CONCORDIA UNIVERSITY

Data and Simulation Models for Route Optimization in Vehicle Routing Problem

by

Monika Sharma

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science

in the

Department of Computer Science and Software Engineering
Faculty of Engineering and Computer Science

November 2018

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Monika Sharma**
Entitled: *Data and Simulation Models for Route Optimization in Vehicle Routing Problem*

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair
Dr. Yann-Gael. Gueheneuc

_____ Examiner
Dr. Emad Shihab

_____ Examiner
Dr. Leila Kosseim

_____ Supervisor
Dr. Brigitte Jaumard

_____ Supervisor
Dr. Tristan Glatard

Approved _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

CONCORDIA UNIVERSITY

Abstract

Department of Computer Science and Software Engineering
Faculty of Engineering and Computer Science

Master of Computer Science

by Monika Sharma

The generalization of online commerce to a wide range of industries is transforming customer's practices by removing the requirement of visiting the physical stores. This has made it essential to develop a system to improve pickup/delivery of items ordered online, in particular, to avoid the extra costs associated with failed home pickup/deliveries. This thesis aims to contribute to the development of such a system by developing failure prediction data models and a prototype of a simulator leveraging these models to improve the planning of routes.

We use Random Forest classifiers to build failure prediction models. Three data re-sampling strategies are used to address class imbalance issue as the proportion of the failed services is much less than that of the successful ones. To interpret classification results, we extract Association Rules where the antecedent is a set of service features and the consequent is a failed service status. To avoid the memory limitations often caused for large datasets, we design a two-step algorithm to first extract Association Rules on the failed services. Then the limited set of rules are obtained based on the frequencies of the antecedents in the complete dataset and in the dataset containing only failed services. The simulation model is developed using the SimGrid library. It simulates route generated by the optimization model, introduces random service failures and computes total traveled distance and time.

We obtain good prediction results on the real dataset (aggregated dataset). Our classifier reaches an average sensitivity of 0.7 and an average specificity of 0.7 for the 5 studied types of failure. Association Rules reassert the importance of confirmation calls to prevent failures due to customers not at home, show the importance of the time window size, slack time, and geographical location of the customer for the other failure types, and highlight the effect of the retailer company on several failure types. The simulation model is successfully validated using sample routes. To reduce the occurrence of service failures, our data models could be coupled with optimizers through simulation or used to define counter-measures to be taken by human dispatchers.

Acknowledgements

First, I wish to express my gratitude to my thesis supervisors, Dr. Brigitte Jaumard and Dr. Tristan Glatard, for all their guidance and support throughout my time as a master's student at Concordia University. They have always been engaged and helpful in my work and ready with feedback and comments that greatly assisted me during my studies.

Second, I would also like to thank my family and friends. I would not have had this amazing opportunity to pursue my dreams if it was not for your continuing support, encouragement and love; and for always believing in me.

Contents

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or
CDP	**C**ollection **D**elivery **P**oint
DC	**D**istribution **C**enter
PDPTW	**P**ickup And **D**elivery **P**roblem with **T**ime **W**indows
RF	**R**andom **F**orest
DT	**D**ecision **T**ree
VRP	**V**ehicle **R**outing **P**roblem
FP-Growth	**F**requent **P**attern **G**rowth
NAH	**N**ot **A**t **H**ome
SMOTE	**S**ynthetic **O**ver-sampling **TE**chnique
ML	**M**achine **L**earning
AI	**A**rtificial **I**ntelligence
OOB	**O**ut **O**f **B**ag

Chapter 1

Introduction

The worldwide increase in electronic commerce (e-commerce) has enabled consumers and businesses to buy and sell a vast variety of items online. Retail shares a large part of e-commerce: as an example, retail e-commerce sales in Canada were worth 24.03 billion US dollars in 2018, which is expected to grow to 31.98 billion US dollars in 2022 [3]. Figure 1.1 shows the growth of retail e-commerce in Canada.

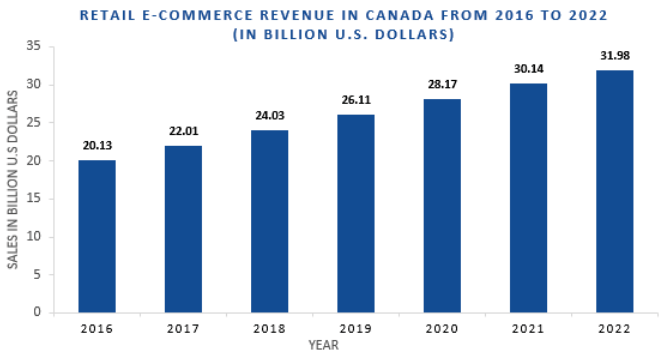


FIGURE 1.1: Retail e-commerce revenue in Canada [3].

With increased e-retail, the demand for home delivery of items also increases. The distribution of items from local distribution centers to end-customers is called *The Last Mile Distribution*. This is an important step in supply chain logistics and comprises 28% of the total delivery cost [26]. However, it is inefficient due to the involvement of many stochastic parameters like traffic, weather, the behavior of the end customer, and failures. Many items ordered online like day-to-day grocery items (which are refrigerated) and

others like electronic items which can not be left unattended, require customers presence for delivery. Failures in such home deliveries are frequent for multiple reasons like the customer is not at home or refusal of an item if the expectations are not met. Incorrect customer address and late driver are other popular causes of failures. Failed home deliveries cause loss to the carriers and retailers and often leave unhappy customers. This can also leave a negative effect on the brand name of the product.

1.1 Background

The route for attended home deliveries/pickups (where the customer is required to be present at home) is generated by solving an optimization problem known as Pickup and Delivery Problem with Time Window (PDPTW). The current approaches to solve PDPTW include heuristics, meta-heuristics and exact methods [7, 27, 30]. Some major limitations of these approaches are that they are not robust to stochastic customer demand and do not take failed home pickups/deliveries into account. The solution suggested in the literature to minimize failure into the last mile include Collection Delivery Point (CDP) [32], Reception Box, and Delivery Box [29]. CDP's are proved very efficient to improve last mile efficiency. However, with CDP's true meaning of home deliveries is not achieved and often customer's satisfaction is compromised. In this thesis work, we study the failures in PDPTW and develop data models which could be used with optimization models to prevent failures.

1.2 Goals and Contributions

E-commerce has revolutionized the traditional shopping style and made it necessary to have cost-effective and efficient pickup/delivery of items. To ensure efficient home delivery of items, a system should be built which takes stochastic parameters into account and avoid service failures for route optimization. It should be sensitive to customer's convenience in order to avoid unhappy customers. Such a system could be developed by combining optimization models with data driven simulation models. An advance simulation model can include service failures data models and stochastic parameters like

bad weather and traffic. Data models could be developed by studying failure reasons and customer's behaviour from existing datasets. The long term goal is to integrate this simulation model into an optimization model to build a system which will generate realistic routes and prevent failures.
The goal of this thesis is to contribute in the development of such a system by:

1. Developing failure prediction data models for different failure types.

2. Implementing a prototype simulation model for simulation optimization.

In this thesis work, we investigate five different failure types for attended home deliveries in Pickup and Delivery Problem with historical data provided by ClearD [11] and develop failure prediction models for each failure type. ClearD is a Montreal based company providing solutions for supply-chain optimization. They provided the dataset which contains 523,643 customer pickup and delivery services scheduled between September 2017 and February 2018 in Canada. A failure model will predict if the service will fail with a specific type. With the failure models, we are able to predict 70% of the failures. We extract Association Rules to better understand the failure reasons for each failure type. The Association Rules provides finer insights of the data and complement the classification results. To facilitate the integration of these failure models into optimization model of ClearD, a simple simulation model is implemented. This simulation model is built to provide a proof of concept for simulation-optimization that simulates a route. The simulation model evaluates the quality of the route in terms of route length, time, and failed pickup/deliveries, of the route generated by optimization model [31].

The contribution of the thesis work on service failure prediction and Association Rules extraction has been accepted for publication in the IEEE Big Data Conference, 2018. This work has also been patented by ClearD [11].

1.3 Plan of the Thesis

The thesis is organized as follows. In Chapter 2, we discuss in details the problem background in PDPTW, methods used to build failure prediction data models and extract

Association Rules, tools used to build simulation model. Chapter 3, introduces problem statement, data set used and implementation of failure prediction data models. Chapter 4 presents the result of failure models and Association Rules. Simulation model is discussed in Chapter 5 followed by simple validation in Chapter 6. The thesis work is concluded in Chapter 7.

Chapter 2

Literature Review and Related Work

In this chapter, we review the background of Pickup and Delivery with Time Window, define services and stops and investigate current approaches to optimize service failures and their limitations. To build a data model for predicting service failure, we first review briefly a few classical supervised machine learning algorithms. Our dataset, provided by ClearD, has more successful services compared to failures. So, we discuss the problems associated with the study of imbalanced datasets and review solutions suggested in literature. To validate the classification results and get a better understanding of failure reasons, we extract Association Rules. The algorithm to find Association Rules is discussed in Section 2.4. In Section 2.6, we review simulation model in general, simulation in PDPTW and their limitations. We discuss the SimGrid library, used to build the simulation model, in Section 2.6.1.

Pickup and Delivery with Time Windows

Pickup and Delivery Problem with Time Windows (PDPTW) [31] is the problem of serving a number of customer requests with time, capacity and precedence constraints. It is a generalization of Vehicle Routing Problem (VRP) to design an optimal set of *routes* to minimize the travelled distance. A *route* is defined as a sequence of *stops* associated

with customer locations, where *services* are delivered with time-window, capacity and precedence constraints. Each *stop* has a pickup or a drop location. The pickup location is generally a local distribution center and the drop location is end customer. A stop is defined with a fixed demand, service time and must be processed in order to respect precedence constraints. As shown in Figure 2.1, consider the route 2, $C2$ is a predecessor pickup stop for stop represented by $C4$. So $C2$ must be visited before to pickup the item which will be delivered at stop $C4$. The service time is an estimate of the time required to make the pickup/delivery. Also, there is a time window attached to each *stop* that defines the time interval to serve the stop. This time window is often required while making attended pickup/deliveries where the customer is required to be present at home when the service is performed. The generated set of routes are evaluated by travelling time and distance, number of vehicles used and number of customer requests fulfilled.

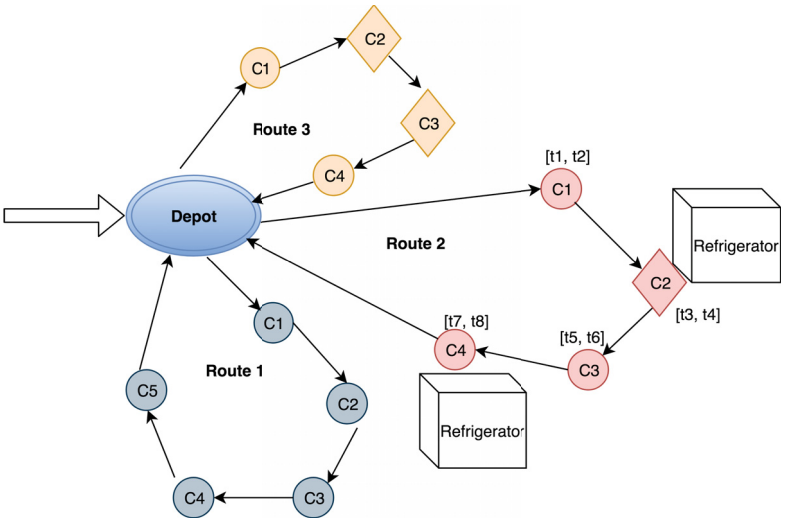


FIGURE 2.1: Pickup and Delivery Problem

In the next section, we discuss current approaches used to solve pickup and delivery problem, measures taken to optimize service failure and their limitation. We also introduce our approach for predicting service failures.

2.1 Failure Prediction Model

Pickup and Delivery Problem with Time Windows is a well studied problem. Many approaches are suggested in the literature including heuristics, and meta-heuristics and

exact algorithms. A good example of efficient meta-heuristic is Adaptive Large Neighbourhood Search (ALNS) [31] method. ALNS uses multiple greedy heuristics to avoid obtaining local solutions. However, this method does not take service failures and stochastic parameters into account. Other existing solutions approaches for PDPTW as suggested in [7, 27, 30], to the best of our knowledge, do not take service failures into account.

Service failure is an important problem in the last mile distribution and several alternative approaches are suggested in the literature to address the problem of failed home deliveries. These approaches are broadly classified into changes in route, changes in location, changes in time and customer behavior by [33]. For attended home deliveries, CDP [32] is suggested as an alternative; Reception Box and Delivery Box [29] are suggested for unattended deliveries but cannot work for "large" electrical appliances.

CDPs like a convenience store, post office, railway station and popular store in the customer area, are used by some carriers as an alternative to dropping the item. Often a notification to the customer is made via message or a note left on the door. This way the customer can collect the item from the CDP and in case of a return, often the item is left to the CDP to be collected by the carriers. CDP as an alternative has been proved an effective way for the first attempt failed home deliveries over traditional methods where the carriers make another delivery attempt in 24 hours. In the work of [32] and [29], an assumption that the customer can always pick up and drop the item from CDP was made. However, certain age groups like elderly, may find it inconvenient to visit CDP and uncomfortable to pickup/drop heavier parcels or may not be able to drive to CDP's. Also, CDP's may not be the best alternative for all parcel, in particular parcels containing large items.

An interesting approach is suggested in [33], where historical delivery data was used to improve the efficiency of the home deliveries by developing address intelligence. The study makes an assumption that the service failures are specific to the geographic region. Delivery efficiency of the geographic area is calculated using the demographic and customer family characteristics using linear regression model. However, this study was conducted in the Netherlands and lacked in exhaustive experiments on the data collected

from multiple countries and carriers. Also throughout the literature, the term parcel has not been clearly defined in terms of the physical characteristics like weight, volume, and cost of the item. Not all the items can be left unattended or at CDP's.

As discussed in Chapter 1, service failure is an important problem and not only causes additional costs to carriers but also damages the product's brand name and retailer's reputation. There are many existing approaches (like CDPs, Delivery Box and Reception Box) to address failures, however, we could not find any relevant data-driven models for predicting the failures in advance to improve route quality.

Advanced techniques like Machine Learning (ML), Artificial Intelligence (AI), and Data Analytic are largely explored in many fields including health care [10], finance stability prediction [22], stock market [8] and analyzing the data generated on social networks. However, the use of these techniques is limited in supply chain logistics. We aim at predicting and explaining the cause of such failures focusing on the last-mile pickup and delivery of items using machine learning algorithm. In the next section, we review the most common supervised machine learning algorithms and ideal scenarios to use them.

2.2 Supervised Machine Learning Algorithms

Supervised learning is the process of learning from the historical data where the training dataset contains both input and output variables. Supervised learning problems are divided into two broad categories: Classification and Regression. A classification problem has a discrete real valued output variable, i.e., the class in which the instance belongs. A regression problem has a continuous real value output variable. Some frequently used supervised machine learning are linear regression, decision tree, support vector machine (SVM) and random forest. There are pros and cons of these algorithms. Linear regression could separate well the data which is linearly separable and works well for non-redundant features. The decision tree is easy to understand but suffers from over-fitting. SVMs are designed to reduce over-fitting but they are complex to explain and communicate. Random Forest [5] is robust to over-fitting and easy to explain, however, takes a lot of time and memory to learn if parameters values are very large. We use

Random Forest supervised machine learning algorithm to forecast the failure types. One important reason to use Random Forest classifier is that it provides list of important features to explain the classification results. These important features help to better understand the dataset which is one of the requirement for our problem. In Section 2.3, we discuss why we selected Random Forest for analysis of our data set. Therein, we also review the Decision Tree & Random Forest algorithm [5] and data re-sampling strategies in details.

2.3 Selected Methods

Random Forest is an ensemble ML method used for both classification and regression learning problems [5]. Ensemble methods are popular ML algorithms which build many individual classifiers and combine the prediction over all the classifiers to predict a new data. Decision trees are basic building blocks of the Random Forest. However, decision trees that are grown really deep to learn highly irregular patterns often overfit the training sets. A slight noise in the data may cause the tree to grow in a completely different manner. This means that a decision tree classifier fits training data too well and may give 100 % accuracy on training data but may perform poorly on the test data. In technical terms, we say that decision tree suffers from high variance [14]. The solution is to use a collection of many high variance decision trees. Random Forest works by training multiple decision trees on a different sub-sample the training data without increasing the bias. The service failure dataset, we are studying in this thesis, is high dimensional and may not be linearly separable which also motivates us to use Random Forest classifier which works well with high dimensional linearly un-separable data [6].

2.3.1 Decision Trees

The Decision Tree is a supervised ML algorithm that learns by splitting the data based on feature values. A Decision Tree is a directed acyclic graph in which any two nodes are connected by only one path, where nodes correspond to some tests on data features, a branch represents a decision path and a leaf corresponds to a class label. A Decision

Tree uses features of training data to split at each node. We use the Classification And Regression Tree (CART) [6] implementation in scikit-learn [28] which makes binary split by considering one feature at a time. We performed experiments to split a node with both *Gini Index and Entropy*. We finally choose *Gini index* [6] to split at a node as it perform best on our dataset.

Gini index: It is the measure of the impurity of the data in a node. Small values of *Gini index* means that the node contains a large proportion of a single class. Let D be the dataset, assume it contains n classes. Let p_1, p_2, .. p_n be the frequencies of classes in D.

$$gini(D) = 1 - \sum_{j=1}^{n} {p_j}^2. \tag{2.1}$$

It is a measure of total variance in all n-classes.

$Gini(D)$ after splitting the dataset into D1 and D2 on an attribute A, is calculated as follows:

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2). \tag{2.2}$$

$$\text{Reduction in impurity } (\Delta gini_A\,(D)) = gini(D) - gini_A(D) \tag{2.3}$$

Consider the same data set D, with two classes represented by '+' and '-'. Assume D has 14 [9+, 5-] samples ('+' and '-' represents positive and negative classes). Let *Wind* (*Weak, Strong*), be an attribute in D which has two values. Gini index for this example is calculated in the following way.

$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = \ 0.4591.$
If we split D based on the attribute *Wind*,
$gini_{Wind}(D) = \frac{|D_{Weak}|}{|D|} gini(D_{Weak}) + \frac{|D_{Strong}|}{|D|} gini(D_{Strong})$
$= \left(\frac{8}{14}\right) \left[1 - \left(\frac{6}{8}\right)^2 - \left(\frac{2}{8}\right)^2\right] + \left(\frac{6}{14}\right) \left[1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2\right]$
$= 0.4284.$
So, reduction in impurity,
$(\Delta gini_{Wind}\,(D)) = gini(D) - gini_{Wind}(D) = 0.4591 - 0.4284 = 0.0307.$

The reduction in impurity is calculated for all the attributes in D and the attribute with the highest value of *Reduction in Impurity* is selected as the best splitting attribute.

Now we describe the CART algorithm to construct a Decision Tree [6]. A Decision Tree is constructed in a top-down fashion, each time selecting the best attribute from the available attributes. At the root node, we have all the data. We summarize the main steps in CART assuming two classes in the data set in Algorithm 1.

Algorithm 1 Classification and Regression Tree (Adapted from [6])

Input- *Samples:* Dataset containing all the instances; *Target_attribute:* List of output classes; *Attributes:* list of attributes to be tested at each node for split.

Output: A Decision Tree.

CART(*Samples, Target_attribute, Attributes*)

1: Create a *Root* node for the tree and assign all the *Samples* to it.

2: If all *Samples* are positive, return the single-node tree *Root*, with label = +

3: If all *Samples* are negative, return the single-node tree *Root*, with label = -

4: Otherwise,

$A \leftarrow$ the attribute in *Attributes* with value v_k that best classifies *Samples*.

The decision attribute to split at Root $\leftarrow A$.

Add two branches to the root node.

Let $Samples_{v_k}$ be the subset of *Samples* that have value $v_i < v_k$ for A.

Add a node to the left branch and assign $Samples_{v_k}$ to it.

CART($Samples_{v_k}$, *Target_attribute, Attributes*).

Add a node to the right branch and assign subset ($Samples - Samples_{v_k}$) to it.

CART($Samples - Samples_{v_k}$, *Target_attribute, Attributes*).

5: Return Root.

The default stopping criteria in Algorithm 1, is when the node has all its instances from the same class. This often leads to over-fitting. So, in practice, the stopping criteria is specified as the maximum depth of the tree or the maximum number of the leaf nodes. The best attribute in Step 4 is calculated using *Gini index* as explained above.

2.3.2 Random Forest

Random Forest is a collection of many Decision Trees. The Random Forest classifier predicts new data on each Decision Tree in the forest. For regression problems, the output of each Decision Tree is combined and averaged to get the final prediction value of the new data. For classification problems, the occurrence of each class in the output of Decision Tree is counted. The data is then classified in the class that occurs more often in the output. In the interest of our study to build classification models, we would limit the discussion to the Random Forest classifier.

Constructing Random Forest

Random Forest is easy to construct once we know how to construct a Decision Tree. From the training data set, a number of sub-samples (a sub-sample is a subset of the training dataset) are drawn with replacement, and a Decision Tree is constructed on each sub-sample. These Decision Trees build up a Random Forest. To build a Random Forest classifier of k trees, k random sub-samples, each containing N (total number of samples in the training dataset) or fewer samples are drawn from the training data. These sub-samples are called bootstrap samples and may have duplicate instances of the data as these instances are chosen randomly with replacement. This process of generating multiple sub-samples is called bootstrap aggregation, i.e., bagging.

When constructing a Decision Tree in Random Forest, a subset of features is selected at random before each split. The size of the subsets has an impact on the prediction accuracy of the classifier. If the subsets have almost all the features of the data then the trees will be highly correlated and the error rate will increase. If the subset of features is too small, the tree will be weak which will again increase the error rate. In scikit-learn,, the square root of the total number of features is used and the number of features remains constant throughout the tree construction process. We summarize the steps explained in [14] to construct a Random Forest in Algorithm 2.

Algorithm 2 Random Forest for Regression or Classification (From [14]).

Input:- D, N: Training dataset and size; p: Total number of features; B: Number of Decision Trees; n_min: Minimum number of samples in a node.

1: **for** $b \leftarrow 1$ *to* B **do**

2: Draw a bootstrap sample Z of size N from the training dataset D.

3: Grow a random-forest tree T_b from Z, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_min is reached.

i. Select a random subset, m variables from p.

ii. Pick the best variable/split-point among the m. (Using *Gini index* or other methods).

iii. Split the node into two child nodes.

4: Output the ensemble of trees $T = \{T_b : b \, \epsilon \, B\}$

5: **end for**

Let $T_b(i)$ denotes the predicted output of Decision Tree T_b, for sample i. To make a prediction at a new point x:

Regression: $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class predicted of bth random-forest tree.

Then $\hat{C}_{rf}^{B}(x) =$ majority vote $\left\{\hat{C}_b(x)\right\}_1^B$.

2.3.3 Re-sampling

The dataset (discussed in Chapter 3) used in our study is very imbalanced as the number of failed samples account for 7.53% of the total number of samples. There are several issues associated with the study of an imbalanced dataset. One of the most common issues is that the learned model is completely biased toward the majority class (class with a large number of instances) and will not detect instances from the minority class (class with the lesser number of instances). The class imbalance problem is well known and has been studied greatly in the literature [9, 23]. There are many approaches to solve this problem like cost-sensitive learning, recognition-based learning and re-sampling of

the training set [25]. In our study, we use re-sampling methods on our dataset because it is suitable for sufficiently large data set [25] and relatively easy to understand.

Re-sizing the training set involves re-sampling the original training data. There are two ways to re-sample the data either by over-sampling or under-sampling. The process of adding data to the minority class is called over-sampling and the process of removing data from the majority class is called under-sampling. The data from the majority class could be removed randomly or strategically to minimize the loss of information in the majority class. On the other hand, to add data in minority class, one way is to simply replicate the minority class data or generate new synthetic data from the available data. We use Synthetic Minority Over-sampling TEchnique (SMOTE) [8] to over-sampling (generate new data in the minority class) and NearMiss [25] for under-sampling (remove data from the majority class).

SMOTE

SMOTE [8] is a data resampling technique in which new minority class data is generated using the training data set. One way to reduce class-imbalance is by adding more samples in the minority class. SMOTE-regular generates new data by finding the k-nearest neighbors for each minority class instance. Then one of the k-nearest neighbors is selected at random and the new minority class data is placed on the line between these two instances. For example, consider an instance $R_i = (6, 4)$ in the minority class and let $R_k = (4, 3)$ be its nearest neighbor selected randomly from the k-nearest neighbors. The new instance (R_{new}) is created in the following way.

$$\begin{aligned} R_{new} &= R_i + rand[0-1] \times (R_k - R_i) \\ &= R_i + rand[0-1] \times (R_{k1} - R_{i1},\ R_{k2} - R_{i2}) \\ &= (6, 4) + rand[0-1] \times (4-6,\ 3-4) \\ \text{or } R_{new} &= (6, 4) + rand[0-1] \times (-2, -1) \end{aligned}$$

where rand [0-1] generates a random number between 0 and 1. We could specify the number of nearest neighbors and an over-sampling ratio to better match the dataset used.

NearMiss

NearMiss [25] is an under-sampling strategy for the majority class which uses k-nearest neighbors for deciding which data to keep in the majority class. There are three variant of the NearMiss algorihm: NearMiss-1, NearMiss-2 and NearMiss-3. NearMiss-1 retains majority class instances whose mean distance to k nearest neighbors (in the minority class) is the lowest i.e., the instance close to some minority class instance are kept. NearMiss-2 keeps majority class instances for which the mean distance to the k farthest point is the lowest i.e., majority class instances which are close to all minority class instances are kept. We use NearMiss-3 which is a two-step algorithm. First, for each minority class instance, k-nearest neighbors from the majority class is selected. Then, samples in the majority class with the highest average distance to the k-nearest neighbor in the minority class are selected. The number of nearest neighbors and sampling ratio are input to the algorithm and adjusted to fit the dataset.

Random Undersampling

We also assessed a straightforward random resampling strategy which is often used for skewed datasets. SMOTE and NearMiss use Euclidean distance to compute the nearest neighbor which may not be efficient for encoded data values. We have performed random under-sampling [4] in which instances from the majority class are selected randomly until the dataset has equal class proportion.

2.3.4 Evaluation Metrics

We consider the classical evaluation metrics to measure the efficiency and reliability of the model [14, 34], which we recall in Table 2.1.

Metric	Definition
True Positive (t_p)	Number of positive samples that are predicted Positive
True Negative (t_n)	Number of negative samples that are predicted Negative
False Positive (f_p)	Number of negative samples that are predicted Positive
False Negative (f_n)	Number of positive samples that are predicted Negative
Positives (P)	$t_p + f_n$
Negative (N)	$t_n + f_p$
Accuracy	$\frac{t_p+t_n}{t_p+t_n+f_p+f_n}$
Sensitivity	$\frac{t_p}{t_p+f_n}$
Specificity	$\frac{t_n}{t_n+f_p}$

TABLE 2.1: Performance Evaluation Metrics

Accuracy measures the portion of test samples classified correctly. In general, accuracy is a good measure, however for an imbalanced dataset, where class proportion differs significantly, accuracy alone is not sufficient to interpret the results. For such datasets, Confusion Matrix [34] parameters are often used to evaluate the performance of the classifier. Sensitivity measures the ability of a classifier to correctly identify positive class samples while specificity measures the classifiers ability to correctly identify negative class samples [14, 34].

Out Of Bag Error (OOB) (Adapted from [14]): Random Forest uses a bootstrap sample to construct a tree (Section 2.3.2). Usually, the size of a bootstrapped sample is smaller than the original data set [14]. The left out data of each bootstrap sample is used to validate the performance of the tree constructed on a bootstrap sample other than the left out sample. The error on the left out data is called OOB error [14] and it is important to analyze the performance of individual trees. This way, each tree is constructed and validated on a different data and there is no need to do cross-validation test to get the test error [14]. In Section 2.4, we review the underlying ideas in association rules and discuss in detail in Chapter 3.

2.4 Association Rules

An important outcome of Random Forests is the list of important features describing the output of the classifier. The importance of a feature X_m is defined as the *total decrease in node impurity* (defined in Section 2.3.1), *weighted by the probability of reaching that node, averaged over all trees of the Random Forest. The probability of reaching a node is approximated by the proportion of samples reaching this node [24].* However, feature importance provides limited insights to interpret classification results. One of the reasons is that correlation among a group features reduces the mean importance in this group of features, as discussed in [15]. To further interpret the results, we extracted the so-called Association Rules [1].

Association Rules are explained with the *market-basket* model to describe relationships in *items* and *baskets (set of items).* The number of items in each basket is less than the number of baskets. An association rule is defined as: $I \rightarrow j$ *where* I *is a set of items and* j *is an item* [21]. *This association rule is interpreted as if all of the items in* I *appear in some basket (consists of a set of items), then* j *is likely to appear in that basket as well.* Here I is called *antecedent* and j is called *consequent.* The confidence of the rule $I \rightarrow j$ is defined as the ratio of the support (number of baskets for which I is a subset) for $I \bigcup \{j\}$ to the support for I. We use the Frequent Pattern Growth (FP-Growth) [18], an efficient and scalable algorithm for mining association rules. It uses an extended prefix tree to find the frequent patterns and overcome the main memory limitation often caused by frequent items in Apriori algorithm [17]. We discuss in detail how to model our dataset to get the basket of items and the algorithm to extract association rules in Chapter 3.

2.5 Tools Used

We used scikit-learn version 0.19.1 [28], a machine learning library in Python for designing the data model using the Random Forest. It is an open source free software, provides a large number of ML algorithms for regression, classification and clustering algorithms. Imbalanced-learn version 0.3 [20] provides support for a large number of

re-sampling techniques for class imbalance problems. We also use Apache Spark version 2.3.1 [35], to find Association Rules using the FP-Growth algorithm.
In the next section (Section 2.6), we review the simulation model in general, advantages of building a simulation model for real life problems and how simulation optimization could be used together to improve the performance of the system. We also review SimGrid [19] library in the next section.

2.6 Simulation Optimization

A simulation model is the representation of a physical or abstract system which mimics a system's important characteristics and behavior. In computer science, a simulation model is a software program that is implemented in some programming language to represent a system. Simulation is the process of executing a simulation model. Simulations can be used for deterministic as well as stochastic problems without mathematical sophistication. In this thesis, we build a prototype simulation model for pickup and delivery of last mile logistics in the supply chain.

A major limitation of mathematical models is that they do not take stochastic parameters into account while route planning. The pickup and delivery system, modeled by optimization models, is complex and involves random parameters like weather, traffic, and service failures. So the routes generated by optimizer are often modified often by the dispatchers to make them more realistic. These parameters are difficult to capture using only an optimization model. ClearD uses databases for integrating real time traffic. Different failures type and weather conditions are not taken into account in the current optimization model. The simulation models can model the randomness and integrate failure models. Such simulation model then can be used into the optimization model guide the route planning process and generate more realistic routes. This is called Simulation-based Optimization (S-O) [13] where the optimization model takes feedback from the simulation model as shown in Figure 2.2.

The use of simulation model in the last mile is not new. An advanced simulation model developed in [16] takes variable number of costs like time window size, manned vs

unmanned delivery, cost of failed deliveries etc to calculate the total cost of the route. A simulation model was developed in [12] to analyze the carbon footprints of vehicles last mile. Feasibility of cargo bike delivery [2] options is studied using simulation to suggest an economic, environment friendly alternative. In all the above work, the simulation model is used to study the effect of introducing new parameters or to see the feasibility of a method. The simulation models are not used with optimization model to improve the solution, obtained with mathematical models.

Our goal is to build a simple simulation model, referred as SimVRP in this thesis, with SimGrid to provide a proof of concept for a data driven simulation model to evaluate the route cost. Different data models to predict failure types and weather conditions could be integrated into our simulation model to flag for service failure or driver arriving late. This simulation model could be integrated in the optimization model to optimize service failures and improve route quality. SimGrid is a versatile network simulator extensively used in many research studies to learn the behavior of large-scale distributed systems such as Grids, Clouds and Peer to Peer systems [19]. It is scalable for all the above mentioned domains.

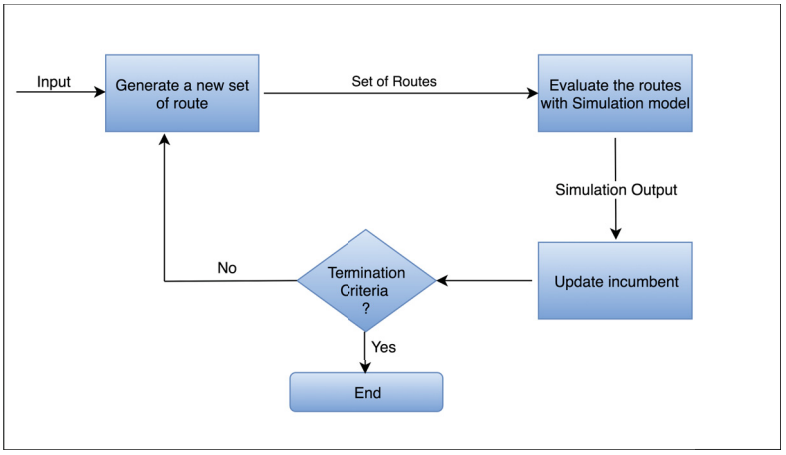


FIGURE 2.2: Optimization Process

2.6.1 Entities in SimGrid

SimGrid [19] allows users to describe the application as a set of independently communicating processes. Entities in SimGrid are agents, location, task, path, and channel.

Agent/Process: A process is defined by a code, private data and the location at which it executes.

Location: A location/host is the place in the simulated topology where an agent runs. It is defined by computational power and mailboxes that enables communication with other agents and private data that can be only accessed by agents at the same location.

Task: A task is an activity in the simulated application and defined by an amount of computing, a data size, and private data.

Path: A path is an agglomeration of communication resources representing a set of physical network links.

Channel: Communication between agents is embedded in the channel abstraction. A channel embodies the concept of communication of ports opened by agents at the location.

2.6.2 Building a simulation model with SimGrid

A SimGrid program should define the resources i.e the virtual platform, process, allocation of processes to host.

Platform: Platform defines objects like hosts, links and routing table to specify the path. A host could be a single computer system, disk, clusters [19]. These hosts are the entities where an event occurs. Links connect two different hosts and are means of communication. The links are characterized by bandwidth and latency. A platform in SimGrid is defined in an XML file.

Process: Each process in the application should be defined by describing the activities to be performed. There could be multiple processes in SimGrid executing at different hosts at the same time.

Deployment: Deployment describes the mapping of the processes to the platform. A process is mapped to the host using "MSG_function_register" when the simulation is started.

In this chapter, we discussed the problem background and solutions suggested in the literature. We discussed different machine learning algorithms that we will use to build

failure prediction data models. We also discussed the SimGrid library used to build the simulator model. In the next chapter we discuss the dataset and implementation of failure prediction models.

Chapter 3

Failure Type Prediction in Pickup and Delivery

Service failures are frequent in Pickup and Delivery and have important consequence in a carrier's business. Failures are not known when the route is planned and difficult to predict. Being able to forecast the most frequent failure types could help carriers to understand and plan routes in an effective way. The carriers could choose to make one more call to the customer and ask to stay at home or make the delivery attempt some other time during the day by changing the time window. Carriers could also decide to change the route of the services in order to better accommodate them in the route.

We study the following frequent failure types:

1. Customer not at home (NAH): This failure type happens when the customer is not at home at the time of pickup/delivery.
2. Stop rescheduled by the dispatcher (SR): When the services are rescheduled due to an unexpected event, for instance, construction in the delivery area, or inbound delays at the dispatch center.

3. Refused by the customer (RC): This failure type happens when the item was delivered to the customer's place, but the customer refused to accept it, e.g., because it did not match the expectations.

4. Canceled by the customer (CC): This failure type happens when the customer cancels the service due to unavailability of cash.

5. Not in stock (NS): The item related to service is not in stock at the local DC on the day of delivery.

In this chapter, we formalize the problem statement and discuss input data, implementation details of the failure models and association rules.

3.1 Problem Statement

The problem of Failure Type Prediction in Pickup and Delivery can be written as follows:

From service data set, for a given failure type, predict if the stop will fail with this type.

This is a binary classification problem for which we will apply Random Forest classifier (see Chapter 2). The dataset contains successful and failed services (due to various reasons). For each failure type, we predict the failure type in one against all fashion i.e., we represent one specific failure type as the target failure class and all the remaining failure types and successful services as the secondary class. In the data set for each failure model, the services failed with specific failure type are represented by 1 and other services are represented by 0. Each failure model will predict 1 if the stop fails with a specific failure type and 0 otherwise.

3.2 Input Data

Data is key for a successful implementation of any machine learning or big data model and its performance heavily relies on the quality and quantity of data. The original dataset for our study contains 523,643 customer pickup and delivery services provided

by ClearD. The data set is very imbalanced with a count of failed services equals 39,438, which is 7.53% of the data set. We discuss the measures taken to overcome class imbalance issue in details in Section 2.3.3.

Id	Feature name	Description	Type
		Customer geographical location (C)	
C1	Longitude	Longitude of the customer location	Numerical
C2	Latitude	Latitude of the customer location	Numerical
C3	Door_number	Door number in the customer address	Categorical
C4	Street_label	Street number in the customer address	Categorical
C5	Apartment_number	Apartment number in the customer address	Categorical
C6	City	City in the customer address	Categorical
C7	Province	Province of the customer location	Categorical
C8	Zip_code	Postal code in the customer address	Categorical
C9	Zone	Zone identifier of the customer location	Categorical
		Route schedule (R)	
R1	Truck	Vehicle allocated to the service	Categorical
R2	Driver	Driver identifier	Categorical
R3	Route_order	Order of the stop in the route	Numerical
R4	Planned_Service_Date_Time	Service time planned by the optimizer	Numerical
R5	Distance_from_Previous_Stop	Distance from the previous stop in the route (km)	Numerical
R6	Time_from_Previous_Stop	Time from the previous stop in the route (min)	Numerical
R7	Time_Window_Start_Time	Start time of the service time window (min since 00:00)	Numerical
R8	Time_Window_End_Time	End time of the service time window (min since 00:00)	Numerical
R9	Time_Window_Size	Service time window span (min)	Numerical
R10	Start_Slack	Time difference between pickup time and start of time window (min). May be negative when the service is performed before the time window starts.	Numerical
R11	End_Slack	Time difference between end of time window and pickup time (min). May be negative when the service is performed after the time window ends.	Numerical
		Phone calls (P)	
P1	Automatic_Calls	If automated calls are allowed (yes or no)	Categorical
P2	Call_Status	Status of phone call to customer (completed: 5, failed: 6)	Categorical
P3	Detailed_Call_Status	Detailed status of the call (answered: 2, voice mail: 3, other failures: 4-21)	Categorical
		Date and time (D)	
D1	Week_of_Year	Week number of the year (1-52)	Categorical
D2	Time_of_Day	Time in the day (early morning: 1, morning: 2, afternoon: 4, early evening: 5, evening: 6, late evening: 7, night: 8)	Categorical
D3	Day_of_Week	Day of the week (1-7)	Categorical

3.1.a. Stops

Id	Feature	Description	Type
S1	Service_Type	Service type (Pickup or Delivery)	Categorical
S2	Retailer	Retailer identifier	Categorical
S3	Item_Volume	Volume of the item to be picked up or delivered (cf)	Numerical
S4	Item_Weight	Weight of the item (lbs)	Numerical
S5	Item_Manufacturer	Item manufacturer	Categorical
S6	Estimated_Service_Time	Estimated service time (s)	Numerical

3.1.b. Services

TABLE 3.1: Features of stops and services (S)

3.2.1 Data pre-processing

Categorical Variables

Scikit-learn [28] implements Classification and Regression Tree (CART) to construct a forest model. In theory, a CART could be constructed using both numerical and

categorical values, however, categorical features must be converted into numeric features. So we encoded non-numeric categorical data and mapped them to numeric values using scikit-learn's function called LabelEncoder. This function sorts the input data and assigns a unique numeric value to each non-numeric data.

Missing Values

The Random Forest classifier does not take input data object with missing entries (NULL values). So in the pre-processing step, we replaced missing values with a constant default value of -100. We also process the data to remove duplicate rows. Normalization is not required for Random Rorest classifier as the splitting criteria based on feature values does not consider the variance in the data values.

Feature Extraction

Feature extraction is the process of extracting new informative features from the existing feature set. In the feature set of the dataset for this study, *TimeWindowPickupEndTime* and *TimeWindowPickupStartTime* contain values in the form *HH:MM:SS*. To use them in the service failure models, we convert them into numeric values. For example, $TimeWindowPickupEndTime = 8:31:42$ is converted into numeric value of 512 minutes. We also formulated some new features which are commonly used and easy to interpret in the context of services in pickup and delivery problem.

$$TimeWindowSize = TimeWindowPickupEndTime - TimeWindowPickupStartTime,$$

$$StartSlack = PickupDateTime - TimeWindowPickupStartTime,$$

$$EndSlack = TimeWindowPickupEndTime - PickupDateTime.$$

PickupDateTime is the expected date and time the customer is be visited. *TimeWindowSize* determines the size of time window provided for attended pickup and delivery (pickup and delivery which require customer to be at home). With this feature, we analyze the effect of small and large time windows on 'customer not at home' failure type.

StartSlack is time difference between expected *PickupDateTime* and *TimeWindowPickupStartTime*. *EndSlack* is the time difference between *TimeWindowPickupEndTime* and *PickupDateTime*. We also want to see the effect of large *StartSlack* and *EndSlack* on 'customer not at home' failure type.

We also extracted *Week_of_Year*, *Time_of_Day* and *Day* from *PickUpDateTime* to observe the impact of any specific time during the day on service failures. It will be interesting to see if failures are specific to a day like weekend or weekday or any specific week in the year.

Table 3.1 shows the features describing the stops and their services. Features describe the customer location, position of the stop in the route produced by the optimizer, phone call status (phone calls are placed to the customer at specific times before the service), date, and service characteristics. Some features may be correlated: for instance, features describing geographical location are strongly interdependent which decrease the importance score produced by Random Forest [15]. This becomes important while interpreting the results based on the feature importance score.

An important note on the choice of the features used in this study, is that the features are not collected explicitly to study the problem of service failure predictions. Some of the features might also seem irrelevant initially for the study of specific failure type. Our goal, as stated in Chapter 1 is to also identify the features which are leading to specific service failures. Additional features for specific failure types could be collected to further improve the performance based on the suggestion made at the end of Chapter 4.

3.2.2 Data Representation

In this section, we describe a key challenge associated with our study. The dataset contains records where each stop is associated with one or more services, represented as a vector of variable size (3 services per stop on average). This is a problem since classifiers cannot work on spaces containing vectors of variable size. A solution could be to create one record for each service and to replicate the stop features in all the

services associated with the stop. However, such a replication would likely lead the classifier to over-fit particular stops, and to rely, for instance, on the exact latitude and longitude of the stop to predict failures. This is not desirable in our case since we aim at predicting failures in situations more general than particular stop locations that may never re-occur.

Dataset Type	Dataset Size	Failure Rate (%)
Original	523,643	7.53
Aggregated	183,872	6.68

TABLE 3.2: Statistics of the Original and Aggregated Dataset

Instead, we aggregated the services of a particular stop in a master service, for which the value of categorical service features (Service_Type, Retailer, and Item_Manufacturer) was determined as the most frequent value among services in the stop, and the value of numerical features (Item_Volume, Item_Weight, and Estimated_Service_Time) was the sum of the values among services in the stop. The status and failure type of the stop were set as the most frequent failure type among its services. The resulting aggregated dataset contains 183,872 stops with 6.68% of failures. We present some important statistics of the original and aggregated dataset in Table 3.2. Indeed, the fraction of failed services is small but it is not negligible in terms of cost. Our investigated failure types account for 55.23% of the failures in the aggregated dataset. Figure 3.1 shows the failure distribution by failure type in the aggregated dataset.

3.3 Implementation and Experimental Setup

As discussed in Section 2.5 of Chapter 2, Scikit-learn is used for implementation of different failure models and Pyspark for finding Association Rules.

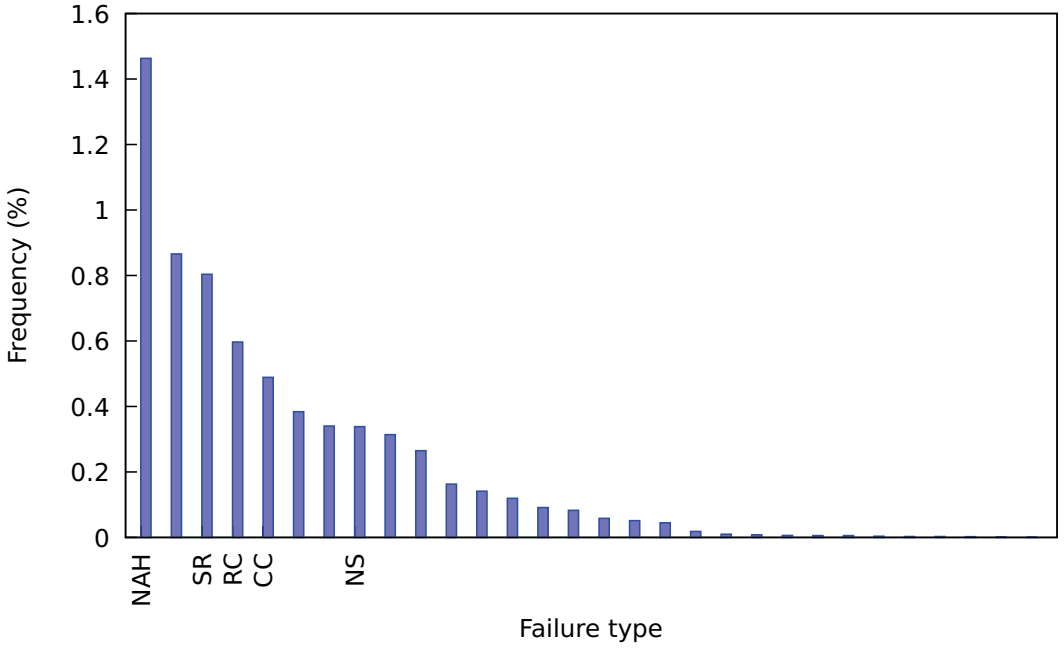


FIGURE 3.1: Distribution of failure types.

3.3.1 Test and Training Datasets

After pre-processing the dataset, we partitioned the aggregated data into test and train data sets for the classifier. The training data set is used for training the classifier and the test data set is used to validate the performance on the unseen data. The data set was randomly split between training and test sets to ensure the similar class proportion using *train_test_split()* method of scikit learn. The training set constitutes 80% of the data and the remaining 20% is used for testing. We constructed ten different training and test datasets from the aggregated dataset using a different random seed in *train_test_split()* method to ensure that the classifier performs consistently.

3.3.2 Creating a Random Forest Classifier

Creating a random forest classifier in Scikit-learn is easy. The *ensemble* package contains a class *RandomForestClassifier* which is imported in the program to create an instance of random forest classifier. To find suitable values of the classifier hyper parameters, we performed a grid search on a training dataset with different numbers of estimators, maximum depths of a tree, split criteria and minimum numbers of samples to split on a node. Table 3.3 shows the selected parameter values. We initialize the parameters

in the classifier with the best parameters obtained from the grid search. The model is trained using *fit()* method on training data set. Predictions for unseen test data is made using *predict()* method (results are discussed in Chapter 4).

Parameter	Description	Selected value	Grid search
# Estimators	Number of trees in the random forest	100	10, 50, 100, 200, 500
Max depth	Maximum depth of the trees	6	5, 6, 7, 8, 9, 10, 20, 50
Split criterion	Impurity criterion used in tree nodes	Gini	Gini, Entropy
Max features	Number of features to consider in each split	$\sqrt{n\_features}$	–
Min samples to split	Minimum number of samples required to split a node	10	–
Min sample in leaf	Minimum number of samples required in a leaf node	5	–
OOB Score	Out of bag error	True	–

TABLE 3.3: Hyper Parameter settings used in Random Forest

3.3.3 Sampling

We used three different re-sampling strategies to address class imbalance of the dataset. Re-sampling is performed on the training data set before the *fit()* method is called.

First, we over-sampled the minority class using SMOTE [8]. SMOTE generates synthetic entries in the minority class, failed stops, using random linear combinations of existing failed stops. We applied SMOTE-regular using 2 nearest neighbors, and an oversampling ratio equal to the ratio between the number of elements in the majority class and the number of elements in the minority class (so that the resulting dataset is evenly distributed among both classes).

For under-sampling, we used NearMiss-3, which, for every element in the minority class, determines its nearest neighbors and keeps only the furthest ones. We applied this method with 3 nearest neighbors.

Finally, we assessed a straightforward random undersampling of the majority class, as we suspected that SMOTE and NearMiss would be disturbed by their use of the Euclidean distance to determine nearest neighbors, which is questionable in our dataset.

3.4 Association Rules

We extracted Association Rules from the aggregated dataset, to check the consistency of the classification results and to provide further insights on their interpretation. To do so, we categorized numerical features into deciles, and we represented stops with vectors containing (1) such categorized features, (2) the initial categorical features, and (3) a binary feature representing the stop status (success or failure type). An Association Rule, written as "antecedent ⇒ consequent", consists of two tuples, an antecedent and a consequent [1]. We focus on the rules where the consequent is a singleton containing a stop status. To represent features in the antecedent, we postfix numerical features with `_Dx`, to indicate that the value is in the $\texttt{x}^{\text{th}}$ decile, and categorical features with `_Vx`, to indicate that the value is `x`. A hypothetical example of rule is:

$$\text{StartSlack\_D3}, \text{Day\_V4} \Rightarrow \text{FAIL\_NAH},$$

which measures the association between failure type "Customer not at home", and stops where Start_Slack is in the third decile and Day has value 4. It should be noted that Association Rules provide a measure of co-occurrence rather than causality.

To measure the relevance of a rule $x \Rightarrow y$, we define its *interest ratio* (IR) by comparing the frequencies of tuple x in the complete dataset C versus in the set F of failed stops (the stops that contain y):

$$\phi = \frac{\sup_F(x)}{|F|} \frac{|C|}{\sup_C(x)},$$

where $\sup_S(x)$, the support of tuple x in set S, is the number of occurrences of x in S. We focus on the cases where $\sup_F(x) \neq 0$, which gives $\phi \neq 0$. Then we define the interest ratio as follows:

$$\text{IR}(x \Rightarrow y) = \max\left\{\phi, \frac{1}{\phi}\right\}.$$

The interest ratio measures the effect of x on the probability to fail with type y. Another way to understand it is to express its relation to the failure probability conditional to the presence of x:

$$P(y|x) = \frac{\sup_F(x)}{\sup_C(x)},$$

which gives:

$$P(y|x) = \mathrm{IR}(x \Rightarrow y)P(y) \quad (\phi \geq 1),$$

or:

$$P(y|x) = \frac{P(y)}{\mathrm{IR}(x \Rightarrow y)} \quad (\phi \leq 1).$$

We also compute the *confidence* of rule $x \Rightarrow y$ with the usual definition:

$$\mathrm{conf}(x \Rightarrow y) = \frac{\mathrm{sup}_C(x \cup y)}{\mathrm{sup}_C(x)}.$$

The following relation should finally be noted:

$$\mathrm{conf}(x \Rightarrow y) = \phi \frac{|F|}{|C|}.$$

We look for rules with high interest ratio (large or small ϕ), and high frequency in the failed set ($\mathrm{sup}_F(x) \geq s$, where s is the desired support threshold in F). We find them using the following approach:

1. Find the set I of items x in F s.t. $\mathrm{sup}_F(x) \geq s$.
2. For every x in I, compute $\mathrm{sup}_C(x)$.

We perform step 1 using the FP-growth algorithm, as implemented in Apache Spark version 2.3.1. Note that finding the frequent itemsets in F requires much less memory than in C since $|F| << |C|$. We implemented step 2 using a single pass on C, which does not raise any memory issue since $|I|$ is small. To obtain a limited set of rules, we then select the Association Rules $x \Rightarrow y$ such that $x \in I$ and:

$$\begin{cases} \mathrm{IR(x \Rightarrow y)} \geq \mathrm{min\_IR} \\ \mathrm{size}(x \Rightarrow y) \leq 2 \\ \mathrm{size}(x \Rightarrow y) = 2 \Rightarrow \exists r \in R_1, r \prec_{\Delta \mathrm{IR}} (x \Rightarrow y) \end{cases}$$

where size($x \Rightarrow y$) is the size of the rule, i.e., the number of elements in x. R_i is the set of rules of size i and $\prec_{\Delta \mathrm{IR}}$ is a partial order on $R = \cup_i R_i$ defined as follows:

$$\forall r_1, r_2 \in R_1 \times R_2, r_1 = (x_1 \Rightarrow y), r_2 = (x_2, x_3 \Rightarrow y) :$$

$$r_1 \prec_{\Delta IR} r_2 \Leftrightarrow \begin{cases} \mathrm{IR}(r_2) - \mathrm{IR}(r_1) \geq \Delta IR \\ x_1 = x_2 \quad \text{or} \quad x_1 = x_3 \end{cases}$$

3.5 Conclusion

In this chapter, we discussed data pre-processing, aggregation and creating a Random Forest classifier. Data aggregation helped to create an un-biased classifier. Re-sampling the training data enabled detection of minority class instances to improve models sensitivity. Association rules are extracted to interpret the results obtained from Random Forest classier and to get more insights of the data set. We discuss the results in Chapter 4.

Chapter 4

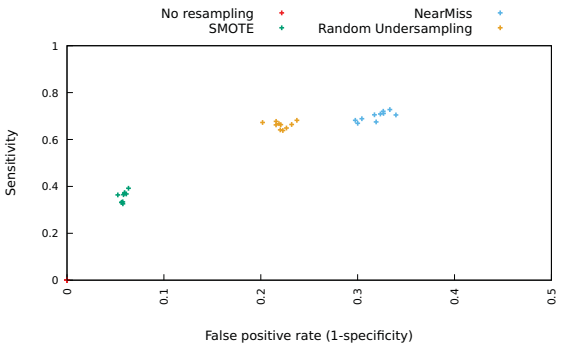
Failure Prediction Model Results

In this chapter, we discuss the results obtained on failure prediction data models. First, we present the predictive results obtained for each failure type classifier with different re-sampling methods. We then discuss important features and association rules. At the end of the chapter, we conclude the Association Rules and classification results.

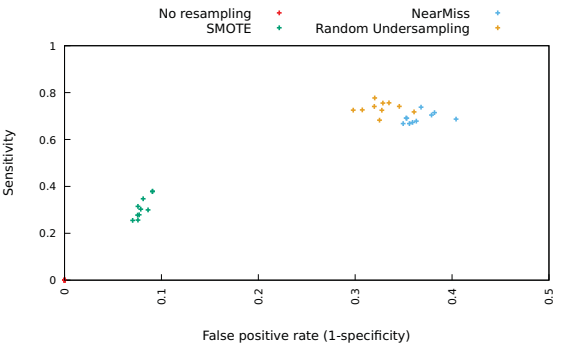
4.1 Classification Results

In this section, we present the results obtained from Random forest classifier. Figure 4.1 shows the sensitivity (ratio of true positives) and specificity (1 - ratio of false positives) obtained for the different failure types and resampling methods. Without resampling, the sensitivity to failure remains 0, as expected in such an imbalanced dataset. Oversampling with SMOTE improves the sensitivity to an average of 0.36 while maintaining a high specificity of 0.92. Undersampling with NearMiss and Random Undersampling further increases the sensitivity to an average of about 0.7, with a specificity close to 0.7. This appears to be the best compromise between sensitivity and specificity. On average, Random Undersampling performs slightly better than NearMiss. In the remainder, we focus on results obtained with Random Undersampling.

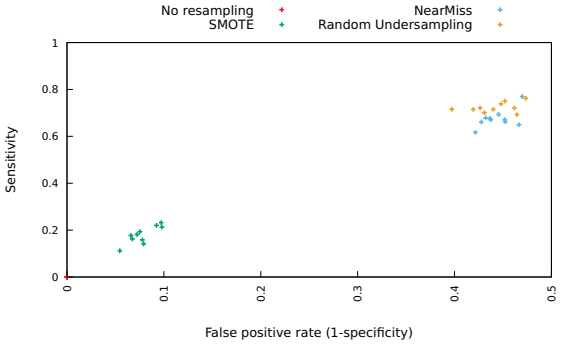
The classification performance is quite stable across failure types. With Random Undersampling, the best specificity values are obtained for NAH and NS failure type, while



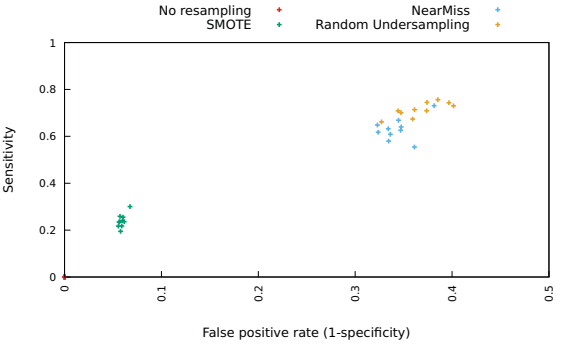
(a) Customer not at home (NAH)



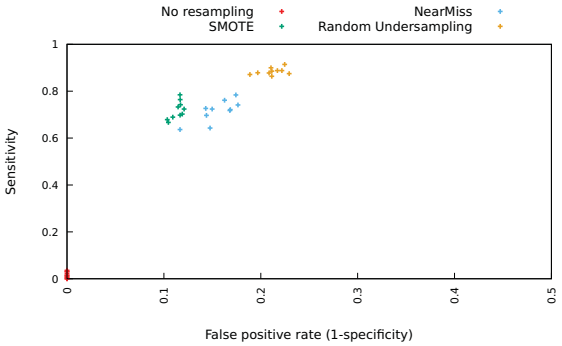
(b) Stop rescheduled (SR)



(c) Refused by customer (RC)



(d) Canceled by customer (CC)



(e) Not in Stock (NS)

	Sensitivity	Specificity
No resampling	0	1
SMOTE	0.36	0.92
NearMiss	0.68	0.67
Random Undersampling	0.74	0.69

(f) Average performance of the resampling methods

FIGURE 4.1: Performance of the classifier for different types of failures and resampling methods

RC failure type is slightly under average. Sensivitiy values are close to average for all failure types, NS being slightly above.

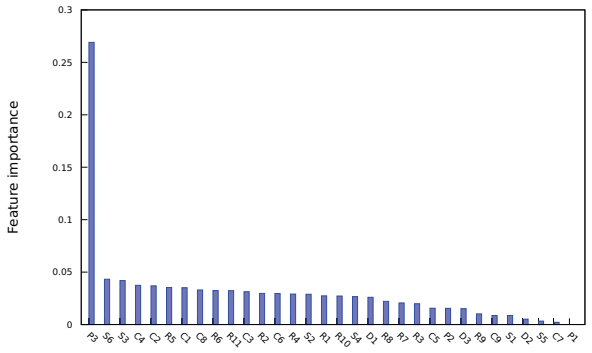
4.2 Important Features and Association Rules

In this section we discuss important feature and association rules for different failure types.

4.2.1 Customer not at home (NAH)

Figure 4.2(a) shows the feature importance resulting from the classification of failures of type "Customer not at home". Feature labels refer to the ones in Table 3.1, and feature importance is computed as explained in Section 2.4. Feature importance is largely dominated by a single feature, Detailed_Call_Status (P3), peaking at an importance of 0.27. The next 18 features have similar importance values, ranging from 0.026 to 0.05. The remaining 13 features are between 0.00 and 0.022. Figure 4.2(b) shows the antecedents of the Association Rules with consequent FAIL_NAH, selected as described in Section 3.4, with their confidence and interest ratio (IR). For clarity, the elements of rules of size 1 are omitted in rules of size 2. For instance, Rule 1 means that (Detailed_Call_Status_V3 $\Rightarrow$ FAIL_NAH) has a confidence of 0.036 and an interest ratio of 2.45, and Rule 2 means that (Detailed_Call_Status_V3, Estimated_Service_Time_D2 $\Rightarrow$ FAIL_NAH) has a confidence of 0.06 and an interest ratio of 4.07. Rules with $\phi \geq 1$ are represented in red, and rules with $\phi < 1$ are in green. For instance, Rule 1, shown in *red*, means that Detailed_Call_Status=3 *increases* failure probability by a factor of 2.45, while Rule 13, shown in *green*, means that Detailed_Call_Status=2 *decreases* failure probability by a factor of 2.06.

The rules in Figure 4.2(b) are consistent with the features importance in Figure 4.2(a), Detailed_Call_Status (P3) being the most important feature. They show that P3=3, which means that a call landed on voicemail, increases the failure probability by 2.45 times (Rule 1). This ratio increases to 3.67 if the call was marked failed (Rule 3: P2=5), to 4.07 if the estimated service time is shorter than 8 minutes (Rule 2: S6 in D2 (240-480]), or to 3.12 if the item volume is low (Rule 5: S3 in D1 [0.0, 0.002]), perhaps because less voluminous items are cheaper on average and customers give less value to them. The failure probability also increases if the item is delivered by specific companies



(a) Feature importance

Id	Rule	conf (%)	IR
1	**(P3)** Detailed_Call_Status_V3	3.6	2.45
2	**(S6)** Estimated_Service_Time_D2	6.0	4.07
3	**(P2)** Call_Status_V5	5.4	3.67
4	**(S2)** Retailer_V3	4.6	3.12
5	**(S3)** Item_Volume_D1	4.6	3.12
6	**(C9)** Zone_V22	4.0	2.77
7	**(D3)** Day_of_Week_V4	4.0	2.73
8	**(S2)** Retailer_V39	4.0	2.72
9	**(D3)** Day_of_Week_V2	4.0	2.72
10	**(R7)** Time_Window_Start_Time_D1	3.9	2.68
11	**(D3)** Day_of_Week_V3	3.9	2.65
12	**(D2)** Time_of_Day_V2	3.8	2.61
13	**(P3)** Detailed_Call_Status_V2	0.7	2.06
14	**(S4)** Item_Weight_D1	0.6	2.4
15	**(C5)** Apartment_number_D1	0.7	2.22
16	**(R9)** Time_Window_Size_D2	0.7	2.17

(b) Association Rules filtered with s=0.1, min_IR=1.4, ΔIR=0.1.

FIGURE 4.2: Customer not at home (NAH)

(Rule 4: S2=3 and Rule 8: S2=39), in the Montreal/Laval area (Rule 6: C9=22), on a Tuesday, Wednesday or Thursday (Rule 9: D3=2, Rule 11: D3=3, Rule 7: D3=4), if the time window starts between 6am and 8am (Rule 10: R7 in D1 (359.99, 480.0]), or if the service is planned between 10am and 12pm (Rule 12: D2=2).

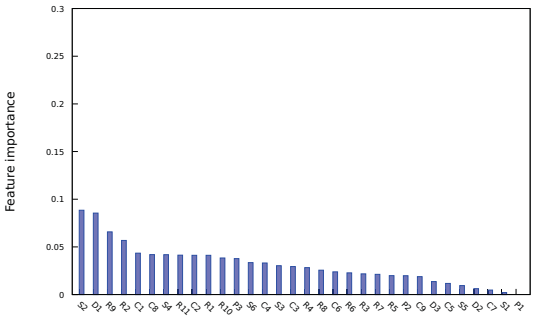
Conversely, P3=2, which means that a call was answered by a human, reduces the failure probability by 2.06 times (Rule 13). Failure probability is further reduced if the item is lighter than 2 lbs (Rule 14: S4 in D1 [0.0, 2.0]), perhaps because drivers can leave small items unattended at the customer's door when they agreed during a phone call. Finally, failure probability is also reduced if the address has no apartment number (Rule 15: C5 in D1) or if the time window provided to the customer is short (Rule 16: R9 in D2 (120.0, 180.0]).

From our analysis, we observe that confirmation call status has a huge impact on NAH failure types. Failed confirmation call along with other parameters like call type, item and time specific information increases the chances of failure.

4.2.2 Stop Rescheduled (SR)

Figure 4.3(a) shows the feature importance resulting from the classification of failures of type "Stop rescheduled". The failure importance is more uniformly distributed than for NAH. Four features stand out: S2 (Retailer), D1 (Week of Year), R9 (Time_Window_Size)

and R2 (Driver). Feature importance remains quite constant for the next 8 features, and it seems to decrease linearly to 0 for the remaining features.



(a) Stop rescheduled (SR)

Id	Rule	conf (%)	IR
1	**(S2)** Retailer_V39	1.5	1.88
2	**(P3)** Detailed_Call_Status_V3	2.5	3.16
3	**(S4)** Item_Weight_D5	1.9	2.41
4	**(D3)** Day_of_Week_V5	1.8	2.30
5	**(D2)** Time_of_Day_V4	1.7	2.07
6	**(C7)** Province_V2	1.7	2.06
7	**(C9)** Zone_V209	1.6	2.05
8	**(P2)** Call_Status_V5	1.6	2.01
9	**(R10)** Start_Slack_D9	1.4	1.80
10	**(C9)** Zone_V209	1.6	1.95
11	**(P2)** Call_Status_V5	1.6	1.94
12	**(C7)** Province_V2	1.6	1.93
13	**(S2)** Retailer_V3	0.5	1.78
14	**(R11)** End_Slack_D6	1.4	1.77
15	**(C9)** Zone_V209	1.6	1.97
16	**(C7)** Province_V2	1.6	1.95
17	**(R9)** Time_Window_Size_D3	1.4	1.77
18	**(P3)** Detailed_Call_Status_V3	2.4	3.03
19	**(S4)** Item_Weight_D5	1.9	2.38
20	**(D3)** Day_of_Week_V5	1.8	2.27
21	**(C1)** Longitude_D3	1.7	2.09
22	**(R7)** Time_Window_Start_Time_D1	1.7	2.07
23	**(R8)** Time_Window_End_Time_D2	1.7	2.05
24	**(C2)** Latitude_D3	1.5	1.92
25	**(S3)** Item_Volume_D3	1.5	1.91
26	**(P2)** Call_Status_V5	1.5	1.9
27	**(R10)** Start_Slack_D9	1.5	1.89
28	**(C2)** Latitude_D2	1.5	1.88
29	**(R9)** Time_Window_Size_D2	0.5	1.74
30	**(P3)** Detailed_Call_Status_V2	0.3	2.30
31	**(C7)** Province_V1	0.4	1.98
32	**(S4)** Item_Weight_D5	1.4	1.74
33	**(P2)** Call_Status_V5	1.5	1.84
34	**(P3)** Detailed_Call_Status_V3	1.4	1.7
35	**(P2)** Call_Status_V5	2.0	2.54
36	**(S6)** Estimated_Service_Time_D2	1.7	2.06
37	**(C9)** Zone_V209	1.6	2.01
38	**(C7)** Province_V2	1.5	1.91
39	**(D2)** Time_of_Day_V1	1.5	1.82
40	**(R10)** Start_Slack_D8	1.3	1.67
41	**(P2)** Call_Status_V5	1.4	1.79
42	**(S3)** Item_Volume_D3	1.3	1.6
43	**(S2)** Retailer_V39	1.5	1.88
44	**(R11)** End_Slack_D7	1.2	1.54
45	**(P2)** Call_Status_V5	1.3	1.68
46	**(R9)** Time_Window_Size_D3	1.3	1.67
47	**(S2)** Retailer_V149	1.2	1.51
48	**(C9)** Zone_V209	1.5	1.87
49	**(R9)** Time_Window_Size_D3	1.5	1.84
50	**(C7)** Province_V2	1.4	1.72
51	**(P2)** Call_Status_V5	1.3	1.66
52	**(S3)** Item_Volume_D2	1.2	1.48
53	**(S2)** Retailer_V39	1.4	1.69
54	**(R9)** Time_Window_Size_D3	1.3	1.64
55	**(R3)** Route_order_D10	1.1	1.43
56	**(S6)** Estimated_Service_Time_D1	1.1	1.42
57	**(S2)** Retailer_V39	1.5	1.90
58	**(R9)** Time_Window_Size_D3	1.4	1.77

(b) Association Rules filtered with s=0.1, min_IR=1.4, ΔIR=0.1.

FIGURE 4.3: Stop rescheduled (SR)

The Association Rules in Figure 4.3(b) confirm the importance of the Retailer: some companies increase the failure rate (Rule 1: S2=39, Rule 47: S2=149), and other ones reduce it (Rule 13: S2=3). The failure rate is also increased by high start slack (defined in Section 3.2.1) times (Rule 9: R10 in D9 (120.0, 129.0], Rule 40: R10 in D8 (115.0,

120.0]), and by high end slack (defined in Section 3.2.1) times (Rule 14: R11 in D6 (116.0, 120.0], Rule 44: R11 in D7 (120.0, 128.0]). The time window size also has an effect on the failure rate: D3 (180.0, 240.0] increases the failure rate (Rule 17, 46, 49, 54 and 58), while D2 (120.0, 180.0] reduces it (Rule 29). As for NAH, a call landing on voicemail (P3=3) increases the failure probability (Rule 34). Interestingly, services executed toward the end of the route tend to be rescheduled more often (Rule 55: R3 in D10 (16.0, 36.0]), and so do services with a short estimated job time (Rule 56: S6 in D1 (0.99, 240.0]). Finally, failures are also more frequent for services with a median weight (Rule 32: S4 in D5(382.0, 4,7016.0]) or for volumes lower than 18.3 cf (Rule 42: S3 in D3 (1.45, 18.3], Rule 52: S3 in D2 (0.002, 1.45]).

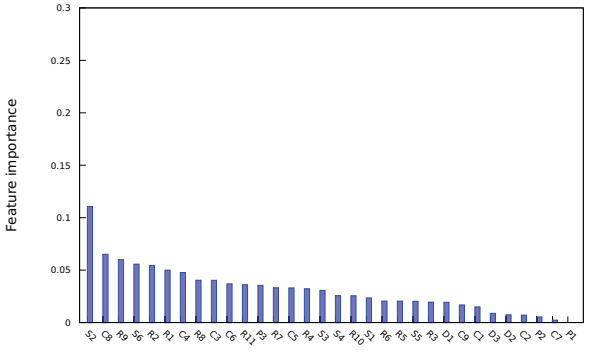
In conclusion, Stop Rescheduled (SR) failure type is more related to route specific parameters, longer start slack and time window size.

4.2.3 Refused by Customer (RC)

Figure 4.4(a) shows the feature importance resulting from the classification of failures of type "Refused by customer". Feature S2, Retailer, is standing out again. The importance seems to decrease linearly for the next features, with a slight increase for C2 (Latitude) and S4 (Item_Weight), and a slight drop between R3 and P3.

The Association Rules in Figure 4.4(b) show the effect of R9 (Time_Window_Size): when in D3 (180.0, 240.0] (Rule 1), it reduces the failure rate by a factor of 1.78, while when in D4 (240.0, 300.0] (Rule 5), it increases it by a factor of 1.74. The failure rate also increases for company 8 (Rule 7: S2=8), for an estimated service time in D3 (480.0, 720.0] (Rule 11), in the Toronto area (Rule 16: C1 in D4 (-79.469, -79.254], Rule 18: C2 in D3 (43.595, 43.737], Rule 30: C9=23), for the highest start slack times (Rule 19: R10 in D10 (129.0, 751.0]), for services scheduled between 11:39am and 12:08pm (Rule 24: R7 in D6 (700.0, 728.0]), and for voluminous items (Rule 28: S3 in D6 (49.02, 59.6]).

We conclude that Refused by Customer (RC) failure type is mostly related to route specific features and high slack time and few geographic regions.



(a) Feature importance

Id	Rule	conf (%)	IR
1	**(R9)** Time_Window_Size_D3	0.3	1.78
2	**(P3)** Detailed_Call_Status_V2	0.3	2.02
3	**(S5)** Item_Manufacturer_V-100.0	0.3	1.95
4	**(C5)** Apartment_number_D1	0.3	1.93
5	**(R9)** Time_Window_Size_D4	1.0	1.74
6	**(P2)** Call_Status_V5	1.1	1.86
7	**(S2)** Retailer_V8	1.0	1.73
8	**(S3)** Item_Volume_D6	1.3	2.24
9	**(S6)** Estimated_Service_Time_D3	1.3	2.24
10	**(P2)** Call_Status_V5	1.1	1.83
11	**(S6)** Estimated_Service_Time_D3	0.9	1.53
12	**(S3)** Item_Volume_D6	1.2	2.09
13	**(C7)** Province_V2	1.1	1.84
14	**(S4)** Item_Weight_D1	1.0	1.75
15	**(P2)** Call_Status_V5	1.0	1.66
16	**(C1)** Longitude_D4	0.9	1.53
17	**(P2)** Call_Status_V5	1.0	1.65
18	**(C2)** Latitude_D3	0.9	1.51
19	**(R10)** Start_Slack_D10	0.9	1.43
20	**(R9)** Time_Window_Size_D4	1.0	1.69
21	**(S2)** Retailer_V8	1.0	1.69
22	**(S4)** Item_Weight_D1	1.0	1.61
23	**(P2)** Call_Status_V5	0.9	1.55
24	**(R7)** Time_Window_Start_Time_D6	0.9	1.43
25	**(D2)** Time_of_Day_V4	1.0	1.68
26	**(S4)** Item_Weight_D1	0.9	1.58
27	**(P2)** Call_Status_V5	0.9	1.56
28	**(S3)** Item_Volume_D6	0.8	1.41
29	**(S4)** Item_Weight_D1	1.0	1.61
30	**(C9)** Zone_V23	0.8	1.41
31	**(P2)** Call_Status_V5	0.9	1.56
32	**(C5)** Apartment_number_D2	0.8	1.41

(b) Association Rules filtered with s=0.1, min_IR=1.4, ΔIR=0.1.

FIGURE 4.4: Refused by customer (RC)

4.2.4 Canceled by Customer (CC)

Figure 4.5(a) shows the feature importance resulting from the classification of failures of type "Canceled by customer". As in the two previous failure types, Retailer (S2) is standing out, and the importance seems to decrease linearly among the other features.

The Association Rules in Figure 4.5(b) show the importance of Time_Window_Size (R9), as in the previous failure type: services tend to fail less when R9 is in D3 (180.0, 240.0] (Rule 1), and they fail more when R9 is in D2 (120.0, 180.0] (Rule 8, 12, 24, 36, 50 and 55) or in D4 (240.0, 300.0] (Rule 19). Two particular companies also have increased failure rates (Rule 13: S2=8, Rule 25: S2=3). The failure rate is also increased in the Montreal area (Rule 5: C6=2118, Rule 5: C2 in D7 (45.452, 45.52], Rule 72: C2 in D8 (45.52, 45.619], Rule 10: C1 in D8 (-73.586, -73.289], Rule 22: C1 in D7 (-73.754, -73.586], Rule 33: C9=22), when a call lands on voicemail (Rule 45: P3=3), when the service is toward the end of the route (Rule 64: R3 in D10 (16.0, 36.0]), when the service time window starts around mid-day (Rule 69: R7 in D6 (700.0, 728.0]) or ends between

4pm and 5pm (Rule 70: R8 in D9 (960.0, 1020.0]), and when the item has a very low or close-to-average volume (Rule 51: S3 in D1 [0.0, 0.002], Rule 59: S3 in D6 (49.02, 59.6]).

In conclusion, Canceled by customer (CC) are associated with route specific feature and specific geographical regions.

4.2.5 Not in Stock (NS)

Figure 4.6(a) shows the feature importance resulting from the classification of failures of type "Not in stock". The most important feature is the week of the year (D1), followed by features related to geographical location (C2, C1, C8 and C9), features related to the route (R9, R11 and R10), the company (S2) and the volume (S3).

This is consistent with the Association Rules in Figure 4.6(b). Note that we used different filtering parameters for this failure type, due to the important number of rules with high IR. Rule 2 has an extremely high IR of 110.43, for a confidence of 37.4%: it means that company 158 had a not-in-stock failure rate of 37.4% in province 1 (Québec). Some geographical locations spanning from Gatineau to Sorel-Tracy have increased failure rates (Rule 8, 42 and 51: C1 in D7 (-73.754, -73.586], Rule 12 and 55: C1 in D6 (-75.601, -73.754], Rule 9: C2 in D8 (45.52, 45.619], Rule 14: C2 in D9 (45.619, 46.328], Rule 60: C2 in D7 (45.452, 45.52]) while other ones have decreased failure rates (Ontario, Rule 49: C7=2). Two specific weeks have increased failure rates: week 36 (Rule 19), which was the week of Labor Day in 2017, and week 44 (Rule 50), which was the week of Haloween. As for route features, time windows shorter than 2 hours (Rule 5: R9 in D1 [0.0, 120.0]), negative end slack times (Rule 32: R11 in D1 (-537.001, 62.0]), and start slack times between 53 and 63 minutes (Rule 21: R10 in D3 (53.0, 63.0]) have increased failure rates, while time windows between 2 and 3 hours (Rule 72: R9 in D3 (180.0, 240.0]) reduce the failure rate. Specific companies increase the failure rate (Rule 1: S2=158, Rule 41: S2=7) while other ones reduce it (Rule 71: S2=3).

We observe that Not in Stock (NS) failure type is specific to certain retailers which could be further investigated to address this failure type.

4.3 Conclusion and Suggested Counter Measures

In this section, we conclude classification results and discuss the performance explanation for SMOTE. We conclude feature importance and Association Rules and suggest some counter measures to prevent failures.

4.3.1 Classification Results

Random Forests showed good performance when applied to the dataset pre-processed with Random Undersampling: they reach an average sensitivity of 0.7 and an average specificity of 0.7. Thus, 70% of the failures of the studied types could be predicted, which represents 4,750 failed stops per year in the studied dataset. This prediction ability is an opportunity to save on pickup and delivery costs.

The classification performance could be further improved by (1) improving the quality of the dataset, in particular through a better definition and separation of failure types, (2) improving the dataset aggregation technique to deal with records of non-uniform sizes; our technique essentially averages the features of the services in a stop, which leads to information loss, (3) improving the strategy to deal with dataset imbalance, perhaps through a more specific oversampling method. Regarding point (3), the poor performance of SMOTE compared to the other resampling methods is illustrated in Figure 4.7. The linear combinations of services generated by SMOTE are not realistic. In particular, the generated services do not respect natural boundaries such as lakes or uninhabitated regions, not to mention roads or actual addresses. This behavior is not surprising, since no such constraints were included in the oversampling method. Similar inconsistencies are also very likely to happen in other features. On the contrary, NearMiss and Random Undersampling maintain a realistic distribution of services, at the cost of reducing the dataset. A more constrained oversampling technique might be able to address this limitation.

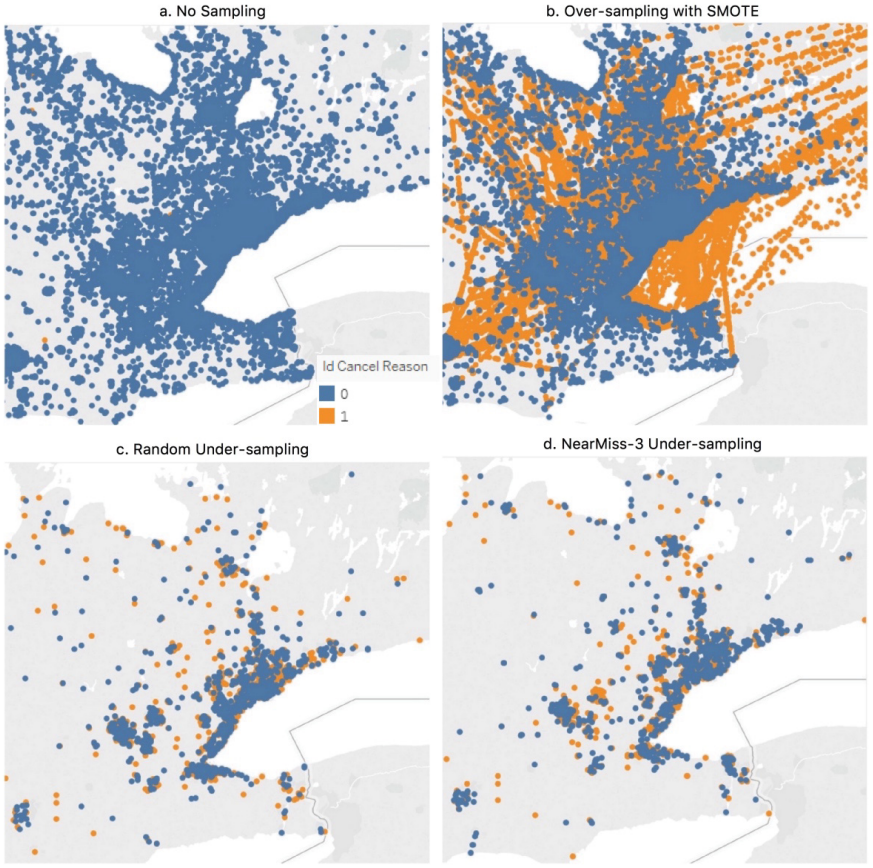


FIGURE 4.7: NAH: 2-D projection of training datasets

4.3.2 Important Features and Association Rules

Overall, we observed a good agreement between the feature importance obtained from Random Forest and the selected Association Rules. Nevertheless, most Association Rules have a low confidence value, below 5%, which shows that failures are predicted from combinations of features rather than straighforward associations. We conclude that Association Rules, computed and selected using the methods we presented, are a relevant addition to RF feature importance to provide finer-grained interpretation.

It should be noted that our extraction of Association Rules focused on rules where the antecedent occurs more than s times among the failed stops. This explains why the selection was biased towards rules with $\phi > 1$ (rules displayed in red).

4.3.3 Suggested Counter-Measures

The Retailer has a measurable effect on all the failure types. Specific investigations among the companies with failure rates higher than average should be conducted, to better understand the failure causes.

Failure of type "Customer not at home" (NAH) are very dependent on confirmation calls. In case such calls are not answered, additional ones should be scheduled, in particular if the estimated service time is short, if the item volume is low, if the item is not delivered on a Monday or Friday, if the time window starts before 8am, or if the service is planned between 10am and 12pm. In addition to these indicators, our trained Random Forest model could be used to recommend additional calls specifically for services predicted to fail. There might even be situations were multiple unanswered calls should result in the service to be removed from the route, if the specificity could be made close enough to 1.

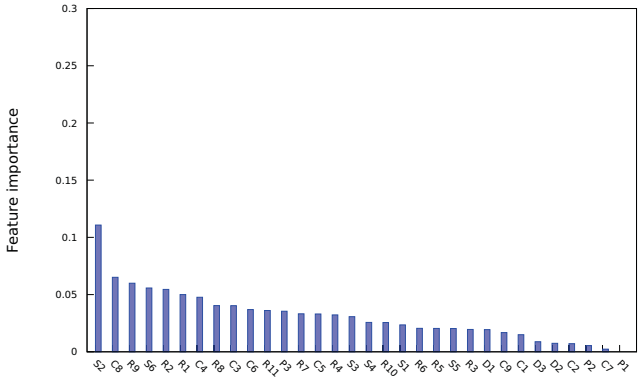
Failures of type "Stop rescheduled" (SR) are associated with many features related to the route (R3, R9, R10, R11) and a few other ones related to the type of service (S3, S4, S6). Such information could be included in optimizers, to facilitate the building of routes with less failures of this type. Start slack times longer than 2 hours lead to increased failure rates, which suggests that failures might happen due to delays in the route: dispatch centers might decide to skip stops when the service won't happen in the time window, which happens with higher probability when the start slack time is high. Likewise, services scheduled toward the end of the route are rescheduled more often than average, perhaps again due to delays in the route. The Time Window Size also has an effect on the failure rate: increased failure rates are observed for window size longer than 3 hours.

Failures of type "Refused by customer" (RC) are also associated with route-related features (R7, R9 and R10), perhaps because delays lead impatient customers to refuse items. In addition, they seem to occur more frequently at specific geographical locations (Toronto area). Again, this information could be used by optimizers to build routes with less of such failures. Such zones might also be further investigated to understand the reasons for refused items. In addition, specific items (volume in D6 and estimated

service time in D3) have an increased failure rate of this type, which might be reported to the manufacturers.

Failures of type "Canceled by customer" (CC) are also associated with route-related and geographical features (Montreal area), which could again be used by optimizers.

Finally, failures of type "Not in Stock" (NS) are strongly related to one specific retailer for which more than 10% of the services fail, and even 37% in Québec. This should be reported to the retailer and further investigated.

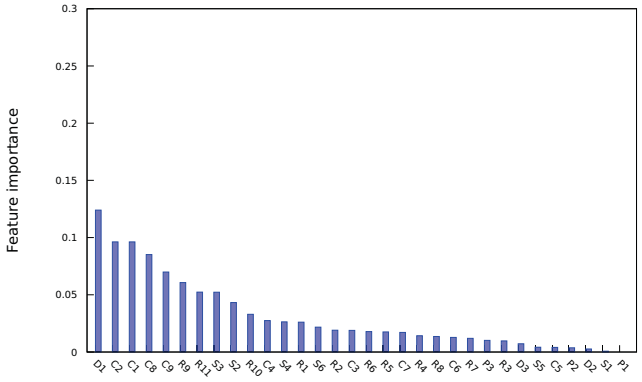


(a) Feature importance

Id	Rule	conf (%)	IR
1	**(R9)** Time_Window_Size_D3	0.2	2.06
2	**(S5)** Item_Manufacturer_V-100.0	0.2	2.41
3	**(C5)** Apartment_number_D1	0.2	2.38
4	**(C7)** Province_V2	0.2	2.18
5	**(C6)** City_V2118	1.0	2.00
6	**(C2)** Latitude_D7	1.0	2.00
7	**(S2)** Retailer_V3	1.5	3.14
8	**(R9)** Time_Window_Size_D2	1.3	2.62
9	**(C9)** Zone_V22	1.1	2.21
10	**(C1)** Longitude_D8	0.9	1.86
11	**(S2)** Retailer_V3	1.3	2.66
12	**(R9)** Time_Window_Size_D2	1.1	2.34
13	**(S2)** Retailer_V8	0.9	1.80
14	**(S3)** Item_Volume_D6	1.1	2.25
15	**(C1)** Longitude_D4	1.1	2.24
16	**(S6)** Estimated_Service_Time_D3	1.1	2.23
17	**(D2)** Time_of_Day_V4	1.0	2.12
18	**(R11)** End_Slack_D10	1.0	1.98
19	**(R9)** Time_Window_Size_D4	0.9	1.79
20	**(D2)** Time_of_Day_V4	1.1	2.18
21	**(R11)** End_Slack_D10	0.9	1.91
22	**(C1)** Longitude_D7	0.9	1.77
23	**(S2)** Retailer_V3	1.4	2.81
24	**(R9)** Time_Window_Size_D2	1.3	2.56
25	**(S2)** Retailer_V3	0.9	1.74
26	**(S4)** Item_Weight_D1	1.3	2.68
27	**(P3)** Detailed_Call_Status_V3	1.3	2.57
28	**(C9)** Zone_V22	1.2	2.50
29	**(C7)** Province_V1	1.0	1.97
30	**(R11)** End_Slack_D4	0.9	1.86
31	**(D2)** Time_of_Day_V1	0.9	1.85
32	**(C5)** Apartment_number_D3	0.8	1.74
33	**(C9)** Zone_V22	0.8	1.60
34	**(P3)** Detailed_Call_Status_V3	1.4	2.90
35	**(S3)** Item_Volume_D1	1.1	2.34
36	**(R9)** Time_Window_Size_D2	1.1	2.30
37	**(D3)** Day_of_Week_V2	1.0	2.05
38	**(C6)** City_V2118	1.0	2.02
39	**(C1)** Longitude_D8	0.9	1.92
40	**(D3)** Day_of_Week_V4	0.9	1.90
41	**(S4)** Item_Weight_D1	0.9	1.86
42	**(C1)** Longitude_D7	0.9	1.78
43	**(S6)** Estimated_Service_Time_D2	0.9	1.76
44	**(D2)** Time_of_Day_V4	0.8	1.74
45	**(P3)** Detailed_Call_Status_V3	0.8	1.57
46	**(S3)** Item_Volume_D1	1.2	2.38
47	**(P2)** Call_Status_V5	1.1	2.35
48	**(S6)** Estimated_Service_Time_D2	1.1	2.27
49	**(C7)** Province_V1	1.0	1.94
50	**(R9)** Time_Window_Size_D2	0.9	1.81
51	**(S3)** Item_Volume_D1	0.8	1.56
52	**(C7)** Province_V1	0.9	1.94
53	**(R11)** End_Slack_D4	0.9	1.84
54	**(S2)** Retailer_V3	0.8	1.74
55	**(R9)** Time_Window_Size_D2	0.8	1.71
56	**(D2)** Time_of_Day_V1	0.8	1.69
57	**(S4)** Item_Weight_D1	0.8	1.69
58	**(C5)** Apartment_number_D2	0.7	1.48
59	**(S3)** Item_Volume_D6	0.7	1.48
60	**(S6)** Estimated_Service_Time_D3	1.1	2.18
61	**(C9)** Zone_V209	1.0	2.08
62	**(C7)** Province_V2	1.0	1.96
63	**(S4)** Item_Weight_D1	0.8	1.65
64	**(R3)** Route_order_D10	0.7	1.47
65	**(S6)** Estimated_Service_Time_D3	0.7	1.47
66	**(C7)** Province_V2	0.8	1.71
67	**(S5)** Item_Manufacturer_V-100.0	0.8	1.66
68	**(S4)** Item_Weight_D1	0.8	1.63
69	**(R7)** Time_Window_Start_Time_D6	0.7	1.47
70	**(R8)** Time_Window_End_Time_D9	0.7	1.46
71	**(S4)** Item_Weight_D1	0.8	1.60
72	**(C2)** Latitude_D8	0.7	1.42
73	**(C1)** Longitude_D4	0.7	1.40

(b) Association Rules filtered with s=0.1, min_IR=1.4, ΔIR=0.1.

FIGURE 4.5: Canceled by customer (CC)



(a) Feature importance

Id	Rule	conf (%)	IR
1	**(S2)** Retailer_V158	10.4	30.81
2	**(C7)** Province_V1	37.4	110.43
3	**(P2)** Call_Status_V5	11.0	32.40
4	**(S4)** Item_Weight_D1	10.9	32.26
5	**(R9)** Time_Window_Size_D1	1.5	4.40
6	**(S2)** Retailer_V158	10.4	30.81
7	**(S4)** Item_Weight_D1	5.9	17.48
8	**(C1)** Longitude_D7	4.9	14.52
9	**(C2)** Latitude_D8	4.2	12.30
10	**(C9)** Zone_V22	3.5	10.29
11	**(C3)** Door_number_D1	3.4	10.07
12	**(C1)** Longitude_D6	3.3	9.73
13	**(S6)** Estimated_Service_Time_D3	3.2	9.32
14	**(C2)** Latitude_D9	3.1	9.19
15	**(C7)** Province_V1	3.0	8.87
16	**(D2)** Time_of_Day_V4	1.9	5.75
17	**(S6)** Estimated_Service_Time_D1	1.7	5.10
18	**(D3)** Day_of_Week_V6	1.7	4.94
19	**(D1)** Week_of_Year_V36	1.2	3.50
20	**(C7)** Province_V1	1.9	5.59
21	**(R10)** Start_Slack_D3	1.2	3.49
22	**(C9)** Zone_V22	2.9	8.45
23	**(C3)** Door_number_D1	2.8	8.22
24	**(S6)** Estimated_Service_Time_D3	2.6	7.75
25	**(C7)** Province_V1	2.4	7.18
26	**(D2)** Time_of_Day_V4	1.8	5.33
27	**(S6)** Estimated_Service_Time_D1	1.6	4.76
28	**(S4)** Item_Weight_D4	1.6	4.62
29	**(R9)** Time_Window_Size_D1	1.5	4.40
30	**(R11)** End_Slack_D1	1.5	4.35
31	**(D3)** Day_of_Week_V6	1.5	4.29
32	**(R11)** End_Slack_D1	1.1	3.35
33	**(C9)** Zone_V22	2.5	7.38
34	**(C7)** Province_V1	2.3	6.83
35	**(S4)** Item_Weight_D4	1.5	4.51
36	**(D2)** Time_of_Day_V4	1.5	4.47
37	**(S6)** Estimated_Service_Time_D1	1.5	4.40
38	**(S4)** Item_Weight_D1	1.4	4.23
39	**(R9)** Time_Window_Size_D1	1.4	4.19
40	**(D3)** Day_of_Week_V6	1.3	3.95
41	**(S2)** Retailer_V7	1.1	3.35
42	**(C1)** Longitude_D7	4.3	12.67
43	**(C9)** Zone_V22	2.8	8.14
44	**(C7)** Province_V1	2.3	6.81
45	**(S6)** Estimated_Service_Time_D1	1.7	5.10
46	**(D2)** Time_of_Day_V4	1.6	4.65
47	**(S4)** Item_Weight_D4	1.6	4.62
48	**(D3)** Day_of_Week_V6	1.3	3.97
49	**(C7)** Province_V2	0.1	2.80
50	**(D1)** Week_of_Year_V44	0.9	2.70
51	**(C1)** Longitude_D7	0.7	2.11
52	**(C9)** Zone_V23	0.7	2.07
53	**(S4)** Item_Weight_D1	1.1	3.26
54	**(C6)** City_V2118	0.7	2.06
55	**(C1)** Longitude_D6	0.7	1.96
56	**(S6)** Estimated_Service_Time_D3	0.7	1.96
57	**(C3)** Door_number_D1	2.8	8.35
58	**(C9)** Zone_V22	1.7	5.14
59	**(C7)** Province_V1	1.4	4.03
60	**(C2)** Latitude_D7	0.7	1.95
61	**(C9)** Zone_V22	0.7	1.94
62	**(C3)** Door_number_D1	4.9	14.54
63	**(S3)** Item_Volume_D4	1.5	4.52
64	**(R8)** Time_Window_End_Time_D1	1.1	3.38
65	**(S4)** Item_Weight_D4	1.1	3.23
66	**(S6)** Estimated_Service_Time_D1	1.0	3.06
67	**(D3)** Day_of_Week_V2	0.9	2.65
68	**(S3)** Item_Volume_D2	0.9	2.62
69	**(R11)** End_Slack_D2	0.8	2.51
70	**(P3)** Detailed_Call_Status_V3	0.8	2.46
71	**(S2)** Retailer_V3	0.2	1.93
72	**(R9)** Time_Window_Size_D3	0.2	1.93

(b) Association Rules filtered with s=0.1, min_IR=1.9, ΔIR=0.5.

FIGURE 4.6: Not in stock (NS)

Chapter 5

Simulation Model

5.1 Introduction on Simulation

The goal of SimVRP is to simulate the routes generated by the optimization model and it is used as a cost function to evaluate the quality of the routes. A route is evaluated based on time and distance taken by a vehicle and the total number of service failures into the route. SimVRP finds the total traveled distance and time for a route and predicts the number of failed services with different failure reasons. The SimVRP prototype, we developed as a part of this thesis, simulates service failures at random. In Chapter 2, we discussed different components needed to build a simulation model in SimGrid. In this Chapter, we discuss the details of the different components in SimVRP like platform, processes and deployment and how they model the route generated by the optimization model.

5.2 Model Description

A simulation model in SimGrid is defined by a platform, a set of processes, and a deployment of the processes on the platform (Chapter 2). In SimVRP, platform models the road network. Processes describe the events in a route like loading the items in the truck, delivering to a customer place and returning back to depot. The deployment file

describes the mapping of processes to hosts in the platform. Figure 5.1 describes the work flow in simulation model.

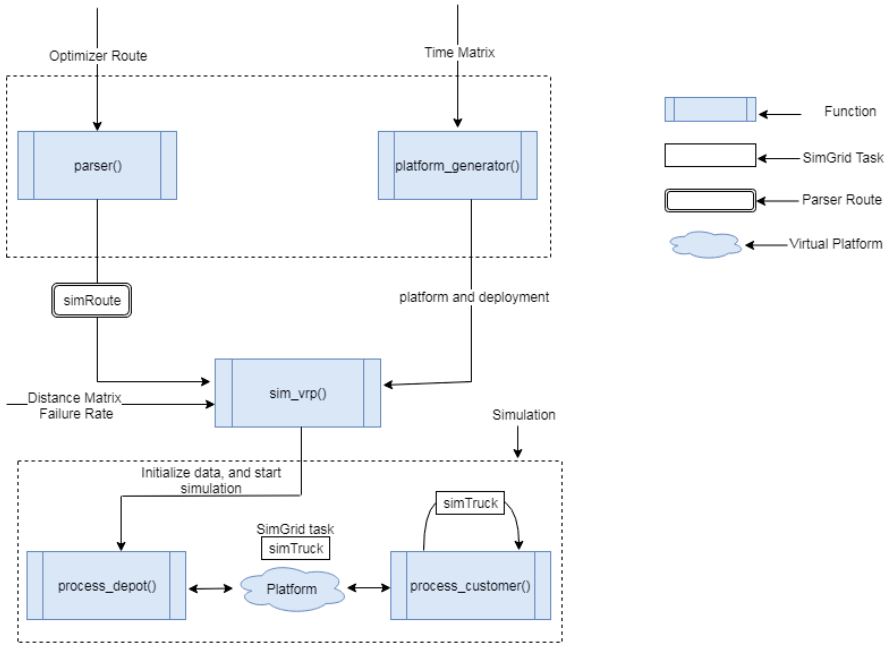


FIGURE 5.1: SimVRP Workflow

In the rest of the chapter, we describe how the road network is modeled by SimGrid platform, different data structures for modeling objects in the road network and processes to be executed on different hosts in the platform.

5.2.1 Platform Model

SimVRP models road network as a computer network and represents different components of road network in the last mile like trucks, local DC or depot(s), end customers and the path connecting them as objects of a computer network. These objects are defined in the SimVRP platform. Depot(s), stores and customers are modeled as hosts in the platform. A host is characterized by a unique name, processing speed and number of cores. The path connecting two physical locations (i.e., Customer-Customer, Depot-Customer or Customer-Depot) is represented by a network link. Each link in the platform is defined by a source and destination host, bandwidth and latency. A truck is modeled as SimGrid task which contains a route representation. A task in SimGrid has

four components as shown in the Figure 5.2.

FIGURE 5.2: SimGrid Task

Host name is the name of the host in the platform represented by a unique host_id. *Processing amount* represents computational size (flops) required to process the task at a host. *Message size* is the task size in bytes. *Data* is user information which can be retrieved at the host. In current version of SimVRP, we do not use *processing amount* and *message size* field in the SimGrid task. So we set them to zero. *User data* contains route representation. SimVRP processes primarily work on *User data* and exchange information contained in this part.

The platform in SimVRP is generated from a time matrix. The time matrix is a square matrix of real values which denotes the smallest amount of time needed to travel between two locations. The travel time is constant and does not consider the traffic and weather conditions. The number of hosts is the number of rows in the matrix and there exists a link between all hosts in the platform. The latency on the links is defined from the travel time between two stops in the time matrix. We assign each link a constant bandwidth of 1bps as there is no capacity constraint on the road and each communication task size is 0 bytes.

In SimVRP, each host has a constant processing speed of 1flop. This is primarily because, we do not execute SimGrid tasks and stops (i.e., depot, customer and stores) are represented by similar hosts. In future version of SimVRP, depot and stores could be separated from individual customer by assigning different processing speed. The platform is generated once before the simulation is started and remains unchanged throughout the simulation.

5.2.2 Processes

There are two main processes in SimVRP, representing the depot and the customer stops. As we discussed in Section 5.2.1, these processes exchange SimGrid tasks containing a route representation. Route-specific information like customer location, different item to be served at a stop, item-specific information like weight and volume, information of the vehicle like maximum capacity, current remaining capacity, driver information are represented in *simTruck*. A *simTruck* is composed of many data structures shown in Figure 5.3 and encapsulated in the *User data* part of SimGrid task. Let's first introduce the data structures used to build *simTruck* and how they are initialized.

Simulated Entities

simItem: This data structure represents items in the last mile and describes item-specific attributes like *size*, *weight* and *volume* and a unique *good_id*. These attributes are initialized with attributes of the item and used to ensure that the sum of the volume of items never exceeds the maximum capacity of *simTruck*. In the last mile, the driver must keep track of the items present in the vehicle for efficiently performing the services. To achieve this, *simTruck* maintains a Boolean variable *in_Truck*, for each item, to determine if the item is available in Truck. *in_Truck* is set to 1 if the item is present and 0 otherwise.

simService: This data structure represents a pickup or delivery service. It has a unique *service_id* and a *simItem* associated to it. *type* describes the service type (Pickup/Delivery). Each service in simVRP has a *predecessor* and *successor* that represents services which must be performed before and after the current service respectively. A service can have three different status: Pending, Completed or Failed which is described by *status* attribute. By default, *status* is set to *Pending* during initialization and changes during simulation. Failed services in SimVRP are tracked with *status_reason*, set to 'undefined' during the initialization process and later updated if the service fails. In the current version of SimVRP, we introduce 5 different service failure types at random which in future could be extended to include predictive data models for a wider range of failure type as developed in Chapter 3.

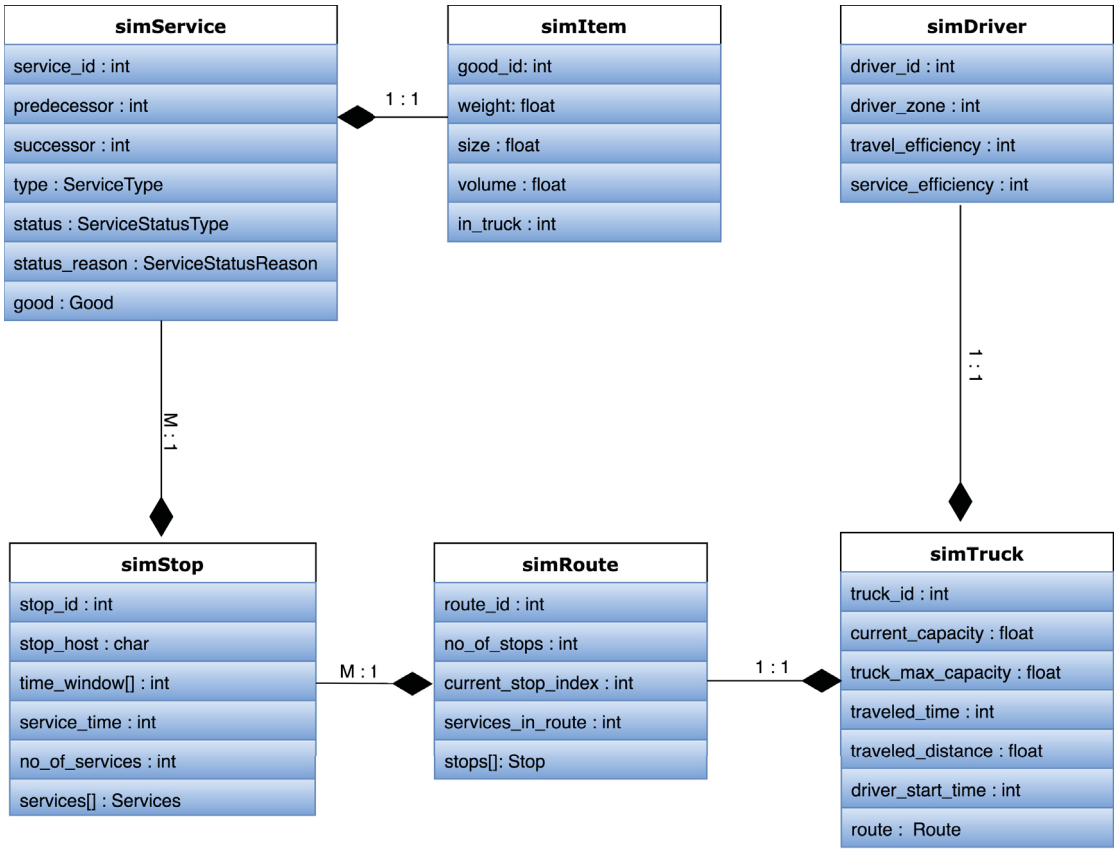


FIGURE 5.3: Data Structure in SimVRP

simStop: This data structure represents an actual physical location like customer address, store or depot and modeled as host in the platform file. Each stop has unique *stop_id* and name *stop_host*. Each *simStop* contains a list of *simService* to be performed on this stop. It has a time window for service processing and this is determined with *time_window* attribute which stores start and end of the time window in seconds. A service is not be processed outside of the service time window. Processing time for each service in seconds is stored in *service_time* attribute.

simRoute: This data structure represents a route in the last mile and is defined by a unique *route_id*. It maintains a list of *simStop* to be served in the route. It also maintains the total number of *simStops* and *simServices* for each route. So a *simRoute* is a collection of *simStop* in the order to be performed in the route.

simDriver: This data structure represents a driver in the last mile. Each driver has a unique Id and is characterized by *travel_efficiency* and *service_efficiency*.

simTruck: This data structure represents a real truck in the road network. It maintains a *unique_id* and encapsulates a *simRoute* and a *simDriver*. In the current version of SimVRP, the driver is just associated with each simTruck and its profile is not taken into account. The attributes *truck_max_capacity* and *current_capacity* determines the maximum and current capacity of the truck.

Route Initialization

From the real routes (generated by the route planning software i.e., optimizer of ClearD), different data structures are created and initialized as discussed above. This modification and initialization happen inside a parser. This function is not a SimVRP process and its task is to create and initialize *simTruck* before the actual simulation process. It accepts a set of routes generated by the optimizer and transforms each route into a *simTruck* by creating instances of the different data structure shown in Figure 5.3. For each route generated by the optimizer, the parser function creates a *simTruck*, *simRoute* and *simDriver*. *simStop* is initialized by the total number of stops in an optimizer route. *simService* data structure, for each stops, holds service to be performed at a stop in the route.

Interaction between two process in SimVRP (*process_depot* and *process_customer*) is shown in Figure 5.4. *process_depot* initiates communication in SimVRP. It creates a SimGrid task with *simTruck* and sends it to the 1st *simStop* in *SimRoute*. *process_customer* receives a SimGrid task from *process_customer* and itself. It processes the services and send the tasks to next *simStop* in the *simRoute*. This process continues and the last stop send the task back to *depot*.

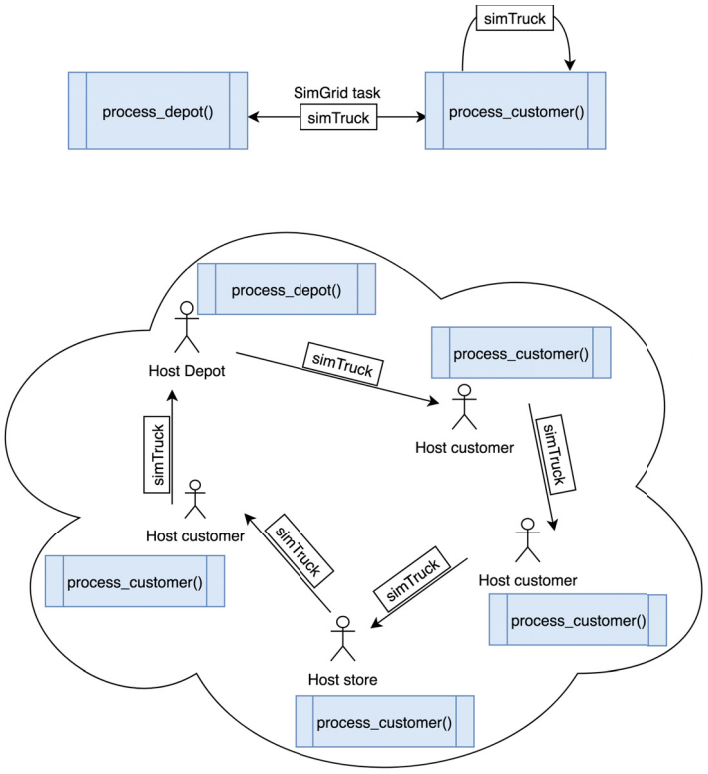


FIGURE 5.4: process_depot and process_customer interaction in SimVRP for a *simRoute*

process_depot: This process is deployed at the hosts representing *depot* in SimVRP platform as shown in Figure 5.4. The main steps in process_depot are explained in Algorithm 3. It creates a SimGrid task (data type msg_task_t) and assigns task's data with *simTruck*. This process is suspended temporarily by *process_sleep()* to keep driver's start time into account when the simulation is started. The task is sent to the mailbox of the host (destination host) representing 1st stop in *simRoute*. Before the task is sent, simulation timestamp is recorded. This process also receives a task sent by **process_customer**. On receiving a task, it updates the traveled time and traveled distance in *simTruck*. *traveled_time* is recorded by reading the difference in simulation time before the task was sent and after the task has been received. The route time is computed from the difference of these two times. *process_depot* is implemented to model behavior of a depot in the last mile where in practice multiple vehicles start their route. It can send multiple tasks (which encapsulates *simTruck*) to different hosts at the same time.

process_customer: This process is deployed at the hosts representing *customer/store*

in SimVRP platform as shown in Figure 5.4. It describes core functionalities in SimVRP. The main steps in this process are summarized in Algorithm 4.

Algorithm 3 process_depot

Input- *simTruck*: Parser generated data structure containing a route representation; *driver_start_time*: Start time for the driver; Distance_Matrix[][]: A two dimentional Distance Matrix.

1: process_sleep(driver_start_time) ▷ Wait until the start time of the driver.
2: $simulation\_start\_time \leftarrow get\_clock()$ ▷ Record the simulation time stamp.
3: $simStop \leftarrow 1st\ stop\ in\ simTruck$
4: $host\_id \leftarrow simStop\ name$
5: $task\ task\_s = task\_create(host\_id, 0, 0, simTruck)$ ▷ Create a task_t with the 1st stop in the parser_route.
6: $task\_send(task\_s, host\_id)$ ▷ Send this task to the stop_id.
7: $rec\_simTruck \leftarrow receive\_task()$ ▷ Recieve the truck sent back to the depot.
8: $simulation\_end\_time = get\_clock()$ ▷ Record tasks arrival time.
9: $traveled\_time \leftarrow simulation\_end\_time - simulation\_start\_time$ ▷ Compute simulation time.
10: $UpdateDistance(simTruck, Distance\_Matrix)$ ▷ Update the distance.
11: Stop simulation.

Receiving a Task: process_customer receives a SimGrid task sent by process_depot in its mailbox by reading the MSG incoming queue. It can receive only one SimGrid task at a time. *simTruck* is extracted from the data part of the received task and the task is destroyed. This state of SimVRP relates to a state in the last mile where a vehicle arrives at the customer location to perform a pickup/delivery service.

Updating travelled distance: Next step in process_customer is to update the traveled distance in *simTruck*. *travel_distance* attribute of *simTruck* is updated to consider the distance travelled between sending host and receiving host. This distance is taken from distance matrix for the associated hosts i.e., the sending host and receiving host as shown in Algorithm 5.

Algorithm 4 process_customer

Input- $failure\_rate$: Service failure rate; $Distance\_Matrix[][]$: A two dimensional distance matrix.

1: $task\_receive(\&task, host\_get\_name(host\_self()))$ ▷ Receive a task.
2: $rec\_simTruck \leftarrow task\_get\_data(task)$ ▷ Extract $simTruck$ from the task.
3: $UpdateDistance(simTruck, Distance\_Matrix)$ ▷ Update the distance.
4: $current\_stop \leftarrow Get\ current\ simStop\ from\ simTruck$ ▷ Get the current stop in the route
5: $now \leftarrow get\_clock()$ ▷ Record task arrival time.
6: $end\_time\_window \leftarrow End\ time\ window\ of\ current\_stop$
7: **if** $end\_time\_window < now$ **then** ▷ Driver arrived late, do not process the stop.
8: $simServices \leftarrow Services\ in\ current\_stop$
9: **for** $service\ in\ simService$ **do**
10: $service\_status \leftarrow Failed$
11: $status\_reason \leftarrow Driver\_late$
12: **end for**
13: **else** ▷ Check if the driver arrived early.
14: $start\_time\_window \leftarrow Start\ time\ window\ of\ the\ current\_stop$
15: $slack \leftarrow start\_time\_window - now$
16: **if** $slack > 0$ **then**
17: $process\_sleep(slack\_time)$ ▷ Wait until start of time window
18: **end if**
19: $process\_service(simTruck, failure\_rate)$ ▷ Process the services at this stop.
20: **end if**
21: Send $simTruck$ to the next stop.

Perform Services: There are two type of services in SimVRP: Delivery and Pickup. On receiving a task, process_customer determine if the task is received with in the service time window of the current *simStop* by looking at the stop specific information like time window from *simStop*. The process extracts 'end time window' from the *simStop* and compares it with the actual arrival time in the simulation. If the task has arrived before the 'end time window' of the *simStop*, it is allowed to process the services (defined in

simService) at the host otherwise, the no service is processed and all the services for the *simStops* are marked fail. This is however very strict constraint as in practice the drivers still make an attempt to perform the services when they are little late. We summarize the step to process the services at a stop in Algorithm 6.

Algorithm 5 UpdateDistance

Input- *simTruck*: Parser generated data structure containing a route representation; Distance_Matrix[][]: A two dimensional Distance matrix.

1: $simRoute \leftarrow Route\ from\ simTruck$ ▷ Get route representation from simTruck.

2: $current\_stop \leftarrow Get\ current\ simStop\ from\ simRoute$ ▷ Get the current stop in the route

3: $sender\_stop \leftarrow Get\ stop\ associated\ to\ sending\ host\ from\ simRoute$

4: $distance \leftarrow Distance\_Matrix[sender\_stop][current\_stop]$ ▷ Get distance between sender and receiver stop from Distance Matrix.

5: $traveled\_distance \leftarrow traveled\_distance + distance$ ▷ Update distance.

The current version of SimVRP simulates 2% service failures randomly for *bad_address*, *item_quality* and *customer_not_at_home* before actually processing any service at a host. Failure rate is passed to SimVRP at execution time which could also be made as an argument in the deployment file.

process_service determines service kind from the *type* attribute of *simService* data structure. If the service is a delivery service, the process looks for the associated item in the *simTruck* by checking *in_truck* variable. It delivers the item to the customer if it is available. In case of a pickup service, the process examines *current_capacity* parameter of *simTruck* to determine if there is enough space available to load the item. The item is successfully loaded into *simTruck* if difference between *maximum_capacity* and *current_capacity* is more than the capacity of the item. If item capacity exceeds available *simTruck* capacity, the item is not picked up at the host.

Algorithm 6 process_service

Input: *simTruck*- Data structure containing a route representation; failure_rate: Service failure rate.

1: $simStop \leftarrow Current\ stop\ in\ simTruck$ ▷ Get the current processing customer stop.
2: $simServices \leftarrow Service\ to\ be\ performed\ at\ the\ simStop$
3: **if** $rand() < RAND\_MAX * failure\_rate$ **then**
4: Introduce failure. ▷ Integrate service failure prediction models here.
5: **else**
6: **for** simService in simServices **do**
7: $simItem \leftarrow item\ associated\ with\ the\ simService$
8: **if** simService is Delivery type **then**
9: **if** *simItem* is in simTruck **then** ▷ Item is in the truck, perform service.
10: $service\_status \leftarrow completed$
11: $simItem \leftarrow Not\ in\ truck$
12: **else** ▷ Item in not in the truck.
13: $service\_status \leftarrow failed$
14: $service\_status\_reason \leftarrow item\_not\_in\_truck$
15: **end if**
16: **else** ▷ Service is a Pickup type.
17: $truck\_current\_capacity \leftarrow truck\_current\_capacity + simItem\_volume$
18: **if** $truck\_max\_capacity \leq truck\_current\_capacity$ **then** ▷ Check if space available in truck.
19: $truck\_current\_capacity \leftarrow truck\_current\_capacity + simItem\_volume$
20: $service\_status \leftarrow completed$
21: $simItem \leftarrow In\ truck$
22: **else** ▷ No space in the truck.
23: $service\_status \leftarrow failed$
24: $service\_status\_reason \leftarrow truck\_capacity\_issue$
25: **end if**
26: **end if**
27: **end for**
28: $process\_sleep(simStop's\ servicetime)$
29: **end if**

In the next step, the service status is updated in *simService*. The service is marked 'successful' or 'failed' with appropriate failure reason depending on the previous step. If the service is successful, current_capacity in *simTruck* is increased/decreased accordingly by the weight of the item picked/delivered. However current_capacity remains same if the status is marked as Failed.

Send simTruck to next Host: Once all the services on a stop is completed, the process creates a new SimGrid task, assigns name and initializes data with the updated *simTruck*. This task is then sent to the host representing next stop in *simRoute*.

5.2.3 Deployment and Execution

The deployment file maps processes to hosts defined in the platform. in SimVRP, *process_customer* and *process_depot* are mapped to the host representing customers/stores and depots respectively. We can pass the input arguments like driver start time and service failure rate through the deployment file.

sim_vrp: This function is the entry point in SimVRP. It initializes the MSG internal data and creates a virtual platform as described in SimVRP platform. *sim_vrp* register the processes to the hosts as defined in the deployment file. It accepts *simTruck* data structure from *parser* as shown in the Figure 5.1. The registered processes are launched from the deployment file and simulation is started.

5.3 Conclusion

In this chapter, we discussed SimVRP implementation with SimGrid. We defined various data structures used to represent different objects in routes and how the road network is represented by the platform. The role of the driver at the depot and customer place is defined by processes and the hosts at which the processes are mapped. The results of SimVRP validation are presented in Chapter 6.

Chapter 6

SimVRP Validation

In this chapter, we present SimVRP validation results on sample routes.

6.1 Experimental Setup

We simulates six sample routes from the prototype route as shown in Figure 6.1, with 5 stops. Each stop (except depot) as a time window interval [a, b] expressed in minutes after midnight and a service time expressed in minutes inside the circle. There is a tuple (x, y) on each link which is the travelling distance (km) and time (minutes) between stop x and y. Stop 2 and Stop 3 have two services and all other stops have one service. All trucks are of the uniform maximum capacity of 500lbs and every driver starts at 7 AM. There is a total of 7 services in any route. The travelled distance for each route is 215 km. The travel time is different based on the number of failed services. In each route, we introduce the following constraints.

1. Route 1 is a straight forward representation of the prototype route shown in Figure 6.1.

2. Route 2 with late driver: We change the time window for Stop 4 to [630, 730] in the route shown in Figure 6.1. The driver arrives at this stop at 778.

3. Route 3 with precedence constraints: Stop 2 has two pickup services and it is predecessor of Stop 3. The item which are picked up at Stop 2 are to be delivered at Stop 3.

4. Route 4 with truck capacity constraints: At Stop 3, we assign two pickup services each with capacity higher than truck's maximum capacity.

5. Route 5 with different travel time:

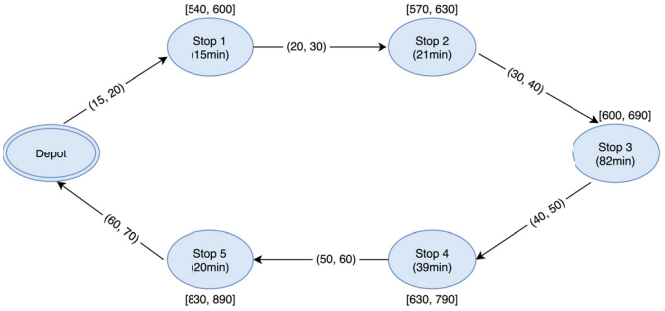


FIGURE 6.1: Sample Route

6.2 Simulation Results

Simulation results are shown in Table 6.2. We simulated six routes and the simulated results are same as the real result.

Route 1 is a simple route where all the stops are successfully completed. In Route 2, the driver arrives late at stop 4 at 778, after the end time window (730) for the stop. So this stop is not processed and the service associated is marked failed with failure reason 'Driver_late' as shown in Table 6.2(b). This affect the total travel time for the route as the processing time for Stop 4 now adjusted to perform service on time at Stop 5.

In Route 3, Stop 2 is predecessor of Stop 3 and the items picked up at this stops are to be delivered at Stop 3. To simulate this, we first simulate a route where Stop 2 is successfully processed which also make Stop 3 successfully processed as shown in Table 6.2(c). One 'Not At Home' failure type in randomly introduced in this route. We then make the driver arrive late by changing the time window of Stop 2 [570, 584]. So Stop 2 is not processed as the driver arrives at 585 at Stop 2. This results in failure of Stop

Real	Traveled Time(Minute)	547
	Traveled Distance (Km)	215
Simulated	Traveled Time(Minute)	547
	Traveled Distance (Km)	215
Service Status	Pending Service	0
	Completed Services	7
	Failed Services	0
	Driver Late	0
Failed Service Status	Item Not in truck	0
	Truck Capacity Issue	0
	Customer Not At Home	0
	Bad Address	0
	Item Quality Issue	0

(a) Route 1 (Simple route)

Real	Traveled Time(Minute)	508
	Traveled Distance (Km)	215
Simulated	Traveled Time(Minute)	508
	Traveled Distance (Km)	215
Service Status	Pending Service	0
	Completed Services	6
	Failed Services	1
	Driver Late	1
Failed Service Status	Item Not in truck	0
	Truck Capacity Issue	0
	Customer Not At Home	0
	Bad Address	0
	Item Quality Issue	0

(b) Route 2 (With late driver)

Real	Traveled Time(Minute)	547
	Traveled Distance (Km)	215
Simulated	Traveled Time(Minute)	547
	Traveled Distance (Km)	215
Service Status	Pending Service	0
	Completed Services	5
	Failed Services	2
	Driver Late	0
Failed Service Status	Item Not in truck	0
	Truck Capacity Issue	0
	Customer Not At Home	2
	Bad Address	0
	Item Quality Issue	0

(c) Route 3.a (With precedence constraints and no failures)

Real	Traveled Time(Minute)	526
	Traveled Distance (Km)	215
Simulated	Traveled Time(Minute)	526
	Traveled Distance (Km)	215
Service Status	Pending Service	0
	Completed Services	3
	Failed Services	4
	Driver Late	2
Failed Service Status	Item Not in truck	2
	Truck Capacity Issue	0
	Customer Not At Home	0
	Bad Address	0
	Item Quality Issue	0

(d) Route 3.b (With precedence constraints and failures)

Real	Traveled Time(Minute)	547
	Traveled Distance (Km)	215
Simulated	Traveled Time(Minute)	547
	Traveled Distance (Km)	215
Service Status	Pending Service	0
	Completed Services	5
	Failed Services	2
	Driver Late	0
Failed Service Status	Item Not in truck	0
	Truck Capacity Issue	2
	Customer Not At Home	0
	Bad Address	0
	Item Quality Issue	0

(e) Route 4 (Truck capacity constraint)

Simulated	Traveled Time(Minute)	5476
	Traveled Distance (Km)	5904.94
Service Status	Pending Service	0
	Completed Services	25
	Failed Services	266
	Driver Late	257
Failed Service Status	Item Not in truck	8
	Truck Capacity Issue	0
	Customer Not At Home	0
	Bad Address	0
	Item Quality Issue	1

(f) Large sample route with 89 stops

FIGURE 6.2: SimVRP simulation results

3 as well as the associated item are not found in the truck. The results are shown in Table 6.2(d).

In Route 4, we introduce capacity constraints at Stop 3. At this stop we assigns two pickup services, each with capacity higher than truck's maximum capacity. So both services fail and items are not picked up as shown in 6.2(e).

6.3 Conclusion

SimVRP simulation results for all the routes are identical to the expected in real life. To ensure the scalability of the model for larger route instances, we successfully simulated a route with 89 stops provided by ClearD as shown in Table 6.2(f). It should be noted that the scope of SimVRP, at present, is limited. It finds the total travelled time and distance and randomly simulates failures. With this piece of work, our goal was to develop a prototype of the data driven simulation model which could be extended to incorporate failure models and randomness in Pickup and Delivery. In future work, more realistic failure models, as developed in Chapter 3 and 4 could be integrated in SimVRP. Every time, a service is to be performed inside SimVRP, the service could be predicted over multiple failure prediction data models designed to address specific failure type. If any of the failure prediction data model flags the service as a failed service, SimVRP should not process this service. Also weather and the driver's profile could be used in the platform to make SimVRP capture real time route execution setup.

Chapter 7

Conclusions

We studied five different failure types and obtained good classification results for failure detection in Pickup and Delivery Problem. We compared different data sampling strategies and showed that for our dataset random under-sampling performs best. To complement the classifier results and better understand the failure reasons, we extracted Association Rules for each failure type. With Important Features obtained from Random Forest and Association Rules, we concluded that phone calls, high start slack and service time, retailer and some geographic regions have high impact on service failures. Different actors in the supply chain can act on these parameters to control failure rate and improve route quality. The simulation model implemented in the thesis was successfully able to simulate route. This simulation model evaluates a route based on travelling time, distance and total number of failed services.

In the future, to extend the work in failure type prediction, the dataset aggregation method could be revised to prevent information loss resulted from simple averaging of service features. The performance of the models could be improved by (1) improving the quality of the dataset and better defining failure types, (2) improving the strategy to deal with imbalanced dataset. The random under-sampling strategy might remove important instances from majority class which can compromise model's performance.

For extending the work in SimVRP, (1) the driver's profile could be considered to include the effect of a new driver, (2) random service failure types could be replaced by the

data models developed in this thesis, (3) bad weather and traffic conditions need be implemented to evaluate routes in more real life environment, (4) finally, SimVRP is to be integrated into the optimization model.

Bibliography

[1] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[2] F. Arnold, I. Cardenas, K. Sörensen, and W. Dewulf. Simulation of b2c e-commerce distribution in antwerp using cargo bikes and delivery points. *European Transport Research Review*, 10(1):2, 2018.

[3] Statista authors. Statista. https://www.statista.com/statistics/289741/canada-retail-e-commerce-sales/, 2018 (accessed on October, 3).

[4] GE Batista, RC Prati, and MC Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[5] Leo Breiman. Random forests. *Machine learning*, pages 5–32, 2001.

[6] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[7] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A. A. Juan. Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*, 47(2):32, 2015.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[9] N. V. Chawla, N. Japkowicz, and A. Kotcz. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[10] M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang. Disease prediction by machine learning over big data from healthcare communities. *IEEE Access*, 5:8869–8879, 2017.

[11] ClearD_authors. Clear Destination. https://www.cleardestination.com/, 2018 (accessed on September, 30).

[12] J. Edwards, A. McKinnon, and S.L. Cullinane. Carbon auditing the last mile: modelling the environmental impacts of conventional and online non-food shopping. *Green Logistics Report, Heriot-Watt University*, 2009.

[13] G. Figueira and B. Almada-Lobo. Hybrid simulation–optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory*, 46:118–134, August 2014.

[14] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[15] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010.

[16] R. Gevaers, E. Van de Voorde, and T. Vanelslander. Cost modelling and simulation of last-mile characteristics in an innovative b2c supply chain environment with implications on urban areas and cities. *Procedia-Social and Behavioral Sciences*, 125:398–411, 2014.

[17] B. Goethals. Survey on frequent pattern mining. *Univ. of Helsinki*, 19:840–852, 2003.

[18] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[19] A. Legrand, L. Marchal, and H. Casanova. Scheduling distributed applications: the simgrid simulation framework. In *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, pages 138–145. IEEE, 2003.

[20] G. Lemaître, F. Nogueira, and C. K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[21] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[22] D. Liang, C.-F. Tsai, A.-J. Dai, and W. Eberle. A novel classifier ensemble approach for financial distress prediction. *Knowledge and Information Systems*, 54(2):437–462, 2018.

[23] R. Longadge and S. Dongre. Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707*, 2013.

[24] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.

[25] I. Mani and I. Zhang. knn-approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126, 2003.

[26] S. Pan, V. Giannikas, Y. Han, E. Grover-Silva, and B. Qiao. Using customer-related data to enhance e-grocery home delivery. *Industrial Management & Data Systems*, 117(9):1917–1933, 2017.

[27] K. F. Doerner Parragh, S. N. and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] M. Punakivi, H. Yrjölä, and J. Holmström. Solving the last mile issue: reception box or delivery box? *International Journal of Physical Distribution & Logistics Management*, 31(6):427–439, 2001.

[30] U. Ritzinger, J. Puchinger, and R. F. Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

[31] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, November 2006.

[32] L. Song, T. Cherrett, F. McLeod, and W. Guan. Addressing the last mile problem: transport impacts of collection and delivery points. *Transportation Research Record*, 2097(1):9–18, 2009.

[33] J.H.R. Van Duin, W. De Goffau, B. Wiegmans, L.A. Tavasszy, and M. Saes. Improving home delivery efficiency by using principles of address intelligence for b2c deliveries. *Transportation Research Procedia*, 12:14–25, 2016.

[34] W. N. Venables and B. D. Ripley. Tree-based methods. In *Modern Applied Statistics with S*, pages 251–269. Springer, 2002.

[35] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.