tree examines all available variables and selects the one that produces the most homogeneous division. The choice of the splitting algorithm is influenced by the type of target variables being used. Several commonly used algorithms in decision trees include ID3 (Iterative Dichotomiser 3), which is a classification algorithm that builds the tree using a greedy approach, C4.5, an extension of ID3 for generating decision trees, CART (Classification and Regression Tree), a versatile algorithm applicable to both classification and regression problems, and CHAID (Chi-square Automatic Interaction Detection), which identifies relationships between categorical outcome variables and categorical predictor variables.

Among the myriad of machine learning algorithms, ensemble methods have gained significant popularity due to their ability to improve model accuracy and robustness. One of the most prominent ensemble methods is the Random Forest model.

### 2.6.3 Random Forest

A Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of classes (classification) or mean prediction (regression) of individual trees[25]. The "forest" in Random Forest refers to the collection of these decision trees.Decision Trees are also the fundamental components of Random Forests.

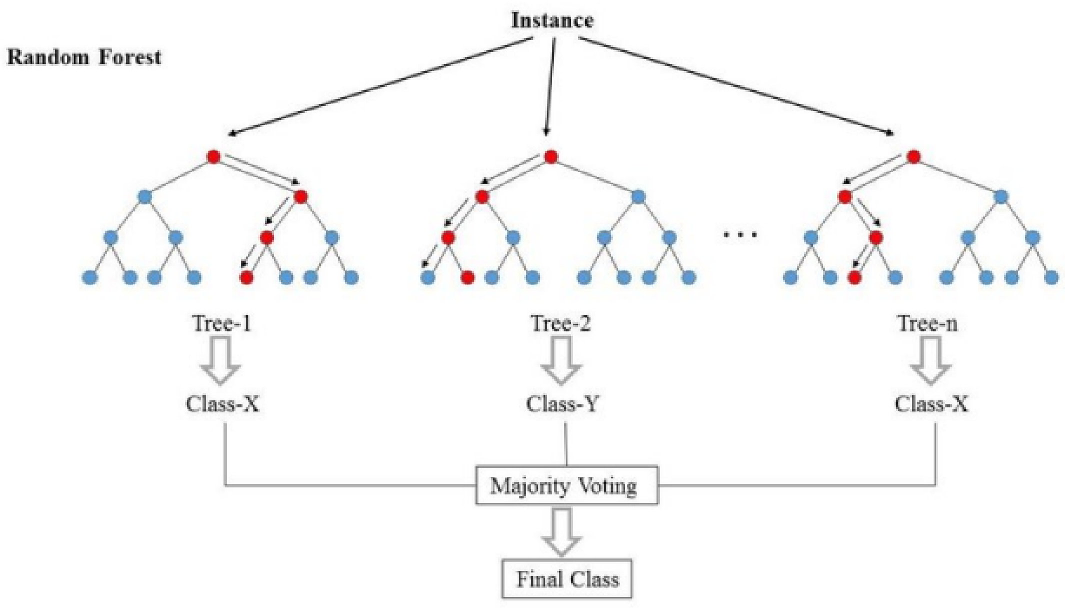


Figure 2.4: Random Forest structure[12].

The Random Forest algorithm can be broken down into the following steps:

1. Data Bootstrapping: The first step involves creating multiple subsets of the original dataset by sampling with replacement. Each subset is used to train a separate decision tree. This technique is known as bootstrapping.
2. Building Decision Trees: For each bootstrap sample, a decision tree is trained. During the training of each tree, a random subset of features is chosen at each split rather than considering all features. This introduces diversity among the trees. The splitting criterion (e.g., Gini Index for classification, Mean Squared Error for regression) is applied to the chosen subset of features to determine the best split.
3. Aggregating Results: After all the trees have been built, predictions are made on the test data. For classification tasks, each tree votes on the class and the majority vote is taken as the final prediction. For regression tasks, the predictions from all trees are averaged to produce the final output.

#### 2.6.3.1 Key Hyperparameters in Random Forest

In the Random Forest model, hyperparameters play a crucial role in determining the structure of the individual decision trees, the ensemble of trees, and the overall performance of the model. Proper tuning of these hyperparameters can significantly impact the accuracy, robustness, and generalizability of the model.

1. **Number of Trees (n_estimators):** The n_estimators hyperparameter specifies the num-

ber of decision trees to be built in the forest. Each tree in the forest contributes to the final prediction by either voting (in classification) or averaging (in regression). A very small number of trees may lead to underfitting, where the model is too simple to capture the underlying patterns in the data. Increasing the number of trees generally improves model accuracy up to a point, but after a certain number, the improvement becomes negligible leading to over-fitting, and the computational cost increases. Common values range from 100 to 1000 trees, but the optimal number depends on the dataset size and complexity.

2. **Maximum Depth of Trees (max_depth):** The max_depth hyperparameter controls the maximum depth of each individual tree in the forest. This determines how deep the tree can grow, or in other words, the maximum number of splits in any branch of the tree. A shallow tree (low max_depth) may not capture all the important features and interactions, leading to underfitting. Very deep trees (high max_depth) can lead to overfitting, where the model becomes too complex and starts to capture noise rather than the underlying pattern. Common values range from 10 to 100, but the optimal depth is highly dataset-dependent. Sometimes, leaving this parameter as None allows trees to grow until they are fully expanded.

3. **Minimum Samples per Split (min_samples_split):** The min_samples_split hyperparameter specifies the minimum number of samples required to split an internal node. It controls the sensitivity of the tree to changes in the data.A small value (e.g., 2) allows the tree to split more often, leading to deeper trees that may overfit the data.Larger values prevent the tree from growing too complex by requiring more data at each split, thus reducing overfitting.The default is usually 2, but values like 10 or higher can be used to prevent overfitting, especially with large datasets.

4. **Minimum Samples per Leaf (min_samples_leaf):** The min_samples_leaf hyperparameter sets the minimum number of samples that must be present in a leaf node. This parameter ensures that leaf nodes are not too small, which can help in regularizing the model. Smaller leaf sizes may lead to more complex trees with many branches, increasing the risk of overfitting. Larger leaf sizes produce simpler trees, which may reduce overfitting but can also lead to underfitting if set too high. Default is usually 1, but higher values (e.g., 5, 10) are often used to improve generalization.

5. **Number of Features (max_features):** The max_features hyperparameter determines the number of features to consider when looking for the best split at each node. By limiting the number of features, Random Forest introduces randomness, which can help to decorrelate the trees and improve performance. A lower value results in more random trees, which can reduce overfitting but may also increase bias. A higher value makes the trees more similar to each other, potentially leading to overfitting, but can reduce bias. Typical values for classification is 'sqrt' and for regression 'number of features/3'

6. **Bootstrap Sampling (bootstrap):** The bootstrap hyperparameter determines whether bootstrap samples (sampling with replacement) are used when building the individual

trees. If set to True, each tree is built on a different random sample of the data; if set to False, the same data is used for all trees. Setting this feature to 'True' introduces diversity in the trees, which can reduce overfitting and improve generalization, if it is set to 'False' can lead to trees that are more similar to each other, which may increase overfitting but can reduce variance. The default is True for most applications, as it generally results in better model performance.

7. **Out-of-Bag (OOB) Score (oob_score):** The oob_score hyperparameter allows the model to calculate the out-of-bag score, which is an internal cross-validation score based on the out-of-bag samples (the samples not used to train a particular tree). Often set to True for internal validation during model training.

## 2.7 Hyper-Parameter Tuning

Hyper-parameter tuning is the process of optimizing the hyper-parameters of a machine learning model to achieve the best possible performance on a given dataset. Unlike model parameters, which are learned during the training process, hyper-parameters are set before the training begins and control various aspects of the learning process, such as the complexity of the model, the learning rate, and the number of iterations. Properly tuned hyper-parameters can significantly improve the model's accuracy, robustness, and generalization ability. Hyper-parameter tuning helps in balancing the bias-variance tradeoff, reducing both underfitting and overfitting. Efficient hyper-parameter tuning can lead to faster training times and more effective use of computational resources. Hyper-parameters can be broadly categorized into two types:

1. Model hyper-parameters are specific to the architecture or structure of the model itself.
2. Algorithm hyper-parameters control the process of learning.

There are several methods for tuning hyper-parameters, each with its advantages and disadvantages. The choice of method often depends on the complexity of the model, the size of the hyper-parameter space, and available computational resources.

### 2.7.1 Manual Search

Manual search involves selecting a set of hyper-parameters by hand, training the model with these hyper-parameters, evaluating its performance, and iterating this process until a satisfactory combination is found. This method relies on the practitioner's experience and intuition to choose the hyper-parameters. To implementing manual search for hyper-parameter optimization, is started by identifying the hyper-parameters that significantly impact the model's performance. Next, define a range or set of values for each of these hyper-parameters. For instance, for tuning the learning rate consider values such as 0.001, 0.01, and 0.1. Then, train the model using the chosen set of hyper-parameters and evaluate its performance using a validation set or cross-validation technique. Based on the performance results, modify the hyper-parameters and repeat the training and evaluation process. Continue iterating through different combinations of hyper-parameters until finding a combination that yields satisfactory performance. Throughout this process, it is beneficial to start with a broad range of values and then narrow down the search based on initial findings. Logging the hyper-parameters tried and their corresponding performance can also help in avoiding redundant evaluations and ensuring reproducibility. While manual search is straightforward and provides full control, it can be time-consuming and may only explore a small portion of the hyper-parameter space, making it essential to have a systematic approach and consider more advanced methods for larger-scale problems. There are more efficient alternatives for hyper-parameter optimization, which are defined in short below.

### 2.7.2 Grid Search

Grid Search is a machine learning technique for optimising hyper-parameters. It involves defining a range of hyper-parameters and creating a grid with all possible combinations of those parameters. The model is then trained using the given hyper-parameters to assess each combination, and a performance metric—such as accuracy, precision, recall, or F1 score—is calculated. The ideal combination of hyper-parameters is then determined by selecting the combination that yields the best performance metric. Grid search is a brute-force technique that can be costly to compute, particularly if there are many hyper-parameters and a wide range of values for each hyper-parameter. However, it is an efficient and simple means to adjust hyper-parameters, which frequently results in better model performance.

### 2.7.3 Random Search

While manual search and grid search are commonly used methods, they can be inefficient and time-consuming, especially when dealing with high-dimensional hyper-parameter spaces. Random search offers an alternative approach that can be more efficient and effective in certain scenarios. Random Search is a more efficient alternative to Grid search. Instead of evaluating all possible combinations, random search samples hyper-parameter combinations randomly from the specified ranges. Random search is a hyper-parameter optimization technique that involves randomly sampling hyper-parameter combinations from a specified distribution. Unlike grid search, which systematically evaluates all possible combinations within a predefined grid, random search explores the hyper-parameter space more broadly by selecting random combinations. This approach can often lead to better results in less time, particularly when only a subset of the hyper-parameter space contains good solutions.

## 2.8 Evaluation Metrics

The evaluation of machine learning models is essential to assess their performance. Accuracy, precision, recall, F1 score, and AUC-ROC are evaluation metrics commonly used for classification tasks. This evaluation will help determine which model performs best for the task of automated identification of incorrectly labelled shipments. The effectiveness of machine learning models can be evaluated based on the evaluation metrics.The assessment measure is used in two stages in a typical data categorization problem: the training stage (learning process) and the testing stage. The classification method was optimised during the training phase using the evaluation metrics. Additionally, the evaluation metric was employed as the evaluator in the testing stage for evaluating the generated classifier's performance when evaluated using unseen data. The evaluation metrics can be categorized into two types based on the type of model, here for our supervised machine learning model mainly classification evaluation metrics and regression evaluation metrics are used. For classification model the results are binary i.e., either 0 or 1, though there are many evaluation metrics for classification models, we use mostly used confusion matrix, accuracy, precision, recall and F1 Score for evaluation of our results in this project. To understand those four terms, we need to know about True Positives(TP), True Negatives(TN), False positives(FP), False Negatives(FN). TP and TN denote the number of positive and negative instances that are correctly predicted by the classification model. Meanwhile, (FP) and (FN) denote the number of negative and positive instances incorrectly predicted by the classification model, respectively. Along with the evaluation metrics analysis, we also considered time taken for the model while training and also to predict the output.

### 2.8.1 Classification Evaluation Metrics

Various classification evaluation metrics are used to measure the performance of classification algorithms. Here are the detailed definitions of some commonly used classification metrics:

#### 2.8.1.1 Confusion Matrix

One popular technique used by academics to assess a model's generalisation performance is the confusion matrix (CM), which can be explained using the matrix below.

| | **Actual Negative** | **Actual Positive** |
|---|---|---|
| **Predicted Negative** | True Negative (TN) | False Negative (FN) |
| **Predicted Positive** | False Positive (FP) | True Positive (TP) |

Table 2.1: Confusion Matrix

The confusion matrix table is displayed in Table 2.1, where the rows indicate the anticipated class labels and the columns reflect the actual class labels. When we examine the cells, the correctly anticipated classes labelled as TN and TP, are indicated by the cells on the other side of the diagonal. Wrongly predicted classes labeled as FP and FN are shown diagonally.

#### 2.8.1.2 Accuracy

The percentage of accurate predictions to the total number of evaluated instances both true and false is represented by accuracy. It is a fundamental and widely used performance metric that provides researchers with an overall understanding of the percentage of correctness, or all instances of a classifier that are correctly classified when tested against unseen data.

However, it has certain limitations, including the inability to examine class imbalance. For example, if a dataset has 1000 examples in class 2 and 10 instances in class 1, the model may be able to predict every entry to be in class 2 with a very high degree of accuracy. Even if it is limited to cases involving minority classes, researchers continue to use it extensively to select the best answers. In general, the accuracy metric measures the ratio of correct predictions over the total number of instances evaluated[10]. The formula for accuracy is given as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

#### 2.8.1.3 Precision

The measure of correctly predicted from the total predicted in a positive class defines precision[10]. In other words, precision quantifies how many of the predicted positive instances are actually positive. The formula for precision is given as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

Precision is a critical metric in scenarios where the cost of false positives is high. High precision indicates that the model has a low false positive rate, meaning it is more reliable when it predicts a positive instance. For example, in fraud detection, a false positive might mean flagging a legitimate transaction as fraudulent, which can inconvenience customers and lead to loss of trust. Similarly, high precision is vital for this project because misclassifying parcels that are less than 20kg as greater than 20kg can lead to unnecessary special assistance, which incurs additional costs and inefficiencies. Ensuring that only the parcels that truly need special handling are identified helps optimize resource allocation and maintain operational efficiency. However, precision alone does not provide a complete picture of the model's performance. It should be considered alongside recall.

#### 2.8.1.4 Recall

The true positive rate, also called recall, measures the proportion of positive instances that are correctly identified[10]. Recall is a metric used to evaluate the performance of a classification model, particularly in binary classification tasks. It measures the proportion of actual positive instances that are correctly identified by the model. In other words, recall quantifies the ability of the model to detect all relevant positive instances in the dataset. The formula for recall/sensitivity is given as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

Recall is a critical metric in scenarios where missing positive instances (false negatives) can have serious consequences. For example, in medical diagnostics, a false negative might mean failing to identify a disease in an affected patient, leading to a lack of necessary treatment. Similarly, high recall is essential to ensure that parcels requiring special assistance are not missed. Missing parcels that are over 20kg can lead to potential legal problems. Ensuring high recall helps in capturing all parcels that need special handling, thereby maintaining legal standards.

#### 2.8.1.5 F1 Score

The problem lies in the fact that, although the model learns to predict the class with the majority of instances accurately, it is unable to recognise the minority class if the dataset is extremely unbalanced. Still, the model will continue to have a very high accuracy value. In order to address the problems of accuracy measurements, scientists developed new methods for determining the mean values that provide equal weight to positive and negative data points. Geometric, harmonic, and mathematical means are a few instances of substitutes. FP and FN are equally significant according to the F-score metric, which is a harmonic mean of the Precision and Recall values[10]. It is stated that this metric is far more helpful than "Accuracy," producing superior results when it comes to correcting. F-measure can be given mathematically as:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

#### 2.8.1.6 PRC-AUC Curve

PRC Area (Precision-Recall Curve Area) is a single metric that encapsulates the performance of a model. The PR AUC Score represents the average of the precision scores computed at various thresholds within the range [0.0, 1.0]. The PRC curve is generated by plotting the Positive Predictive Value (Precision) against the True Positive Rate (Recall) at each threshold. For every threshold, both the Positive Predictive Value and the True Positive Rate are determined, and the respective point is plotted on the graph as shown in below figure.

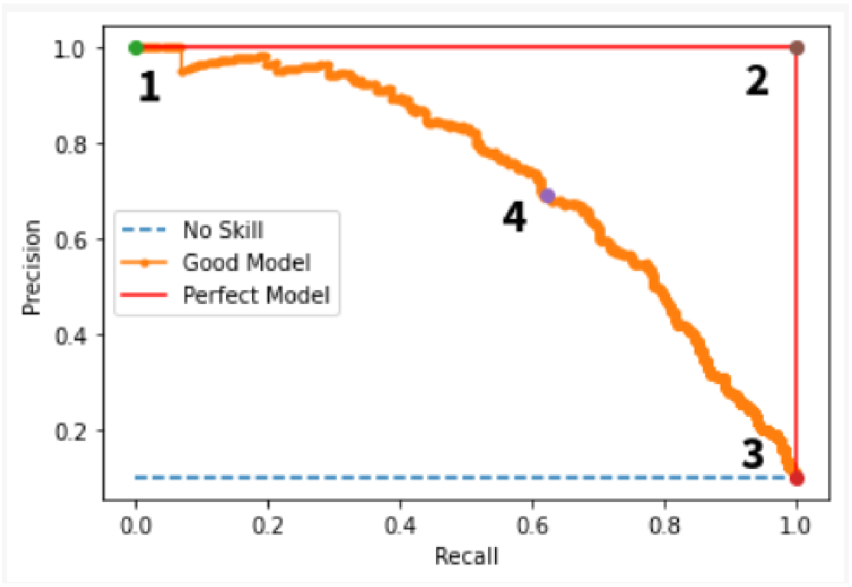


Figure 2.5: Precision - Recall Curve [15].

Ideally, an algorithm should exhibit high precision and high recall. However, these two metrics are not independent, necessitating a trade-off between them. A superior PRC curve will have a higher AUC, indicating better model performance. Studies have demonstrated that PRC curves provide more informative insights than ROC curves when evaluating binary classifiers on imbalanced datasets[28].

### 2.8.2 Regression Evaluation Metrics

Regression evaluation metrics are used to assess the performance of regression models, which predict continuous outcomes. Here are some commonly used metrics:

#### 2.8.2.1 R- Squared score

The R-squared ($R^2$) metric, also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. It is one of key performance metric for evaluating the regression models.By calculating the fraction of explained variance, it gives a measure of the model's quality of fit and consequently, how well unseen data are likely to be predicted. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). In the general case when the true y is non-constant. A constant model that always predicts the expected (average) value of y, disregarding the input features, would get an score of 0[24].

$$R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{2.5}$$

Where N is the number of training datasets, $\bar{y}$ is the mean of the actual $y_i$ and $\hat{y}_i$ is the model's prediction. Generally speaking, the better the model fits the data, the greater the $R^2$ value. However, there are a few noteworthy observations. For instance, adding additional explanatory factors to the model will always result in a higher R2 value, even if these variables don't really improve the prediction. Therefore, a regression model may have a higher R2 not because it makes better predictions, but rather because it has more variables that explain the data. Because of this, the adjusted R2 was created, which takes into account the quantity of variables in your model.

#### 2.8.2.2 Mean Absolute Error(MAE)

The average of the variation between the true and anticipated values is known as the mean absolute error, or MAE. It calculates the deviation between the actual output data and the forecasts. It's crucial to note, though, that this metric does not indicate the precise direction of the error i.e., whether the model over or under-predicts the data. MAE is calculated using the following formula.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{2.6}$$

where N is the number of training datasets, $y_i$ is the actual output and $\hat{y}_i$ the model's prediction.

#### 2.8.2.3 Mean Squared Error(MSE)

The most popular and basic statistic for evaluating regressions is the mean squared error (MSE), which calculates the average squared error over the predictors. The squared difference between the actual value and the predictions is computed for every record, after the summation and averaging of these values across the whole input space. Since the actual and predicted differences are squared before being added together, the mean square error (MSE) can never go negative and, in the case of a perfect model, would be zero. The formula for the MSE is as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \sum (y_i - \hat{y}_i)^2 \tag{2.7}$$

Where N is the number of training datasets,$y_i$ is the actual output and $\hat{y}_i$ the model's prediction.

## 2.9 Cross Validation

Cross-validation is a method for estimating the performance of a model by training it on multiple different subsets of the available data and then validating it on the complementary subsets. This provides a more reliable estimate of the model's performance compared to a single train-test split. Cross-validation is used to assess the generalizability and robustness of a machine learning model. It involves partitioning the data into multiple subsets, training the model on some of these subsets, and validating it on the remaining subsets. This process helps in evaluating the model's performance on unseen data, thereby mitigating the risks of overfitting and underfitting.

Overfitting is recognised as a major contributing factor to the poor performance of machine learning algorithms. Overfitting happens when a model's output correlates too strongly with a set of training data, which might make the model less accurate in predicting future observations. The fundamental principle behind overfitting is to take some residual variance, or noise, and use it to infer the model's underlying structure. When data is large, one way to solve the issue is to train a set of models or a single model with a set of values for its complexity parameters using some of the available data. Then, compare the models on a separate set of data, also known as a validation set, and choose the model with the best predictive performance. It can be necessary to set aside a third test set on which the performance of the chosen model is ultimately assessed if the model design is repeatedly iterated using a small dataset due to the possibility of over-fitting to the validation data.

### 2.9.1 K-Fold Cross-Validation

K-Fold Cross-Validation is a widely used technique for evaluating machine learning models. In this method, the dataset is randomly partitioned into ( k ) equally sized folds or subsets. The model is then trained on ( k-1 ) folds and validated on the remaining fold. This process is repeated ( k ) times, with each fold used exactly once as the validation set. The performance metrics from each iteration are averaged to produce a final estimate of the model's performance. This approach ensures that every data point is used for both training and validation, providing a comprehensive assessment of the model's ability to generalize to unseen data. However, K-Fold Cross-Validation can be computationally expensive, particularly for large datasets and complex models.

### 2.9.2 Stratified K-Fold Cross-Validation

Stratified K-Fold Cross-Validation is a variation of K-Fold Cross-Validation that is particularly useful for imbalanced datasets. In this method, the dataset is divided into ( k ) folds such that each fold maintains the same proportion of classes as the original dataset. This stratification ensures that each fold is representative of the overall class distribution, providing a more accurate and reliable estimate of the model's performance. By preserving the class distribution in each fold, Stratified K-Fold Cross-Validation mitigates the risk of biased performance estimates that can arise from imbalanced datasets. However, implementing this method can be slightly more complex compared to standard K-Fold Cross-Validation.

### 2.9.3 Leave-One-Out Cross-Validation

Leave-One-Out Cross-Validation (LOO-CV) is an exhaustive cross-validation method in which each data point in the dataset is used once as the validation set while the remaining data points are used for training. This process is repeated for each data point, resulting in ( n ) iterations, where ( n ) is the number of data points in the dataset. LOO-CV provides a very thorough evaluation of the model's performance, as it utilizes the maximum amount of data for training and ensures that each data point is tested exactly once. However, this method is extremely computationally intensive, especially for large datasets, as it requires training the model ( n ) times.

### 2.9.4 Repeated K-Fold Cross-Validation

Repeated K-Fold Cross-Validation is an extension of K-Fold Cross-Validation that aims to provide a more robust estimate of model performance by reducing variance. In this method, the K-Fold Cross-Validation process is repeated multiple times with different random splits of the data. For each repetition, the dataset is divided into ( k ) folds, and the model is trained and validated as in standard K-Fold Cross-Validation. The results from all repetitions are then averaged to produce a final performance estimate. This approach helps to ensure that the performance estimate is not overly dependent on a single partitioning of the data, leading to more reliable results. However, Repeated K-Fold Cross-Validation increases the computational cost due to the multiple repetitions.

### 2.9.5 Out-of-Bag (OOB) Validation

Out-of-Bag (OOB) Cross-Validation is a technique used predominantly with ensemble methods like Random Forests to estimate model performance without requiring a separate validation set. In ensemble methods that use bootstrap aggregating (bagging), such as Random Forests, each tree in the ensemble is trained on a bootstrap sample, which is a random subset of the data selected with replacement. As a result, some data points are not included in the bootstrap sample for a given tree; these are referred to as Out-of-Bag samples. OOB Cross-Validation leverages these out-of-bag samples to evaluate the model's performance. Specifically, each data point in the dataset is predicted using only the trees that did not include that point in their bootstrap sample. The predictions are then aggregated, and the OOB error is calculated by comparing these predictions to the actual values. This method provides an unbiased estimate of the model's generalization error and is inherently integrated into the training process of the ensemble method, making it efficient and straightforward to use. However, OOB Cross-Validation is specific to ensemble methods like Random Forests and cannot be directly applied to other types of models. Additionally, for very small datasets, the OOB estimate may be less reliable due to the limited number of out-of-bag samples for each data point.

Cross-validation is an essential technique in machine learning for assessing model performance, tuning hyperparameters, and selecting the best model. It provides a robust estimate of a model's ability to generalize to unseen data by leveraging multiple subsets of the data for training and validation. Various types of cross-validation methods exist, each with its advantages and disadvantages, making it crucial to choose the appropriate method based on the specific characteristics of the dataset and the problem at hand. By incorporating cross-validation into the model development process, practitioners can ensure that their models are both accurate and generalizable, ultimately leading to better performance in real-world applications.

## 2.10 Resampling

Resampling is a vital technique used in machine learning to address the challenge of imbalanced datasets, where one class significantly outnumbers the other(s). This imbalance can lead to biased models that perform poorly on the minority class. Resampling involves either oversampling the minority class, undersampling the majority class, or a combination of both[27]. Oversampling techniques, such as SMOTE (Synthetic Minority Over-sampling Technique), create synthetic data points for the minority class, while undersampling reduces the number of instances in the majority class to balance the dataset. By applying these methods, resampling ensures that the model learns equally from all classes, improving its ability to generalize and make accurate predictions across different categories.

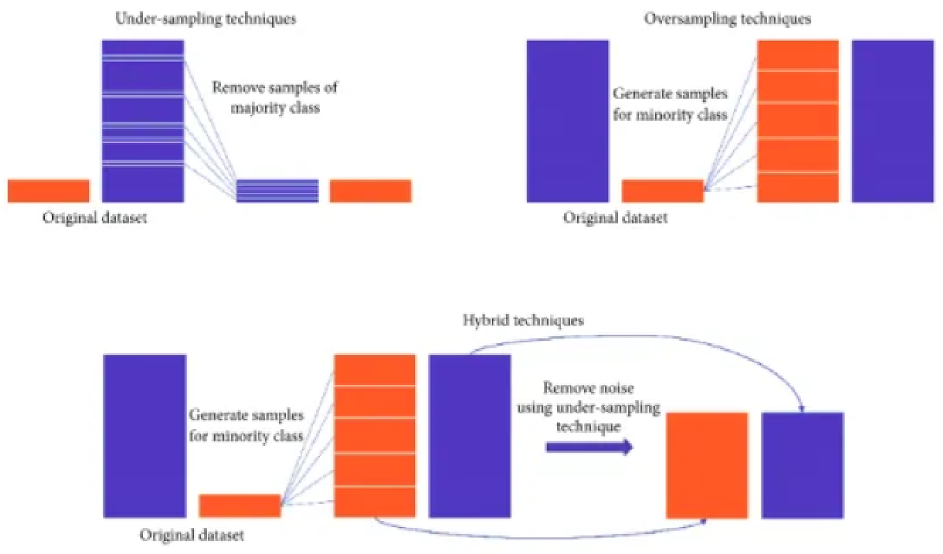

Under-sampling techniques
Remove samples of majority class
Original dataset
Oversampling techniques
Generate samples for minority class
Original dataset
Hybrid techniques
Generate samples for minority class
Remove noise using under-sampling technique
Original dataset

Figure 2.6: Understanding Resampling Methods[29].

## 2.11 Environment and Technologies

The software applications used for the implementation: Windows OS version 22H2
All tests were run on the same hardware specification:
Processor : 1.60GHz Intel Core i5-8365U, 64 Bit
Memory : 8GB
Used programming languages: Python-3.11, SQL
Used libraries: Scikit-learn
Used packages: NumPy, Pandas, Matplotlib

# Chapter 3

# Methodology for National Shipments

The methodology employed in this project encompasses several steps, each designed to facilitate the development and evaluation of the machine learning-based approach for automated identification of incorrectly labeled shipments.

## 3.1 Business Understanding

In the general process of parcel shipment within DHL Germany, data collection occurs at various stages. It all begins at the parcel drop-off or pickup stage, where the customer either drops off the parcel at a DHL service point or schedules a pickup. At this juncture, data regarding the sender, recipient, parcel size, and weight are likely to be collected and this data is known as PAN data. Following this, the parcel is transported to a local sorting center. At the sorting center, the parcel is sorted based on its destination, which may involve scanning barcodes or QR codes on the parcel. This process might involve recording data about the sorting procedure and the parcel's destination. Here, real weight of the parcel is recorded when the parcel is sent over the conveyor belt from the parcel centres only with weighing scales, also dimensions of the parcel and other information of the parcel known as PZA event. The sorted parcel then undertakes another journey, this time to the sorting center nearest to its destination. Like the previous transport stage, data about weight, dimensions, volume, shape and other features of parcel could be collected here as well. This event is known as PZE event. Data about the parcels such as weight, dimensions, transport vehicle, route, international border crossing, time and other relevant information could be collected. The parcel is then sorted for the final time at the destination sorting center before it is delivered to the recipient. For export parcels, after PZA event, it is transported to the destination country. Data about the parcels such as weight, dimensions, transport vehicle, route, international border crossing, time and other relevant information could be collected. The parcel is then sorted for the final time at the destination sorting center before it is delivered to the recipient.

The overall process can be represented in pictorial form as below:

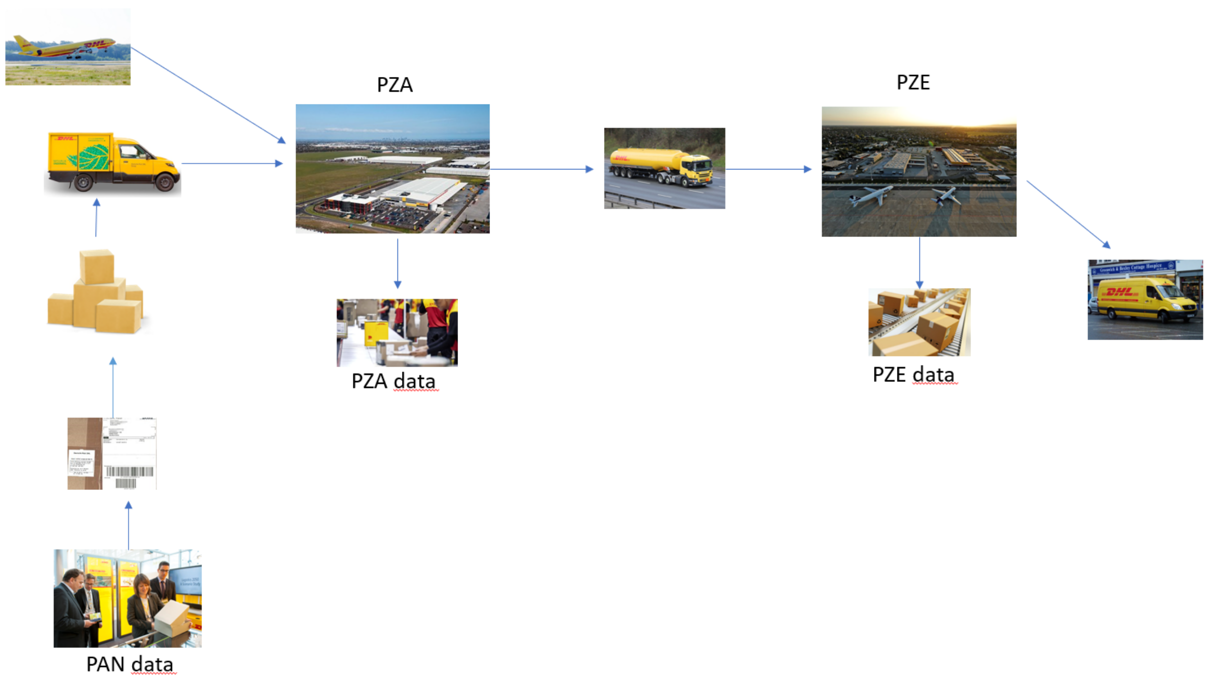


Figure 3.1: Shipment flow of national parcels

## 3.2 Data Sources and Description

In this section, the data sources used for data collection are described. There are different data sources from which all data for this research is collected. These data sources are linked to the software systems used in the operational process of DHL. These software systems also has a storage function as all shipment data are collected in specific data warehouse(DWH). The data sources used for this research can be distinguished on three shipment process levels: customer(PAN), origin parcel centre(PZA) and destination parcel centre(PZE). The data collection is done using Teradata SQL. By combining all available features of the three data source tables, these data features could be used to describe a shipment. The objective is to create new features using data features known from initially collected data. Moreover, these new features are useful in prediction tasks on parcel weights. Some data features cannot be used for this predictive objective because these features are not relevant for weight prediction. For instance, the final delivery time. Moreover, data features unrelated to the parcel weight, such as the house number should be excluded. The domain knowledge is used to extract the relevant information regarding parcel weights, which results in selection criteria that are used to make an initial feature selection on the available features. These initially selected features can be distinguished on shipment process level, i.e., customer, PZA and PZE level. All three dimensions could impact the probability of a parcel weighing more than 20kg.

### 3.2.1 Customer (PAN) Level

Every parcel gets a code which is a unique number when it is collected or registered by the customer commonly referred as ’sendungs_code’ throughout this project. The customer could update the PAN data leading to new information or data of the shipment with same sendungs_code but different data of the shipments. This leading to having redundant data, so I filtered them and selected the information with the latest date and time. Customer level data is aggregated per order, every row in the data source corresponds to a distinct shipment. As a result, the PAN level aggregation is equivalent to the shipment level aggregation. PAN level data, provides details on the shipment from the customer,such as weight, method code, ekp number, PAN ID, verfahren number and more. An overview of the extracted PAN level data features, including a description and data type, is shown in Table 3.1.

| **Feature name** | **Description** | **Type** |
|---|---|---|
| sendungs_code | Unique identifier of an shipment | INTEGER |
| PAN_ID | Unique Id of the the PAN information | INTEGER |
| ekp_no | customer number | INTEGER |
| Verfahren_Nr | Information of the parcel such as retoure, WaPo parcel or other | STRING |
| Teilnahme_Nr | Unique identifier of customer which is part of billing number | INTEGER |
| data_source_type_id | ID where the parcel was scanned and the information was created. Eg. 1 = parcel store, 2 = PZA, 3 = PZE, 5 = delivery | INTEGER |
| event_dt | Date when the customer given information about the parcel | DATE |
| Pruefziffer | Additional number of sendungs_code | INTEGER |
| Gewicht_PAN | Weight of parcel given by the customer | FLOAT |

Table 3.1: PAN data Feature Descriptions

The parcels when reached to PZA centre(origin ) or PZE centre(destination) we get more information for that parcels like real weight(if the parcel centre has the weighing scale), length, breadth, height, volume, parcel centre number as ERFASSUNGSORT, WEIGHT_LEGAL_CODE, Gewicht_Quelle and other information. Each feature from PZA or PZE event are explained below. As, I considered same features from both PZA and PZE event, these feature are described in single section.

### 3.2.2 PZA and PZE Level

Since PZA and PZE level data is aggregated per shipment, every row in these data source tables corresponds to a distinct order. Each row corresponds to a single shipment because only single shipments are listed. As a result, the PZA and PZE level aggregation is equivalent to the shipment level aggregation. PZA, or PZE level data, provides details on the shipment, including dimensions such as weight, length, breadth, height, volume, weighing scale, and route. An overview of the extracted PZA and PZE level data features, including a description and data type, is given in Table 3.2.

| **Feature name** | **Description** | **Type** |
|---|---|---|
| sendungs_code | Unique identifier of an order | VARCHAR |
| ERFASSUNGSORT | ID of the parcel centre | INTEGER |
| gewicht_real | The actual weight of the item as measured | FLOAT |
| gewicht_quelle | weighing scale information | STRING |
| volumen_pals | Volume of the parcel | INTEGER |
| laenge | Length of the parcel | INTEGER |
| breite | Breadth of the parcel | INTEGER |
| hoehe | Height of the parcel | INTEGER |
| ereignis_zeitstempel | Time stamp of parcel arrived or departed at the parcel centre | TIME |
| ereignis_datum | Date | DATE |
| ereignis_typ | type | STRING |
| zeit | Time | VARCHAR |
| ziel_pz | Destination Identification number of parcel centre | INTEGER |
| ZIELKENNUNG_AGNR | Destination Identification number of parcel centre in case ziel_pz column does not have data, it it recorded here | INTEGER |
| WEIGHT_LEGAL_CODE | Whether the parcel is Legal for trade or not | STRING |
| rc_produkt_code | Product code | INTEGER |
| routing_code | which route the parcel is shipped | INTEGER |

Table 3.2: PZA and PZE data Feature Descriptions

If the parcel with Gewicht_Quelle, either ’W’ or ’Y’, this indicates that parcel is gone through a parcel centre where we have weighing scales and therefore have a real weight. And I only considered parcels with WEIGHT_LEGAL_CODE = ’LFT’, which means these parcels are legal for trade.

## 3.3 Collection of Initial Data

The first step for building a machine learning model is to collect appropriate data.

For National project, a *for* loop is designed to iteratively collect data for a specified number of months, starting from the second previous month. Each iteration moves one month further back in time, allowing for the collection of historical data over the specified period. I have considered past 5 months data for creating history features, I run the loop for 5 months which are from ’July 2024’ to ’March 2024’ backward. For this project, the data is collected from various tables from various databases throughout the shipment process.The reason for running the loop and collecting data in many tables is that there is mot enough computing power to run it in a single query as the data is too big. As mentioned earlier in the business understanding section and data sources section above, the data is recorded from the customer to end delivery at three different stages. I took the parcels which are national by using where clause *method_code* in (’01’, ’06’) from PAN table and the parcels at PZA and PZE which have weighing scales for both training, testing and cross validation. These shipments are filtered by applying where clause *Gewicht_Quelle* in (’W’, ’Y’) and *WEIGHT_LEGAL_CODE* = ’LFT’ from PZA or PZE tables. Also I only considered the shipments whose weight and volume data are greater than 0 from PZA and PZE events. In addition, shipments from parcel centre (PZ=62) are excluded from the data because of inaccurate weighing information from weighing scale. The query used to collect and merge these tables from different data sources is listed in Appendix A.1, A.2, A.3, A.4. It can be seen that primary key sendungs_code, is used to join the datasets. On average I got around 40 million datasets for each month.

The visual understanding of how data is collected as base data table from these different sources is shown in below figure.

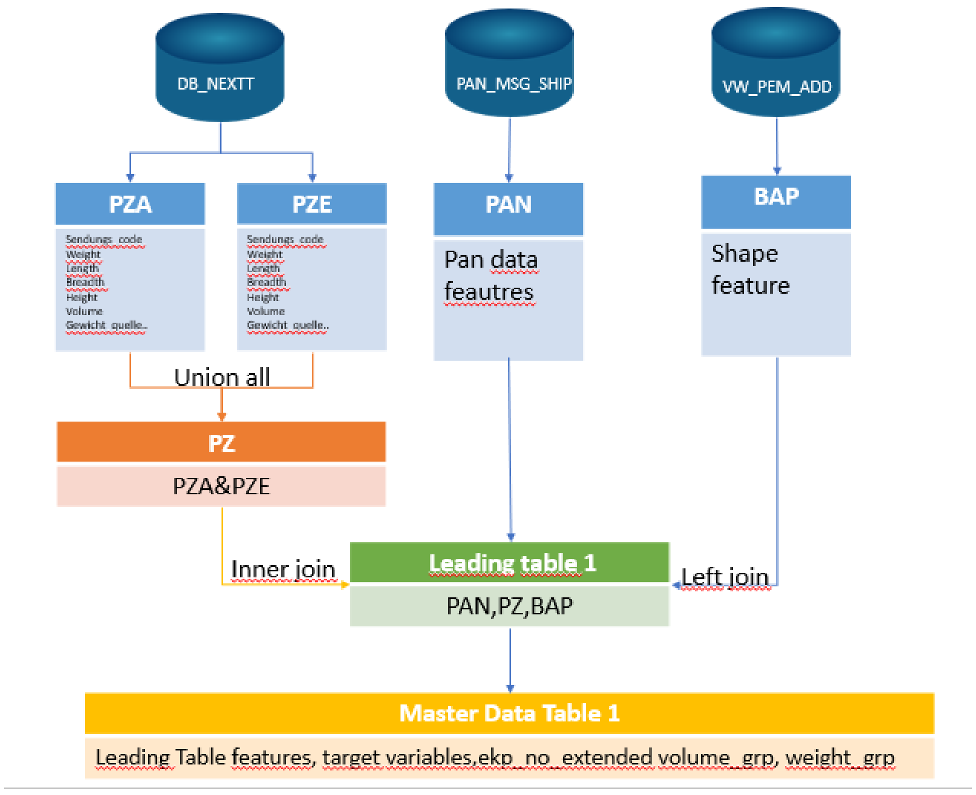


Figure 3.2: Initial collection of data into Base data Table

The above figure represents data collected for one month into single table which is Base Table 1. As I run the loop for 5 months I got five base tables each for respective month. I combined these five months data tables into single table which results into single main table by using *union all* function. After data collection, a raw dataset is obtained containing almost 2 billion shipments, described by 21 features. This raw data set is used for initial data analysis. The initial exploratory data analyses and obtained figures are performed in Python. For confidentiality the exact numbers are hidden.

## 3.4 Data Analysis

### 3.4.1 Exploration of Data

An exploratory data analysis is performed to determine the data quality and explore what data preprocessing is necessary. This exploratory data analysis is performed on five months of shipments (July 2024 and March 2024), with shipments having real weights. It was decided to extract the data in dedicated tables instead of using live data connections. This decision was made as the relevant data is historical and will not change. Moreover, using an active connection would put more strain on querying servers and local servers than necessary. I analyzed for some shipments the weight given by customers was leading to some outliers as the weight entered was recorded as Kg in our database, whereas the customer entered in grams. So with this analysis I could remove weight outliers by using the filter gewicht_pan less than or equal to 50 Kg. With this I considered the shipments whose maximum weight of the parcel entere by customer is 50Kg. Also it was found there are some outliers whose specific weight is more than 5 which is mereley not possible to have parcels weighing more than 5kg per unit volume, so I filtered these outliers by selecting shipments whose specific weights are less than or equal to 5. This exploratory data also helped understanding data types of each features, and noticed three features gewicht_pan_grp, volumen_grp and parcel_shape are categorical therefore need to be transformed into numerical values as the machine learning models I will be using does not accept categorical data as input.

### 3.4.2 Verifying the quality of data

With verifying he quality of data analysis, I confirmed that all the data collected initially were correct.

1. First analysis was to check of all the parcels collected were national or not by using the using 'group by method_code' function. The results were either '01' or '06', this confirms the parcels are delivered only within Germany.
2. I analysed if all the parcels taken are from the Parcels centre with weighing scale or not as the data I considered is only from the parcel centres with weighing scales by using 'group by gewicht_quelle' function which performs grouping all the parcels by gewicht_quelle. This query results only either by 'W' or 'Y', then I concluded that all the data collected was having real weight.
3. Analysed if all the parcels are legal for trade or not, by using 'group by' function which performs grouping of parcels by *WEIGHT_LEGAL_CODE*. This query result were only 'LFT' then I finalized that all the data collected are legal for trade.

## 3.5 Feature Engineering

Once the data is collected, relevant features were selected or engineered to serve as input variables for the machine learning models, and also creation of new relevant features using the existing features from history data. Features may include product descriptions, weights, volumes, dimensions like length, breadth and height, parcel shape,origins, destinations, shipping labels, and any other attributes deemed informative for the task of classifying the weights of the parcels more than 20kg. Feature selection techniques such as feature creation, statistical analysis, domain knowledge, and feature importance algorithms were employed to identify the most predictive variables.

### 3.5.1 Initial Feature Creation

1. *Target Features*: In this section, I define the target variable *Flag_Gewicht_20*, which is crucial for our machine learning model. This binary variable is derived from the weight data (Gewicht_Real and Gewicht_PAN) and is used to categorize the data into two distinct classes for further analysis. The target variable *Flag_Gewicht_20* is designed to identify whether the weight is greater then 20kg but the customers labeled the parcel with a weight less then 20kg. The threshold of 20 units (presumably kilograms) is chosen based on new regulations from the government as well as business requirement. I also created another target variable *Flag_Gewicht_10* which is not important for this thesis, but might be useful for future requirements or future developments. This target variable *Flag_Gewicht_10* is designed to identify whether the weight of an item meets a threshold of 10kg. These two target variables are binary variable, so I use them for classification models. Another feature *Gewicht_Real* is considered as target variable for regression model where the model predicts the real weight of the parcel based on predictive features.

2. *EKP_NO_EXTENTED*: I created the feature *EKP_NO_EXTENTED*, which is derived from the concatenation of EKP_No, verfahren_no and Teilnahme_no. This feature is used to create a unique identifier by combining the above mentioned features which serves as unique identifier of the customer and used for data aggregration in later history feature creation.

3. *Volumen_grp*: I created the new feature *volumen_grp*, which is created by categorizing volume measurements into predefined groups. This feature is used to simplify and segment continuous volume data into meaningful categories for analysis and modeling. The feature *volumen_grp* is designed to facilitate the analysis of volume data by grouping it into discrete categories. The specific volume thresholds are chosen based on distribution of data and domain knowledge. By clearly defining volume categories, I can gain better insights and make more informed decisions based on volume data.The creation of the volumen_grp feature involves categorizing the volume (volumen) into the following groups:

   *'V30'*: Volumes less than 30. *'V40'*: Volumes between 30 and 39. *'V50'*: Volumes between 40 and 49. *'V70'*: Volumes between 50 and 69. *'V90'*: Volumes between 70 and 97. *'V99'*: Volumes 98 and above. *'?'*: A fallback category for any other cases, ensuring that the feature always has a value.

4. *Gewicht_PAN_grp*: I created the feature *Gewicht_PAN_grp*, which is created by categorizing weight measurements (Gewicht_PAN) into predefined groups. This feature helps in simplifying and segmenting continuous weight data into meaningful categories for analysis and modeling. The feature *Gewicht_PAN_grp* is designed to facilitate the analysis of weight data by grouping it into discrete categories. This categorization helps in

understanding and interpreting the weight data more easily and can also improve the performance of machine learning models by reducing the complexity of continuous weight measurements. The specific weight thresholds are chosen based on distribution of data and domain knowledge.

### 3.5.2 Features creation on Data aggregation

This section describes the process of creating and aggregating features from the master data table. The resulting features are stored in the separate history table. The aggregation is based on the combination of*EKP_NO_EXTENTED*, volumen_grp, and parcel_shape fields. The data is sourced from the master data table, which contains detailed records of shipment events. The subset of records for aggregation is determined by the presence of *EKP_NO_EXTENTED*
Event and Weight Counts
in another table where the total number shipments count is considered to greater than 3 for each *EKP_NO_EXTENTED*. The detailed explanation of new features are described below:

1. **Event and Weight Counts:**

   *Zeitraum_Tage:* Number of distinct event dates, representing the active period in days.
   *D_Anz_diff_pan_weights:* Number of distinct PAN weights.
   *D_Anz_diff_Real_weights:* Number of distinct Real weights.
   *Anzahl:* Total count of shipment codes.

2. **Average Measurements:**

   *D_Hoehe, D_Laenge, D_Breite:* Average height, length, and width of the parcels.
   *D_Gewicht_real, D_Gewicht_pan:* Average real and PAN weights.
   *D_Volumen:* Average volume.

3. **Min/Max Measurements:**

   *D_hoehe_min, D_hoehe_max:* Minimum and maximum height.
   *D_laenge_min, D_laenge_max:* Minimum and maximum length.
   *D_breite_min, D_breite_max:* Minimum and maximum width.
   *D_Gewicht_real_min, D_Gewicht_real_max:* Minimum and maximum real weight.
   *D_gewicht_pan_min, D_gewicht_pan_max:* Minimum and maximum PAN weight.
   *D_Volumen_min, D_Volumen_max:* Minimum and maximum volume.

4. **Derived Features:**

   *D_Volumengewicht_real, D_Volumengewicht_pan:* Volume-weight ratios for real and PAN weights.
   *D_LBH_Summe:* Sum of length, breadth, and height.
   *D_Volumengewicht_real2, D_Volumengewicht_pan2:* Adjusted volume-weight ratios.
   *D_Gewichtsabweichung, D_Gewichtsabweichung_min & D_Gewichtsabweichung_max:*

Weight deviations.

5. **Flag Counts and Proportions:**

   *Anzahl_GU_20, Anzahl_GU_10:* Counts the errors the customer made in the history. Counts number of time the customer gave as a parcel with a label less the 20kg but in reality the parcel was greater then 20kg and labeled less than 10kg but in reality the parcel was greater than 10kg respectively.
   *D_Anteil_GU_20, D_Anteil_GU_10:* Perceentage of shipments under specific weight flags.
   *Anzahl_pan20_korrekt, Anzahl_pan10_korrekt, Anzahl_pan_korrekt:* Counts of correctly flagged PAN weights.
   *Anteil_PAN20_korrekt, Anteil_PAN10_korrekt, Anteil_PAN_korrekt:* Proportions of correctly flagged PAN weights.

6. **Weight Difference:**

   *Diff_kg_sum:* Sum of the differences between real and PAN weights, with a condition to set small average differences to zero.

By grouping data based on EKP_NO_EXTENTED, volumen_grp, and parcel_shape aggregations, a rich set of features that capture essential aspects of the shipment data were created. These features are then joined to training data that can be used for predictive modeling. Now I created features on EKP_NO_EXTENTED, volumen_grp, and parcel_shape aggregrate level, I filtered only one month data from the main table for training the model. To this table I joined these features using *left join* created above on aggregate levels resulting in Training table 1. This training table 1 consists of initially collected 21 features from raw data table and aggregated history features.

### 3.5.3 Final Features Creation

In this section, I described the process of creating new features by calculating ratios and derived values from existing measurements. These new features are intended to capture relationships between different physical dimensions and weights of parcels, which can be valuable for predictive modeling.

1. **Ratios Involving Length, Height, and Breadth:**

   *VH_Laenge_Hoehe:* Ratio of length to height.
   *VH_Laenge_Breite:* Ratio of length to breadth.
   *VH_Hoehe_Breite:* Ratio of height to breadth.

2. **Ratios Involving Volume and Dimensions:**

   *VH_Volumen_Laenge:* Ratio of volume to length.
   *VH_Volumen_Breite:* Ratio of volume to breadth.

*VH_Volumen_Hoehe:* Ratio of volume to height.

3. **Ratios Involving Dimensions and Deviations:**

   *VH_Laenge_DLaenge:* Ratio of length to its deviation.
   *VH_Breite_DBreite:* Ratio of breadth to its deviation.
   *VH_Hoehe_DHoehe:* Ratio of height to its deviation.
   *VH_Volumen_DVolumen:* Ratio of volume to its deviation.

4. **Weight Ratios and Deviations:**

   *VH_Gewicht_DGewicht:* Ratio of PAN weight to real weight.
   *VH_Gewicht_DGewicht_PAN:* Ratio of PAN weight to average PAN weight.
   *DGewichtsabweichung1, DGewichtsabweichung2, DGewichtsabweichung3, DGewichtsabweichung4, DGewichtsabweichung5, DGewichtsabweichung6:* Various weight deviations.

5. **Sum of Dimensions and Related Ratios:**

   *LBH_Summe:* Sum of length, breadth, and height.
   *LBH_VH:* Adjusted sum of dimensions ratio.

6. **Volume-Weight Ratios:**

   *Volumengewicht_pan, Volumengewicht_pan2, Volumengewicht_pan3:* Various volume-weight ratios.
   *VH_Volumengewicht_pan, VH_Volumengewicht_pan2:* Adjusted volume-weight ratios.

All these above new features are joined using left join to Training table 1 leading to Final training table.

## 3.6 Loading Data from Teradata

Efficient data import is crucial for handling large datasets in machine learning projects. This section describes the process of importing data from a Teradata database into a Python environment, including timing the operation to measure performance. The dataset is sampled to ensure manageability during initial explorations and development stages. The time module is used to measure the duration of the data import process. All the data from the final training table is downloaded from teradata into data frame *GS_trainset*. The time module is used to capture the start time before the import and the end time after the import to calculate the total duration of the import process. A sample of 5,000,000 shipments only was imported from the specified table due to memory and run time limitations. This subset is used for further data cleaning and transformation followed by training. For better understanding on how these features were created and joined to training table can be visullay represented as diagram below:

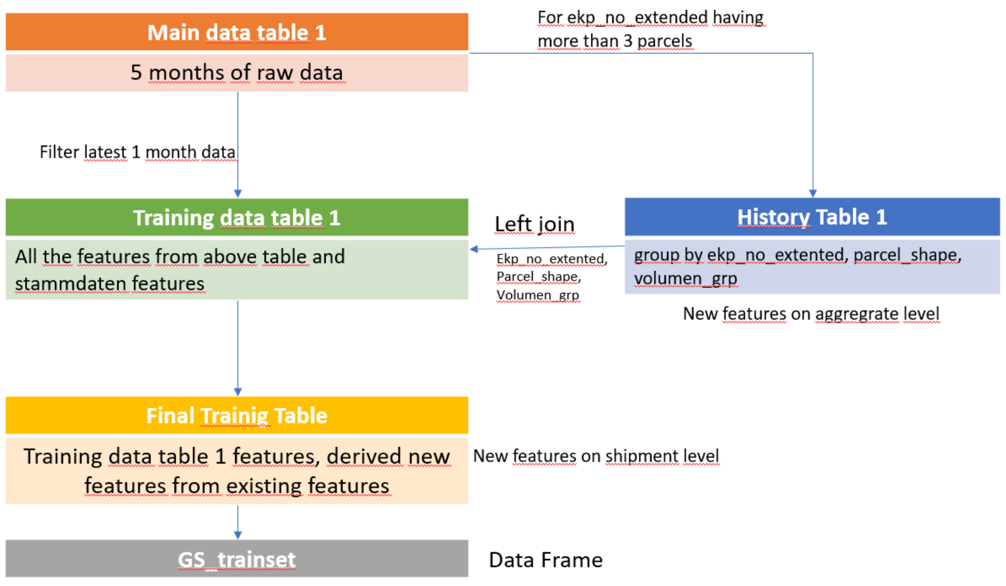

Main data table 1
5 months of raw data
For ekp_no_extended having more than 3 parcels
Filter latest 1 month data
Training data table 1
All the features from above table and stammdaten features
Left join
Ekp_no_extented, Parcel_shape, Volumen_grp
History Table 1
group by ekp_no_extented, parcel_shape, volumen_grp
New features on aggregrate level
Final Trainig Table
Training data table 1 features, derived new features from existing features
New features on shipment level
GS_trainset
Data Frame

Figure 3.3: training data collection

Next step is to clean the data, for which I downloaded dataset, that is prepared and manipulated using MySQL, into dataframe which is used to train and test the prediction model.

## 3.7 Data Cleaning

The quality of the data has a major impact on how well a prediction model works. Handling missing numbers and addressing data errors reduces noise in the data. This is known as data cleaning. The encoding of features into the appropriate data format and data creation are included in the data transformation. An outcome from the data cleaning stage is a dataset was utilised in the modelling stage that follows.

### 3.7.1 Dropping irrelevant features

Before training a machine learning model, it is crucial to clean and prepare the dataset. This involves removing any columns that are not needed for the training process. Dropping unnecessary columns helps to reduce the dimensionality of the dataset, potentially improving model performance and training efficiency. I identified a set of columns that are not relevant for the training process. These columns include identifiers, dates, and other attributes that do not contribute to the predictive power of the model. The *drop* method from the pandas library is used to remove these columns from the dataset. The drop_col0 list contains the names of columns that are deemed unnecessary for the training process. This process ensured that the resulting model was built on a more focused and meaningful set of features, improving both its performance and interpretability. To ensure the integrity of the analysis, a systematic process was followed to identify and remove the irrelevant features. This process involved a combination of domain knowledge, statistical techniques, and feature importance analysis. Domain knowledge played a crucial role in identifying 13 features that were logically unrelated to the problem at hand. These 13 features, although present in the original dataset, were determined to have no direct impact on the target variable and were therefore dropped.The code in python used to drop these features are listed in Appendix A.8. This step ensures that the dataset is clean, concise, and ready for the subsequent modeling process.

### 3.7.2 Missing Values

Appendix A.7, display the number of missing values for each feature. It is evident that, when joining the historical features on sendungs_code to the main data, only the features on *ekp_no_extended*, *volume_grp*, and *parcel_shape* aggregration have a high percentage of missing values which counts to 53884. The supervisor's discussion suggested that the new customers not having historical data is linked to the degree of missingness. For these features, zero is assigned to the missing values. A large dataset has a low percentage of missing values (per feature) after imputing. There are multiple ways to deal with missing values in the numerical. The numerical could handle missing values in a number of ways. The most popular techniques are:

1. Filling null values with zero, this simple, implementable strategy fills null values with zero. Simple methods, such as *'fillna(0)'*, are available in most programming languages and libraries to replace null values with zeros. Time and effort can be saved by this simplicity, particularly when working with big datasets.
2. Delete the observations; in this method, the rows with missing data are eliminated. But this approach can lead to the loss of important data, and it works best when the amount of missing data is small.
3. Using the mean or mode value in place of missing values is another method for handling null values. Because all observations with missing data have the same value, filling in the missing values with the mean or mode might help the observations look more similar to one another. This may cause the similarity of observations to be overestimated and negatively impact the performance of some models, which is not what is wanted.

Due to its benefits, I decided to use the first approach of filling with null values.

### 3.7.3 Data Transformation

Apart from numerical data, I also have three features *parcel_shape*, *volume_grp* and *gewicht_pan_grp* with categorial values. So I have transform these categorial values into numerical values. A number of machine learning models require input variables to have numerical values. In order to incorporate categorical variables into machine learning models, it is crucial to transform these values. In this project, the categorical variables are nominal instead of ordinal. Ordinal encoding and one-hot encoding are often used techniques for data transformation of categorical data. All of the distinct categories are changed into dummy variables during one-hot encoding. For this category, instances will receive a value of 1, but for all other categories, they will receive a value of 0. Conversely, ordinal encoding designates an integer to every category that is, if it is known how many categories are there. One-hot encoding is said to improve performance, even at the expense of larger dimensional data, according to research. Therefore, one-hot encoding is used to transform all categorical features.

### 3.7.4 Data Normalization

Normalization is a crucial preprocessing step in machine learning that ensures each feature contributes equally to the model. For training a Support Vector Machine (SVM) model, it is especially important to scale the features, as SVMs are sensitive to the magnitude of the input data. To normalize the data, I used the *StandardScaler* from the sklearn.preprocessing module, which standardizes features by removing the mean and scaling to unit variance. This process transforms the features to have a mean of zero and a standard deviation of one. After applying the normalization process, each feature in the training dataset has been transformed to have a mean of zero and a standard deviation of one. This standardization ensures that features with larger magnitudes do not dominate the training process of the SVM model, leading to improved model performance and stability.

## 3.8 Defining Target Variables

In machine learning, it is essential to clearly distinguish between features (input variables) and target variables (output variables) before training a model. This section describes the process of defining target variables and preparing the dataset by dropping these target variables from the feature set. *Flag_Gewicht_20*: The target variable *Flag_Gewicht_20* is binary feature, used to identify whether the weight is greater then 20kg but the customers labeled the parcel with a weight less then 20kg target variable for classification models. Another feature *Gewicht_Real* is considered as target variable for regression model where the model predicts the real weight of the parcel based on predictive features. So our main target variable for this project is flag_gewicht_20 columns which is used for classification model as output and gewicht_real column as target variable for regression model. After defining these target variables, I dropped these columns from feature set which I used for training the machine learning model. After dropping these three target variables, finally I have 87 features in training set based on which our model predicts the output.

For National shipments, after data pre-processing, I considered the final dataset size of 5M with 87 predictive features and three target variables. Each row represents one parcel. This dataset used for modeling contains parcels with real weight greater than 20kg which represents 0.3% of the total dataset.

## 3.9 Train-Test split

To evaluate the performance of the machine learning model, it is essential to split the dataset into training and testing sets. The training set is used to train the model, while the testing set is used to assess the model's performance on unseen data. I divided our dataset into two distinct subsets: a training set and a test set. The test set was allocated 30% of the total data (test_size = 0.3), while the remaining 70% was used for training the model. The supervised machine learning classifiers in this project are learned on a labeled training set. It is chosen to use a 70% training and 30% test data split because of the small sample size of minority instances. The 70% training data is used for training and hyperparameter tuning, while 30% remains untouched and is solely used as a test set. It is expected that a large training size, with many minority instances, results in a better understanding of the underlying relationships of the data. Moreover, it is expected that 30% test data provides a valid performance indication because the dataset size is sufficiently large.

## 3.10 Training

A Random Forest Classifier was initialized with a random_state set to 33 to ensure reproducibility and n_jobs set to 44 to utilize parallel processing across 44 CPU cores, thereby enhancing computational efficiency. The classifier was trained on the provided training dataset (X_train, y_train) using the fit method. Post-training, the model's performance was assessed on the test dataset (X_test, y_test) by employing a custom evaluation function designed to compute accuracy, precision, recall and F1 score. This methodology ensured a structured and reproducible approach to model training and evaluation while leveraging parallel processing capabilities for efficiency.

# Chapter 4

# Methodology for International Shipments

For international parcels,there are two cases, one is import parcels and other is export parcels. For this project, we only need import parcels to take into consideration as the new law applies to the parcels within Germany. So all the parcels imported to Germany are considered. we divided the international import parcel weight prediction model into two parts:

1. First part is where we don't have a PAN-Dataset(Data from the customers). These customers were referred as Private customer and we developed a model for these international private customers as a part of this thesis.
2. For all other import parcels we should have a PAN-Dataset. These customers are referred as Business Customers. Due to lack of time, a model for this part is not developed under this thesis, it will be developed in future and therefore considered as Future work.

## 4.1 Business Understanding

The process of importing parcels into Germany and their subsequent delivery is an intricate system involving multiple stages. Understanding this process is vital for developing a machine learning model to predict the weight of imported parcels accurately. This section provides a comprehensive overview of this process.

1. **Parcel Dispatch from the Origin Country:** Parcels are collected from senders, which could be businesses or individuals, through various DHL collection points or pick-up services. As a initial processing, at the origin country's DHL facilities, parcels are sorted and tagged with unique tracking numbers. This information is fed into DHL's global tracking system.
2. **International Transportation:** Before leaving the origin country, parcels undergo customs clearance to ensure compliance with export regulations. Depending on the service level and urgency, parcels are transported by air or sea.
3. **Arrival in Germany:** Upon arrival in Germany, parcels are sent to DHL's central sorting

hub. The sorting centre is based on the origin country. At the sorting hub, parcels are scanned, sorted, and routed based on their final destinations within Germany. Here we get information of parcel such as weight, dimensions, volume and others. This information is stored in database under PZA event data. When the parcel is reached to the final destination parcel centre, the parcels are agained scanned, sorted where we get information of the parcel stored in database as PZE event data.

So all these import parcels with weight grater than 20kg should be ejected from the conveyor belt at destination parcel centre. For this a machine learning model is developed using the data from the parcel centre at the initial sorting to predict the parcel weight before it reaches the destination parcel centre. As we do not have any information from the customer regarding parcel, the model has insufficient data to predict if it is more than 20kg. So we use slightly different approach to collect data and creating history features when compared to national project.

## 4.2 Data Sources and Description

In this section, the data sources used for data collection are described. There are different data sources from which all data for this research is collected. These data sources are linked to the software systems used in the operational process of DHL. These software systems also has a storage function as all shipment data are collected in specific data warehouse(DWH). The data sources used for this research can be distinguished on two shipment process levels: initial parcel centre after arrival to Germany(PZA) and destination parcel centre(PZE). The data collection is done using Teradata SQL. By combining all available features from these data source tables, these data features could be used to describe a shipment. The objective is to create a new features using data features known from initially collected data. Moreover, these new features are useful in prediction tasks on parcel weights. Some data features cannot be used for this predictive objective because these features are not relevant for weight prediction. For instance, the final delivery time. Moreover, data features unrelated to the parcel weight, such as the house number should be excluded. The domain knowledge is used to extract the relevant information regarding import parcel identifiers, weights, which results in selection criteria that are used to make an initial feature selection on the available features. All these dimensions could impact the probability of a parcel weighing more than 20kg. As we do not have any information from the customer, we do not have any PAN data like in national data. So we considered the data from PZA and PZE parcel centres without weighing scales for these import parcels as PAN data.

| **Feature name** | **Description** | **Type** |
|---|---|---|
| sendungs_code | Unique identifier of the shipment | INTEGER |
| ereignis_datum | Date when the parcel arrived the parcel centre in germany after arrival | DATE |
| Gewicht_PAN | Weight of parcel from parcel centre without weighing scale | INTEGER |

Table 4.1: PAN Data Description

The parcels when reach to PZA centre(origin) or PZE centre(destination) we get more information for that parcels like real weight, length, breadth, height, volume, parcel centre number as ERFASSUNGSORT, WEIGHT_LEGAL_CODE, Gewicht_Quelle and other information. Each feature from PZA or PZE event are explained below. As, we consider same features from both PZA and PZE event, these feature are described in single section.

### 4.2.1 PZA and PZE Level Data Description

Since PZA and PZE level data is recorded per order, every row in these data source tables corresponds to a distinct order. Each row corresponds to a single shipment because only single shipments are listed. As a result, the PZA and PZE level aggregation is equivalent to the shipment level aggregation. PZA, or PZE level data, provides details on the shipment, including dimensions such as weight, length, breadth, height, volume, weighing scale, and route. An overview of the extracted PZA and PZE level data features, including a description and data type, is given in Table 4.2.

| **Feature name** | **Description** | **Type** |
|---|---|---|
| sendungs_code | Unique identifier of an order | INTEGER |
| ERFASSUNGSORT | Identification number of parcel centre in Germany | INTEGER |
| LEIT_PRODUKT_CODE | ID of the post product. E.g 80 = international parcel, 10 = bulky good, 0 = normal national parcel. | INTEGER |
| gewicht | The actual weight of the item as measured depending on gewicht quelle data | INTEGER |
| gewicht_quelle | Information where weight data is coming from. If it W or Y then weight is from weighing scale | STRING |
| volumen_pals | Volume of the parcel | INTEGER |
| laenge | Length of the parcel | INTEGER |
| breite | Breadth of the parcel | INTEGER |
| hoehe | Height of the parcel | INTEGER |
| ereignis_zeitstempel | Time stamp of parcel arrived or departed at the parcel centre | STRING |
| ereignis_datum | Date | DATE |
| ereignis_typ | type | STRING |
| zeit | Time | STRING |
| ziel_pz | Destination Identification number of parcel centre | INTEGER |
| ZIELKENNUNG_AGNR | ID of destination parcel centre if it is not recorded in ziel_pz column. | INTEGER |
| WEIGHT_LEGAL_CODE | Whether the order is Legal for trade or not | STRING |

Table 4.2: PZA and PZE data Feature Descriptions for import parcels

If the parcel with Gewicht_Quelle, either 'W' or 'Y', this indicates that parcel is gone through a parcel centre where we have weighing scales and therefore have a real weight. And we only considered parcels with WEIGHT_LEGAL_CODE = 'LFT', which means these parcels are legal for trade.

## 4.3 Collection of Initial Data

The critical part of developing a machine learning model for import parcels was to identify import parcels as the database has combined data of national and international export and import parcels together. For identifying these parcels,I seeked the help of experts with better domain knowledge of international parcels. From their inputs I filtered international import parcel for private customer by using following filters. First identifed the retour parcels which are not needed for training the model. Then in other query select the import parcels excluding the above selected re-toure parcels. Select the shipment with Method_code = 87, Target_country = DE and Product_code in (82, 96). The code for identifying the international import parcels are listed in Appendix A.10. International import parcels are identified by following the below criteria.

Criteria for postal import parcels:

1. Sendung_code, starts with C, do not ends with DE
2. Sendung_code, starts with EP, ends with CH or IT
3. Sendung_code, starts with EE or EK, ends with GB
4. Sendung_code, starts with EH, ends with GB (these are usually shipments from Germany that were undeliverable in GB, but for which the original IDC is unfortunately no longer recognizable)
5. Sendung_code, starts with EG-EO or EQ-ER, ends with IE

**Note:** If only the shipment numbers are used then, DHL Returns International will also be recorded. These returns can be recognized on the Product code 74, on the ORI-PAN and on dedicated prefixes from the different sending countries. Import packages that run through the IPZ (I+II) are marked there, i.e. no rejection is necessary. Import parcels with customs import duties (Product Code 22) that pass through Speyer or Leipzig are expected to be marked there, i.e. no discharge necessary. As mentioneed earlier, the retoure parcels are identified with these conditions and eliminated when collecting required import parcels.

The visual understanding of how data is collected as base data table from these different sources is shown in below figure.

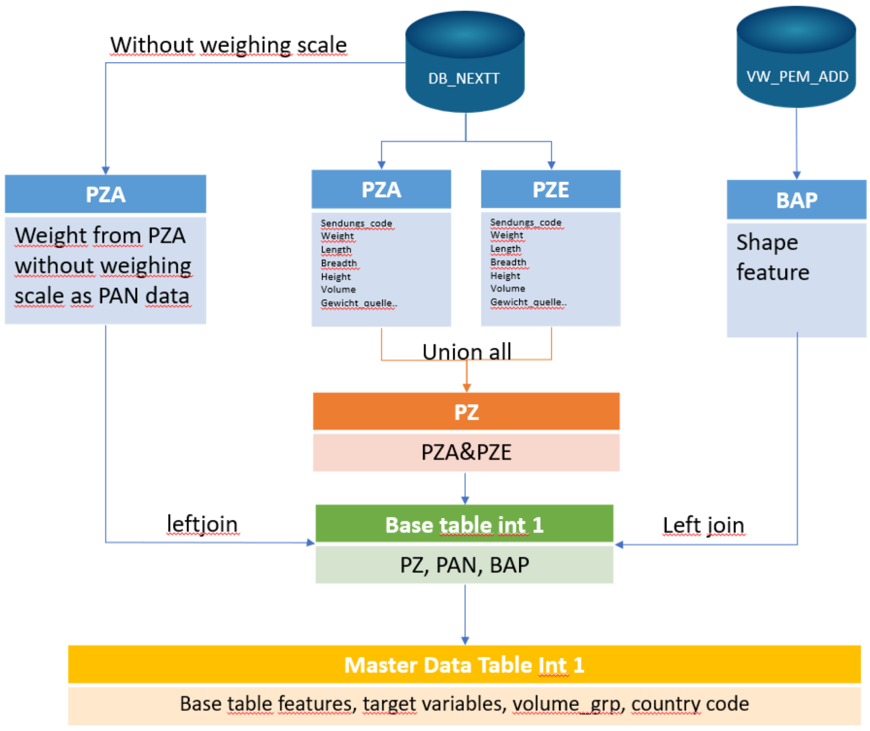


Figure 4.1: Initial collection of data into Base data Table

The above figure represents data collected for one month into single table which is Base Table 1. For international project, a *for* loop is designed to iteratively collect data for 10 months, starting from the second previous month. Each iteration moves one month further back in time, allowing for the collection of historical data over the specified period. We have considered past 10 months data for creating history features, we run the loop for 10 months which are from 'July 2024' to 'October 2023' backward. For this project, the data is collected from various tables from various databases throughout the shipment process. As mentioned earlier in the business understanding section and data sources section above, the data is recorded from the parcel centre in germany after import to end delivery at different stages. We took the parcels which are international by using where clause *Leit_Product_code* not in ('72', '73') from PZA, PZE, PZA_Hist, PZE_Hist, tables which have weighing scales for both training, testing and cross validation. These shipments are filtered by applying where clause *Gewicht_Quelle* in ('W', 'Y') from PZA or PZE tables. Also we only considered the shipments whose weight and volume data are greater than 0 from PZA and PZE events. The query used to collect and merge these tables from different data sources is listed in Appendix A.11, A.12, A.13, A.14, A.15. It can be seen that primary key sendungs_code, is used to join the datasets. On average we got around 85000 datasets for each month. By combining all these datasets for 10 months, the final table consists of 1M datasets which is used to create history features and also for training.

## 4.4 Data Analysis

### 4.4.1 Exploration of Data

An exploratory data analysis is performed to determine the data quality and explore what data preprocessing is necessary. This exploratory data analysis is performed on ten months of shipments (July 2024 and October 2023), with shipments having real weights. It was decided to extract the data in dedicated tables instead of using live data connections. This decision was made as the relevant data is historical and will not change. Moreover, using an active connection would put more strain on querying servers and local servers than necessary. Randomly selcted sendungs_codes from this raw data table after initial data collection were analyzed using another software tool (internal) for better analysis of shipment data, which contains data of each parcel at all different levels in the shipment process. This exploratory data also helped understanding data types of each features, and noticed three features country_code, volumen_grp and parcel_shape are categorical therefore need to be transformed into numerical values as the machine learning models we will be using does not accept categorical data as input.

### 4.4.2 Verifying the quality of data

With verifying he quality of data analysis, we confirmed that all the data collected initially were correct.

1. First analysis was to check of all the parcels collected were international or not by using the using 'group by method_code' function. The results were either '72' or '73', this confirms the parcels are international.
2. we analysed if all the parcels taken are from the parcel centres with weighing scale or not as the data we considered is only from the parcel centres with weighing scales by using 'group by gewicht_quelle' function which performs grouping all the parcels by gewicht_quelle. This query results only either by 'W' or 'Y', then we concluded that all the data collected was having real weight.

## 4.5 Feature Engineering

Once the data is collected, relevant features will be selected or engineered to serve as input variables for the machine learning models, and also creation of new relevant features using the existing features from history data. Features may include product descriptions, weights, volumes, dimensions like length, breadth and height, parcel shape,origins, destinations, shipping labels, and any other attributes deemed informative for the task of classifying the weights of the parcels more than 20kg. Feature selection techniques such as feature creation, statistical analysis, domain knowledge, and feature importance algorithms may be employed to identify the most predictive variables.

### 4.5.1 Initial Feature Creation

1. *Target Features*: In this section, we define the target variable *Flag_Gewicht_20*, which is crucial for our machine learning model. This binary variable is derived from the weight data (Gewicht_Real and Gewicht_PAN) and is used to categorize the data into two distinct classes for further analysis. The target variable *Flag_Gewicht_20* is designed to identify whether the weight of an item meets a threshold of 20kg. The threshold of 20 kilograms is chosen based on new regulations from the government as well as business requirement. We also created another target variable *Flag_Gewicht_10* which is not important for this thesis, but might be useful for future requirements or future developments. This target variable *Flag_Gewicht_10* is designed to identify whether the weight of an item meets a threshold of 10kg. These two target variables are binary variable, so we use them for classification models. Another feature *Gewicht_Real* is considered as target variable for regression model where the model predicts the real weight of the parcel based on predictive features.
2. *volumen_grp*: This feature is same as we defined in national project. This feature we used as one of aggregrating features to create history features data.
3. *Country_code:*The sendungs_code is a unique identifier for each parcel, and it contains embedded information, including the country code of the parcel's origin. By extracting this information, we can create a new feature called country_code, which can be used for further analysis and modeling.

### 4.5.2 Aggregated Features Creation from data with weighing scale

The data is grouped by country_code, volumen_grp, and parcel_shape.

1. **Count and Days Calculation:**

   *Anzahl_GM:* Counts the number of parcels (sendungs_code).
   *anzahl_days:* Counts the number of distinct days (EREIGNIS_DATUM) parcels were recorded. If no days are counted, it defaults to 1.

2. **Daily Average Count:**

   *D_Anzahl_GM:* Computes the daily average count of parcels by dividing Anzahl_GM by anzahl_days.

3. **Height Metrics:**

   *D_Hoehe_GM, D_hoehe_min_GM, D_hoehe_max_GM:* Average, minimum, and maximum height of parcels (hoehe).

4. **Length Metrics:**

   *D_Laenge_GM, D_laenge_min_GM, D_laenge_max_GM:* Average, minimum, and maximum length of parcels (laenge).

5. **Width Metrics:**

*D_Breite_GM, D_breite_min_GM, D_breite_max_GM:* Average, minimum, and maximum width of parcels (breite).

6. **Weight Metrics:**

   *D_Gewicht_real_GM, D_Gewicht_real_min_GM, D_Gewicht_real_max_GM:* Average, minimum, and maximum real weight of parcels (gewicht_real), ensuring a minimum value of 1 for the average.
   *D_Gewicht_pan_GM, D_gewicht_pan_min_GM, D_gewicht_pan_max_GM:* Average, minimum, and maximum of assumed pan weight of parcels(gewicht_pan).

7. **Volume Metrics:**

   *D_Volumen_GM, D_Volumen_min_GM, D_Volumen_max_GM:* Average, minimum, and maximum volume of parcels (volumen), ensuring a minimum value of 1 for the average.

8. **Volume Weight Metrics:**

   *D_Volumengewicht_real_GM:* Calculated as the ratio of real weight to volume, multiplied by 100.
   *D_Volumengewicht_pan_GM:* Similar calculation using the alternative weight measure.

9. **Dimension Sum and Volume Weight Ratios:**

   *D_LBH_Summe_GM:* Sum of length, width, and height, constrained between 4 and 20.
   *D_Volumengewicht_real2_GM:* Ratio of the dimension sum to the real weight.
   *D_Volumengewicht_pan2_GM:* Similar ratio using the alternative weight measure.

10. **Weight Deviation:**

    *D_Gewichtsabweichung_GM:* Difference between the real weight and the alternative weight, constrained between -20 and 20.

11. **Weight Flag Counts and Percentages:**

    *Anzahl_GU_20, Anzahl_GU_10:* Counts of parcels flagged for weight deviations of 20 and 10, respectively.
    *D_Anteil_GU_20, D_Anteil_GU_10:* Percentages of these flagged parcels relative to the total.

12. **Dimension Differences:**

    *D_laenge_diff_GM, D_breite_diff_GM, D_hoehe_diff_GM:* Differences between maximum and minimum values for length, width, and height.
    *D_Volumen_diff_GM:* Difference between maximum and minimum volume.
    *D_gewicht_pan_diff_GM, D_gewicht_real_diff_GM:* Differences between maximum and minimum weights.
    *D_Volumengewicht_diff_GM:* Difference between volumetric weights calculated using the real weight and the alternative weight.

### 4.5.3 Aggregated Features Creation from data without weighing scale

This section describes the process of creating and aggregating features from the master data table we considered from the parcel centres without weighing scale. The resulting features are stored in the separate history table. The data is grouped by country_code, volumen_grp, and parcel_shape features. The data is sourced from the master data table, which contains detailed records of shipment events from parcel centres without weighing scale. The detailed explanation of new features are described below:

1. **Event and Weight Counts:**

   *Zeitraum_Tage:* Number of distinct event dates, representing the active period in days.
   *Anzahl:*Total count of shipment codes.
   *D_Anzahl:*Total count of shipment codes per year.

2. **Average Measurements:**

   *D_Hoehe, D_Laenge, D_Breite:* Average height, length, and width of the parcels.
   *D_Gewicht_real* Average real weight.
   *D_Volumen:* Average volume.

3. **Min/Max Measurements:**

   *D_hoehe_min, D_hoehe_max:* Minimum and maximum height.
   *D_laenge_min, D_laenge_max:* Minimum and maximum length.
   *D_breite_min, D_breite_max:* Minimum and maximum width.
   *D_Gewicht_real_min, D_Gewicht_real_max:* Minimum and maximum real weight.
   *D_Volumen_min, D_Volumen_max:* Minimum and maximum volume.

4. **Derived Features:**

   *D_Volumengewicht_real:* Volume-weight ratios for real weights.
   *D_LBH_Summe:* Sum of length, breadth, and height.
   *D_Volumengewicht_real2:* Adjusted volume-weight ratios.
   *D_Gewichtsabweichung, D_Gewichtsabweichung_min, D_Gewichtsabweichung_max:* Weight deviations.

5. **Flag Counts and Proportions:**

   *Anzahl_GU_20, Anzahl_GU_10:* Counts of items under specific weight flags.
   *D_Anteil_GU_20, D_Anteil_GU_10:* Proportions of items under specific weight flags.
   *Anzahl_pan20_korrekt, Anzahl_pan10_korrekt, Anzahl_pan_korrekt:* Counts of correctly flagged PAN weights.
   *Anteil_PAN20_korrekt, Anteil_PAN10_korrekt, Anteil_PAN_korrekt:* Proportions of correctly flagged PAN weights.

6. **Weight Difference:**

   *Diff_kg_sum:* Sum of the differences between real and PAN weights, with a condition to set small average differences to zero.

Before proceeding to final feature creation, we joined all the created features from data with and without weighing scales using left join on country_code, parcel_shape,volume_grp into a single table.

### 4.5.4 Final Features Creation

1. *VH_Laenge_Hoehe:* This feature represents the ratio of the length to the height of the parcel. If the height is null or zero, the value is set to 0.
2. *VH_Laenge_Breite:* This feature captures the ratio of the length to the width of the parcel. Similar to the previous feature, it defaults to 0 if the width is null or zero.
3. *VH_Hoehe_Breite:* This is the ratio of the height to the width of the parcel, with a default value of 0 if the width is null or zero.
4. *VH_Laenge_DLaenge:* This feature compares the length of the parcel to the average length (D_Laenge). If the average length is null or zero, the value is set to 0.
5. *VH_Breite_DBreite:* The ratio of the width of the parcel to the average width (D_Breite), defaulting to 0 if the average width is null or zero.
6. *VH_Hoehe_DHoehe:* This feature captures the ratio of the height of the parcel to the average height (D_Hoehe), with a default value of 0 if the average height is null or zero.
7. *VH_Volumen_DVolumen:* The ratio of the volume of the parcel to the average volume (D_Volumen), defaulting to 0 if the average volume is null or zero.
8. *VH_Gewicht_DGewicht:* This feature compares the weight of the parcel (Gewicht_PAN) to the average real weight (D_Gewicht_real_gm). If the average real weight is null or zero, the value is set to 0.
9. *VH_Gewicht_DGewicht_PAN:* The ratio of the parcel's weight to the average weight (D_Gewicht), constrained to a maximum value of 10. If the average weight is null or zero, the value is set to 0.
10. *DGewichtsabweichung3:* This feature calculates the deviation between the parcel's weight and the average weight (D_Gewicht), constrained between -10 and 15, and then adjusted by adding 10.
11. *DGewichtsabweichung4:* Similar to DGewichtsabweichung3 but compares the parcel's weight to the average real weight (D_Gewicht_real_gm).
12. *DGewichtsabweichung5:* This feature calculates the deviation between the average weight (D_Gewicht) and the average real weight (D_Gewicht_real_gm), constrained between -10 and 10, and then adjusted by adding 10.
13. *DGewichtsabweichung6:* Similar to DGewichtsabweichung3 but compares the parcel's weight to the average weight (D_Gewicht_pan_gm).
14. *LBH_Summe:* This feature sums the length, width, and height of the parcel, constrained between 4 and 20.

15. *LBH_VH_gm:* The ratio of the dimension sum (LBH_Summe) to the average dimension sum (D_LBH_Summe_GM), constrained between 70 and 150.

16. *Volumengewicht_pan:* This feature calculates the volume weight of the parcel, defined as the ratio of the parcel's weight to its volume, constrained to a maximum value of 50.

17. *Volumengewicht_pan2:* Similar to Volumengewicht_pan but uses the average real weight (D_Gewicht_real_gm) instead of the parcel's weight.

18. *Volumengewicht_pan3:* This feature calculates the ratio of the average dimension sum (D_LBH_Summe_GM) to the parcel's weight, constrained to a maximum value of 15.

    I joined these final features to the training table with history features created as mentioned above using left join on sendungs_code level. These features are created for each shipment based on the base features and history features.

## 4.6 Loading Data from Teradata

This section describes the process of importing data from a Teradata database into dataframe, including timing the operation to measure performance. The dataset is sampled to ensure manageability during initial explorations and development stages. All the data from the final training table is downloaded from teradata into data frame *HK_Trainset_int*. All the shipments only was imported from the specified table as the size of the dataset is only around 1 Million.The query for downlaoding the dataset to dataframe is presented in the appendix A.16. This subset is used for further data cleaning and transformation followed by training. For better understanding on how these features were created and joined to training table can be visullay represented as diagram below:

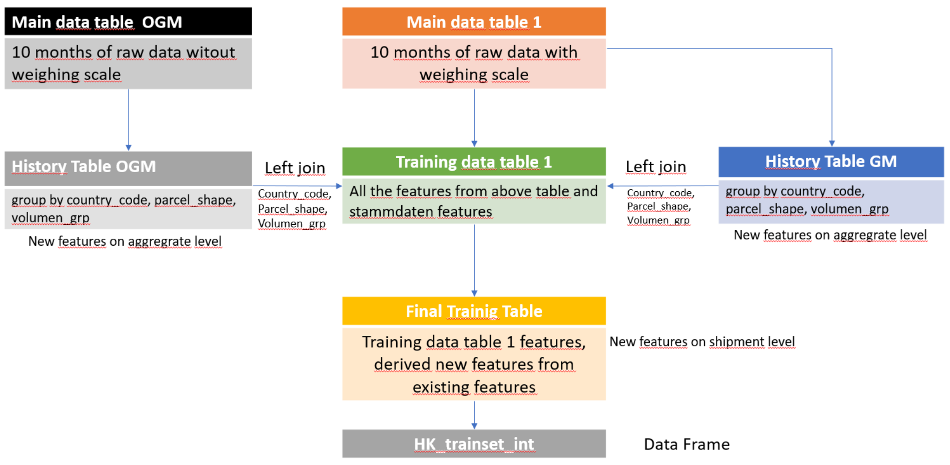

Main data table OGM
10 months of raw data witout weighing scale
Main data table 1
10 months of raw data with weighing scale
History Table OGM
group by country_code, parcel_shape, volumen_grp
New features on aggrerate level
Left join
Country_code, Parcel_shape, Volumen_grp
Training data table 1
All the features from above table and stammdaten features
Left join
Country_code, Parcel_shape, Volumen_grp
History Table GM
group by country_code, parcel_shape, volumen_grp
New features on aggregrate level
Final Trainig Table
Training data table 1 features, derived new features from existing features
New features on shipment level
HK_trainset_int
Data Frame

Figure 4.2: Features creation tables for international import shipments

Next step is to clean the data, for which I downloaded the dataset into dataframe which is going to be used for train and test the prediction model.

## 4.7 Data Cleaning

The quality of the data has a major impact on how well a prediction model works. The downloaded dataframe consists of 1 Million samples with 98 features including target variables. And this dataframe consists of 5,838,524 missing values. For cleaning these missing values I followed same methods as I did in national project to clean the data. Apart from numerical data, I also have three features *parcel_shape*, *volume_grp* and *country_code* with categorial values. So I have transform these categorial values into numerical values similar to the method followed in the national project mentioned in chapter 3. I dropped 10 features which are not relevant for prediction model. And filled null values with zero as I did in the national project. After performing these data cleaning and transormation methods the final dataframe consists of 146 predictive features in total with zero null values.

## 4.8 Defining Target Variables

So our main target variable for this project is flag_gewicht_20 columns which is used for classification model as output and gewicht_real column as target variable for regression model. Another target variable flag_gewicht_10 was also defined as a part of new postal law. After defining these three target variables, I dropped these columns from feature set which I used for training the machine learning model. After dropping these three target variables, finally I have 143 features in training set based on which our model predicts the output.

For international shipments I considered around 1 M dataset with 143 predictive features and 3 target features. Each row represent one shipment. The dataset used for modeling contains 5733 shipments with real weight greater than 20kg which is 0.58% of the total training dataset.

## 4.9 Train-Test split

I divided our dataset into two distinct subsets: a training set and a test set. The test set was allocated 30% of the total data (test_size = 0.3), while the remaining 70% was used for training the model similiar to the national project. This splitting resulted in training set of 674363 samples and testing set of 289013 samples.

## 4.10 Training

I trained the dataset using SVM and Random Forest model by fitting the training samples and evaluating the results on testing samples same way as I did in national project.

# Chapter 5

# Results & Evaluation

In this chapter, I will focus on the evaluation techniques used to compare the results of the testing set and the out-of-bag cross-validation (CV) set for the classification and regression models of Random Forest and Support Vector Machine (SVM) for both national and international projects and analyze the results from each model. This evaluation will help determine which model performs best for the task of automated identification of incorrectly labelled shipments. The two different supervised learning models introduced in chapter 2 are evaluated through experiments on the datasets. This evaluation also includes the optimization of the hyper-parameters of the finalised machine model. As I have two parts in the project as mentioned in the previous chapter, first analyze the results from the national project.

## 5.1 Results from ML Models for National Shipments

For training and testing the machine learning model with national shipments, I have considered data from month of June with dataset size of training set has 1,386,000 samples and testing set with 594,000 samples which is 7:3 ratio for both Random Forest model and also the SVM model. Initially we considered only 1,386,000 samples for training of these two models as the run time and computational power required for training model is very high.

### 5.1.1 Comparison of Random Forest and SVM Classifier Test Results

In this section, I compared the performance of the SVM classifier and the Random Forest classifier on the testing set. The comparative analysis between the SVM classifier and the Random Forest classifier is summarized in the table below:

| Model | Accuracy | Precision | Recall | F-Measure | Training Time(in sec) |
|---|---|---|---|---|---|
| **Random Forest** | 0.997 | 0.687 | 0.445 | 0.49 | 57 |
| **SVM** | 0.997 | 0.658 | 0.235 | 0.23 | 2414 |

Table 5.1: Comparison of Evaluation Metrics between SVM and Random Forest Classification ML Models

The graphical representation of comparision of evaluation metrics of two models is presented below:

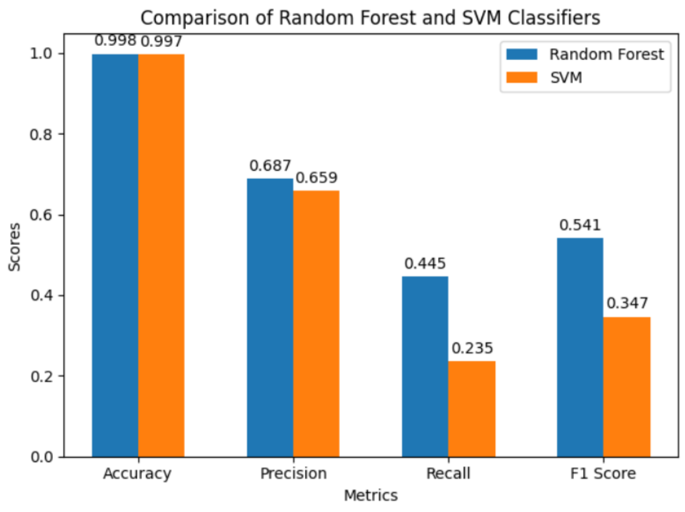


Figure 5.1: Graphical representation of RF results and SVM results

From the above table and the graph, it is evident that the Random Forest classifier consistently outperforms the SVM classifier. The higher precision, recall and F1 score of the Random Forest classifier highlight its superior ability to correctly classify samples while maintaining a balance between precision and recall. Additionally, the Random Forest classifier has significant shorter training time and does not require data normalization, making it more efficient and easier to implement. The Random Forest classifier's superior performance can be attributed to its ensemble learning approach, which leverages multiple decision trees to improve classification performance and robustness. In contrast, the SVM classifier, while effective in certain scenarios, struggled with the large dataset size and complexity, resulting in longer training times and the need for data normalization. The SVM classifier demonstrated adequate performance in terms of accuracy and precision evaluation metrics. However, it faced significant challenges in terms of computational efficiency, pre-processing requirements and also low recall score when compared with Random Forest model. The shorter training time and lack of need for data normalization make the Random Forest classifier a more practical choice for large datasets, as it reduces computational overhead and simplifies the pre-processing pipeline. In conclusion, the comparison between the SVM classifier and the Random Forest classifier clearly indicates the latter as the better-performing model. The Random Forest classifier's higher accuracy, precision, recall, F1 score, shorter training time, and reduced pre-processing requirements demonstrate its effectiveness and reliability, making it the preferred choice for further optimization and deployment.

### 5.1.2 Hyper-parameter tuning of Random Forest model

Having established the Random Forest classifier as the superior model, I proceeded to optimize its performance by tuning its hyper-parameters which are explained in detail in section 2.6.3.1. By manually tuning key hyperparameters such as the number of trees, maximum depth, and minimum samples split, I aimed to achieve better classification results.

The tuning process involved adjusting several hyper-parameters and evaluating their impact on the model's performance using the F1 score as the evaluation metric. I used dataset containing 5 million samples for training, with a test size of 30% to ensure a robust evaluation during the tuning process. The key hyper-parameters considered were the number of estimators (n_estimators), maximum depth (max_depth), minimum samples split (min_samples_split), minimum samples leaf (min_samples_leaf), and maximum features (max_features). For each hyper-parameter, a range of values was tested: n_estimators [100, 150, 200], max_depths [10, 15, 20, 25], min_samples_splits [100, 150, 200], min_samples_leafs [10, 50, 80, 100], and max_features [30, 40, 50]. Using a loop, I trained a Random Forest model for each value of the hyper-parameters on the training data and evaluated the model's performance on the test data by calculating the F1 score. The F1 scores were plotted against the respective hyper-parameter values to visualize their impact. These plots provided clear insights into the optimal values for each hyper-parameter, guiding in selecting the best configuration for model. Based on the graph results, I selected the following optimal hyper-parameters: n_estimators = 150, max_depth = 20, min_samples_split = 150, min_samples_leaf = 50, and max_features = 40. The graph below illustrates the F1 scores for each tested value of the hyper-parameters, highlighting the chosen optimal values.

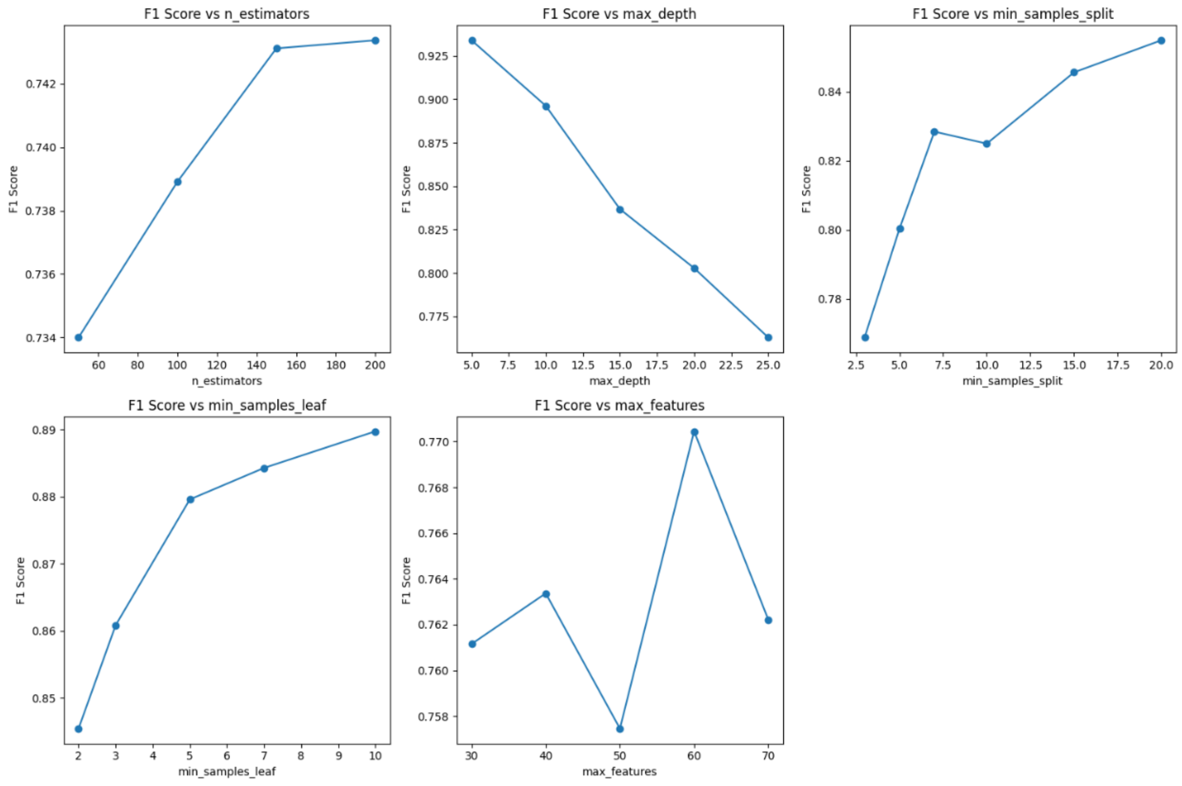


Figure 5.2: Plots showing the optimal values of chosen hyper-parameters

The optimization process led to a notable improvement in the model's performance metrics. Specifically, the precision of the Random Forest classifier increased by 3.95%, demonstrating the effectiveness of the hyper-parameter adjustments. This enhancement underscores the importance of fine-tuning machine learning models to achieve optimal performance. The improved results further validate the choice of the Random Forest classifier for this dataset, confirming its robustness and reliability in providing accurate predictions.

### 5.1.3 Feature Importance Analysis

I analyzed the importance of various features used in the tuned Random Forest model. Feature importance helps to identify which variables have the most significant impact on the model's predictions and provides insights into the underlying patterns in the data. The graph below visually represents the top 35 important features of tuned random forest model.

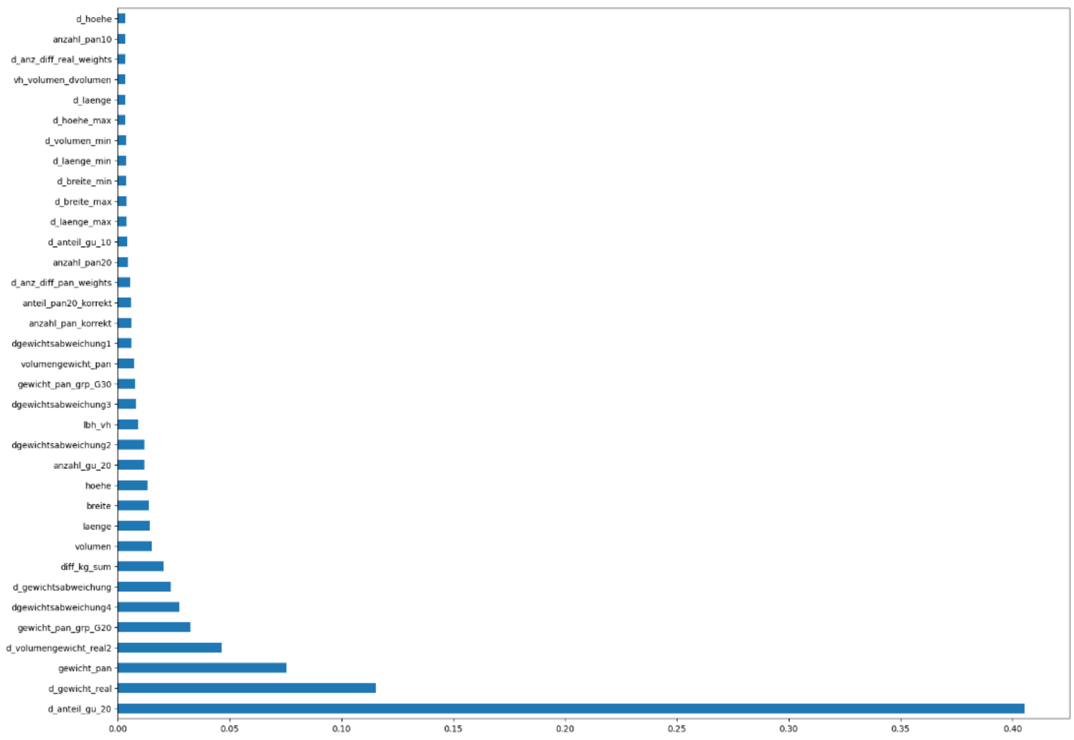


Figure 5.3: The feature importance of the optimized random forest model

These features significantly contribute to the model's predictive power, and their high importance scores indicate their strong relationship with the target variable. The feature importance analysis provides valuable insights into the factors that drive the model's predictions. Identifying the most important features, helps in better understanding of the underlying patterns in the data and improve the model's interpretability. In my tuned Random Forest model, the features with the highest importance scores were d_anteil_gu_20(percentage of parcels greater than 20kg on aggregrate level), d_gewicht_real(average real weights of parcels on aggreagration) , gewicht_pan(weight given by the customer), and other. These features should be given special consideration in any further analysis or modeling efforts. Future work may involve further exploring the relationships between these key features and the target variable to enhance the model's performance and reliability.

### 5.1.4 RF Classifier results before and after tuning

The performance comparison of the Random Forest classifier with default parameters and optimized parameters is summarized in the table below:

| Metric | Default Parameters | Optimized Parameters |
|---|---|---|
| Accuracy | 0.997 | 0.997 |
| Precision | 0.687 | 0.742 |
| Recall | 0.445 | 0.452 |
| F1 Score | 0.540 | 0.561 |

Table 5.2: Performance Comparison of Random Forest Classifier with Default and Optimized Parameters

The table clearly shows that hyperparameter optimization resulted in slightly higher precision, recall, and F1 score, demonstrating the effectiveness of the manual search process.

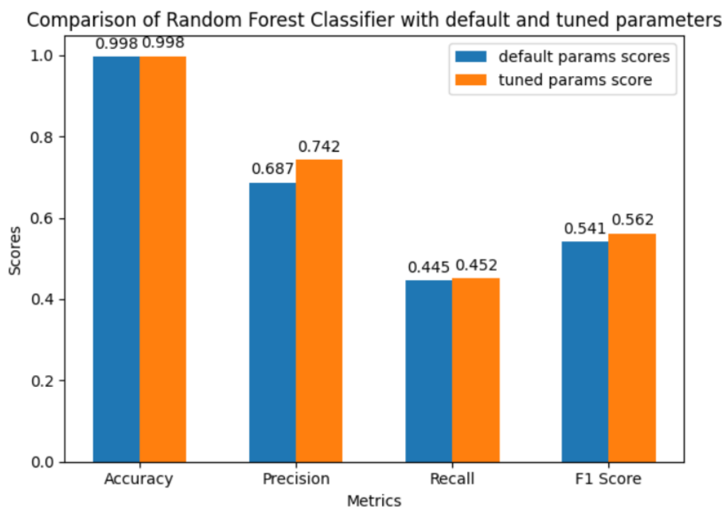


Figure 5.4: Graphical representation of RF classification model results with default and optimized parameters

The results clearly indicate that hyper-parameter tuning significantly improved the performance of the Random Forest model. Specifically, the tuned model outperforms the default model in terms of precision, recall and F1-score. This improvement can be attributed to the optimal selection of hyperparameters, which enhances the model's ability to capture the underlying patterns in the data. The objective of the model is keeping high precision and recall which is important to consider while evaluating the various models and choosing the top performing one. Since the goal is to identify every package weighing more than 20 kg, the recall is significant. This should not, however, come at the expense of anticipating that every package weighs more than 20 kg. Accurate insight into the total performance of this target is provided by the acceptable precision and PRC-AUC. In Figure, the AUC plot for the top-performing model is displayed. The performance of the tuned random forest model is much improved and it displays the largest area under the curve.

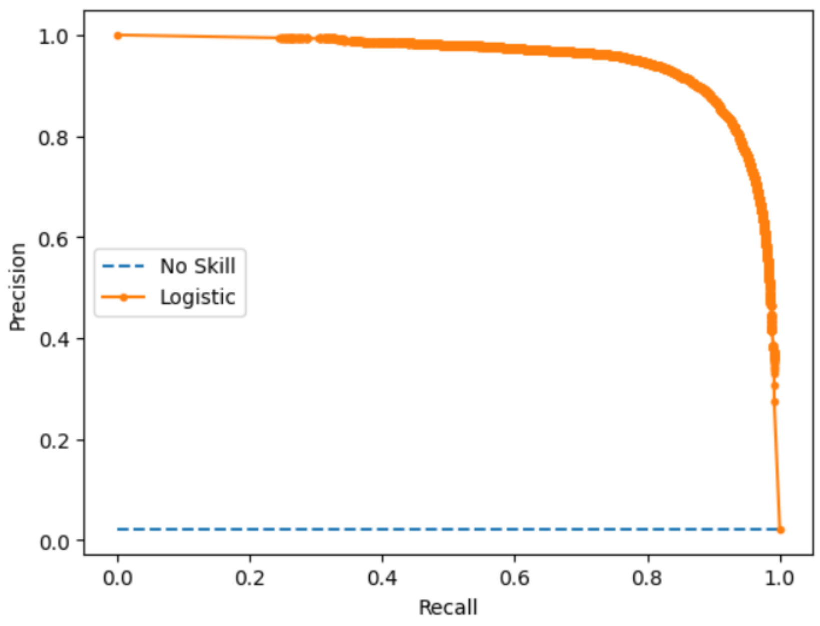


Figure 5.5: PRC curve

### 5.1.5 Random Forest Regression Model Results

Similiarly, I also trained a Random Forest Regression model with target of predicting the real weight of the shipment which is continuous value, here the target variable for the regression model is *GS_Trainset_targetvar_GR*. The size of the dataset used for training this regression model is also same as SVM model mentioned above, i.e., 1M samples with 30% of it is split for testing the model. Since the run time for regression model is high, I considered only 1M data samples. I also tuned this regression model with the similar hyper-parameters I finalized for classification model. The results from the regression model indicate its effectiveness in predicting the target variable. The table below summarizes the evaluation metrics for both the default and tuned regression models:

| **Metric** | **Default Parameters** | **Tuned Parameters** |
|---|---|---|
| R-squared ($R^2$) | 0.947 | 0.949 |
| Mean Absolute Error (MAE) | 0.383 | 0.373 |
| Mean Squared Error (MSE) | 1.181 | 1.124 |

Table 5.3: Comparison of Regression Metrics for Default and Tuned Parameters

### 5.1.6 Comparision tuned RF classifier results on test and Validation set

In this section, I compared the performance of the optimized Random Forest classifier model on the test set and the cross-validation set. This comparison helped to evaluate the model's generalizability and robustness. For Cross Validation, I considered data for the month of August with the size of 3M, these data is never seen by the model.

| **Dataset** | **Accuracy** | **Precision** | **Recall** | **F1- Score** |
|---|---|---|---|---|
| **Test set** | 0.997 | 0.742 | 0.452 | 0.561 |
| **Validation set** | 0.997 | 0.722 | 0.350 | 0.471 |

Table 5.4: Comparison of Evaluation Metrics between test set and validation set using optimized Random Forest Classification Model

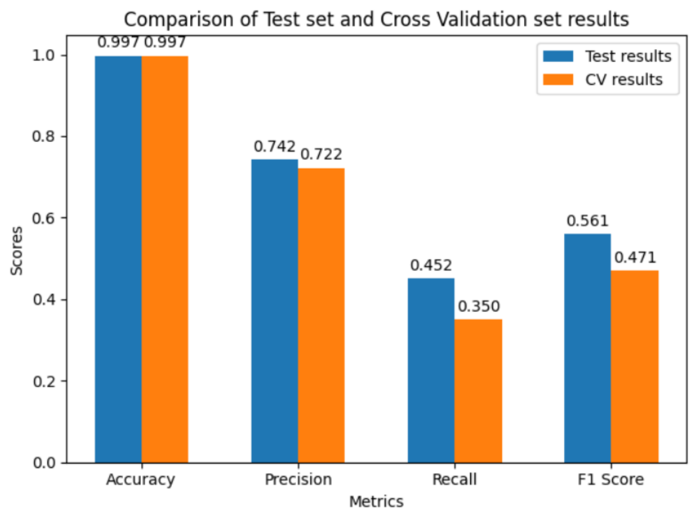


Figure 5.6: Graphical representation of RF classification model results on Test set and Validation set

From the above comparision of classification results of tuned Random Forest model on test set and validation set, it is observed that the random forest model exhibited signs of overfitting. Overfitting refers to a model's poor ability to generalise from observed data to unknown data. The model fits badly on the testing set but works flawlessly on the training set due to overfitting. This is a result of the over-fitted model's inability to handle information from the testing set that may differ from that from the training set. Conversely, overfitted models have a propensity to acquire the discipline concealed beneath the data via memorisation of all the data, including inevitable noise on the training set. This was evident from the significant discrepancy between the training and cross-validation (CV) results as seen above, where the model performed exceptionally well on the training data but poorly on the validation data. To overcome this problem, I employed a new validation strategy combining the results from classifier model such as probability of forecasting and prediction weight from regression model. In the analysis, I utilized a combination of a random forest classifier and a random forest regression model to predict shipment outcomes. The classifier model outputs a probability (p_PGU_20), while the regression model predicts a weight (pred_weight_20). I combined these outputs using a series of conditional statements to derive the final predicted flag (Flag_weight_model_20), the code for these statements which can be seen in detail in the appendix A3.11.

### 5.1.7 Comparison of Validation results between Classifier model and Combined model

Along with validating classifier model on the validation set, we combined the predicted probability of classifier and regression model which predicted the weight of the shipment to analyse if the combined model performs better than the individual classifier. The code used for classifying the shipment using combined models is listed in the Appendix A.9. The results with these methods are listed in below table.

| Dataset | Accuracy | Precision | Recall | F1- Score |
|---|---|---|---|---|
| **Results on Validation set by individual RF Classifier** | 0.997 | 0.742 | 0.452 | 0.561 |
| **Results on Validation set by combined RF classifier and Regression model** | 0.997 | 0.722 | 0.350 | 0.471 |

Table 5.5: Comparison of Evaluation Metrics between RF classifier model and combined model on Validation set

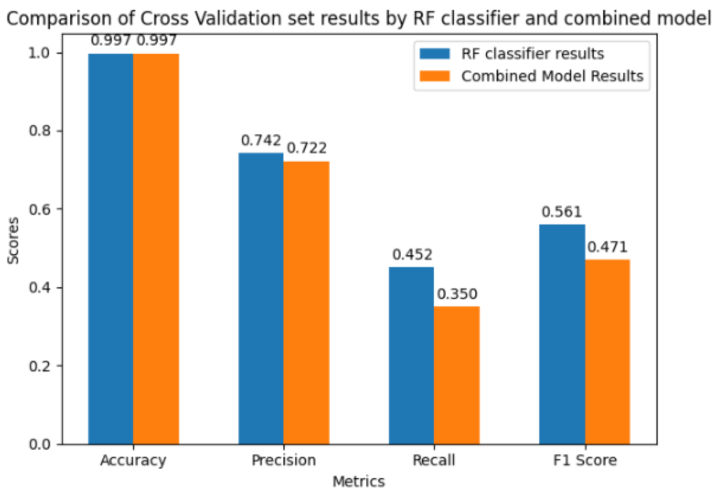


Figure 5.7: Graphical representation results of RF classification model and combined model on Validation set

### 5.1.8 Results after Resampling

To mitigate the problem of overfitting, I used a resampling technique explained in deatil in the section 2.10, to address the class imbalance in our dataset. Specifically, I increased the representation of the majority class of training data to 1.3% form 0.3%. By doing so, I aimed to provide the model with a more balanced training set, thereby improving its generalization capabilities.

Still I noticed the individual classification model is overfitting, there by resulted in poor precision on cross validation dataset. However the results of combined model were better than overall results achieved so far. Following the resampling process, I observed a marked improvement in the combined model's performance. The new results of combined model demonstrated a better validation metrics, indicating that the overfitting problem had been effectively addressed.

The final results are as below:

| Dataset | Accuracy | Precision | Recall | F1- Score |
|---|---|---|---|---|
| **Results on Validation set by combined model before re-sampling** | 0.997 | 0.722 | 0.350 | 0.471 |
| **Results on Validation set by combined RF model after re-sampling** | 0.997 | 0.677 | 0.577 | 0.623 |

Table 5.6: Comparison of Evaluation Metrics on Validation set by combined model before and after resampling

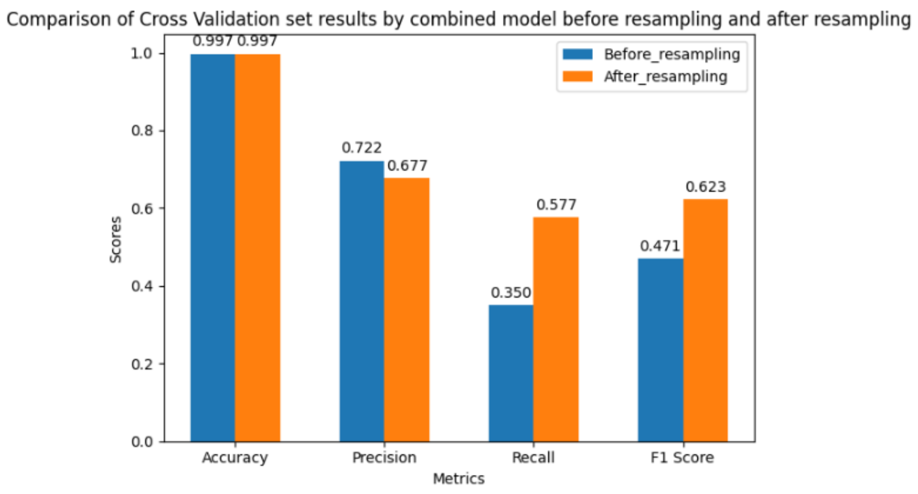
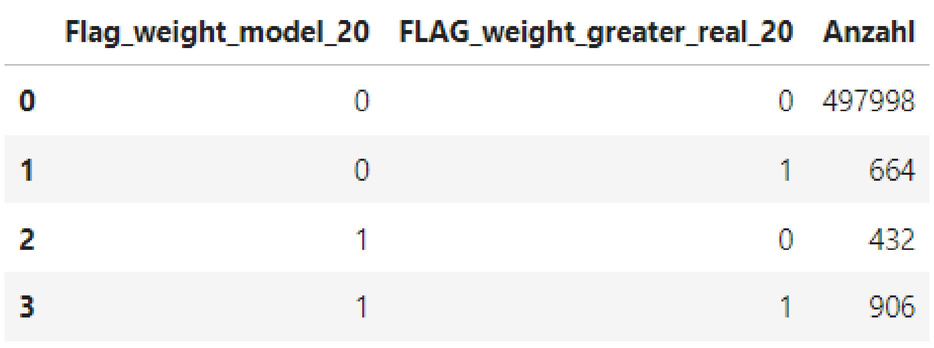


Figure 5.8: Graphical representation of combined model results on Validation set before and after resampling

With resampling precision score is reduced by 5 %, However, recall score could be increased by 12% which is significant for the project. So I finally chose to go with the combined model of random forest model after discussion with my supervisor. The Random Forest (RF) tuned model on the cross-validation set demonstrated a high accuracy of approximately 99.85%, indicating that the model performs well in correctly classifying both positive and negative instances. However, the precision and recall values suggest a trade-off between false positives and false negatives. The precision of approximately 67.75% indicates that when the model predicts a positive class, it is correct about 68% of the time. The recall of approximately 57.73% indicates that the model correctly identifies about 58% of the actual positive cases. The F1-Score, which balances precision and recall, is approximately 62.38%, suggesting a moderate balance between the two metrics. This analysis indicates that while the model is generally accurate, there is room for improvement in its ability to correctly identify positive cases without increasing the false positive rate. For detailed figures the following confusion matrix is helpful for analysis.



| | Flag_weight_model_20 | FLAG_weight_greater_real_20 | Anzahl |
|---|---|---|---|
| **0** | 0 | 0 | 497998 |
| **1** | 0 | 1 | 664 |
| **2** | 1 | 0 | 432 |
| **3** | 1 | 1 | 906 |

Figure 5.9: confusion matrix of combined model

For further improvement, fine-tuning the model parameters or employing different techniques such as balancing the dataset or utilizing different algorithms might be beneficial.

## 5.2 Results from ML Models for International Import Shipments

I trained both SVM and Random Forest model with final pre-processed data as mentioned in Chapter 4.

### 5.2.1 Comparison of Random Forest and SVM Classifier Test Results

In this section, we compare the performance of the SVM classifier and the Random Forest classifier on the testing set. The comparative analysis between the SVM classifier and the Random Forest classifier is summarized in the table below:

| **Model** | **Accuracy** | **Precision** | **Recall** | **F-Measure** | **Training Time(in sec)** |
|---|---|---|---|---|---|
| **Random Forest** | 0.99 | 0.70 | 0.31 | 0.43 | 1.779 |
| **SVM** | 0.99 | 0.91 | 0.16 | 0.28 | 1553.32 |

Table 5.7: Comparison of Evaluation Metrics between SVM and Random Forest Classification ML Models on international import shipments

From analysing above results from Classification model of Random Forest and SVM, it is clear that Random Forest model outperformed SVM model with default parameters. Therefore I chose to consider Random Forest model for classifying and predicting the weight of international import shipments. With deafult parameters of Random Forest model, the accuaracy was exceptional, but the precision and recall were not to the expectation, so I tuned random forest model by using manual search similar to the method followed in national project and the performance is increased to 4.10%. The results from the tuned RF Classifier are as follows.

| **Model** | **Accuracy** | **Precision** | **Recall** | **F-Measure** | **Training Time(in sec)** |
|---|---|---|---|---|---|
| **Tuned Random Forest Classifier on test set** | 0.995 | 0.767 | 0.325 | 0.457 | 40 |

Table 5.8: Tuned RF Classifier Model Results of international import shipments on test set

### 5.2.2 Cross Validation Results

To check the robustness of trained model on test set, performing cross validation is crucial. For cross validation, I considered the data for the month of August with dataset size of 63,525 samples.

| **Model** | **Accuracy** | **Precision** | **Recall** | **F-Measure** |
|---|---|---|---|---|
| **Tuned Random Forest Classifier on Validation set** | 0.99 | 0.82 | 0.40 | 0.54 |

Table 5.9: Tuned RF Classifier Model Results of international import shipments on Validation set

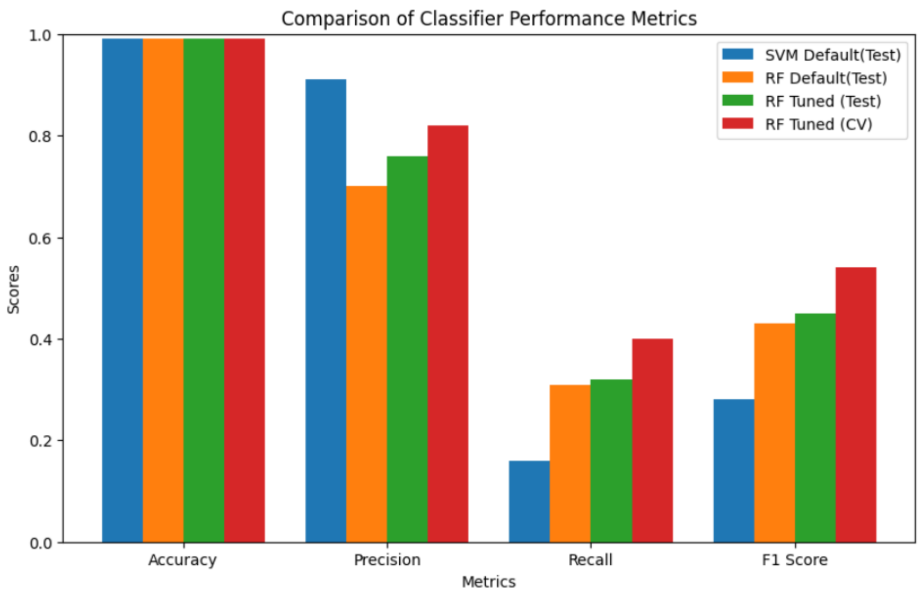
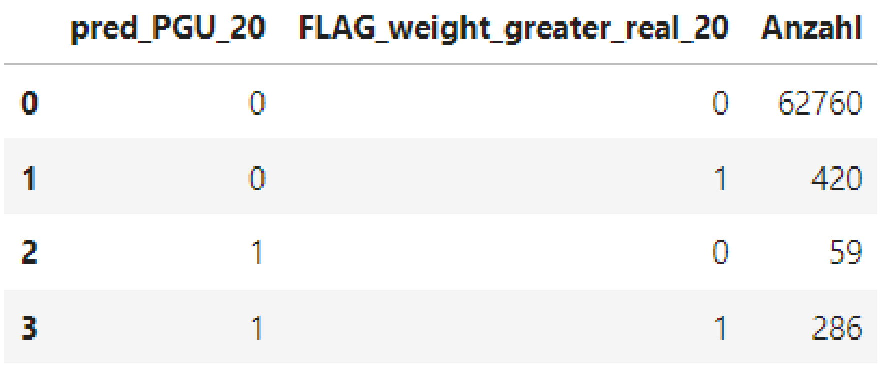


Figure 5.10: Graphical comparison of two model's performance on test and Validation set

A confusion matrix was used to visualize and summarize the performance of the tuned random model on the cross-validation set. The confusion matrix provides insights into the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions made by the model. The confusion matrix in Figure 5.8 demonstrates the distribution of predicted and actual class labels. Each cell in the matrix represents the count of instances that fall into the predicted versus actual category.



| | pred_PGU_20 | FLAG_weight_greater_real_20 | Anzahl |
|---|---|---|---|
| **0** | 0 | 0 | 62760 |
| **1** | 0 | 1 | 420 |
| **2** | 1 | 0 | 59 |
| **3** | 1 | 1 | 286 |

Figure 5.11: Confusion Matrix of Tuned Random Forest Model on Validation Set

The confusion matrix and the performance metrics indicate that while the model achieves high precision (82.90%), its recall (40.51%) is comparatively lower. This suggests that the model is more effective at correctly identifying positive instances but misses a significant portion of them, as evidenced by the 420 false negatives. The F1 score of 54.32% reflects the balance between precision and recall, highlighting the trade-offs involved.

The high accuracy (99.25%) is largely influenced by the substantial number of true negatives (62,760), indicating that the model performs well in identifying negative instances. However, the imbalance in the dataset, with a majority of negative instances, contributed to this high accuracy.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In summary, we have developed machine learning models for the problem of classification of shipments with a label less than 20 kilograms for both national and international shipments. In this thesis, we explored the development and evaluation of machine learning models to identify shipments with weights greater than 20kg using both Support Vector Machine (SVM) and Random Forest (RF) models. Our initial approach with a smaller subsample of final data revealed that the Random Forest model outperformed the SVM model in terms of precision, recall and overall performance. Upon selecting the Random Forest model, we expanded our dataset to 4 million samples for training and applied hyperparameter tuning. This tuning process resulted in a performance improvement of approximately 4 percent, demonstrating the effectiveness of model optimization techniques. We conducted an out-of-bag cross-validation to check the robustm´ness of the model against overfitting ot underfitting and to ensure how well the model is performing on new data, for the month of August, utilizing a test set of 3 million samples. The evaluation involved comparing results from both individual classifiers and a combined model, which integrated the probabilities from the classifier and regression models. The results show that the combined Random Forest classifier and regression model after resampling is a viable for the problem of classification of shipment weight. I have noticed it outperforms SVM Classifier and also individual Random Forest Classifier in the classification problem of parcel weights.

Despite these efforts, we observed that the classification model exhibited signs of over-fitting, particularly when applied to the larger dataset.

## 6.2 Limitations

Some limitations were identified during the course of this research:

1. Dataset Size Constraints: The study was constrained by runtime and memory issues when considering datasets larger than 5 million samples. This limitation hindered the

ability to fully explore the model's potential performance on even larger datasets.

2. Insufficient International Shipment Data: The model faced challenges in accurately predicting the weight of international shipments due to insufficient data. The lack of customer data further complicated the prediction process for these shipments.

## 6.3 Future Work

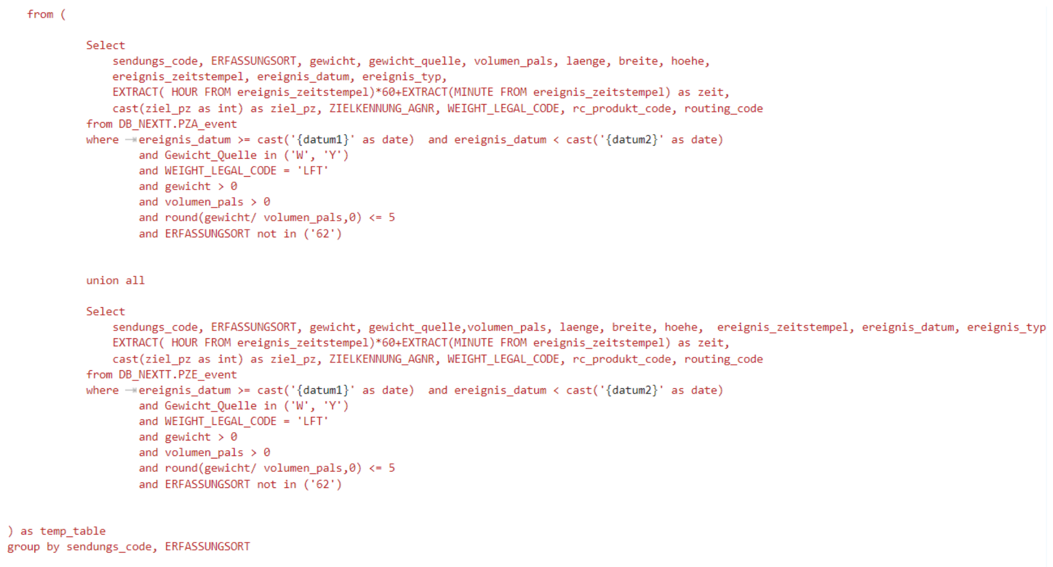
To build upon the findings of this thesis, some recommendations for future research are proposed as follows.

1. Addressing Over-fitting: Future work should focus on identifying the underlying reasons for model over-fitting and implementing strategies to mitigate this issue. Techniques such as regularization, dropout, or the inclusion of more diverse training data could be explored.
2. Data Augmentation: Efforts should be made to acquire more comprehensive data, particularly for international shipments. This could involve partnerships with international logistics providers or enhanced data collection methods from customers.
3. Enhanced Feature Engineering: Further analysis should be conducted to understand why the model is making incorrect predictions. This could involve exploring additional features or employing more sophisticated feature engineering techniques to improve model performance.
4. Scalability Solutions: Investigating scalable machine learning solutions that can handle larger datasets efficiently would be beneficial. This may include the use of distributed computing frameworks or more advanced hardware resources.

In conclusion, while the combined Random Forest model demonstrated superior performance over the individual RF Classification model for identifying shipments greater than 20kg, challenges such as overfitting and data limitations were encountered. By addressing these challenges and implementing the recommended strategies, future research can enhance the accuracy and reliability of shipment weight prediction models, ultimately contributing to more efficient logistics and supply chain management.

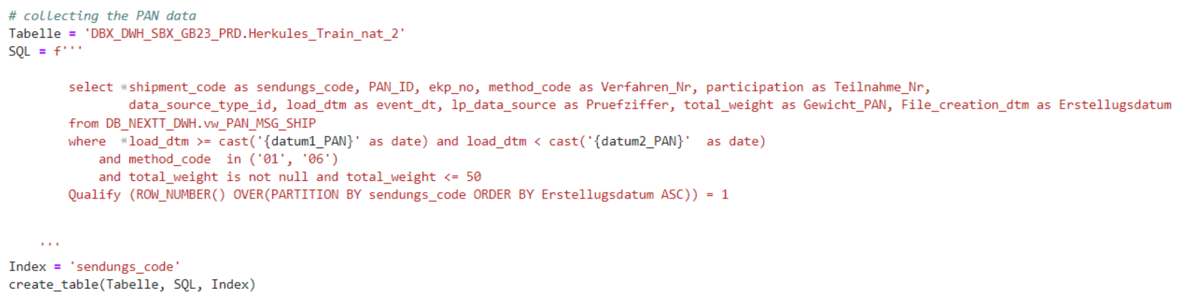
# Chapter 7

# Appendix



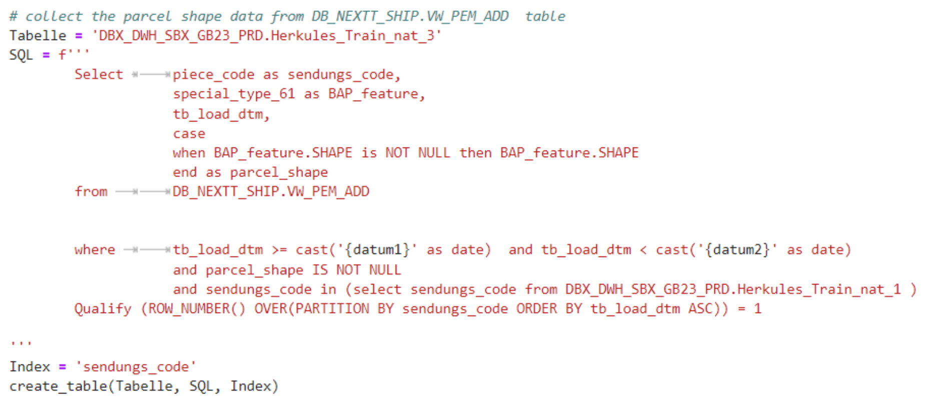
```
   from (

          Select
              sendungs_code, ERFASSUNGSORT, gewicht, gewicht_quelle, volumen_pals, laenge, breite, hoehe,
              ereignis_zeitstempel, ereignis_datum, ereignis_typ,
              EXTRACT( HOUR FROM ereignis_zeitstempel)*60+EXTRACT(MINUTE FROM ereignis_zeitstempel) as zeit,
              cast(ziel_pz as int) as ziel_pz, ZIELKENNUNG_AGNR, WEIGHT_LEGAL_CODE, rc_produkt_code, routing_code
          from DB_NEXTT.PZA_event
          where   ereignis_datum >= cast('{datum1}' as date)  and ereignis_datum < cast('{datum2}' as date)
                  and Gewicht_Quelle in ('W', 'Y')
                  and WEIGHT_LEGAL_CODE = 'LFT'
                  and gewicht > 0
                  and volumen_pals > 0
                  and round(gewicht/ volumen_pals,0) <= 5
                  and ERFASSUNGSORT not in ('62')


          union all

          Select
              sendungs_code, ERFASSUNGSORT, gewicht, gewicht_quelle,volumen_pals, laenge, breite, hoehe,  ereignis_zeitstempel, ereignis_datum, ereignis_typ
              EXTRACT( HOUR FROM ereignis_zeitstempel)*60+EXTRACT(MINUTE FROM ereignis_zeitstempel) as zeit,
              cast(ziel_pz as int) as ziel_pz, ZIELKENNUNG_AGNR, WEIGHT_LEGAL_CODE, rc_produkt_code, routing_code
          from DB_NEXTT.PZE_event
          where   ereignis_datum >= cast('{datum1}' as date)  and ereignis_datum < cast('{datum2}' as date)
                  and Gewicht_Quelle in ('W', 'Y')
                  and WEIGHT_LEGAL_CODE = 'LFT'
                  and gewicht > 0
                  and volumen_pals > 0
                  and round(gewicht/ volumen_pals,0) <= 5
                  and ERFASSUNGSORT not in ('62')


) as temp_table
group by sendungs_code, ERFASSUNGSORT
```

A.1



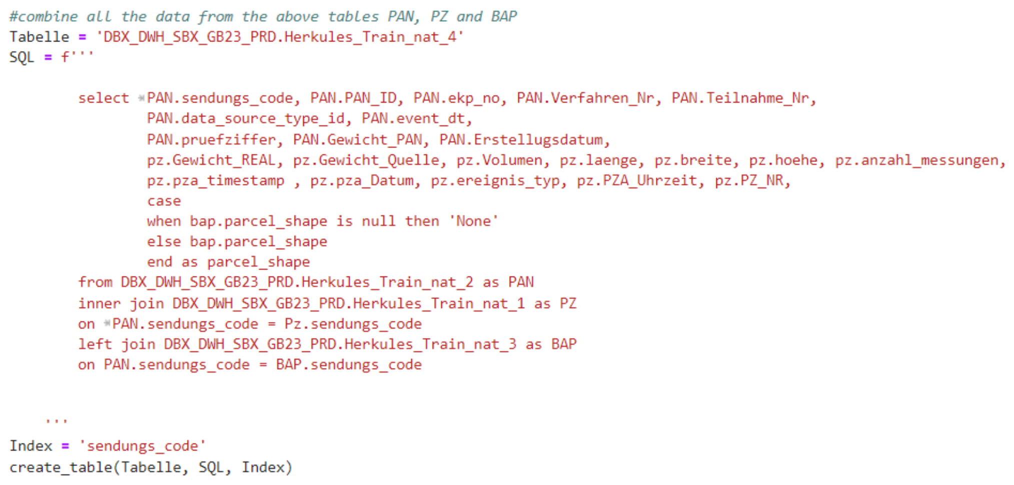
```
# collecting the PAN data
Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_2'
SQL = f'''

        select  shipment_code as sendungs_code, PAN_ID, ekp_no, method_code as Verfahren_Nr, participation as Teilnahme_Nr,
                data_source_type_id, load_dtm as event_dt, lp_data_source as Pruefziffer, total_weight as Gewicht_PAN, File_creation_dtm as Erstellugsdatum
        from DB_NEXTT_DWH.vw_PAN_MSG_SHIP
        where   load_dtm >= cast('{datum1_PAN}' as date) and load_dtm < cast('{datum2_PAN}'  as date)
            and method_code  in ('01', '06')
            and total_weight is not null and total_weight <= 50
        Qualify (ROW_NUMBER() OVER(PARTITION BY sendungs_code ORDER BY Erstellugsdatum ASC)) = 1


    '''
Index = 'sendungs_code'
create_table(Tabelle, SQL, Index)
```

A.2



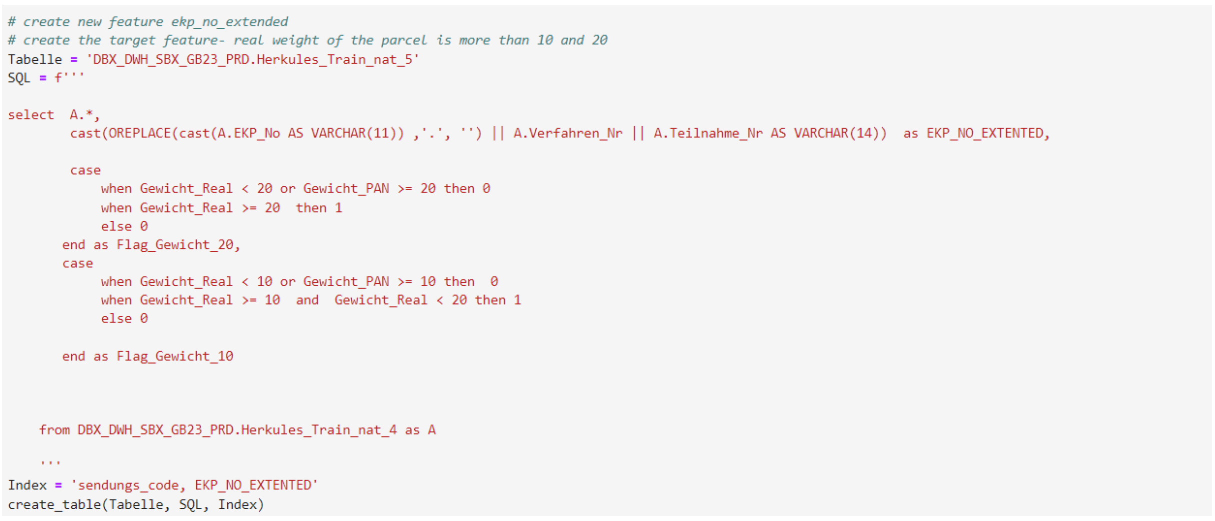
```
# collect the parcel shape data from DB_NEXTT_SHIP.VW_PEM_ADD  table
Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_3'
SQL = f'''
        Select          piece_code as sendungs_code,
                        special_type_61 as BAP_feature,
                        tb_load_dtm,
                        case
                        when BAP_feature.SHAPE is NOT NULL then BAP_feature.SHAPE
                        end as parcel_shape
        from            DB_NEXTT_SHIP.VW_PEM_ADD


        where           tb_load_dtm >= cast('{datum1}' as date)  and tb_load_dtm < cast('{datum2}' as date)
                        and parcel_shape IS NOT NULL
                        and sendungs_code in (select sendungs_code from DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_1 )
        Qualify (ROW_NUMBER() OVER(PARTITION BY sendungs_code ORDER BY tb_load_dtm ASC)) = 1

'''
Index = 'sendungs_code'
create_table(Tabelle, SQL, Index)
```

A.3



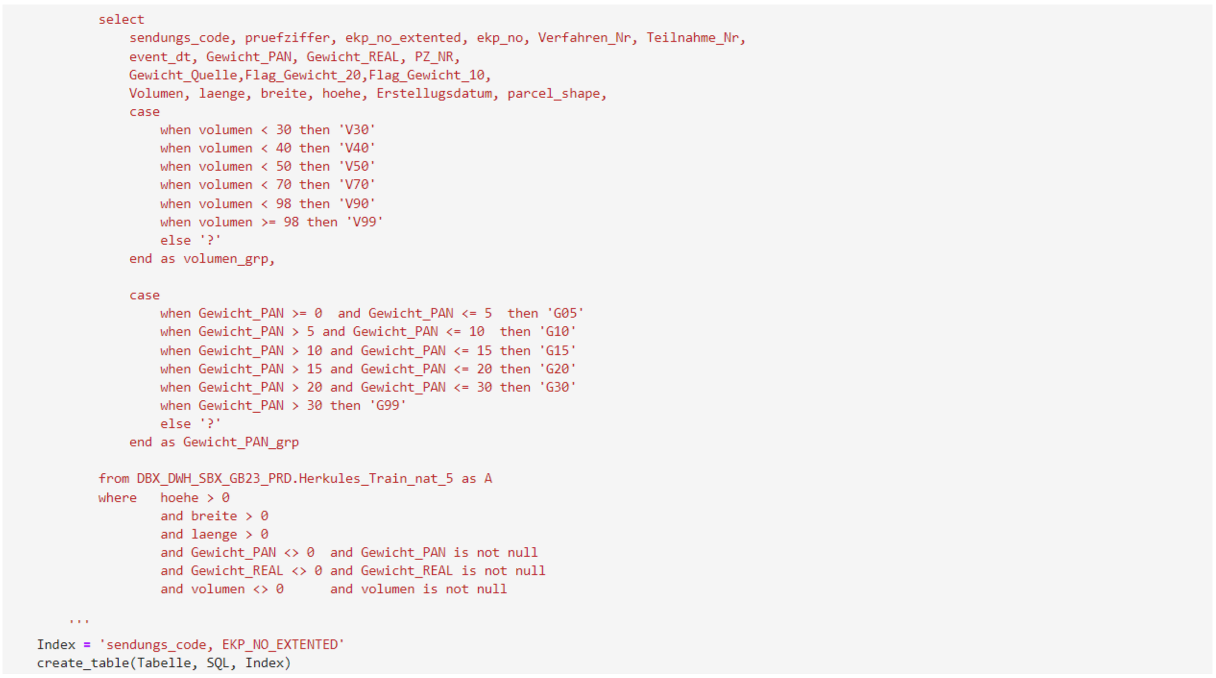
```
#combine all the data from the above tables PAN, PZ and BAP
Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_4'
SQL = f'''

        select PAN.sendungs_code, PAN.PAN_ID, PAN.ekp_no, PAN.Verfahren_Nr, PAN.Teilnahme_Nr,
                PAN.data_source_type_id, PAN.event_dt,
                PAN.pruefziffer, PAN.Gewicht_PAN, PAN.Erstellugsdatum,
                pz.Gewicht_REAL, pz.Gewicht_Quelle, pz.Volumen, pz.laenge, pz.breite, pz.hoehe, pz.anzahl_messungen,
                pz.pza_timestamp , pz.pza_Datum, pz.ereignis_typ, pz.PZA_Uhrzeit, pz.PZ_NR,
                case
                when bap.parcel_shape is null then 'None'
                else bap.parcel_shape
                end as parcel_shape
        from DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_2 as PAN
        inner join DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_1 as PZ
        on PAN.sendungs_code = Pz.sendungs_code
        left join DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_3 as BAP
        on PAN.sendungs_code = BAP.sendungs_code


    '''
Index = 'sendungs_code'
create_table(Tabelle, SQL, Index)
```

A.4



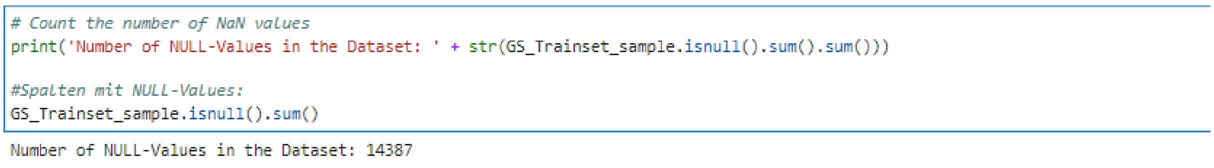
```
# create new feature ekp_no_extended
# create the target feature- real weight of the parcel is more than 10 and 20
Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_5'
SQL = f'''

select  A.*,
        cast(OREPLACE(cast(A.EKP_No AS VARCHAR(11)) ,'.', '') || A.Verfahren_Nr || A.Teilnahme_Nr AS VARCHAR(14))  as EKP_NO_EXTENTED,

        case
            when Gewicht_Real < 20 or Gewicht_PAN >= 20 then 0
            when Gewicht_Real >= 20  then 1
            else 0
       end as Flag_Gewicht_20,
       case
            when Gewicht_Real < 10 or Gewicht_PAN >= 10 then  0
            when Gewicht_Real >= 10  and  Gewicht_Real < 20 then 1
            else 0

       end as Flag_Gewicht_10



    from DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_4 as A

    '''
Index = 'sendungs_code, EKP_NO_EXTENTED'
create_table(Tabelle, SQL, Index)
```

A.5



```
            select
                sendungs_code, pruefziffer, ekp_no_extented, ekp_no, Verfahren_Nr, Teilnahme_Nr,
                event_dt, Gewicht_PAN, Gewicht_REAL, PZ_NR,
                Gewicht_Quelle,Flag_Gewicht_20,Flag_Gewicht_10,
                Volumen, laenge, breite, hoehe, Erstellugsdatum, parcel_shape,
                case
                    when volumen < 30 then 'V30'
                    when volumen < 40 then 'V40'
                    when volumen < 50 then 'V50'
                    when volumen < 70 then 'V70'
                    when volumen < 98 then 'V90'
                    when volumen >= 98 then 'V99'
                    else '?'
                end as volumen_grp,

                case
                    when Gewicht_PAN >= 0  and Gewicht_PAN <= 5  then 'G05'
                    when Gewicht_PAN > 5 and Gewicht_PAN <= 10  then 'G10'
                    when Gewicht_PAN > 10 and Gewicht_PAN <= 15 then 'G15'
                    when Gewicht_PAN > 15 and Gewicht_PAN <= 20 then 'G20'
                    when Gewicht_PAN > 20 and Gewicht_PAN <= 30 then 'G30'
                    when Gewicht_PAN > 30 then 'G99'
                    else '?'
                end as Gewicht_PAN_grp

            from DBX_DWH_SBX_GB23_PRD.Herkules_Train_nat_5 as A
            where   hoehe > 0
                    and breite > 0
                    and laenge > 0
                    and Gewicht_PAN <> 0  and Gewicht_PAN is not null
                    and Gewicht_REAL <> 0 and Gewicht_REAL is not null
                    and volumen <> 0      and volumen is not null

        '''
    Index = 'sendungs_code, EKP_NO_EXTENTED'
    create_table(Tabelle, SQL, Index)
```

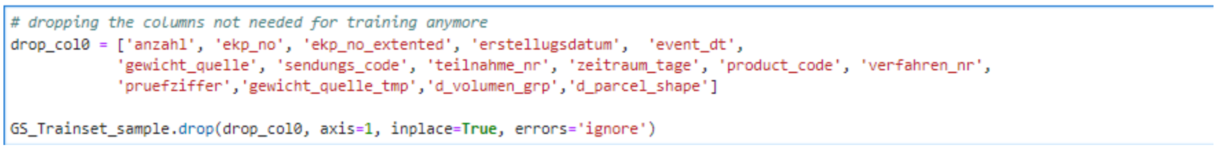
A.6



```
# Count the number of NaN values
print('Number of NULL-Values in the Dataset: ' + str(GS_Trainset_sample.isnull().sum().sum()))

#Spalten mit NULL-Values:
GS_Trainset_sample.isnull().sum()
```

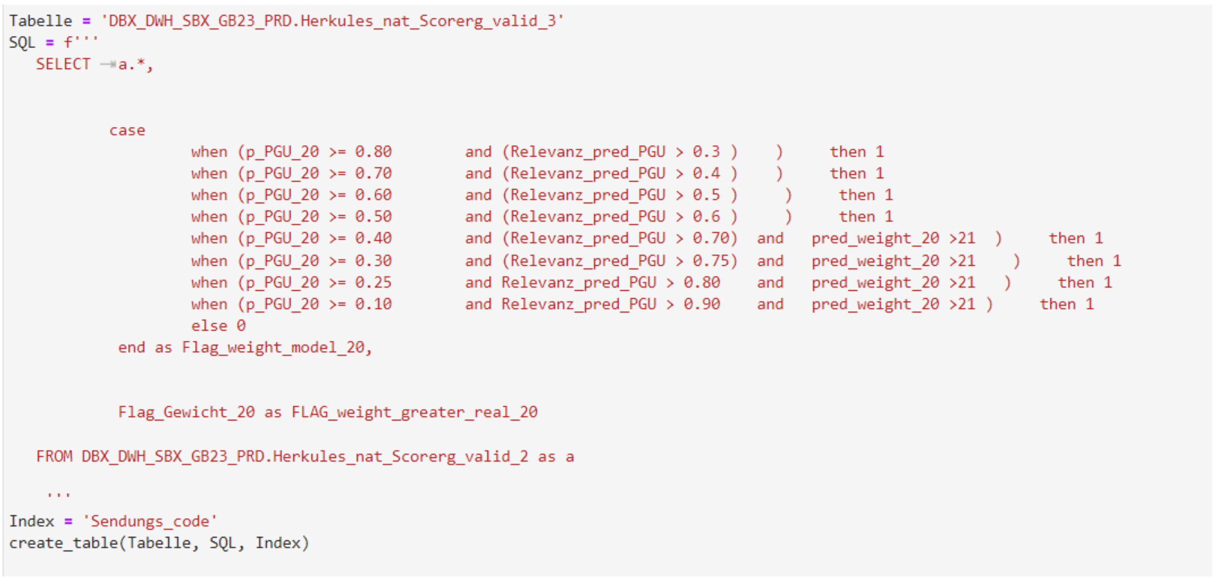
```
Number of NULL-Values in the Dataset: 14387
```

A.7



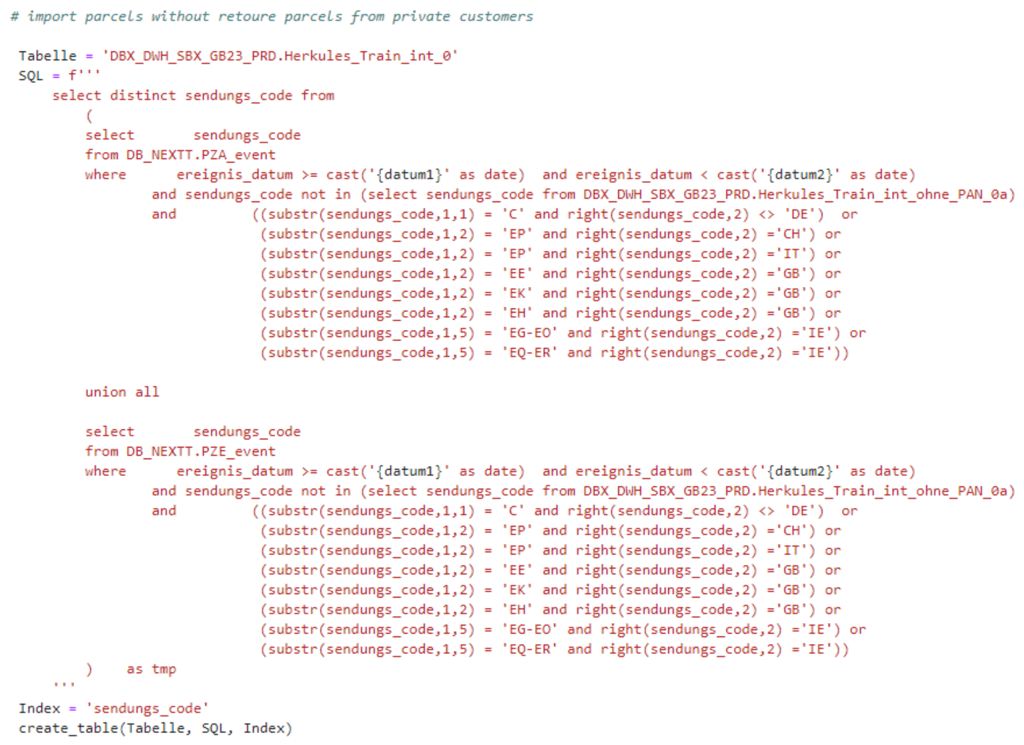
```
# dropping the columns not needed for training anymore
drop_col0 = ['anzahl', 'ekp_no', 'ekp_no_extented', 'erstellugsdatum',  'event_dt',
             'gewicht_quelle', 'sendungs_code', 'teilnahme_nr', 'zeitraum_tage', 'product_code', 'verfahren_nr',
             'pruefziffer','gewicht_quelle_tmp','d_volumen_grp','d_parcel_shape']

GS_Trainset_sample.drop(drop_col0, axis=1, inplace=True, errors='ignore')
```

A.8



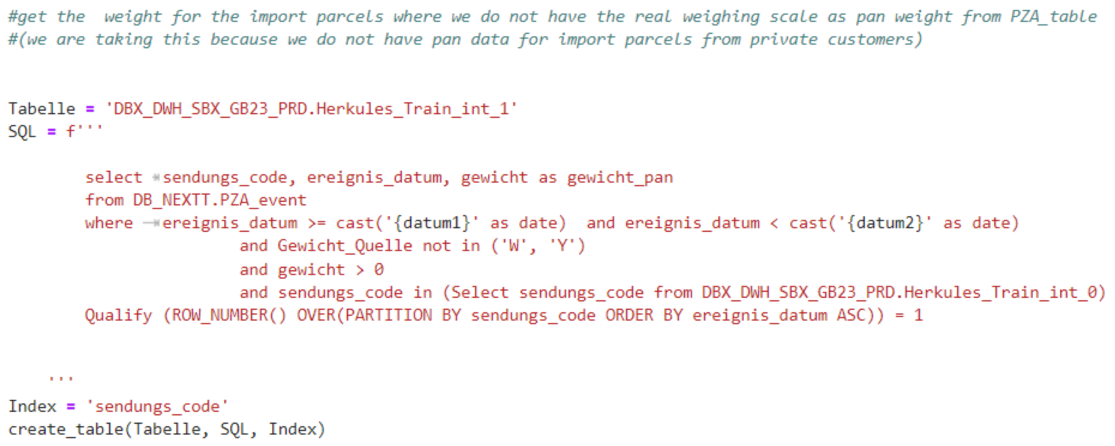
```
Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_nat_Scorerg_valid_3'
SQL = f'''
   SELECT →a.*,


           case
                    when (p_PGU_20 >= 0.80       and (Relevanz_pred_PGU > 0.3 )    )     then 1
                    when (p_PGU_20 >= 0.70       and (Relevanz_pred_PGU > 0.4 )    )     then 1
                    when (p_PGU_20 >= 0.60       and (Relevanz_pred_PGU > 0.5 )     )     then 1
                    when (p_PGU_20 >= 0.50       and (Relevanz_pred_PGU > 0.6 )     )     then 1
                    when (p_PGU_20 >= 0.40       and (Relevanz_pred_PGU > 0.70)  and   pred_weight_20 >21  )     then 1
                    when (p_PGU_20 >= 0.30       and (Relevanz_pred_PGU > 0.75)  and   pred_weight_20 >21    )     then 1
                    when (p_PGU_20 >= 0.25       and Relevanz_pred_PGU > 0.80    and   pred_weight_20 >21   )     then 1
                    when (p_PGU_20 >= 0.10       and Relevanz_pred_PGU > 0.90    and   pred_weight_20 >21 )     then 1
                    else 0
            end as Flag_weight_model_20,


            Flag_Gewicht_20 as FLAG_weight_greater_real_20

   FROM DBX_DWH_SBX_GB23_PRD.Herkules_nat_Scorerg_valid_2 as a

    '''
Index = 'Sendungs_code'
create_table(Tabelle, SQL, Index)
```

A.9



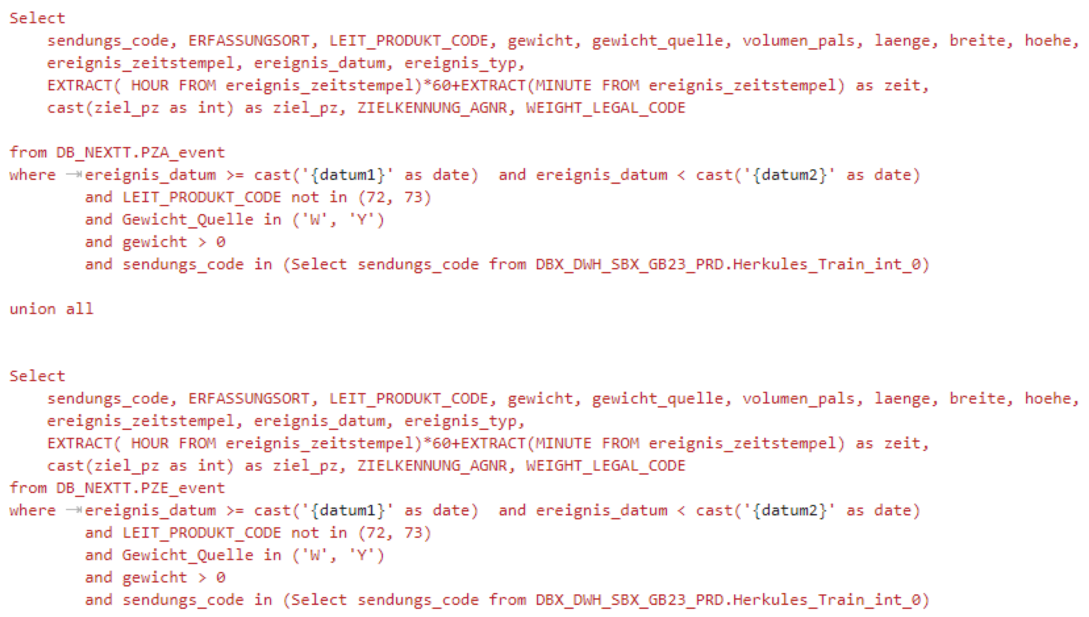
```
# import parcels without retoure parcels from private customers

 Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_0'
 SQL = f'''
     select distinct sendungs_code from
         (
         select       sendungs_code
         from DB_NEXTT.PZA_event
         where      ereignis_datum >= cast('{datum1}' as date)  and ereignis_datum < cast('{datum2}' as date)
                 and sendungs_code not in (select sendungs_code from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_ohne_PAN_0a)
                 and         ((substr(sendungs_code,1,1) = 'C' and right(sendungs_code,2) <> 'DE')  or
                              (substr(sendungs_code,1,2) = 'EP' and right(sendungs_code,2) ='CH') or
                              (substr(sendungs_code,1,2) = 'EP' and right(sendungs_code,2) ='IT') or
                              (substr(sendungs_code,1,2) = 'EE' and right(sendungs_code,2) ='GB') or
                              (substr(sendungs_code,1,2) = 'EK' and right(sendungs_code,2) ='GB') or
                              (substr(sendungs_code,1,2) = 'EH' and right(sendungs_code,2) ='GB') or
                              (substr(sendungs_code,1,5) = 'EG-EO' and right(sendungs_code,2) ='IE') or
                              (substr(sendungs_code,1,5) = 'EQ-ER' and right(sendungs_code,2) ='IE'))

         union all

         select       sendungs_code
         from DB_NEXTT.PZE_event
         where      ereignis_datum >= cast('{datum1}' as date)  and ereignis_datum < cast('{datum2}' as date)
                 and sendungs_code not in (select sendungs_code from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_ohne_PAN_0a)
                 and         ((substr(sendungs_code,1,1) = 'C' and right(sendungs_code,2) <> 'DE')  or
                              (substr(sendungs_code,1,2) = 'EP' and right(sendungs_code,2) ='CH') or
                              (substr(sendungs_code,1,2) = 'EP' and right(sendungs_code,2) ='IT') or
                              (substr(sendungs_code,1,2) = 'EE' and right(sendungs_code,2) ='GB') or
                              (substr(sendungs_code,1,2) = 'EK' and right(sendungs_code,2) ='GB') or
                              (substr(sendungs_code,1,2) = 'EH' and right(sendungs_code,2) ='GB') or
                              (substr(sendungs_code,1,5) = 'EG-EO' and right(sendungs_code,2) ='IE') or
                              (substr(sendungs_code,1,5) = 'EQ-ER' and right(sendungs_code,2) ='IE'))
         )    as tmp
     '''
 Index = 'sendungs_code'
 create_table(Tabelle, SQL, Index)
```

A.10



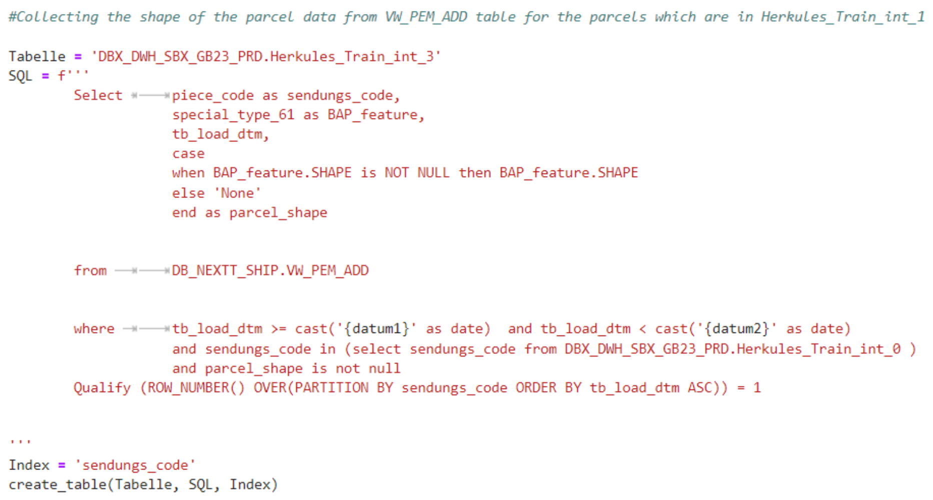
```
#get the  weight for the import parcels where we do not have the real weighing scale as pan weight from PZA_table
#(we are taking this because we do not have pan data for import parcels from private customers)


Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_1'
SQL = f'''

        select sendungs_code, ereignis_datum, gewicht as gewicht_pan
        from DB_NEXTT.PZA_event
        where   ereignis_datum >= cast('{datum1}' as date)  and ereignis_datum < cast('{datum2}' as date)
                        and Gewicht_Quelle not in ('W', 'Y')
                        and gewicht > 0
                        and sendungs_code in (Select sendungs_code from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_0)
        Qualify (ROW_NUMBER() OVER(PARTITION BY sendungs_code ORDER BY ereignis_datum ASC)) = 1


    '''
Index = 'sendungs_code'
create_table(Tabelle, SQL, Index)
```

A.12



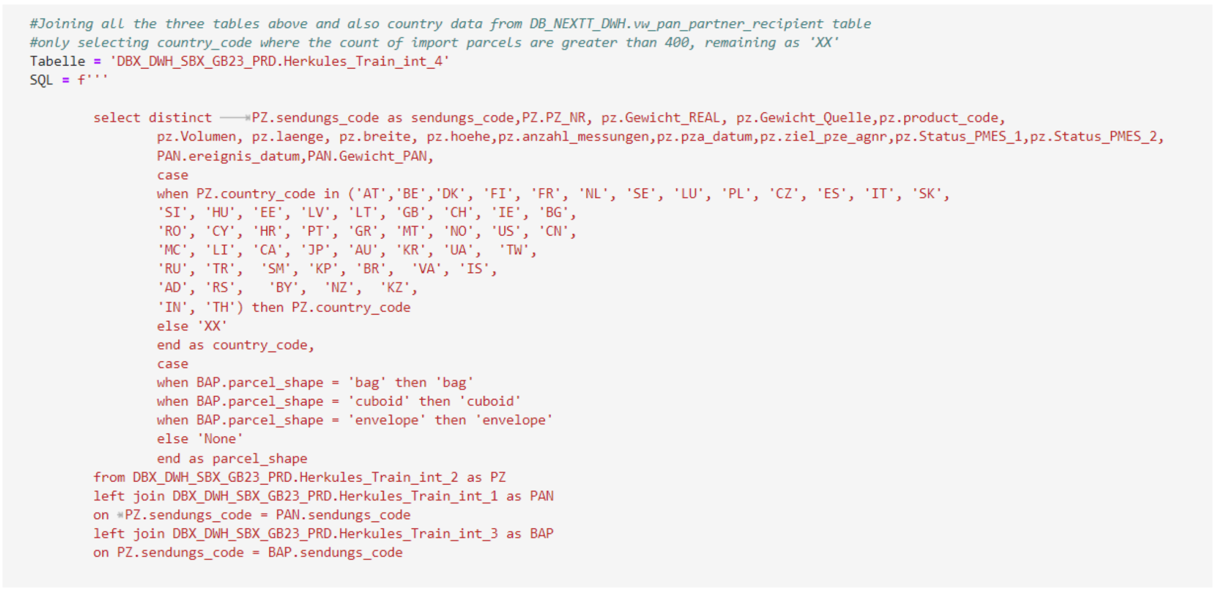
```
Select
    sendungs_code, ERFASSUNGSORT, LEIT_PRODUKT_CODE, gewicht, gewicht_quelle, volumen_pals, laenge, breite, hoehe,
    ereignis_zeitstempel, ereignis_datum, ereignis_typ,
    EXTRACT( HOUR FROM ereignis_zeitstempel)*60+EXTRACT(MINUTE FROM ereignis_zeitstempel) as zeit,
    cast(ziel_pz as int) as ziel_pz, ZIELKENNUNG_AGNR, WEIGHT_LEGAL_CODE

from DB_NEXTT.PZA_event
where   ereignis_datum >= cast('{datum1}' as date)  and ereignis_datum < cast('{datum2}' as date)
        and LEIT_PRODUKT_CODE not in (72, 73)
        and Gewicht_Quelle in ('W', 'Y')
        and gewicht > 0
        and sendungs_code in (Select sendungs_code from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_0)

union all


Select
    sendungs_code, ERFASSUNGSORT, LEIT_PRODUKT_CODE, gewicht, gewicht_quelle, volumen_pals, laenge, breite, hoehe,
    ereignis_zeitstempel, ereignis_datum, ereignis_typ,
    EXTRACT( HOUR FROM ereignis_zeitstempel)*60+EXTRACT(MINUTE FROM ereignis_zeitstempel) as zeit,
    cast(ziel_pz as int) as ziel_pz, ZIELKENNUNG_AGNR, WEIGHT_LEGAL_CODE
from DB_NEXTT.PZE_event
where   ereignis_datum >= cast('{datum1}' as date)  and ereignis_datum < cast('{datum2}' as date)
        and LEIT_PRODUKT_CODE not in (72, 73)
        and Gewicht_Quelle in ('W', 'Y')
        and gewicht > 0
        and sendungs_code in (Select sendungs_code from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_0)
```

A.11



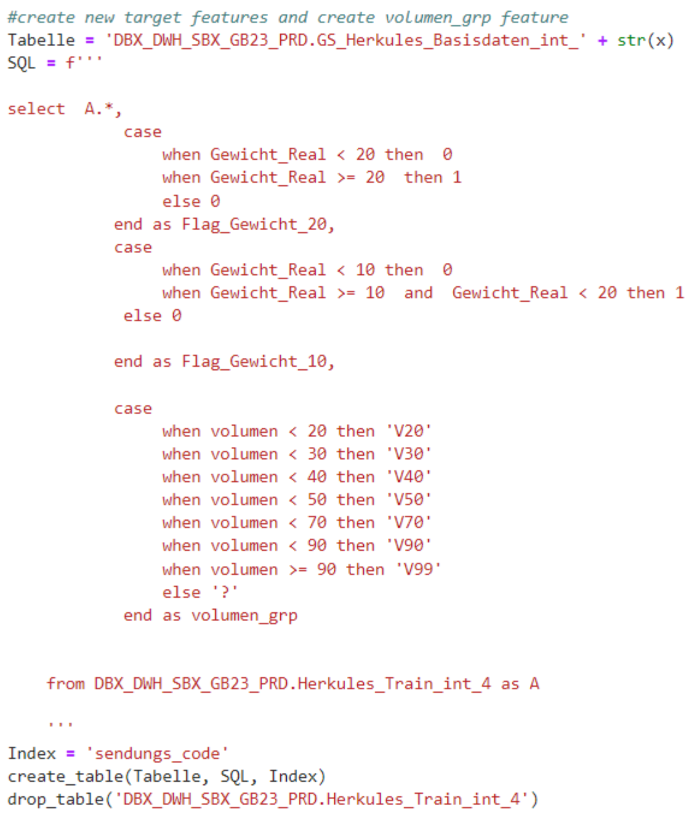
```
#Collecting the shape of the parcel data from VW_PEM_ADD table for the parcels which are in Herkules_Train_int_1

Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_3'
SQL = f'''
        Select          piece_code as sendungs_code,
                        special_type_61 as BAP_feature,
                        tb_load_dtm,
                        case
                        when BAP_feature.SHAPE is NOT NULL then BAP_feature.SHAPE
                        else 'None'
                        end as parcel_shape


        from            DB_NEXTT_SHIP.VW_PEM_ADD


        where           tb_load_dtm >= cast('{datum1}' as date)  and tb_load_dtm < cast('{datum2}' as date)
                        and sendungs_code in (select sendungs_code from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_0 )
                        and parcel_shape is not null
        Qualify (ROW_NUMBER() OVER(PARTITION BY sendungs_code ORDER BY tb_load_dtm ASC)) = 1


'''
Index = 'sendungs_code'
create_table(Tabelle, SQL, Index)
```

A.13



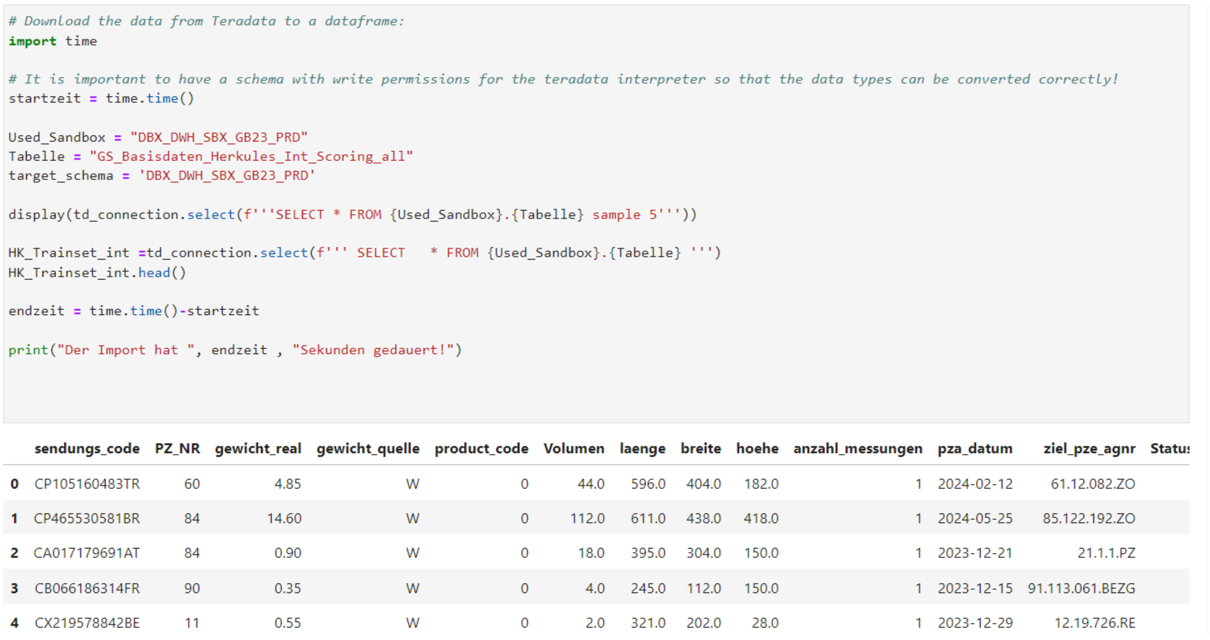
```
#Joining all the three tables above and also country data from DB_NEXTT_DWH.vw_pan_partner_recipient table
#only selecting country_code where the count of import parcels are greater than 400, remaining as 'XX'
Tabelle = 'DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_4'
SQL = f'''

        select distinct     PZ.sendungs_code as sendungs_code,PZ.PZ_NR, pz.Gewicht_REAL, pz.Gewicht_Quelle,pz.product_code,
                pz.Volumen, pz.laenge, pz.breite, pz.hoehe,pz.anzahl_messungen,pz.pza_datum,pz.ziel_pze_agnr,pz.Status_PMES_1,pz.Status_PMES_2,
                PAN.ereignis_datum,PAN.Gewicht_PAN,
                case
                when PZ.country_code in ('AT','BE','DK', 'FI', 'FR', 'NL', 'SE', 'LU', 'PL', 'CZ', 'ES', 'IT', 'SK',
                'SI', 'HU', 'EE', 'LV', 'LT', 'GB', 'CH', 'IE', 'BG',
                'RO', 'CY', 'HR', 'PT', 'GR', 'MT', 'NO', 'US', 'CN',
                'MC', 'LI', 'CA', 'JP', 'AU', 'KR', 'UA',  'TW',
                'RU', 'TR',  'SM', 'KP', 'BR',  'VA', 'IS',
                'AD', 'RS',   'BY',  'NZ',  'KZ',
                'IN', 'TH') then PZ.country_code
                else 'XX'
                end as country_code,
                case
                when BAP.parcel_shape = 'bag' then 'bag'
                when BAP.parcel_shape = 'cuboid' then 'cuboid'
                when BAP.parcel_shape = 'envelope' then 'envelope'
                else 'None'
                end as parcel_shape
        from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_2 as PZ
        left join DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_1 as PAN
        on  PZ.sendungs_code = PAN.sendungs_code
        left join DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_3 as BAP
        on PZ.sendungs_code = BAP.sendungs_code
```

A.14



```
#create new target features and create volumen_grp feature
Tabelle = 'DBX_DWH_SBX_GB23_PRD.GS_Herkules_Basisdaten_int_' + str(x)
SQL = f'''

select  A.*,
            case
                when Gewicht_Real < 20 then  0
                when Gewicht_Real >= 20  then 1
                else 0
           end as Flag_Gewicht_20,
           case
                when Gewicht_Real < 10 then  0
                when Gewicht_Real >= 10  and  Gewicht_Real < 20 then 1
            else 0

           end as Flag_Gewicht_10,

           case
                when volumen < 20 then 'V20'
                when volumen < 30 then 'V30'
                when volumen < 40 then 'V40'
                when volumen < 50 then 'V50'
                when volumen < 70 then 'V70'
                when volumen < 90 then 'V90'
                when volumen >= 90 then 'V99'
                else '?'
            end as volumen_grp


    from DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_4 as A

    '''
Index = 'sendungs_code'
create_table(Tabelle, SQL, Index)
drop_table('DBX_DWH_SBX_GB23_PRD.Herkules_Train_int_4')
```

A.15



```
# Download the data from Teradata to a dataframe:
import time

# It is important to have a schema with write permissions for the teradata interpreter so that the data types can be converted correctly!
startzeit = time.time()

Used_Sandbox = "DBX_DWH_SBX_GB23_PRD"
Tabelle = "GS_Basisdaten_Herkules_Int_Scoring_all"
target_schema = 'DBX_DWH_SBX_GB23_PRD'

display(td_connection.select(f'''SELECT * FROM {Used_Sandbox}.{Tabelle} sample 5'''))

HK_Trainset_int =td_connection.select(f''' SELECT   * FROM {Used_Sandbox}.{Tabelle} ''')
HK_Trainset_int.head()

endzeit = time.time()-startzeit

print("Der Import hat ", endzeit , "Sekunden gedauert!")
```

| | sendungs_code | PZ_NR | gewicht_real | gewicht_quelle | product_code | Volumen | laenge | breite | hoehe | anzahl_messungen | pza_datum | ziel_pze_agnr | Statu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | CP105160483TR | 60 | 4.85 | W | 0 | 44.0 | 596.0 | 404.0 | 182.0 | 1 | 2024-02-12 | 61.12.082.ZO | |
| **1** | CP465530581BR | 84 | 14.60 | W | 0 | 112.0 | 611.0 | 438.0 | 418.0 | 1 | 2024-05-25 | 85.122.192.ZO | |
| **2** | CA017179691AT | 84 | 0.90 | W | 0 | 18.0 | 395.0 | 304.0 | 150.0 | 1 | 2023-12-21 | 21.1.1.PZ | |
| **3** | CB066186314FR | 90 | 0.35 | W | 0 | 4.0 | 245.0 | 112.0 | 150.0 | 1 | 2023-12-15 | 91.113.061.BEZG | |
| **4** | CX219578842BE | 11 | 0.55 | W | 0 | 2.0 | 321.0 | 202.0 | 28.0 | 1 | 2023-12-29 | 12.19.726.RE | |

A.16

# Bibliography

[1] Akash. Decision tree, 2024. URL https://imakash3011.medium.com/decision-tree-91adec4370d1. Accessed: 2024-08-09.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Jason Brownlee. Overfitting and underfitting with machine learning algorithms. *Machine Learning Mastery*, 21:575, 2016.

[4] Adithi D. Chakravarthy, Sindhura Bonthu, Zhengxin Chen, and Qiuming Zhu. Predictive models with resampling: A comparative study of machine learning algorithms and their performances on handling imbalanced datasets. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1492–1495, 2019. doi: 10.1109/ICMLA.2019.00245.

[5] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT press, 2006.

[6] Wikipedia contributors. Unsupervised learning, 2023. URL https://en.wikipedia.org/wiki/Unsupervised_learning. [Online; accessed on 30.07.2024].

[7] DataDrivenInvestor. Data preprocessing, 2024. URL https://medium.datadriveninvestor.com/data-preprocessing-3cd01eefd438. Accessed: 2024-08-09.

[8] Mohammed Amine El Mrabet, Khalid El Makkaoui, and Ahmed Faize. Supervised machine learning: a survey. In *2021 4th International conference on advanced communication technologies and networking (CommNet)*, pages 1–10. IEEE, 2021.

[9] GeeksforGeeks. Types of machine learning. https://www.geeksforgeeks.org/types-of-machine-learning/, 2023. Accessed: 26 July 2024.

[10] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2019.

[11] Ramin Ghorbani and Rouzbeh Ghousi. Comparing different resampling methods in predicting students' performance using machine learning techniques. *IEEE Access*, 8: 67899–67911, 2020. doi: 10.1109/ACCESS.2020.2986809.

[12] Deniz Gunay. Random forest, 2024. URL https://medium.com/@denizgunay/random-forest-af5bde5d7e1e. Accessed: 2024-08-09.

[13] Michal Haltuf. *Support Vector Machines for Credit Scoring*. PhD thesis, 08 2014.

[14] R Hossain and Douglas Timmer. Machine learning model optimization with hyper parameter tuning approach. *Glob. J. Comput. Sci. Technol. D Neural Artif. Intell*, 21(2):31, 2021.

[15] Analytics India Magazine. Complete guide to understanding precision and recall curves. https://analyticsindiamag.com/developers-corner/complete-guide-to-understanding-precision-and-recall-curves, 2024. Accessed: 2023-10-05.

[16] Batta Mahesh. Machine learning algorithms -a review, 01 2019.

[17] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 243–248, 2020. doi: 10.1109/ICICS49469.2020.239556.

[18] Ingmar Nitze, Urs Schulthess, and H Asche. Comparison of machine learning algorithms random forest, artificial neural network and support vector machine to maximum likelihood for supervised crop type classification. 05 2012.

[19] Eric Nyakundi. *Using support vector machines in anomaly intrusion detection.* PhD thesis, University of Guelph, 2015.

[20] A. Omar and T. Abd El-Hafeez. Quantum computing and machine learning for arabic language sentiment classification in social media. *Scientific Reports*, 13:17305, 2023. doi: 10.1038/s41598-023-44113-7. URL https://doi.org/10.1038/s41598-023-44113-7.

[21] Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions, 5 2021. ISSN 26618907.

[22] Cullen Schaffer. Selecting a classification method by cross-validation. *Machine learning*, 13:135–143, 1993.

[23] Towards Data Science. Data types from a machine learning perspective with examples. https://towardsdatascience.com/data-types-from-a-machine-learning-perspective-with-examples-111ac679e8bc, n.d. Accessed: 26 July 2024.

[24] scikit-learn developers. $R^2$ score. https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score, n.d. Accessed: 2023-10-05.

[25] Black Shadow. Exploring random forests: A beginner's guide to ensemble learning, 2024. URL https://blackshadow.hashnode.dev/exploring-random-forests-a-beginners-guide-to-ensemble-learning. Accessed: 2024-07-25.

[26] Lily-belle Sweet, Christoph Müller, Mohit Anand, and Jakob Zscheischler. Cross-validation strategy impacts the performance and interpretation of machine learning models. *Artificial Intelligence for the Earth Systems*, 2(4):e230026, 2023.

[27] Chinedu L Udeze, Idongesit E Eteng, and Ayei E Ibor. Application of machine learning and resampling techniques to credit card fraud detection. *Journal of the Nigerian Society of Physical Sciences*, pages 769–769, 2022.

[28] Zeljko Vujovic. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, Volume 12:599–606, 07 2021. doi: 10.14569/IJACSA.2021.0120670.

[29] Hersan Yağcı. Random resampling methods for imbalanced data with imbalanced-learn (imblearn). https://hersanyagci.medium.com/random-resampling-methods-for-imbalanced-data-with-imblearn-1fbba4a0e6d3, 2023. Accessed: 2024-08-05.

[30] Hongwei Zhao, Mingzhao Li, Taiqi Wu, and Fei Yang. Evaluation of supervised machine learning techniques for dynamic malware detection. *International Journal of Computational Intelligence Systems*, 11(1):1153–1169, 2018.

[31] Xiaojin Zhu and Andrew B Goldberg. *Introduction to Semi-Supervised Learning*, volume 3. Morgan & Claypool Publishers, 2009.