MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master‘s Thesis

„Data Mining for efficient Request Bundle Generation in Auction – based Transportation Collaborations“

verfasst von / submitted by

Edoardo Baggio, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 977
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Business Analytics
Betreut von / Supervisor:	Univ.-Prof. Dr. Jan Fabian Ehmke
Mitbetreut von / Co-Supervisor:	

Table of Contents

List of Tables

List of Figures

Zusammenfassung

Eine Kooperation zwischen Logistikdienstleistern stellt verschiedene Vorteile dar. Beispiele aus der Industrie und der Literatur zeigen, wie die horizontale Kollaboration in der Logistik zu Kosteneinsparungen im Transport, Umsatzerhöhungen und CO2-Einsparungen für das gesamte Netzwerk führen. Wesentlicher Teil der Kollaboration zwischen Logistikdienstleistern ist der Austausch von nicht-profitablen Kunden. Diese können sich für einen Dienstleister als nicht profitabel erweisen oder nicht optimal zu dessen existierenden Routen passen. Hingegen kann es eine Möglichkeit zur Umsatzerhöhung für einen anderen Dienstleister im Netzwerk darstellen, wenn der Kunde in die existierenden Routen effizienter integriert werden kann. Die Literatur unterscheidet zwischen dem Austausch von Einzelkunden und dem Austausch von Kundenbündeln, d.h. Gruppen von Kundenanfragen. Letzteres bietet mehrere Vorteile, vor allem die unterschiedliche Profitabilität eines Kunden, je nach dem, ob er einzeln oder in einem Bündel betrachtet wird. Demnach könnte ein Kunde allein unprofitabel für alle Teilnehmer eines Netzwerkes sein, jedoch einen Mehrwert für einen Dienstleister bieten, wenn das ganze Bündel wirtschaftlich attraktiv ist. Allerdings ist der Kundenaustausch in Bündeln mit einem kombinatorischen Problem verbunden. Für den Fall, dass n unprofitable Einzelkunden ausgetauscht werden müssen, bestehen $2^n - 1$ verschiedene Kombinationen, in denen sie in Bündeln zusammengefasst werden können. Aus der Literatur geht hervor, dass solche Bündel mittels kombinatorischer Auktionen unter den Netzwerkmitgliedern gehandelt werden. Das bedeutet, dass Logistikdienstleister alle Bündel einer Auktion auswerten müssten, um einen Gebotspreis auszuhandeln. Da dieser Wert der Profitabilität eines Bündels entsprechen soll, wird er nach den Kosten für das Einfügen eines Bündels in eine bestehende Route berechnet. Das trägt dazu bei, dass ein Vehicle Routing Problem für jedes Bündel gelöst werden soll. Angesichts der Tatsache, dass das Vehicle Routing Problem als NP-Hard gilt, ist eine Gebotspreisberechnung für alle Bündelmöglichkeiten zeitlich undenkbar. Es wird in der Literatur anerkannt, dass nicht alle Bündel gleich profitabel sind und dass es ausreichend ist, wenn die Netzwerkteilnehmer nur für die attraktivsten Bündel ein Gebot abgeben können. Das kann die Problemgröße deutlich reduzieren und die Berechnung des Preisgebots wieder möglich machen. Das zweite Problem umfasst die Durchführbarkeit der Lösungen. Demnach ist darauf zu achten, dass alle Einzelkunden einmal erscheinen, wenn der Austausch in Bündeln erfolgt. Daher sind in der Literatur Methoden entstanden, bei welchen nur die attraktivsten Bündel, die zu durchführbaren Lösungen führen, für die Auktion generiert werden können. In den letzten Jahren wurden verschiedene Heuristik-Methoden zu diesem Zweck vorgestellt. Darunter fällt der Ansatz von Gansterer & Hartl (2018), der einen effizienten evolutionären Algorithmus zur Generierung attraktiver durchführbarer Kundenbündel bereitstellt. Alle Methoden zu diesem Zweck basieren auf Heuristiken. Nach derzeitigem Wissensstand verwendet keiner Business Analytics Methoden, um attraktive Bündel für Auktionen zu generieren. Diese Arbeit soll diese Lücke füllen.

Vergangene Auktionen bieten historische Daten von Kundenbündeln und dazugehörigen Gebotspreisen, die genutzt werden könnten, um ein Regressionsmodell zu trainieren. Damit können Gebotspreise für zukünftige Auktionen vorhergesagt werden, um nur attraktive Bündel den Netzwerkteilnehmern für die Gebotspreisbestimmung anzubieten. Dafür müssen neue Features berechnet werden, die die Bündel numerisch beschreiben, welche möglicherweise mit den Gebotspreisen korreliert sind, um diese effektiv vorherzusagen. Das Verfahren von Gansterer & Hartl (2018) wurde schon nachimplementiert und mittels existierender Instanzen getestet. Ein neues Business Analytics basierendes Verfahren kann daher implementiert werden und mit denselben Instanzen getestet werden, um seine Wirksamkeit zu prüfen. Ziel dieser Masterarbeit ist es zum einen, herauszufinden, ob die Implementierung einer solchen alternativen Methode zum vorgestellten Zweck möglich ist und zum anderen der Vergleich der Lösungsqualität und der Laufzeit beider Verfahren.

Abstract

Cooperation between logistics service providers presents various advantages. Examples from industry and literature show how horizontal collaboration in logistics leads to cost savings in transport, increase in turnover and CO2 emissions savings for the entire network. An essential part of collaboration between logistics service providers is the exchange of non-profitable customers. These may not be profitable for a service provider or may not fit optimally within the existing routes. On the other hand, it can represent an opportunity for another service provider in the network to increase revenue, if the customer can be integrated more efficiently into its existing routes. The literature distinguishes between the exchange of individual customers and the exchange of customer bundles, meant as groups of customer requests. The latter offers several advantages, most notably the different profitability of a customer depending on whether it is considered individually or in a bundle. Accordingly, a customer alone could be unprofitable for all participants in a network but offer added value for a service provider if the whole bundle is economically attractive. However, customer exchange in bundles is associated with a combinatorial problem. If n unprofitable individual customers need to be exchanged, there are $2^n - 1$ different combinations in which they can be combined in bundles. The literature shows that such bundles are traded among network members by means of combinatorial auctions. This means that logistics service providers would have to evaluate all bundles in an auction in order to negotiate a bidding price. Since this value is supposed to correspond to the profitability of a bundle, it is calculated according to the cost of inserting a bundle into an existing route. This implies that a Vehicle Routing Problem should be solved for each bundle. Given that the Vehicle Routing Problem is NP-Hard, a bidding price calculation for all bundle possibilities is unthinkable in terms of time. It is recognised in the literature that not all bundles are equally profitable and that it is sufficient if network participants can bid only for the most attractive bundles. This can significantly reduce the problem size and make the calculation of the bidding price feasible again. The second problem involves the feasibility of the solutions. Specifically, it must be ensured that all individual customers appear exactly once when the exchange is made in bundles. Therefore, methods have emerged in the literature where only the most attractive bundles leading to feasible solutions can be generated for the auction. In recent years, various heuristic methods have been presented for this purpose. These include the approach of Gansterer & Hartl (2018), which provides an efficient Genetic Algorithm for generating attractive feasible customer bundles. All methods for this purpose are based on heuristics.

To the best of my knowledge, none uses business analytics methods to generate attractive bundles for auctions. This thesis aims to fill this gap.

Past auctions provide historical data of customer bundles and associated bidding prices that could be used to train a regression model. This can be used to predict bidding prices for future auctions in order to offer only attractive bundles to the network participants for price determination. To achieve this, new features that numerically describe the bundles that are possibly correlated with the bidding prices must be calculated in order to effectively predict them. The method by Gansterer & Hartl (2018) has already been re-implemented and tested using existing instances. A new business analytics-based procedure can therefore be implemented and tested with the same instances to verify its effectiveness. The aim of this master thesis is firstly to find out whether the implementation of such an alternative method is possible for the purpose presented and to compare the solution quality and the runtime of both methods.

1 Introduction

In a competitive environment, companies may either classically directly compete against each other, or be part of collaborative networks. The latter has several advantages, such as the reduction of expected costs, possible revenue increase of the whole network, as well as increased efficiency, increased service levels and market shares, while contributing to the fulfilment environmental goals [19]. Collaborating companies sharing parts of their resources for more effective utilization can be referred as alliances [24]. Nowadays many companies or big corporations are part of collaborative networks or strategic alliances [22], the Deutsche Lufthansa for example is part of the Star Alliance, a strategic alliance of airlines with main hubs placed in different parts of the world. This enables the airlines to sell tickets to destinations which are not directly served by each of them, since every member can sell tickets for seats on other carriers' aircrafts. Through these alliances, airlines can rely on a truly global network which guarantees the reachability of every point on earth, fostering economies of scale and above all economies of scope of the single alliance members [12].
In general, literature distinguishes between vertical and horizontal collaborations [9]. In the first case, companies which do not directly operate within the same business collaborate to take advantage of each other's business area. An example of vertical collaboration is the alliance between the Deutsche Lufthansa AG and the Deutsche Bahn AG, whose seats on some trains are sold by the airline to make its customer reach secondary cities within Germany from its main hubs [46].

Horizontal collaboration on the other hand describes cooperation between companies which operate within the same business area and would hence be direct competitors if they would not collaborate [9]. Wang et al. (2014) [43] considers horizontal collaboration to efficiently optimise the internal resources of each single member. Transportation logistics companies may leverage from this kind of collaboration, since it may be the case that, due to the operational planning of deliveries, which often occurs on daily basis, some customer requests do not efficiently fit in the route of one carrier but may be optimally delivered by another member of the network [5]. A horizontal collaborative network in transportation logistics is hence beneficial for the internal exchange of unprofitable customers within the members with the aim of maximising the total revenue of the whole alliance [18] or, alternatively, of minimising its total transportation costs [1]. In the context of collaboration among logistics carriers, Berger & Bierwirth (2010) [5] distinguishes two ways of exchanging customer

requests within such networks, namely single customer reassignment, and reassignment in customer bundles. In the first case, customers are reassigned one by one, a single unprofitable request for one carrier is transferred and integrated into the route of another member. In the second case, customers are grouped together into bundles, referred as groups of customer requests, and reassigned as such; this implies that the single unattractive requests may come from different members of the transportation network, but after these have formed a bundle, they are integrated into the route of one single carrier. It means, only one transporter takes over the whole requests placed in a bundle [18].
Literature states that companies have the interest of acquiring sets of customers instead of single ones [11], as in this way carriers are able to directly fulfil combination of requests. Bundles are particularly attractive since the value of a request could change significantly if other requests are combined [39]. Gansterer & Hartl (2018) [18] presents an example showing how a single extra request, which would represent a significant detour from the already predefined route of a carrier, would probably not be profitable alone; but suddenly becomes attractive if combined with another set of requests which bring extra revenue.

2 Problem Definition

Customer bundles may hence represent an opportunity for collaboration networks in transportation logistics, but this kind of customer reassignment is associated with issues of mathematical and combinatorial nature. The problem arises when dealing with the quantity of possible customer bundles that can be formed out of single unattractive requests, because if there are n customers to be reassigned, there are $2^n - 1$ possibilities of generating possible bundles; this means that the problem size increases exponentially [18]. Since it is assumed that the attractiveness and the willingness to pay for the added customers contained in each of the bundles is based on the insertion cost of the latter into the existing route of a carrier [5], the more bundles are generated, the more Vehicle Routing Problems (VRP) must be solved to state these values. This increases the computational time of this problem significantly, given the NP-Hard nature of the VRP [30]. Generating all possible bundle combinations is hence not possible for large problem sizes, but on the other hand, different bundles have distinctive levels of attractiveness for the companies of the network [18]. The way how bundles of requests can be traded among the network's members can be conceptualized as a decentralized combinatorial auction-based exchange system. Berger & Bierwirth, (2010) [5] presents a 5-phase combinatorial auction which aims at exchanging customer bundles. For the scope of this thesis, only the first four phases are presented in figure 1.

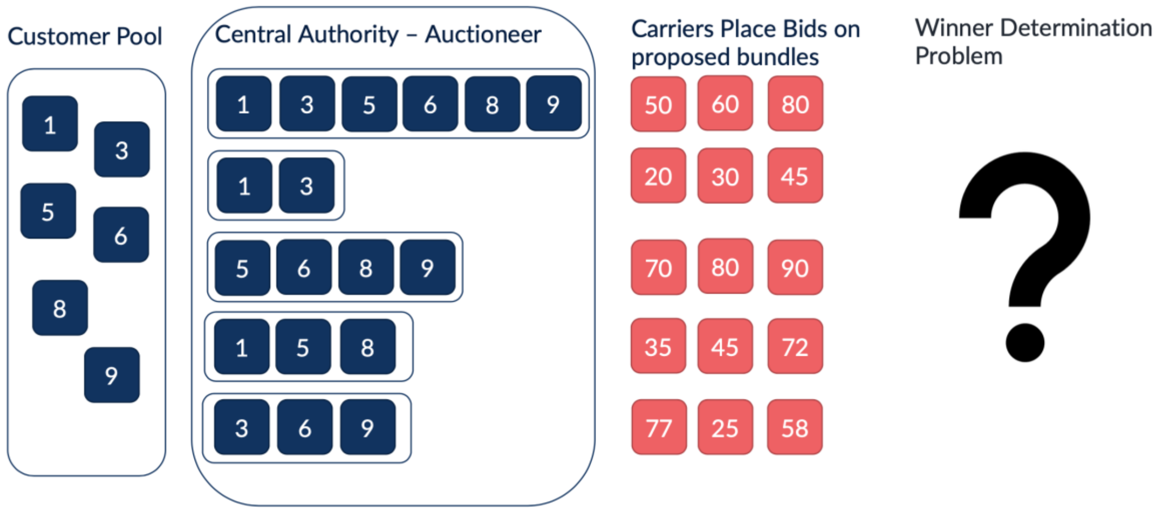
Customer Pool
1
3
5
6
8
9
Central Authority – Auctioneer
1 3 5 6 8 9
1 3
5 6 8 9
1 5 8
3 6 9
Carriers Place Bids on proposed bundles
50 60 80
20 30 45
70 80 90
35 45 72
77 25 58
Winner Determination Problem
?

Figure 1 5-Phase Combinatorial Auction Procedure

In the first phase, each carrier states which single requests will be placed in the customer pool, the blue boxes in figure 1 represent customer requests and the numbers represent their respective request IDs; it is assumed, that single requests that do not optimally fit in the existing routes of the carriers are placed in this virtual basket. Secondly, a central authority, which represents the auctioneer, generates the bundles out of the single requests in the customer pool. These bundles are then offered to the carriers as inseparable group of requests. In the third phase, carriers place bids on the proposed bundles according to the attractiveness of each single bundle for each of them. Following the example in figure 1, assuming that there are three carriers in a network, each of them places a bid for each proposed bundle, hence there will be three bids for every bundle. The red boxes in figure 1 represent the bids of each single carrier, and the numbers inside represent the bidding values. The fourth phase called Winner Determination Problem states which carrier won which bundle according to the placed bids. In the fifth and last phase, profit is shared among the network's members.

The role of the auctioneer is crucial for the presented auction procedure since, as stated above, the auctioneer cannot simply offer all possible bundle combinations to the carriers, since it is assumed that the bidding prices, which reflect the attractiveness of each bundle for each of the carriers, are the results of the calculated VRPs to fit the bundles into their existing routes.

A possible solution to this problem is offering only a limited number of bundles in the bidding phase, which leads to a drastic problem size reduction and a consequential computational time reduction for the carriers to come up with bidding prices. More specifically, the auctioneer should propose only those bundles which are meant to be attractive for the carriers (i.e., 'good bundles') and directly discard all other bundles which are believed to be not of interest for the carriers [18].

A second problem for the auctioneer arises when dealing with the fourth phase of the described auction procedure, since the proposed bundles not only have to be attractive for the carriers but must also be able to build feasible solution sets, as described in [18]. A feasible solution consists of a set of bundles, in which each customer appears exactly once in the whole set, and no customers are missing. Since each bundle of the winning set will be assigned to a carrier, the feasibility of a solution implies that, after reassigning customers into bundles, no customers are delivered by multiple carriers and no customers are left out.

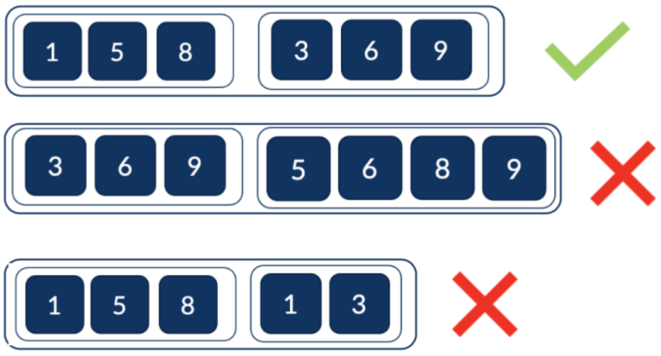

Figure 2 Example of Sets of Bundles

Figure 2 shows a feasible solution set and two infeasible ones, the outer boxes represent a set, while the inner boxes represent the bundles. Assuming that there are 6 customers that have to be reassigned, with respective IDs 1, 3, 5, 6, 8, 9 and two carriers in a network, the first set represents a feasible way of reassigning these customers into two bundles. On the other hand, the second and the third sets are not feasible solutions, since customer 9 would be delivered by both carriers after the reassignment in the second example, and in the last set, customer 9 would not be delivered at all.

The auctioneer hence must consider that the proposed bundles are able to build feasible sets with each other.

The literature acknowledged the potential of customer exchange in request bundles as well as its combinatorial difficulty. In view of these facts, some recent papers present several creative ways of generating bundles which fulfil the described conditions.

Existing literature makes use of different kinds of heuristic methods to solve the bundle generation problem. To the best of my knowledge, none of the papers that deal with customer bundle generation in the context of transportation logistics solves this problem with alternative techniques.

Business analytics and data mining techniques can represent an alternative to heuristics, leveraging historical data and making use of machine learning algorithms to predict and generate attractive bundles.

The Genetic Algorithm framework presented in Gansterer & Hartl (2018) [18] has already been implemented on artificially generated instances; the scope of this master thesis is to implement a new framework based on data mining procedures and a machine learning algorithm which aims at finding attractive feasible solutions for the bundle generation problem, without any size limitation constraints. The developed framework is then applied to the same instances on which the Genetic Algorithm has already been tested, in order to compare and contrast the effectiveness of the two approaches in terms of solution quality and of computational time. It must be pointed out, that the managerial objective of [18] is the maximisation of total network profit, the application of the algorithm on the proposed instances aims at minimising total transportation costs, the comparison of the two approaches in terms of solution quality will be hence based on this last criterion. Additionally, Gansterer & Hartl (2018) [18] considers a Pick-up and Delivery Problem, the re-implemented version of their approach can also solve a pure VRP, hence the development of the new framework focusses on the solution of this problem.

To pursue the machine learning approach for the determination of attractive bundles, historical data is essential to train a machine learning model. Historical data are the bidding prices placed by the carriers for bundles in past auctions. In order to use these prices in a machine learning algorithm, the bundles must be numerically described by some features which depict the attractiveness of the bundles and can be used by the model to precisely predict the assignment costs of the bundles. The attractiveness of a bundle can be estimated by predicting its future bidding price. Since the latter is a continuous value, a regression model in machine learning must be chosen to solve this problem [44].

For a fair comparison with the approach presented in Gansterer & Hartl (2018) [18], the regression model approach has to be developed under the constraint that carriers do not want to reveal sensitive pieces of information to the central auctioneer to assess the attractiveness of a bundle. For example, the carriers do not reveal any information about their existing routes, which would be a major piece of information to state whether a bundle of customers may be of interest for a carrier or not. Since this thesis aims at comparing the solution quality and the computational time of bundle generation of two described approaches, the focus is on the role of the auctioneer which offers the bundles, under which criteria customer requests are placed in the customer pool and the price generation in the proper bidding phase are out of scope.

This thesis is structured as follows, section 3 presents an overview of recent literature on collaborative networks in transportation and reassignment in customer bundles, particularly focussing on their generation strategies; section 4 presents a brief theoretical background of the chosen machine learning models and their selected evaluation criteria. The data processing, the application of the machine learning models, and their evaluation is presented in section 5. Section 6 compares the performances of the machine learning-based bundle generation approach with the already present Genetic Algorithm. Final conclusions are derived in section 7.

3 Literature Review

Collaboration in different fields of transportation has been widely studied by the literature during the past decades; in the field of city logistics, this has an influence on sustainability issues as well, and it is therefore welcome by municipalities, which fund research in this field [18]. It is also acknowledged, that although transportation collaboration is a well-known topic, horizontal collaboration is not as deeply considered by the literature as vertical collaboration [29]. Horizontal collaboration is considered as a way to reduce costs by resource optimisation; asset repositioning or empty runs are considered potential *hidden costs* of each single logistics carrier, this kind of cost of doing business however can be drastically reduced if the carriers horizontally cooperate, as reported in [16]. The paper models carrier collaboration as a lane covering problem aiming at minimising asset repositioning; it implements a greedy algorithm to solve this problem and states that as the number of lanes increases, the asset repositioning ratio decreases, resulting in more efficient routes for a collaborating network of carriers. Laporte et al. (2007) [29] also explores the potential of horizontal collaboration among carriers, acknowledging that being part of a network brings to cost reductions achieved by a better and more equal distribution of resources such as the

minimisation of empty truck runs. The paper models the routes of the carriers as a pick-up and delivery problem with time window, which are considered as single depot if the carriers operate autonomously but become multi depot if they cooperate. The difference in the solutions of both approaches can be measured to assess the benefits of potential collaborations. The authors use a large neighbourhood search heuristic to solve the problem, given the NP-hard nature of the pick-up and delivery problem with time window; the heuristic is tested also on a small instance of two carriers with 50 requests each, and it is directly shown how collaborating routes bring to 12,46% cost reduction compared to a situation in which they would serve their requests completely on their own. The paper tests its approach on larger instances and on a real-world instance from Germany as well, also taking profit sharing into consideration, and concluding that single carriers are better off when pooling resources than operating singularly.
Ergun et al., 2007 [15] also identifies empty relocations of trucks as a cost that a carrier may drastically reduce by taking part to a network in order to optimise routes striving for continuous loaded movements, reducing the need of relocation of the carriers. The paper acknowledges that collaboration networks are already present on the US market, which in real life made its member save on routing costs. Theoretically, the authors model this problem as a lane covering problem, which is solved using a heuristic method.
Dahl & Derigs (2011) [10] as well acknowledges the potential of collaboration among carriers and developed a real-time collaborative decision support system which the authors call pool.tour. This decision support system leverages from data of the dispatchers of the network's members, and a heuristic which processes information about orders, routes and vehicle positions makes continuous proposals for inserting requests into different tours. The authors tested the decision support system on a real-world currier network based in Europe for prolonged time and report the benefits of the usage of such system in terms of cost savings compared with isolated planning. The authors found out that such a system is beneficial for the network total cost saving, but they also acknowledge that the compensation system implemented in the decision support system is crucial for the success of such a tool and that different compensation schemas bring to different results in terms of total cost savings.
Literature has dealt with the bundle generation problem alone as well, although this problem has not been deeply studied in recent years [42]. Berger & Bierwirth (2010) [5] not only introduces the auction-based decentralised planning system described above, but also tries to solve the bundle generation problem assuming size constraints; they offer all bundle

combinations in the auction, but they let the bidders place only one request into the customer pool to keep the problem size manageable.

More recent papers allow larger problem sizes and use different methodologies to come up with the feasible-attractive bundles out of the whole possible combinations. Los et al. (2020) [32] overcomes the combinatorial auction with a central auctioneer and considers a multi-agent approach, studying the effects of request bundling with such a method. The paper proposes a bundle generation based on a relatedness measure, whose idea is acknowledged to be taken from Ropke & Pisinger (2006) [38]. Since the authors consider the model as a pick-up and delivery problem, the selected bundles must contain requests whose pick-up or delivery locations of one order are similar to the pick-up or delivery locations of another order, so that the orders can be efficiently served together. The similarity of two pick-up and delivery locations are defined based on travel time and time windows. To test their bundling approach and the impact of bundling in request exchange, the authors created instances of 2.000 orders and 150 carriers, the instances were tested by auctioning single requests to set a benchmark, and then the bundling procedure is introduced. The paper reports that when introducing the bundles, the travel costs decrease by roughly 4%. The authors directly acknowledge that the bundling brings benefits in terms of cost reduction, but it slows down the solving time of the instances by about 10 times. The authors also further tested the bundle benefits by changing the parameters of the instances, eventually considering 500 carriers, and concluded that bundling brings to cost reductions of 1,7% compared to single request reassignment, with the trade-off of increase computational time.

Mancini & Gansterer (2022) [34] considers a Vehicle Routing Problem with occasional drivers and order bundles, the paper also acknowledges the fact that the bundle possibilities grow exponentially with the number of orders, and hence propose an agglomerative clustering procedure and a more innovative *corridor-based* bundle generation approach. The first method bundles requests together based on their geographical location, it is assumed that a bundle in which all requests are close to each other is profitable, since the driver can serve them all with relatively small marginal costs and therefore it brings benefits in terms of costs to place requests into bundles. The second more innovative method uses the direction of a driver as an attractiveness measure instead of the proximity of the clients, assuming that drivers may accept to deliver customers even if they are distant from each other, given that they are located on the original way of the driver or on the way home. They solve this problem using a Large Neighbourhood Search heuristic. As far as the computational study is concerned, the authors test the performance of the heuristic as well as the performance of the two presented bundling methods on some generated instances. They conclude that the newer corridor-based bundling method outperforms the classical clustering-based approach, since the total costs decrease by 2,7%; in addition, they could demonstrate that this method generates few and attractive bundles, so that the computational time is reduced by 86,7%. It must be however pointed out that the authors consider a very special case of the VRP, namely the inclusion of occasional drivers, which perform the pick-up at the depot, drive to their destination and may choose whether to transport further goods on behalf of the shipping company. It is hence comprehensible that requests that are even distant from each other, but which lay on the way of the occasional drivers, are attractive for them. This approach however may have a different effect on regular drivers.

Existing literature presents approaches in which bundles are generated by the auctioneer, like described in the introduction, or directly by the carriers themselves. This is the case presented in Triki et al. (2014) [42], which measures bundle synergy for effective bundle generation and implemented a sequential ascending and a sequential descending heuristic. Two different ways for measuring the synergy of a bundle are presented; a distance-based pairwise synergy and a hop-based pairwise synergy. The first method considers the distance between loads expressed in travel time between the origin of a load and the destination of its precedent load, so to measure time compatibility. The second measure is based on the number of empty relocations in a tour, the less relocations a truck needs in a tour the higher the synergy. As far as the two heuristics are concerned, the first method considers the bundle of maximal cardinality and computes the benefit of removing loads to the bundle, the second proposed method does the opposite, namely it starts with an empty set, and computes the incremental profit by adding requests to the bundles. Results show the superiority of the second method. Gansterer & Hartl (2018) [18] points out how the bundle generation made directly by the carriers enables them directly to generate attractive bundles, this comes however with the difficulty of finding a solution for the Winner Determination Problem, if for instance multiple carriers propose attractive bundles whose requests overlap each other.

Recent literature focusses on efficient bundle generation acknowledging the fact that collaborative networks do bring benefits to their members, but the latter are not willing to share sensitive strategic pieces of information to the direct competitors. This is reported by Los et al. (2020) [32] and Gansterer & Hartl (2018) [18], the latter follows the 5-phase approach presented in [5], and lets a central authority generate the bundles out of the single requests placed in the pool by each carrier. To produce the bundles, the authors developed a Genetic Algorithm which generates attractive *candidate solutions*, which are sets of bundles that satisfy the requirements described in the introduction. Acknowledging the reluctancy of the carriers to reveal sensible pieces of information to the central authority, the latter has to find an alternative way to approximate the attractiveness of the proposed bundles. The authors assume that attractive bundles have a high density, isolated from other bundles and whose requests can be fulfilled within short travel times. Accordingly, a proxy function to state the attractiveness of customer bundles based on non-sensitive parameters such as centroid of the bundle, its density, and its isolation with regards to the rest of the requests has been developed. This proxy function is then used as the fitness function of the Genetic Algorithm to determine the quality of the candidate solutions. To test their approach, the authors use instances which simulate a network of three carriers. The papers first presents a situation in which each carrier can submit 4 requests to the initial pool, and hence 4.095 different bundles combinations can be generated; several bundle pool size limitations are tested out, showing how the larger the pool, the smaller the solution quality lost in percentage, although limiting the bundle pool to 500 has an impact of only 5,2% on average on solution quality, but the running time decreases from 47,6 to 5,5 seconds. The authors also test the case in which the carriers can submit 5 unattractive requests into the initial pool, leading to 32.767 possible bundle combinations. The results confirm initial assumptions made by the authors, it is sufficient to generate few, attractive feasible bundles to achieve high quality results with a very compact running time. Due to the promising results in terms of running time of the framework, the Genetic Algorithm is applied to real-world size instances with 210 requests, comparing the performances of an initial request pool of 45 and 90 requests, which can be processed in roughly 45 and 70 minutes respectively.

4 Machine Learning Models & Evaluation Criteria

As described in the section above, a dataset must be first created before a machine learning algorithm can be trained. More specifically, features which describe the bundles, that can be referred as independent variables in machine learning, must be engineered in order for a model to predict the bundle prices, referred as dependent variable. Since at this stage it is still unknown how the distribution of the data will look like and how complex the final model will be, machine learning algorithms of different natures are tested and compared. In order to state the most suitable machine learning algorithm for the described problem, 5 quality criteria are presented in section 4.2.

4.1 Machine Learning Models

As simplest and quickest regression model, a linear regression is trained and used as initial benchmark for more complex algorithms.
Furthermore, other 5 models have been chosen for this comparison, two additional linear models and three ensemble tree-based models. The linear models are Bayesian Ridge and Elastic Net Regressor, whereas the tree-based models are Gradient Boosting Regressor, XGB Regressor and LGBM Regressor. The ensemble models more complex and advanced than the linear models, and hence require higher training times, the reason of the comparison between these two kinds of models is to check whether more complex models bring benefits in terms of prediction performance compared to the linear models.

Linear regression

For regression it is meant the study of the statistical relationship between variables. A linear regression finds the linear relationship between one dependent variable and some independent variables. The goal of the linear regression model is to fit a line in the data so that the sum of squared residuals is minimised [45].
A dependent variable Y can be estimated by a linear function of the independent variables X_i with i representing the number of the independent variables. Assuming that there are three independent variables, a linear regression function may be expressed as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

with ε being a random error quantity with mean 0.
The β_i are the regressions coefficients that have to be estimated when fitting the model to the data to find the regression line which minimises the squared residuals [3]. For this work, the *Linear Regression* Class has been imported from the Python library scikit-learn [52].

Elastic Net Regressor

The Elastic Net Regressor is a hybrid linear regression model which combines the features of Lasso and Ridge regression [14]; hence these two special versions of linear regressions are briefly introduced first.
Generally, if the regression coefficients in linear regression are unbounded, they can become very large, resulting in very high variance [6]. High variance may derive from overfitting the

regression line to the data, which is referred as the case in which a model is too well fitted for the training data, but is not able to generalize [37, Chapter 5], i.e., properly fit to the test data as well. Lasso and Ridge regression reduce model complexity and prevent overfitting [53].

In Ridge regression the objective function of the linear regression is altered by adding the sum of the squared coefficients, multiplied by a parameter λ. λ is used as penalty term, since it multiplies the squares of the regression coefficients, and the more the latter increase, the more penalized the objective function will be [53]. Since the square of the coefficients are taken, Ridge regression can be intended as a linear regression with the ℓ_2 regularization [54]. Adapted from [55], Ridge regression can be mathematically represented as

$$\min_{w} ||Xw - y||_2^2 + \lambda ||w||_2^2$$

Lasso regression works similarly to Ridge regression; a parameter λ regularizes the regression coefficients. This time however the absolute values of the latter are taken. Differently from the ℓ_2 regularization, taking only the absolute values of the coefficients as penalty can make the latter become 0, and hence not part of the model anymore. It hence has the double benefit of reducing overfitting and automatically helps in feature selection [53].

Adapted from [56], Lasso regression adds term $\lambda ||w||_1$, which is equivalent to the ℓ_1 norm of the coefficient vector [56].

Since Elastic Net is a hybrid of Lasso and Ridge regression, the model is trained with both ℓ_1 and ℓ_2 norm for the coefficients' vector [57]. The Elastic Net regression model is applied in this work by importing the associated class from the Python library scikit-learn [58]. The objective function that the algorithm minimises is the linear regression function with both penalty functions [57]. The advantage of such a regression model is that it comprises the benefits of the two original models Elastic Net is built upon, it is able to reduce the number of features used by the model, and at the same time construct more robust models for the generalization purpose [59].

Bayesian Ridge

In contrast with the classical linear regression, where the current data is the only piece of information processed by the model to fit the regression line, the Bayesian approach fits a model to the data not only considering the current data, but also leveraging from past conditions [3].

Bayesian regression takes the uncertainty into consideration [4] using conditional probabilities. The prediction of a new value in Bayesian regression is derived by a probability distribution and is assumed to be Gaussian distributed with variance σ^2 [21, 47]. Mathematically, the probability distribution of the dependent variable can be formulated as reported in [46].

$$y \sim \mathcal{N}(\beta^T X, \sigma^2)$$

The mean of the normal distribution is the product of the unknown parameters β multiplied by the independent variables X [47].

In Bayesian linear regression, not only the dependent variables but also the β are assumed to be derived from a probability distribution [3]. The Bayesian linear regression is able to leverage from past experience, since it estimates the regression parameters β calculating the conditional probability of the β values given the data. This can be estimated using the Bayesian Rule. As stated above, the y are normally distributed given a set of parameters β. This can be referred as the *conditional distribution* of y given known values for the parameters β [3]. As mentioned, the parameters β themselves also have an independent distribution themselves, the literature refers to this as *prior distribution* [3]. Following the notation of [33], let D represent the data and w the set of unknown parameters. Given these pieces of information, Bayes Rule allows to calculate the conditional probability of observing w given the data with the formula

$$P(w|D) = \frac{P(D|w)P(w)}{P(D)}$$

MacKay (1992) [33] simplifies the Bayes Rule presenting the following interpretation of the formula.

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$

The probability of observing the parameters w given the Data is referred as *Posterior Probability*. This is calculated as the probability of observing the data given the parameters, referred as *Likelihood* multiplied by the prior probability, normalised by the probability of observing the data, described as *Evidence*.
Bayesian Linear Regression allows to choose a predetermined probability for the prior if some domain knowledge about the data and its distribution is available. In case no significant pieces of information are known about the data, then a so-called *non-informative* prior, which decreases the influence on the outcome of the posterior distribution, can be chosen [4].
For this work the model is used importing the *Bayesian Ridge* class from the Python library scikit-learn [48]. The scikit-learn implementation of this model is based on the algorithm presented in [41]. It uses non-informative [49] conjugate priors [50] with parameters w, α and λ, which are estimated during the model fitting [49].
The priors over α and λ have a Gamma distribution, there are four hyperparameters of these distributions, $\alpha_1, \alpha_2, \lambda_1, \lambda_2$, which are also chosen in a non-informative manner with an initial default value of 10^{-6} [49].
In conclusion, the advantage of a Bayesian Linear Regression is that the model not only learns from the available data but can also incorporate prior knowledge and depict the uncertainty. This is possible as the model constructs an initial estimate, which is updated as soon as more data and past information become available [51].

Gradient Boosting Regressor

The Gradient Boosting Regressor is an ensemble-based machine learning algorithm. An ensemble-based method is generally more complex than the linear methods described above since it can combine multiple algorithms with the aim of improving predicting performances. Combining many models is a way of fitting a model to more complex data structures without letting the single models become too complicated [28].
Gradient Boosting belongs to the Boosting algorithms, it combines multiple weak learners to compose the ensemble model. A weak learner is to be intended as a single model which performs only marginally better than the random [28].

Gradient Boosting is an ensemble method whose learners are decision trees. The algorithm can be used for classification and regression purposes; as far as the latter is concerned, the Gradient Boosting Regressor initiates the estimation of the target variable just by calculating its mean. Successively it computes the error, namely the difference between the actual value of each datapoint and the estimated prediction value. This algorithm builds each successive learner based on the error of the previous one. From the second learner onwards, the decision trees do not aim at predicting the actual value of the target variable but the error. To prevent overfitting, each tree is scaled by a constant learning rate. By adding multiple scaled trees to the model, which iteratively predict the error of the previous learner, the ensemble model is created [28].
A mathematical formulation of the algorithm is presented in [17], simplified explanations are presented in [60] and [28], the latter summarizes the procedure with the following 8 steps.

1. Definition of the learning rate
2. Calculation of the target variable's mean as initial prediction value
3. Calculation of the residuals between the actual values and the mean
4. Build a decision tree that predicts the residuals using existing features of the dataset
5. Predict the residuals
6. Add the new predicted values to the previous ones, all added values are scaled by the predefined learning rate
7. Calculate the new residuals given the newest predicted values
8. Keep building decision trees until the predefined ensemble size is reached

Given a constant learning rate $l,$ if the user specified the ensemble size being t, then the final predicted value p_t is the result of the addition [28]

$$p_t = mean + lp_1 + lp_2 + lp_3 + \cdots + lp_t$$

For this work, the *GradientBoostingRegressor* available in the Python library scikit-learn [61] has been used. The parameters have been left to the default ones to train the model; hence the loss function is defined as the squared error, the learning rate of 0,1 and the ensemble size is 100 [61].

XGB Regressor

The Extreme Gradient Boost or XGBoost is a tree-based regression and classification algorithm presented by Chen & Guestrin (2016) [8]. Since the algorithm is presented in the mentioned paper, the latter can be considered as the reference for the entire subsection. The algorithm is available for regression and classification, given the objective of this thesis, only the first case is considered, referred as XGB Regressor. The purpose of the development of such algorithm was the necessity of creating an optimised version of the existing tree boosting algorithms in order to handle large amount of data in reasonable time; in fact, the authors claim that the proposed algorithm is ten times faster than existing popular solutions on single machine. The ways XGBoost optimises computational time are manifold, they comprehend algorithmic and machine usage optimisations as well as the ability of handling sparse data, and a novel procedure for tree split finding used in an *Approximate Algorithm*. There are various possibilities of constructing trees, depending on where the data is split; in order to evaluate the tree structures, a scoring function is used to assess its quality, the goal is to find the trees that best define the data. Finding the best split to construct a new branch however is an issue with large dataset, since each datapoint of each feature is a potential candidate for a new split. The paper acknowledges this problem and presents two *Split Finding Algorithms*. The first one is defined as the *exact greedy algorithm*, which enumerates all possible split candidates of all features, although this is extremely computationally expensive, XGBoost supports this kind of algorithm in its single machine version. The second method is presented as the *Approximate Algorithm*, which proposes candidate splits according to percentiles of feature distributions. Aggregating the data in such a manner is beneficial, since it significantly accelerates the computational time by reducing the number of split candidates. In addition, XGBoost is also optimised for split finding when the dataset is characterized by sparse data; the paper presents an additional algorithm for this purpose. The algorithm is claimed to be extremely efficient, since it processes only the non-missing values for a split and then calculates the gain of placing the missing values into the left or into the right branch, calculates the gain for both cases and returns the split and the direction with the highest score. XGBoost further optimises its runtime by leveraging from more efficient system settings. As far as the tree learning is concerned, the authors claim that data sorting is one of the most time-consuming parts of the entire procedure, hence the algorithm stores the data in in-memory units defined as *blocks*, the latter are stored in compressed column format, where the data is sorted by the feature values. When the exact greedy algorithm for split search is used, the entire dataset is pre-sorted and stored in a single block, which is then scanned by the split algorithm; this data format is computed only once and is used for all ensemble iterations. Finally, the algorithm further optimises the runtime by storing the data required for the calculation of the similarity scores in the cache, for faster data access by the CPU.

XGB Regressor is available as an open-source library for Python [62], which has been used for this work.

The reason of the consideration of such algorithm for this work is not to exploit its full potentials of large data handling and parallelization, but to test whether a more efficient and quicker algorithm than the classical Gradient Boosting Regressor achieves the same result quality with lower computational time.

LGBM Regressor

Light Gradient Boosting Machine or LightGBM is another decision tree-based boosting algorithm developed by Microsoft Research in 2016 [63], which aims improving the performances of the classical Gradient Boosting Machine algorithm, especially in terms of computational time. The algorithm is presented in Ke et al., (2017) [27], hence the reference of this subsection can be referred to this paper. The strategies how LightGBM reduces its training time are manifold. Firstly, it is acknowledged that one of the main slowing factors of the Gradient Boosting Machine lies in training the decision trees, since this implies that the optimal splitting points must be found. In the base case, all datapoints are scanned and all splitting points are tested, until the best one is found; this allows to find the best split of a node, but it is extremely inefficient in terms of computational time and memory usage, if the size of the data is significantly large. To overcome this issue, LightGBM uses a *histogram-based* splitting algorithm, which groups continuous values into bins and uses the latter to construct histograms; the split points are then tested on the feature histograms. The runtime complexity of this procedure is dependent on the amount of data multiplied by the number of features for constructing the histograms, and on the number of bins multiplied by the number of features to find the best splitting points. Since the number of bins is lower than the original number of datapoints, this procedure drastically reduces the computational time. Since this split finding procedure is dependent on the number of datapoints and the number of features, the authors developed procedures which reduce both and further increase the training performance of LightGBM. To reduce the number of datapoints while constructing new trees, the algorithm relies on *Gradient-based One-side Sampling*, which exploits the gradients for data sampling. It is based on the idea that an instance which has a small gradient is already well-trained, since the training error is small, and it should not be processed any further. On the other hand, instances with high gradient are a sign of undertraining and should be used for additional information gain. Gradient-based One-side Sampling hence first sorts the data instances according to the absolute value of their gradient in descending order and selects a percentage α of the top part of the sorted set, then it selects another percentage β of the bottom part of the set to create the new sample for further training. Since selecting majorly only the instances with large gradient would change the distribution of the data, the algorithm scales the sample with small gradient by a constant $\frac{1-\alpha}{\beta}$. This new sample is then used for further learning.

As far as dimensionality reduction is concerned, the authors developed a second algorithm which is referred as *Exclusive Feature Bundling*. The algorithm processes sparse mutually exclusive features, such as one-hot encoded features which describe categorical variables and bundles them into one single feature which preserves the meaning of the old ones. Since the number of bundles will be lower than the number of features, dimensionality can be reduced without any loss in accuracy, but significantly improving the training time.

For this work, the *LGBMRegressor* class has been imported from the *lightgbm* library for Python [64], by default the model trains 100 boosted trees.

As far as the training time is concerned, Ke et al., (2017) [27] uses XGBoost as the benchmark, since it was the fastest among the tested models; the authors show that their implemented new machine learning model outperforms the competitors.

As for XGBoost, this model is considered in this work not to exploit its full potentials in handling very large and sparse data, but to study the performance of quicker alternatives than the classical Gradient Boosting Regressor for the scope of this work.

4.2 Evaluation Criteria

Once the dataset will be created and there will be a validation that the developed features are correlated with the bidding prices, a regression model must be chosen to make predictions for future auctions. The trade-off for the chosen model should be its performance with regard to the required regression task, its understandability and its efficiency in terms of training time. For this purpose, five quality criteria have been chosen and partially newly developed to select the best possible model. These criteria are the pure accuracy of the model in terms of r^2, its training time, the ability to recognise the most attractive bundles, the model rank accuracy in terms of Spearman's ρ, and the ability to find the best possible winning bundle set.

r^2

The classical r^2 explains how well a model fits to the data, more specifically, how well the predicted values from the regression model match the actual values the model tried to predict. Mathematically, this is explained by variance of the predicted values divided by the variance of the data [20]. If the value of the r^2 is equal to 1, it means that the model perfectly fits the data.

Training time

All models are evaluated according to the training times, independently from their prediction accuracies. Since the scope of this study is to find alternative solutions to the existing ones for the bundle generation problem, the chosen model must be able to be fitted to the existing data in reasonable time, to compete with the computational times of the other existing solutions and to be quicker than the base mathematical solution of complete bundle enumeration.

Ability to recognise the most attractive bundles

This quality criterion has been developed for the specific scope of this work. The ability to recognise the most attractive bundles is referred as the accuracy of a determined model in correctly predicting the exact bundles which are going to be offered in the bidding phase. For the scope of application of the models, this measurement is more meaningful than the mere r^2, since it focusses on the accuracy in predicting the bidding price for the bundles which are of interest for the next phase, instead of the whole dataset.
Firstly, it must be stated what is meant by attractive bundles for the purpose of this study. As stated above, the variable that the models try to predict is the bidding prices for all carriers; this value however does not reveal much information about the attractiveness of a bundle, if not put into context. As stated in the introduction, the purpose of trading customer requests grouped into bundles instead of as single ones is the different level of attractiveness of the same request alone or into a bundle. This piece of information can be directly derived from the bidding prices. As mentioned above, the bidding prices are assumed to reflect the cost

of including a bundle in the existing routes of each carrier, hence, the lower the bidding price, the better. The bidding price itself however is not an indicator of the attractiveness of inserting customer requests into a route of a specific carrier, since it does not reveal how many customers can be inserted into a route at the specific cost. Dividing the bidding prices by the number of customers reveals how much it would cost for a carrier on average, to insert each customer into the existing route. The price per customer can be now used as a measure of attractiveness for each bundle, since it tells whether each request is cheaper to be served alone, or in combination with other ones. Given that a bundle A, which contains only one request a is traded at a price p; if a bundle B, which also contains request a and other additional requests is also traded at the same price p, it implies that on average a carrier would be better off delivering request a in bundle B, since the average delivery price per customer is lower, given the higher number of customers in bundle B.

The values of the predicted bidding prices using the regression models can also be divided by the number of customers in the bundles to derive a predicted price per customer. Since the lower the price per customer the more attractive a bundle is meant to be, the dataset can be sorted ascendingly according to the price per customer to have a rank of bundles sorted by their attractiveness. The same can be done with the predicted price per customer; in this way the dataset is sorted by the predicted attractiveness. Once that the dataset has been sorted according to the level of attractiveness, the bundles in the top part of the dataset are the ones that will be offered to the carriers in the bidding phase. The first n bundles from the sorted list can be selected for further processing. To state how well a model is able to recognise the n most attractive bundles, the list of n most attractive bundles according to the actual price per customer and the list of n most attractive ones according to the predicted price per customer are compared. The comparison consists of checking how many bundles present in the first list have been preserved in the second one. Dividing this number by n gives a value between 0 and 1, with 0 indicating that the two sorted lists contain completely different bundles, and that no bundle has been classified correctly by the model, whereas a value of 1 would imply that the two lists are identical, and that the model classified all n best bundles correctly.

The best n bundles according to the predicted price per customer are collected to build feasible sets for the bidding phase.

Rank accuracy – Spearman's ρ

The ability to recognise the most attractive bundles gives an overview about how many bundles are correctly classified among the very best n ones according to the predicted price per customer. The measurement however is not able to tell whether the first best bundle according to the price per customer is also the first best one according to the predicted price per customer. In other words, it is not able to specify whether the ranking within the best n bundles has also been preserved. For this purpose, Spearman's ρ reveals how much of the ranking has been preserved from the actual to the predicted list of best bundles. Spearman's rank correlation coefficient takes the ranking of two sorted lists and the number of entries as input and gives a correlation value which ranks from 1 to -1, depending on whether the rankings of the two lists are in the exact same or opposite order, respectively. The rank correlation coefficient in terms of Spearman's ρ is given by the relationship

$$\rho = 1 - 6\frac{\sum_{i=1}^{n}(r(x_i) - r(y_i))^2}{n(n^2 - 1)}$$

where $r(x_i)$ is the rank of the value x_i in the first sorted list with $i = 1 \dots n$, and $r(y_i)$ is the rank of the value y_i in the second sorted list with $i = 1 \dots n$ [2, chapter 4].

Ability to find the best possible winning set

Finally, the models are evaluated according to the ability of finding the best possible winning bundle sets. This criterion has also specifically been designed for the scope of this work. The models are compared by the ability of finding the best winning bundles in the same instance. As part of the computational study, this criterion will also be used to compare the performance of the final chosen model with the solutions given by the already implemented Genetic Algorithm.

After that a model is fitted to the training dataset, which consists of a bunch of concatenated instances, a completely new one is used as test set. The bidding prices are predicted for each single carrier, the predicted price per customer is calculated and the dataset is then sorted according to these values. Successively, the best n bundles are selected as candidates to build feasible sets. This is repeated for each carrier, if the alliance is composed by m carriers, the final list of best bundles will have length mn. Predicting bundle prices does not assure that all bundles are able to build feasible sets when combined; hence, all feasible sets of bundles are built from the final list of best bundles, and the bundles which are meant to be attractive, but are not able to build any feasible set combined with other bundles are directly discarded and not processed any further. Before the bidding takes place, it is hence already known that the offered bundles can build feasible sets. As far as the actual bidding phase is concerned, for sake of simplicity, it is assumed that the bidding prices of the new auction are the actual original prices of the instance, hence the values that the models try to predict. Given that each carrier can receive up to one bundle, each set can consist of up to m bundles and will have a related bidding price matrix, as shown in table 1, which represents a situation where $m = 3$. Assuming that there are 10 requests with IDs from 1 to 10 that must be reassigned, a possible winning set could look as following.

Table 1 Example of a Bidding Price Matrix for a Feasible Set

Bundle Set			
Bundles	Bids Carrier 1	Bids Carrier 2	Bids Carrier 3
[3,4,5,6]	50	80	90
[1,2]	30	20	25
[7,8,9,10]	60	70	80

For each set of bundles, it must be assessed which carrier will acquire which bundle so that the total assignment cost is minimised.

This issue refers to the well-known generalized assignment problem [35]. In the generalized assignment problem, there are n jobs (the bundles in a set) that have to be assigned to m workers (the carriers). c_{ij} is defined as the cost of assigning job i to worker j (the bidding price). The set of jobs is defined as $I = \{1 \dots n\}$, and the set of workers is defined as $J = \{1 \dots m\}$. The generalised assignment problem aims at minimising the total cost required to assign each job to one worker. For this, the binary variables x_{ij} are introduced, taking the value of 1 if bundle i is assigned to carrier j, or 0 otherwise.

The generalised linear assignment problem aims at solving the following LP [35].

$$minimise \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$subject\ to \sum_{i \in I} x_{ij} = 1 \qquad \forall\, j \in J$$

$$x_{ij} = 1\ or\ 0 \qquad \forall\, i \in I, \qquad j \in J$$

This LP minimises the assignment cost of jobs to workers, the first constraint indicates that each job needs to be assigned to exactly one worker [35].

The generalized assignment problem also implies a constraint that the job assignments cannot exceed the number of resources available; in this study, since sets of only up to m bundles are built and m is constant, this constraint can be omitted. Note that a set can also consist of $< m$ bundles; in this case, not all carriers will receive a bundle. This is also a feasible solution.

The solution for this LP applied to very small matrices like the ones considered in this study can be found very quickly by the linear sum assignment function offered by the library *scipy.optimise* in Python [65].

This small LP is solved for each set in the bidding phase. Each set will consequentially have an assignment cost, these are then sorted and the set with the smallest assignment cost is the one that is considered to include the bundles that have won the auction. Customer requests will be hence reassigned grouped into the bundles that are contained in the winning set. As mentioned above, the same procedure is also conducted for finding the winning set with the framework described in [18]; this framework is able to generate bundles which are already able to build feasible sets with other generated bundles, hence there is no need of any feasibility check, as it happens in the regression model-based approach. The sets are therefore built, the original bidding prices are added, the LP is solved for each set and the set with the smallest assignment cost is the winning one.

5 Data Processing & Evaluation

5.1 Data Description

In the instances used to conduct this study, the alliance consists of three carriers operating in the federal state of Vienna, Austria. In total there are 74 customers, and each carrier submits five presumably unattractive customers to the customer pool, the central authority then creates the bundles and offers them to the carriers.

The only pieces of information known about the submitted customers are their geographical coordinates and two matrices reporting the distances and the travel times between the customers. The geographical locations of the depots of the carriers are also known. In total, 20 instances are considered in this thesis.

Figure 3 represents the original setting of one example instance. The blue markers represent the customers, whereas the markers with colours other than blue represent the depots of the carriers.

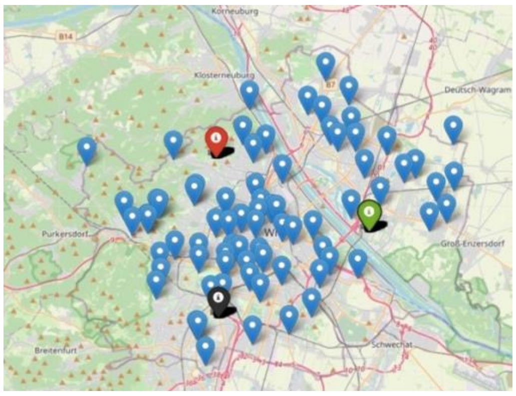

Figure 3 Geographical Representation of Customers and Depots

Historical bidding data can be thought as a table of possible bundles and respective historical bidding prices placed by the network members, as shown in table 2.

Table 2 Starting Situation of an Example Instance

Bundles	Bids Carrier 0	Bids Carrier 1	Bids Carrier 2
[0]	295	365	2176
[11]	1091	1018	2666
[0, 1, 11, 18, 25]	4567	5470	6181
[1, 11, 12, 29, 40]	5869	5507	6645
…	…	…	…

Starting from this initial situation and according to the available data, new features that numerically depict the bundles and have a positive correlation to the bidding prices must be engineered in order to train the regression model and predict prices for bundles prices in future auctions. The features should describe the bundles as accurately and effectively as possible, always taking the computational times into consideration. In the best case, a few and very simple to calculate features are sufficient to accurately predict bidding prices for future bundles.

5.2 Feature engineering

The section below describes the new engineered features. The assumption that attractive bundles may be compact, dense, and cheap to deliver is taken from [18], but the ideas for the implementation of these specific features are taken from literature, personal knowledge in transportation logistics and personal thoughts. The effectiveness of the engineered features will be tested in future steps. All features have been engineered in Python.

Number of Customers

As shown in table 1, a customer bundle is represented by none other than a list of customer IDs, it can be hence quickly determined how many customers there are in each bundle by counting the number of elements in each list. This is the first and simplest calculated feature, called `number_of_customers`.

Bundle Centroid

Each bundle can be also considered as a group of customers; since the coordinates of each customer are known, bundles can be represented as a set of points in a Euclidean space. As for a cluster of points in a two-dimensional space, bundles can also be represented by their centroids [31, Chapter 7]. These are calculated by taking the mean of all x and y coordinates of all customers belonging to a bundle. As a result, each bundle is described by just one point. A tuple representing the coordinates of the centroid of each bundle is temporarily added to the dataset for further calculations. An example of a bundle and its centroid is shown in figure 4, the blue markers represent the customers, whereas the green point marks the location of the centroid.

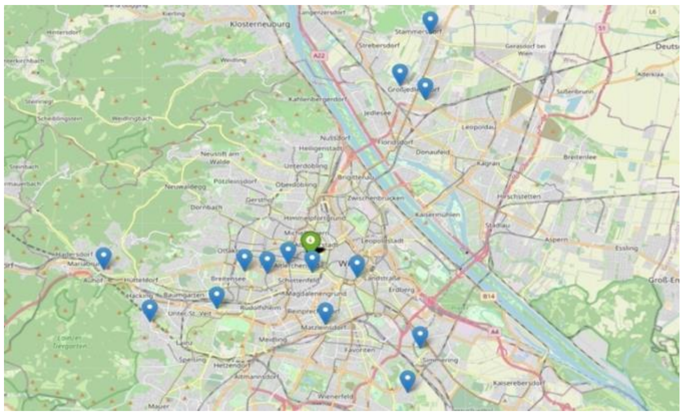

Figure 4 Example of Customer Bundle and its Centroid

Originally, working with the bundles' medoids, namely the most centrally located customer [40], was also taken into consideration. The idea was to save computational time by not calculating an extra point in the space, but to leverage from the existing ones, since the centroid is hardly ever a point in the data, whereas the medoid is by definition already contained [40]. However, to state which point is considered as medoid, the distances between all customers have to be calculated to determine the most centrally located one. Since this also requires computational time, and the bundle's centroids were still needed to compute further features, the medoid idea was directly discarded.

Sum of Squared Distances & Sum of Distances in Kilometres

The idea of k-means clustering, whose idea is finding clusters considering their compactness, namely the sum of squared Euclidean distances from every point in the cluster to a cluster representative, the centroid [25], can be also applied to the customer bundles, where the points in the cluster are the customers' requests. The objective function of k-means clustering aims at minimising this sum when finding clusters; in the case of the bundles, clusters are already given and for each of them the sum of squared distances from each customer to the centroid is calculated, following the idea that small values may indicate attractive bundles, since the customers are close to each other, and the bundle is hence very compact. Large values on the other hand indicate that customers in a bundle are far from each other, and it may not be so attractive for a carrier to deliver them as a single bundle. The sum of squared distances for each bundle is stored as a feature called `distance_sum_squared`. This idea can be further adapted to the purpose of this study. Since the instances contain real coordinates, the geographical distances between customers and centroids can be measured with an actual unit of measure and used as a real-world compactness measure. With the same lines of code and the same idea, the distance between each customer in a bundle and the centroid can be also calculated using the distance function of the Python built-in library *Geopy*. This function processes two points and respective tuples of real geographical coordinates and calculates the real distance in a standard unit of measure, in this case kilometres. The sum of real distances in kilometres from each customer in a bundle to its centroid used in this study is stored in the feature `distance_sum_[Km]`.

Distances between Centroid and Depots

Bundles' centroids may also be used to assess the average distance of each bundle to the depots of the carriers, assuming that bundles which are closer to the depots may be more of interest, and hence more attractive for a carrier, compared to some other ones whose centroid lays further away. This piece of information can be acquired in very short computational time, since only one distance must be calculated for each bundle. The distances from each depot of the carriers to all bundles' centroids are generated as three new features named `distance_from_depot_1`, `distance_from_depot_2` and `distance_from_depot_3`, indicating the distances to the centroids of all bundles from depots of carriers 0, 1 and 2 respectively. Including the depots in the bundle description is also used to effectively describe bundles which contain only one customer; when considering the pure inter-bundle compactness, these special kinds of bundles will reasonably have 0 values for all measures, it does not necessarily mean though, that these kinds of bundles are the most attractive ones.

Radius (from Gansterer & Hartl (2018))

Gansterer & Hartl (2018) [18] also develops descriptive features for the bundles for the objective function of their approach, one of them is the radius of a bundle. The same feature has been computed for this dataset as well. The authors consider the radius of a bundle as the average of the distances from each customer in a bundle to its centroid. This feature has been named `radius` in the dataset.

Radius (Geometrical)

A more classical geometrical concept of radius has also been calculated to describe the compactness of each bundle. The distance from the centroid to the furthest customer in a bundle is stored as new feature, implying that if this distance is small, the furthest customer in the bundle is still close to the centroid and hence the bundle is compact, if this distance is significant, the bundle will be loose and potentially unattractive. This measure is referred as `max_distance`.

Bundle Density

Gansterer & Hartl (2018) [18] also defines a concept of density for each bundle; however, since they consider a pick-up and delivery problem, they consider the density being the average direct travel distance of all requests divided by the maximum distance of all requests to the bundle's centroid.

Since this study does not consider such a problem, the idea of calculating bundles' density has been taken from the aforementioned paper, but the calculation approach is different. For the development of this dataset, the bundle density is calculated following the idea of the calculation of the population density of a city or Country. The population density is calculated as the number of inhabitants, or population size, divided by the unit of area [36]. In the case of a bundle, the number of inhabitants is represented by the number of customers in a bundle, whereas the area could be defined in many ways, since there are no legal boundaries of a bundle. For this feature, the surface of a bundle is considered as the area of the circle which includes all bundles in it; hence the radius r of this circle is represented by the distance of the furthest customer from the bundle's centroid. Then the area A of the circle is calculated by the given formula $A = \pi r^2$. Once this value is calculated for all bundles, since the number of customers in the bundles is already given as first described feature, it is sufficient to divide the latter by the area of the circle of each bundle to obtain the bundle density. This piece of information is stored in a column named `bundle_density`.

Standard Deviation

Since the centroid of the bundle is represented by the mean of the coordinates of all customer locations, a measurement of the compactness of a bundle can be represented by the standard deviation of the distances between the customers and the centroid. Since the standard deviation describes the dispersion of a set of values from the mean, as far as the bundles are concerned, a low standard deviation implies that the customers are very close to the centroid and hence the bundle is compact, a significant dispersion on the other hand is sign of a loose bundle. This feature is referred as `standard_dev` in the dataset.

Bundle Tour Estimation

Lastly, as for [18], a rough idea of the cost to deliver each customer in a bundle in terms of the solution of a Travelling Salesman Problem (TSP) may give an additional hint about the attractiveness of the bundles. The TSP describes the problem of visiting each of the foreseen points exactly once, with the shortest possible tour [26]. Since the TSP is a NP-Hard problem [26], solving it to optimality for each single bundle would vanish the competitive advantage in terms of computational time of alternative methods compared to the full bundle enumeration in the bidding phase. The cost of the TSP should be hence approximated taking solution quality and computational time into consideration. For this, two approaches have been selected for testing; the one which brings the best performances for the regression models will be chosen for the final dataset.

Regression-based Tour Length

Firstly, a regression-based tour length approximation method presented in Çavdar & Sokol (2015) [7] has been implemented. This feature is referred as `Tour_length_approx`. The approach does not take distances between customers into consideration, the TSP approximation is based on calculations between customer coordinates and the central cartesian axes built on the centroid of each bundle. This method considers the dispersion in each dimension, namely the standard deviation of the coordinates in both dimensions; the closeness to the centre in terms of the average distance of the customers to the central cartesian axes, and the dispersion around the central axes, considered as the standard deviation of the distances from the nodes to the central cartesian axes. Based on these measurements, the authors estimate the cost of a TSP with the following formula:

$$T \approx 2{,}791 \times \sqrt{n(cstdev_x cstdev_y)} + 0{,}2669 \times \sqrt{n(stdev_x stdev_y) \times \frac{A}{c_x c_y}}\,.$$

The formula has the following notation.

- T: tour length
- n: number of customers in a bundle
- $cstdev_x,\ cstdev_y$: standard deviation of the absolute distances of the nodes to the horizontal and vertical axis
- $stdev_x,\ stdev_y$: standard deviation of the nodes' horizontal and vertical coordinates
- A : area of the bundle, calculated with the method stated above

- $c_x,\ c_y$: average distance of the nodes to the central horizontal and vertical cartesian axes

Tour Length with Cheapest Insertion Heuristic

In the second place, a more classical TSP cost estimation has been calculated implementing a cheapest insertion heuristic for each bundle, since the distances between customers are known. The cheapest insertion heuristic iteratively inserts new customers in a graph calculating the cost of inserting the customer in every possible position of the existing route and picking the insertion position with the cheapest insertion cost [26]. In contrast to the first method, the cheapest insertion heuristic considers the actual distances between all customers, its solution is hence not an approximation as for the regression-based method, but the actual sum of distances of the found optimal solution.

As far as the runtime complexities of the two methods are concerned, the regression-based method outperforms the cheapest insertion heuristic. The first method in fact does not consist of a proper algorithm, its computational complexity in Python is given by the calculation of the distances from each customer to the central cartesian axes and above all by the calculation of the standard deviations needed for the formula. The Cheapest Insertion Heuristic is a proper algorithm and could be most efficiently executed in time $O(n^2 \log n)$ [26].
The used code for the cheapest insertion heuristic has three nested *loops*, it has therefore a runtime complexity of $O(n^3)$.
The cheapest insertion heuristic is hence the feature with the highest computational complexity of the whole dataset.
Both versions of the TSP cost approximation have been calculated for each bundle alone, and considering each of the single depots as the starting and ending point of each TSP. Including the depots in the TSP cost calculation highlights the different levels of attractiveness of the same bundle for the different carriers; for example, it may be assumed that bundles whose customers are close to each other, but generally very distant from the depot of one carrier are not as attractive as other bundles whose customers generally lay closer to the depot. The second reason for including the depots in the TSP calculations is to provide bundles with only one customer with a numerical description of the TSP other than 0.

5.3 Features Evaluation

5.3.1 Pearson's Correlation Coefficient

Since the features have been engineered according to assumptions and ideas from the existing papers, whose scopes however differ from the one of the presented methodology, it is still unknown whether they are able to describe the attractiveness of the past bundles. In other words, it is still not known how they are correlated to the bidding prices of the different carriers.

Calculating the correlation coefficients of all features with regard to the bidding prices represents a way of understanding if the assumptions made in the feature engineering phase are reflected when considering the bidding prices. For this, Pearson's Correlation Coefficient is used. Pearson's Correlation Coefficient is a way of measuring the relationship between two statistical variables X and Y, which is assumed to be linear; it hence measures the linear correlation strength of the considered variables [23 Chapter 6]. The correlation coefficient is given by their covariance divided by the product of the standard deviations of the two variables [13 Chapter 4]. The correlation coefficient is hence represented by the following relationship.

$$r = \frac{cov_{xy}}{s_x s_y}$$

Ideally, the absolute value of the correlation coefficients should be as close as possible to 1, to indicate a strong positive correlation between two variables. The closer to 0, the more uncorrelated two features are [23 Chapter 6]. The values of the Pearson's correlation coefficients for all variables with regard to the bidding prices are reported in tables 3, 4 and 5 below. The correlation coefficients for all created features have been sorted in a descending order. Since some features are calculated taking information about the depots of each single carrier into consideration, the correlation coefficients have been calculated with regard to the bidding prices of each carrier singularly. Correlations for features which include carrier information have been calculated regarding the bidding prices of the respective carrier only. The tables report the average correlation values of the features for all 20 instances.

Table 3 Feature Correlations Carrier 0

Feature	Correlation
Bids Carrier 0	1,000000
cheapest_insert_depot_0	0,885510
cheapest_insert	0,812964
distance_sum_squared	0,803919
Tour_length_approx_with_depot_0	0,800156
distance_sum_[Km]	0,791232
Tour_length_approx	0,765242
number_of_customers	0,755851
bundle_density	0,742186
standard_dev	0,590554
radius	0,580096
max_distance	0,536066
dist_from_depot_1	-0,054171

Table 4 Feature Correlations Carrier 1

Feature	Correlation
Bids Carrier 1	1,000000
cheapest_insert_depot_1	0,927401
cheapest_insert	0,874118
Tour_length_approx	0,859674
Tour_length_approx_with_depot_1	0,856256
distance_sum_[Km]	0,826611
distance_sum_squared	0,802072
number_of_customers	0,733959
bundle_density	0,715502
standard_dev	0,614350
radius	0,600777
max_distance	0,524052
dist_from_depot_2	-0,020447

Table 5 Feature Correlations Carrier 2

Feature	Correlation
Bids Carrier 2	1,000000
cheapest_insert_depot_2	0,939851
Tour_length_approx_with_depot_2	0,881408
Tour_length_approx	0,872513
cheapest_insert	0,872067
distance_sum_squared	0,835236
distance_sum_[Km]	0,833249
bundle_density	0,752626
standard_dev	0,709238
number_of_customers	0,706295
radius	0,691614
max_distance	0,619474
dist_from_depot_3	0,306131

It can be observed that almost all generated features have a positive correlation coefficient with the bidding prices of all carriers. The coefficients clearly show that the features which have the highest correlations to the bidding prices are the ones describing the bundles in terms of tour length. For all carriers, it can be observed that the tour length of a bundle estimated using the cheapest insertion heuristic and taking its respective depot as starting and ending point of the TSP, is the feature with the highest correlation to the bidding prices. Even not considering information concerning the depots of the carrier, the TSP estimation of the isolated bundle with the cheapest insertion has a very strong correlation to all bidding prices, being the second strongest feature for two carriers, directly after the TSP estimation including the depot. The TSP length approximation using the regression-based approach is also relatively strongly correlated to the bidding prices; in two cases, including the depot as starting and ending point increases the correlation value to the target variable. For carrier 1, the correlation values of the two variants are almost identical. Its performance in terms of ranking however is not as stable as the cheapest insertion heuristic; for carriers 1 and 2, this feature is the one with the highest correlation after the TSP calculated with cheapest insertion, whereas for carrier 0, another feature has a slightly higher correlation than the regression-based TSP. It is also worth noticing that the regression-based TSP with no depot information has a slightly higher correlation than its Cheapest Insertion equivalent for carrier 2. Features which describe the bundles in terms of compactness are also significantly correlated to the bidding prices for all carriers; the compactness measured as the sum of distances to the centroid also brings good correlation values, indicating that compact bundles are generally more attractive than loose ones. Describing a bundle in terms of its density taking the idea from the measurement of population density has also a positive correlation to the bidding prices for all carriers, with a correlation of over 0,7 for all cases. Other features do not bring the desired level of fitness to the bidding prices, the distance from the depots to the bundles' centroid for example has low correlations in all three cases. It must be also mentioned that the correlation ranking is similar for all carriers, but the actual correlation values also vary from carrier to carrier, even for the features which do not take any information about the depot into consideration and are hence equal for all carriers. Since the data is artificially generated, a possible natural explanation of this phenomenon is not possible at this stage. To give a better visual overview of the correlations between the different types of created features and the bidding prices, the scatter plots of three example features, one describing the bundles in terms of TSP solution, one in terms of compactness and one in terms of density are presented in figures 5, 6 and 7 below. The visualizations have been generated in the visualization software Tableau.

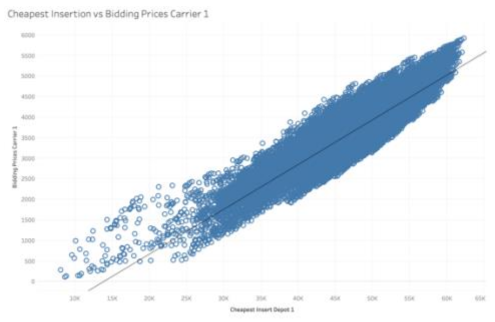

Figure 5 Correlation of Tour Length feature

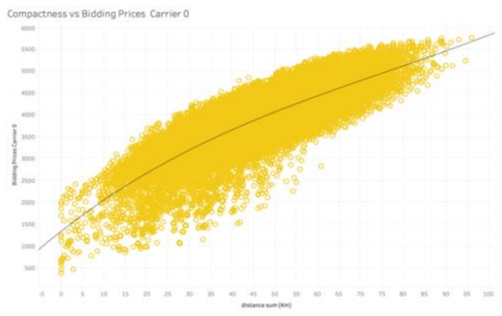

Figure 6 Correlation of Compactness feature

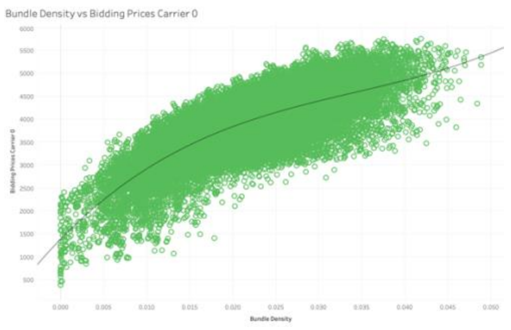

Figure 7 Correlation of Density feature

These examples describe only one randomly chosen feature per category for one random carrier in one instance. The very high correlation between the bidding prices and the TSP solved with cheapest insertion including the carrier depot is described by the shape of the data in the first scatter plot, whereas it can be observed how the features of the other two categories are directly proportional, but the scatter plot does not present a clear linear shape like in the case of the TSP, hence the positive correlation value, but not as high as for the first category.

5.3.2 Sequential Feature Selector

After that the features have been created and it has been stated that they are all differently, but positively correlated to the bidding prices for all three carriers, it is still unclear which ones really bring benefits in predicting these prices in terms of model accuracy, and since all studied models work differently, it may also be the case that some features are more beneficial than others when applied to different models. In general, the prediction accuracy of a model may not improve if more and more features are added to the model [37 Chapter 5]. To accurately understand which variables help improving the prediction accuracy in terms of r^2, a Sequential Feature Selector has been applied when fitting all models. The Sequential Feature Selector is a tool available in Python from the library scikit-learn [66], it starts with an empty set and iteratively adds one feature after the other into the model according to which ones bring the best performance improvement and evaluates the latter in terms of a scoring parameter after inserting each feature [67]. For this work, r^2 has been chosen as scoring parameter. When adding new features does not bring any further benefit to the model, the selection stops [37 Chapter 5]. The tool then reports the model best r^2 value together with the combination of feature which made the model achieve this value. The reason of the usage of such a tool is not only to understand which model achieves the best accuracy, but also which one requires the lowest number of features to be optimally trained, or which one requires features that are less expensive to compute than others.

5.4 Evaluation of the Models and Dataset

The Sequential Feature Selector evaluates the models in terms of r^2. For the purpose of this study, the models will be tested according to the criteria described above. Besides the comparison of the training times, the study on the other quality criteria have been supported by the Sequential Feature Selector. This will not only provide an indication on which features to choose for the final study after that the best regression model will be chosen, but will also allow a fair comparison of the models among the different criteria, since the tool makes sure that each machine learning algorithm is always optimally trained. In order to evaluate the model with the support of the Sequential Feature Selector, some instances are used as test set and some different ones are used for testing.

r^2

The Sequential Feature Selector has been applied on 7 different test instances for each of the models chosen for this study, since the features which have been calculated taking depot information of a specific carrier into consideration must be used only when calculating the bids of the corresponding member, the model fitting has been performed for each carrier singularly. The tables in the appendix report the best r^2 values and the list of features which let the model achieve the reported r^2 for each instance and each model. For each carriers the models have been divided into two tables, one reports the performance of the linear models, the other of the ensemble models. In total there are 126 r^2 values and groups of features to evaluate.

From the tables it can be clearly noticed that the application of different models has an influence in the best achievable r^2 for each instance, and that the list of features which enables each model to achieve the best scoring value also varies depending on which model is trained. There are however some similarities, as anticipated in the correlation analysis, the feature which has the highest correlation with the bidding prices, namely the cheapest insertion including the depot, contributes to an improvement of the r^2 values for all models in 100% of the cases.
Table 6 reports the average number of features needed from each model in order to achieve the best possible r^2 values among the tested instances.

Table 6 Average Number of Features required by each Model

Model	Average number of features
Linear Regression	8,0
Bayesian Ridge	8,142857
Elastic Net	8,428571
XGB Regressor	6,0
LGBM Regressor	6,142857
Gradient Boosting Regressor	6,571429

Elastic Net is the model which on average requires the highest number of features to achieve to be optimally trained, whereas the XGB Regressor needs the lowest number of features. In general, it can be observed how the linear methods on average must be trained with more features to reach the best accuracy score compared the ensemble algorithms. This may represent an advantage in choosing such a model, however, the significant different in training times between ensemble and linear model must also be taken into consideration. In addition, as stated above, despite the difference in the number of features needed by the different models, the most computational expensive one, namely the cheapest insertion, is the one always contributing to a r^2 improvement. If the difference in the number of features chosen by the Sequential Feature Selector is represented by variables which are computationally cheap to calculate, the variation of the training times between the two main regression methods treated in this study may not justify the choice of an ensemble method only for the lower number of features needed.

Despite the dissimilarities in the nature of the various regression models and the deriving difference in the features needed by each of the models to be optimally trained, the resulting average r^2 values derived from the tested instances do not significantly differ among the different models. Table 7 reports the average r^2 values derived by applying Sequential Feature Selector to all models for the tested instances.

Table 7 Average r^2 achieved by each Model

Model	Average $\boldsymbol{r^2}$
Linear Regression	0,785952
Bayesian Ridge	0,786372
Elastic Net	0,724430
XGB Regressor	0,760721
LGBM Regressor	0,775621
Gradient Boosting Regressor	0,761217

It can be noticed how ensemble and linear models produce very similar results in terms of r^2, nevertheless, the Bayesian Ridge is the model achieving the highest average r^2. All models were also able to achieve r^2 values higher than 0,9 in some instances. Since the predictors perform roughly equally in terms of r^2 and they require the most computationally expensive features to be optimally trained, looking at the training times may provide a better overview for preferring a model instead of another.

Training Times

The performance of the models in terms of training times is remarkably contrasting when comparing the different models. Table 8 reports the average training times for the three carriers of all models in one instance expressed in milliseconds. Since the size of all tested instances is identical, the training times of the models throughout the different instances are extremely stable.

Table 8 Average Training Time required by each Model

Model	Training Time (ms)
Linear Regression	123,546680
Bayesian Ridge	292,697350
Elastic Net	732,926448
XGB Regressor	34.003,606002
LGBM Regressor	3.594,304800
Gradient Boosting Regressor	165.912,721634

The table reports the critical difference in training times between the linear and the ensemble models. The simplest predictor, namely the linear regression, is also the one with the lowest training time; in general, all linear models can be trained within less than a second, whereas the Gradient Boosting regressor requires almost 166.000 milliseconds, slightly less than 3 minutes to be trained. It can be also noticed how the training times grow exponentially if compared to the linear regression. Compared to the simplest model, the Bayesian Ridge requires 2,3691 times more to be trained, the elastic net 5,9323 times more, the LGBM Regressor 29,0926, XGB Regressor 275,2288, and finally the Gradient Boosting Regressor is slower than the Linear Regression by a factor of 1.342,9152. Due to the extremely long training time of this last model and its non-outstanding performances compared to the other model, Gradient Boosting Regression has been directly discarded as optimal model choice from this moment of the study onwards.

Ability to recognise the most attractive bundles

Having acknowledge that the r^2 values achieved by the models is relatively similar, but the linear models require in some cases more than 1.000 times less training time than the ensemble models, at first sight it may be concluded that the first ones are more suited for this regression problem. However, as stated above, the accuracy throughout the whole dataset is not primarily of interest for this study, as the models should be able to predict the attractive bundles.
When considering the performance for predicting the best 500 bundles compared to the best 500 actual ones, the models behave as shown in table 9. The reported numbers are the average percentage of recognized attractive bundles for each of the three carriers. This measure has been tested on 7 different instances.

Table 9 Percentage of correctly-recognised Attractive Bundles

Model	% Attractive bundles
Linear Regression	54,3905
Bayesian Ridge	54,5048
Elastic Net	50,7238
XGB Regressor	53,5238
LGBM Regressor	52,4190

For this measure as well, the models tend to behave similarly, when considering the best 500 bundles, all regressors are able to classify more than the half of the predicted bundles correctly. This result is very promising for all predictors, considering that this performance measure refers to the classification of 500 out of the 32.767 bundles of an instance, namely the 1,53% of the whole dataset. Since the regressors behave similarly, the difference in training times still provides a driving factor for opting for a linear model instead of an ensemble one.

Rank accuracy – Spearman's ρ

As far as the rank preservation is concerned, the models behave differently from each other, as reported in table 10. This reported number are the Spearman's ρ average values of the tree carriers among the same tested instances of the measure above.

Table 10 Spearman's ρ achieved by each Model

Model	Spearman's ρ
Linear Regression	0,414450
Bayesian Ridge	0,432178
Elastic Net	0,404134
XGB Regressor	0,366029
LGBM Regressor	0,339750

The table shows how on average the linear models can better preserve the ranking than the ensemble models, since the latter reach a Spearman's ρ of below 0,4, whereas all linear models achieve results above 0,4, with the Bayesian Ridge being the model outperforming as far as this measure is concerned.

Ability to find the best possible winning set

With this measure the models are tested in their ability to find the bundles with successively lead to the cheapest possible bundle reassignment. The models have been tested on 8 different instances, and 500 best bundles per carrier have been chosen to form the feasible sets. Table 11 reports the assignment costs of the winning set of bundles after that the auction has been run after the application of each model.

Table 11 Result of the Winning Set after applying each Model

Linear Regression	XGB Regressor	LGBM Regressor	Bayesian Ridge	Elastic Net
4442	4402	4402	4442	4402
3319	3297	3115	4059	2872
3840	3840	3840	3840	3840
2573	2676	2772	2573	2573
4512	4512	4512	4512	4512
3959	2850	2850	3959	2850
3700	3945	3849	3700	3700
4896	4896	4896	4896	4300

Results show how the regression models differ from each other in terms of finding optimal assignment costs. In some instances, all algorithms find the same winning set with equal cost, whereas in other cases the ensemble models find different optimal solutions than the simpler regressors. It can be also clearly seen how more complex predictors do not necessarily imply better solutions than the linear methods, providing a hint that the choice of the latter is sufficient for the purpose of this study. It is worth noticing that the model which often outperforms the other regression models is the Elastic Net.

Since the scope of this study is finding the best possible model which is able to solve the bundle assignment problem as optimally as possible; the positive performances of the Elastic Net in finding the best winning set make the regressor an ideal candidate for the final choice.

5.5 Model Selection

The different quality criteria for choosing the best model show that each one has its strengths and weaknesses, ensemble predictors require less features to be optimally trained, but the training time is significantly higher than the linear regressors; all models classify more than 50% of the best 500 bundles correctly, but the linear ones can better preserve the rank, and as far as the last quality criterium is concerned, Elastic Net often finds the best assignment. Although Elastic Net performed well also when considering the other criteria, it never outperformed so clearly like in this last measure. In fact, when considering all other quality criteria, Bayesian Ridge always slightly performed better than any other model. To better understand why Elastic Net is able to find better winning partitions than other models, it is worth considering the same studied measures, this time applied to the extreme top part of the dataset, namely where the very best bundles should be found. To do so, the ability to find the most attractive bundles and the Spearman's ρ measurements are repeated taking only the best 50 bundles into consideration. Tables 12 and 13 report the values of the average percentage of recognised best bundles and the rank accuracy for the best 50 bundles respectively, comparing the Bayesian Ridge and the Elastic Net.

Table 12 Percentage of the correctly-recognised 50 most Attractive Bundles

Model	% Attractive Bundles
Bayesian Ridge	30,5714
Elastic Net	31,5238

Table 13 Spearman's ρ for the 50 most Attractive Bundles

Model	Spearman's ρ
Bayesian Ridge	0,150012
Elastic Net	0,376402

When taking only the ability of classifying and preserving the rank of the 50 most attractive bundles, the superiority of the Elastic Net is clearly noticeable. In fact, the two models behave relatively similar when considering the percentage of the recognised attractive bundles, but when taking the ranking into consideration, it can be clearly seen how the rank accuracy of the Bayesian Ridge drastically sinks compared to its performance for the best 500 bundles, while the Elastic Net is able to preserve its performance when switching from 500 to only the top 50 bundles. This may explain the outstanding performances of Elastic Net in finding the best bundles to best solve the instances, although the results of the other measurements were not always the best for this regression model. Indeed, the scope of this study is not finding the best model to predict all prices as accurately as possible, but the one which is able to work best in the top part of the sorted dataset, where the most attractive bundles are to be found. Finding the regression model which can best preserve the ranking of the very best bundles is of primary interest, since preserving the ranking is for the scope of this study more significant than accurately predicting the prices. If the ranking of the sorted original dataset is preserved correctly, the actual predicted prices are not influent for solving the bundle assignment problem, since the regression model only serves the scope of predicting the most attractive bundles out of which candidate solutions for the final auction are built, and a correct ranking is sufficient to recognise the most attractive bundles.
Since the Elastic Net fulfils this need, and due to its very efficient training time, it can be chosen as optimal regression model for this study.

5.6 Feature Choice for Elastic Net

After that the regression model has been chosen, the last step for defining the overall model for the auctioneer is to define which features should be computed when a new set of bundles must be evaluated before an auction. To achieve this, it can be looked back at the data derived by the Sequential Feature Selector for Elastic Net and choose only the features which bring an improvement in the model performances and discard all the others for the final model. Since some features are related to one single carrier, the features which bring benefits to the prediction performances have been evaluated for each single member separately. Tables 14, 15 and 16 report the relative percentage frequency of how often each feature brought benefits in terms of predicting performances among the tested instances. For sake of completion, the Sequential Feature Selector has been applied on five further instances on top of the ones already tested in the section above, for a total of 13 instances. If a feature always improved the performances of the regression model for all tested instances, the frequency will be 100%.

Table 14 Relative Frequency of the Model Performance Increasing Features Carrier 0

Sequential Feature Selector Elastic Net Carrier 0	
Feature	Relative Frequency (%)
cheapest_insert_depot_0	100,000000
standard_dev	92,307692
number_of_customers	84,615385
distance_sum_[Km]	84,615385
dist_from_depot_1	76,923077
max_distance	76,923077
cheapest_insert	76,923077
bundle_density	69,230769
Tour_length_approx_with_depot_0	69,230769
Tour_length_approx	53,846154
radius	15,384615

Table 15 Relative Frequency of the Model Performance Increasing Features Carrier 1

Sequential Feature Selector Elastic Net Carrier 1	
Feature	Relative Frequency (%)
cheapest_insert_depot_1	100,000000
dist_from_depot_2	100,000000
standard_dev	92,307692
distance_sum_[Km]	84,615385
max_distance	76,923077
Tour_length_approx	76,923077
number_of_customers	76,923077
cheapest_insert	61,538462
bundle_density	61,538462
Tour_length_approx_with_depot_1	53,846154
radius	15,384615

Table 16 Relative Frequency of the Model Performance Increasing Features Carrier 2

Sequential Feature Selector Elastic Net Carrier 2	
Feature	Relative Frequency (%)
cheapest_insert_depot_2	100,000000
standard_dev	92,307692
max_distance	84,615385
distance_sum_[Km]	76,923077
Tour_length_approx	76,923077
Tour_length_approx_with_depot_2	76,923077
number_of_customers	69,230769
cheapest_insert	69,230769
bundle_density	53,846154
dist_from_depot_3	46,153846
radius	30,769231

From this percentage values it can be chosen which features it is worth keeping for the final model and which ones can be discarded. For example, both the cheapest insertion heuristic and the regression-based tour length approximation aim at describing a bundle in terms of TSP solution, hence it is sufficient taking only one measure for this description. As anticipated by the correlation analysis, the cheapest insertion heuristic including the depot is the feature with the highest correlation to the bidding price; this is confirmed by the Sequential Feature Selector, which identifies this feature as always improving the prediction performances in 100% of the tested instances and for all carriers. It can be also noticed how the information related to the depot is crucial in order to make this feature extremely powerful in predicting bundle prices. In fact, while the cheapest insertion heuristic including the depot in the solutions improves the r^2 in 100% of the cases, the cheapest insertion describing the single bundle improves the model performance in 76,92% of the cases for carrier 0, while for the other two carriers its frequency stays below 70%. For these reasons, it is worth sacrificing computational time to calculate the cheapest insertions with depot information for all three carriers in the feature calculation phase, keep this feature for the final model, and discard the regression-based tour length approximation. It is worth noticing how other features tend to contribute to a performance improvement in almost the totality of the tested instances, like the bundle density expressed in terms of standard deviation, which is added to the Elastic Net regressor by the Sequential Feature Selector in more than 92% of the cases for all carriers; it is hence a reasonable decision to keep this feature for the final model. Bundle compactness expressed in terms of sum of distances from the centroid to each customer also brings positive effects to the predictions in most of the cases, more than 84% for the first two carriers, and in roughly 77% of the cases for carrier 2. Since the calculation of the centroid of each bundle is already needed for the calculation of the Standard Deviation feature, this feature can be included with very little computational effort. For the same reason, the distance from the centroid of each bundle to each of the three depots can be calculated extremely quickly, since it is only necessary to compute one distance per bundle, and this feature is partly extremely powerful, like in the case of carrier 1, where this kind of feature contributes to a model performance improvement in 100% of the cases. There are also good reasons to keep the number of customers in the final model, since the feature can be calculated within fractions of a second in *Pandas*, and for its effortless computational complexity and the ease of its concept, it provides good results in the improvement of the regression model, namely in almost 77% of the cases for carriers 0 and 1, and almost 70% of the instances for carrier 2. The bundle density is related to the features which have already been chosen for being part of the final model, since it is calculated as the number of customers divided by the area of a bundle, whose radius is simply described as the distance to the furthest customer from the centroid, a measure which has already been computed for other chosen features such as the standard deviation. The bundle density contributes to an improvement of the r^2 values in roughly 69%, 62% and 54% of the cases for the three carriers respectively, but the added computational effort for computing this extra feature is only a multiplication for the bundle area calculation and a division with the already present number of customers in a bundle, operations which can be performed by

Pandas extremely efficiently within fractions of a second. Lastly, it is worth mentioning how the new concept of radius for this study, namely furthest distance of a customers from the centroid, also improve the model performances in the majority of the cases. This feature improves the r^2 in about 77% of the cases for carriers 0 and 1, 85% for carrier 2. It can be noticed how some other features almost do not bring benefits to the model, such as the radius from [18], which for all carriers is the useless feature of the set; for this reason, this feature is directly discarded independently from its computational complexity.
For all the reasons stated above, the following features are chosen for the final model.

- Cheapest insertion including the depot
- Standard deviation
- Sum of distances from centroid in Kilometers
- Furthest customer from centroid
- Distance of from each depot
- Number of customers
- Bundle density

6 Comparison of Machine Learning Approach vs. Genetic Algorithm

The final model for generating attractive feasible bundles out of a new customer pool consists of the generation of all bundle possibilities, the calculation of the chosen features described above, the prediction of the bundle prices for each carrier, the sorting of the dataset according to the predicted bidding price, the selection of the best n bundles for each carrier, and the construction of all feasible set of bundles according to the definition described in the introduction. The chosen machine learning model is the Elastic Net Regressor. To save computational time when applying the whole model, it is possible to exclude the training of the regression model when generating new bundles by letting the algorithm use already precomputed regression coefficients to predict new bidding prices.
To do so, several historical training sets with all the calculated features are combined, and the Elastic Net is then fitted to this resulting dataset, and the resulting regression coefficients and the point of intercept are extracted. This procedure has been done using all the 20 considered instances together, in order to make the extracted regression coefficients as robust as possible. Since there are features that only relate to one single carrier, and as

detected by the Sequential Feature Selector, features are sometimes not perceived equally by the different carriers, the regression coefficients and the points of intercept have been extracted for each single carrier separately. The regression coefficients and the point of intercept are used to predict the bidding prices for each bundle, replicating the procedure of the *predict* function of scikit-learn; the predicted bundle price is given by the dot product of the feature values and the regression coefficients, adding the point of intercept. Note that this procedure with the regression coefficient has been used only within computational study, since one the scopes of the latter is measuring the runtime of the bundle generation of both the procedure proposed in this thesis with the already given Genetic Algorithm, the runtime comparison of the different regression models has already been performed. The usage of the regression coefficients for predicting the bundle prices allows a direct running time comparison between the feature generation and the extraction of the feasible attractive bundles of the presented approach and the Genetic Algorithm, independently of the chosen regression model. To finalize the model, it must first be stated how many attractive bundles after the prediction should be considered in order to build feasible sets. To do so, the performances of the regression model in terms of finding the best winning set considering different n best bundles have been compared to the performance of the Genetic Algorithm, integrating the new code generated for the presented approach into the existing framework.

6.1 Choice of n best bundles

The number of bundles that allows the construction of feasible sets depends on how many best bundles are picked per carrier, after that the dataset has been sorted by the predicted price per customer. As mentioned in the introduction, a set of bundles can be considered feasible if and only if no customers are missing; considering a very few number of best bundles may decrease the computational time of the whole procedure, since the number of bundles which are able to build feasible sets will be presumably very low, but the possibility that within the very best bundles not all customers are present, and hence no feasible set can be built is also very likely. Table 17 reports the values of the best possible winning sets of the chosen regression model compared to the solution of the Genetic Algorithm as well as the pool size of best feasible bundles which enabled the model to come up with a solution. After the application of the regression model, 50, 100, 200, 500, 750 and 1000 best bundles per carrier have been selected, to study the variation in the solution quality as well as in the size of the bundle pool. It is already known that the Genetic Algorithm always produces 100 attractive feasible bundles. Each instance with varying n is directly compared with the solution of the Genetic Algorithm to state how many best bundles the regression model requires to cope with the performance of the existing benchmark. This study has been conducted for all considered instances. A 'worst-case' set of bundles has been added to all instances, namely the original requests that each carrier placed in the customer pool before the attractive bundle generation. If no further feasible attractive bundles can be found, or if the solution quality of other sets of bundles is worse than the original assignment, the solution of the instance will be the original assignment.

Table 17 Feasible-best Bundles' Pool Size & Winning Set Value for different values of n and Genetic Algorithm

Instance	EN $\boldsymbol{n}$ best bundles /GA	Feasible bundle pool size	Winning Set Value
0	50	7	2676
0	100	14	2676
0	200	55	2676
0	500	321	2622
0	750	592	2622
0	1000	876	2622
0	GeneticAlgorithm	101	2676
1	50	64	3169
1	100	129	3169
1	200	257	3169
1	500	769	3169
1	750	1372	3169
1	1000	1927	3169
1	Genetic Algorithm	101	3169
2	50	3	3201
2	100	32	3192
2	200	98	3180
2	500	314	3099
2	750	474	3099
2	1000	668	3099
2	Genetic Algorithm	100	3180
3	50	3	4318
3	100	13	4318
3	200	75	4318
3	500	344	4318
3	750	644	4318
3	1000	941	4318
3	Genetic Algorithm	100	4318
4	50	6	3746
4	100	41	3746
4	200	118	3430
4	500	568	3317
4	750	862	3317
4	1000	1120	3317

4	Genetic Algorithm	101	3430
5	50	3	4306
5	100	3	4306
5	200	33	3530
5	500	151	3530
5	750	295	3336
5	1000	506	3336
5	Genetic Algorithm	101	3530
6	50	3	3664
6	100	3	3664
6	200	18	3255
6	500	356	3075
6	750	954	3075
6	1000	1496	3000
6	Genetic Algorithm	100	3180
7	50	3	4132
7	100	4	4132
7	200	50	3937
7	500	142	3807
7	750	278	3807
7	1000	423	3807
7	Genetic Algorithm	100	3914
8	50	3	4621
8	100	48	4275
8	200	120	4087
8	500	322	4087
8	750	543	3895
8	1000	805	3895
8	Genetic Algorithm	101	3568
9	50	3	3516
9	100	3	3516
9	200	61	3516
9	500	344	3289
9	750	590	3289
9	1000	1089	3245
9	Genetic Algorithm	101	3245
10	50	3	3777
10	100	3	3777
10	200	26	3617
10	500	264	3081
10	750	342	3081
10	1000	454	3081
10	Genetic Algorithm	100	3341
11	50	3	3501
11	100	3	3501
11	200	7	3066
11	500	302	3066
11	750	656	3066
11	1000	1125	3066
11	Genetic Algorithm	101	3066
12	50	3	3507

12	100	3	3507
12	200	11	3507
12	500	88	3507
12	750	284	3507
12	1000	450	3507
12	Genetic Algorithm	101	3507
13	50	3	4197
13	100	26	4197
13	200	62	4197
13	500	324	3843
13	750	554	3843
13	1000	778	3843
13	Genetic Algorithm	101	3970
14	50	3	3444
14	100	3	3444
14	200	3	3444
14	500	136	3187
14	750	310	3187
14	1000	517	3154
14	Genetic Algorithm	102	2584
15	50	3	5073
15	100	3	5073
15	200	3	5073
15	500	3	5073
15	750	3	5073
15	1000	5	4915
15	Genetic Algorithm	101	3475
16	50	3	3392
16	100	36	3392
16	200	138	3376
16	500	470	3376
16	750	946	3304
16	1000	1432	3304
16	Genetic Algorithm	100	3392
17	50	3	3429
17	100	3	3429
17	200	28	2702
17	500	266	2702
17	750	664	2702
17	1000	1096	2702
17	Genetic Algorithm	100	2702
18	50	3	3666
18	100	16	3666
18	200	55	3666
18	500	223	3666
18	750	606	3666
18	1000	958	3666
18	Genetic Algorithm	100	3666
19	50	3	3611
19	100	18	3217
19	200	77	3217

19	500	521	3217
19	750	1152	3217
19	1000	1703	3217
19	Genetic Algorithm	101	2999

If considering only the best 50 attractive bundles per carrier to build feasible sets, it can be clearly seen how more than the half of the instances could not be solved if the original solution before the auction would not automatically belong to the feasible bundles set. A pool size of 3 bundles implies that no other bundles other than the original ones are able to build feasible sets, hence applying the whole algorithm would be pointless, since after all the computations, the solution would be equal to the original one. On the other hand, it must be also noticed, that 3 instances have a bundle pool greater than 3 after considering only 50 bundles, implying that the algorithm already found alternative solutions other than the original one. Table 18 reports how many instances experienced an improvement in solution quality after increasing the number of considered best bundle per carrier.

Table 18 Number of Instances whose solutions have been improved after increasing n

n Best Bundles	n. of Improvements
100	3
200	11
500	9
750	3
1000	4

When doubling the number of best bundles considered after the bundle price prediction, considering 100 best bundles, the number of non-solvable instances, if not with the original solutions, drastically reduces. However, many instances are still not solvable without the 'worst-case scenario', and sometimes the pool sizes include new bundles, but the value of the best set does not change, implying that returning the requests back to the original carriers is still the best solution. For other instances on the other hand, it can be appreciated how the increasing size of the bundle pool brings to an improvement of the solution quality, meaning that there are more feasible attractive bundles than the original solution; this can be seen in instance 2, 8 or 19, where considering 18 best bundles instead of 3 improves the solution from 3611 to 3217. Nevertheless, in only three instances out of 20 an improvement in solution quality is visible. When considering 200 instead of 100 best bundles per customers to form feasible sets, the number of per-se unsolvable instances becomes very small, the bundle pool size increases, but more than the half of the considered instances experience an improvement in solution quality, in fact, in 11 cases the solution quality is better than when considering only 100 best bundles. This implies that including more and more bundles per customer, increases the bundle pool on the one hand, but increases the chances of finding better sets. This is further confirmed when considering 500 best bundles per carrier; in this case, 9 instances out of 20 return a better solution than when considering 200 best bundles. When further increasing the number of best considered bundles however, the scenario slightly changes, now only three instances have a better solution than before, and when considering 1000 best bundles, hence when the best bundles pool consists of 3000 ones, only in 4 cases the solution could be further improved. These numbers reveal that increasing the size of the best bundles pool often brings to an improvement of the solution quality at the beginning, but then further less attractive bundles do not bring significant benefits.

In addition, the improvement factor for varying n best bundles should also be taken into consideration; for this, the average percentage solution improvement of all instances when augmenting the number of initially considered best bundles has been calculated, the results are reported in Table 19.

Table 19 Percentage of Solution Improvement of the Instances for increasing n

n Best Bundles	Average % improvement
100	0,933991195
200	4,272348062
500	2,693174423
750	0,616313725
1000	0,396340073

It can be clearly seen how switching from 100 to 200 best bundles brings the major average improvement in percentage for all instances; nonetheless, increasing n to 500 best bundles improves the solution on average by further 2,69%. Further increase in n do not bring any significant improvements. This can be effortlessly visualized in figure 8. The chart represents the sum of average percentage improvements with increasing n best bundles.

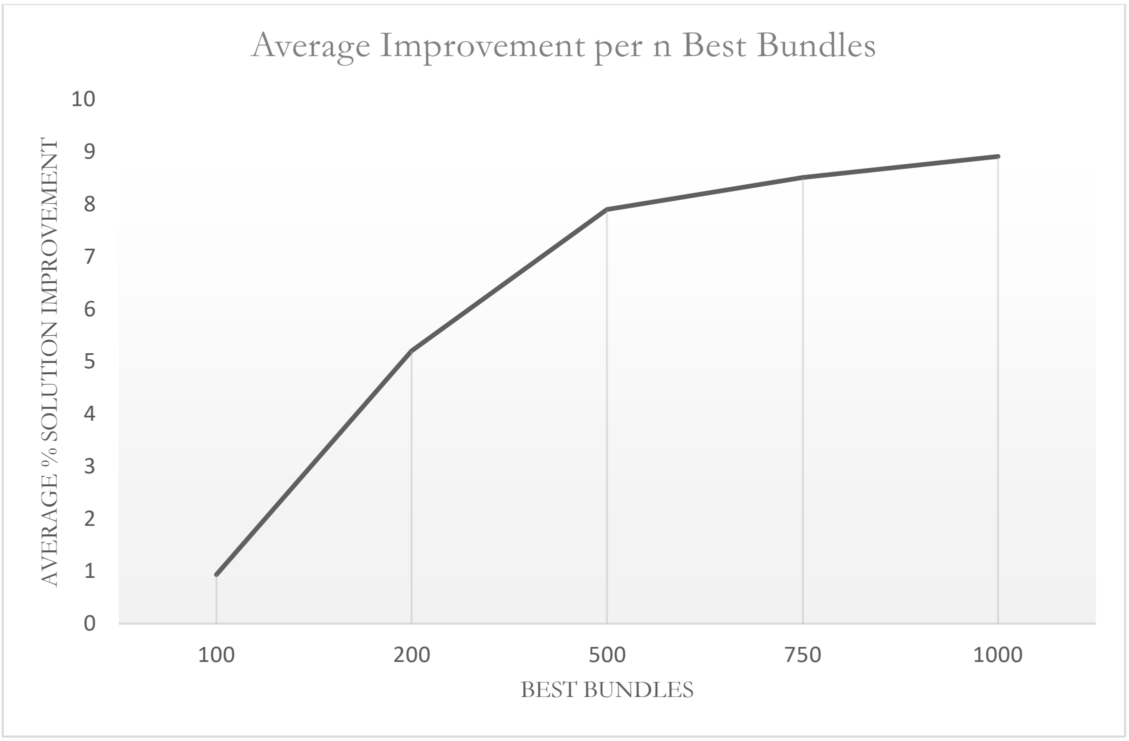

Figure 8 Average Percentage Improvement Curve for Increasing n

The curve clearly shows how the solutions steadily improve up to 500 best bundles, until the curve flattens, this implies that it is worth considering up to 500 best bundles to build feasible sets, since it can be assumed that the solutions are on average better than with sets built on starting pools of 200 best bundles per carrier but, considering $n > 500$ does not bring any further significant improvement.

Furthermore, the feasible-best bundle pool size has also to be considered, since this influences the computational time of the whole process. Table 20 reports the average feasible-best bundle pool sizes out of all instances when considering each studied number of best bundles per carrier after the bidding price prediction. The third column reports the pool size increase factor with regard to the previous n best bundles step.

Table 20 Average Feasible-best Bundles' Pool Size & Enlargement Factor for Increasing n

$\boldsymbol{n}$ Best Bundles	Average Pool Size	Pool Enlargement Factor
50	6,4	-
100	20,2	3,15625
200	64,75	3,20544554
500	311,4	4,80926641
750	606,05	1,94621066
1000	918,45	1,51546902

The average bundle pool size experiences significant enlargements with the increasing number of considered best bundles per carrier. When up to 200 best bundles are considered, the average bundle pool size is smaller than the one considered by the Genetic Algorithm, whereas when considering 500 best bundles and upwards the pool size skyrockets. The analysis of the effects of increasing n best bundles revealed that the solutions improve significantly with $n = 500$, this would imply an average bundle pool size of 311,4, which is almost 5 times larger than the pool built on the 200 best bundles. To understand whether this may compromise the running time of the whole process, an analysis on the variation of the running times of the auction process and winner determination process is conducted. Table 21 reports the running times of the bidding phase and of the winner determination for all instances and considered n best bundles.

Table 21 Running Times for Bidding & Winner Determination for different values of n and Genetic Algorithm

Instance	EN $\boldsymbol{n}$ best bundles /GA	Runtime Bidding	Runtime Winner Determination
0	50	0,102513	0,034438
0	100	0,148881	0,051251
0	200	0,589238	0,097654
0	500	3,504832	0,190361
0	750	6,767382	0,162610
0	1000	9,998785	0,244080
0	Genetic Algorithm	1,206343	0,101044
1	50	0,746952	0,107011
1	100	1,572485	0,168867
1	200	3,303166	0,190005
1	500	9,780774	0,203515
1	750	16,641416	0,349627
1	1000	24,873133	0,434436

1	Genetic Algorithm	1,009394	0,117837
2	50	0,029838	0,045163
2	100	0,386419	0,034331
2	200	1,054712	0,040295
2	500	3,813429	0,083633
2	750	5,489330	0,150858
2	1000	7,912376	0,292706
2	Genetic Algorithm	1.041722	1,152036
3	50	0,029545	0,087474
3	100	0,121848	0,085558
3	200	0,866045	0,856927
3	500	4,224689	0,129039
3	750	7,212685	0,210619
3	1000	11,185489	0,242957
3	Genetic Algorithm	1,109804	0,335982
4	50	0,089309	1,093371
4	100	0,425162	0,040169
4	200	1,380552	0,057897
4	500	7,071213	0,223545
4	750	9,876173	0,304413
4	1000	13,206844	0,273875
4	Genetic Algorithm	1,155031	0,072935
5	50	0,031962	0,018372
5	100	0,030629	0,076225
5	200	0,499456	0,077478
5	500	1,814540	0,149199
5	750	3,898076	0,171253
5	1000	6,391513	0,220582
5	Genetic Algorithm	1,194048	0,084619
6	50	0,029141	0,073611
6	100	0,030367	0,086837
6	200	0,240297	0,041980
6	500	4,026330	0,148429
6	750	11,296577	0,323217
6	1000	17,262951	0,331931
6	Genetic Algorithm	1,028025	0,064366
7	50	0,029701	0,055059
7	100	0,045888	0,024186
7	200	0,636533	0,115846

7	500	1,644680	0,092414
7	750	3,692092	0,138265
7	1000	5,005581	0,218915
7	Genetic Algorithm	1,054660	0,066714
8	50	0,030346	0,111976
8	100	0,554280	0,081911
8	200	1,382931	0,085533
8	500	4,096567	0,128680
8	750	6,868336	0,202770
8	1000	9,890025	0,261241
8	Genetic Algorithm	1,080345	0,081976
9	50	0,032366	0,086017
9	100	0,031828	0,055685
9	200	0,947667	0,091795
9	500	4,020680	0,182157
9	750	7,024912	0,270379
9	1000	13,179972	21,745357
9	Genetic Algorithm	1,132958	0,058953
10	50	0,033166	0,100077
10	100	0,031614	0,101458
10	200	0,408896	0,206822
10	500	2,815920	0,138822
10	750	3,991750	0,435843
10	1000	5,657592	0,172052
10	Genetic Algorithm	1,162351	0,089550
11	50	0,121116	0,086001
11	100	0,029918	0,167795
11	200	0,069322	0,020568
11	500	3,462877	0,118072
11	750	8,167918	0,210797
11	1000	12,908495	0,318743
11	Genetic Algorithm	1,089403	0,101044
12	50	0,032401	0,091053
12	100	0,029951	0,018751
12	200	0,148349	0,051470
12	500	1,171004	0,103186
12	750	3,305224	0,147963
12	1000	5,077821	0,158676
12	Genetic Algorithm	1,090753	0,058673

13	50	0,030557	0,081085
13	100	0,271470	0,092050
13	200	0,654003	0,126206
13	500	4,065536	0,128996
13	750	6,606134	0,212244
13	1000	9,666402	0,209202
13	Genetic Algorithm	1,921175	0,173209
14	50	0,047120	0,057523
14	100	0,030631	0,087395
14	200	0,048835	0,097996
14	500	1,399440	0,116965
14	750	3,434272	0,078184
14	1000	5,903776	0,209644
14	Genetic Algorithm	1,063834	0,068587
15	50	0,030824	0,116500
15	100	0,030133	0,027916
15	200	0,027563	0,081511
15	500	0,028831	0,050824
15	750	0,029253	0,048227
15	1000	0,046742	0,088092
15	Genetic Algorithm	1,154673	0,146545
16	50	0,026955	0,057851
16	100	0,291400	0,018095
16	200	1,365389	0,093873
16	500	4,948384	0,118579
16	750	9,469863	0,151469
16	1000	14,546157	0,205001
16	Genetic Algorithm	1,065736	0,085091
17	50	0,027235	0,038001
17	100	0,027212	0,014425
17	200	0,259852	0,029434
17	500	2,372467	0,066527
17	750	6,421924	0,103988
17	1000	10,649934	0,161618
17	Genetic Algorithm	1,178284	0,068417
18	50	0,027345	0,025172
18	100	0,159241	0,017355
18	200	0,562317	0,029227
18	500	2,236021	0,080326

18	750	5,879341	0,092313
18	1000	9,128290	0,151619
18	Genetic Algorithm	1,342298	0,053873
19	50	0,027964	0,017642
19	100	0,157375	0,023567
19	200	0,714622	0,036566
19	500	5,073695	0,100903
19	750	10,922360	21,954228
19	1000	17,293613	52,980389
19	Genetic Algoritm	1,006419	0,084173

The running times do not experience significant variations with the increasing n best bundles, although the numbers slightly increase when increasing n. Table 22 reports the average running times for bidding and winner determination of all instances with all possible increasing n.

Table 22 Average Running time of Bidding & Winner Determination for different Values of n

$\boldsymbol{n}$ Best Bundles	Average Runtime Bidding	Average Runtime Winner Determination
50	0,08	0,12
100	0,22	0,06
200	0,76	0,12
500	3,58	0,13
750	6,85	1,29
1000	10,49	3,95

The averages of the runtimes of the bidding phases show that bundle pool size and runtime for bidding are positively correlated, whereas as far as the winner determination is concerned, its running time seems to be independent on the feasible-best bundle pool size up to 500 best bundles, after this boundary, the running time increases as well. In every case, the running times are in absolute values efficient, apart from the case of 1000 best bundles, the sum of the running times of both steps stays below 10 seconds. Considering bundle pools made of 500 best bundles per carrier slows down the runtime of the bidding phase by almost 5 times compared to 200 bundles, but in absolute values the runtime remains manageable. According to the average improvement curve and considering that a large feasible-best bundle pool does not significantly slow down the running time of the bidding process, it can be concluded that selecting $n = 500$ is the best choice for the scope of this study.

6.2 Runtime Comparison

The final regression model uses 500 best bundles per carrier to construct feasible sets. As far as the runtime comparison of the bundle generation approaches is concerned, the two algorithms have been applied to all 20 considered instances and the running times have been extracted. The running times expressed in seconds of the two approaches are presented in table 23.

Table 23 Running Time Comparison of the two Approaches

Instance	Regression Model Approach	Genetic Algorithm
0	369,1589498	29,57036327
1	371,6046634	28,58620914
2	370,3192576	27,66807321
3	371,1045674	30,2087037
4	363,3090003	28,78536869
5	366,3798671	30,24629144
6	369,4167036	28,75963512
7	370,1900364	30,25718478
8	372,4060662	29,05922398
9	362,271004	28,27399371
10	362,7168789	29,62923289
11	370,9454699	29,34155391
12	362,5039806	28,28933318
13	363,9000103	28,42990875
14	367,9944328	29,24233868
15	371,3369544	28,40571918
16	359,554863	24,78009645
17	359,2637531	25,68611899
18	358,4440302	24,60742282
19	359,4329124	23,91247534

Unsurprisingly, the running times are relatively stable for both approaches, since the number of customers in the pool and the features to generate do not change. In terms of computational time, it can be clearly acknowledged how the Genetic Algorithm significantly outperforms the methodology proposed in this thesis. On average, an instance can be solved in 28,1896 seconds using the Genetic Algorithm, compared to the average 366,1127 seconds needed by the regression model approach. The average running time hence increases by a factor of 12,98872. The reasons for the significantly weaker performances in terms of running time of the regression model approach are related to the nature of the created algorithm. Firstly, the running time of the algorithm highly depends on the chosen features to train the regression model with. For this choice, the Sequential Feature Selector has been utilized; this evaluates the effects of a determined feature on a regression model exclusively taking the prediction performance into consideration. Since this is a crucial factor for a successful prediction model, the features which brought most benefits to the model have been chosen for the final algorithm, although it has already been acknowledged that the features which bring more benefit to the regression model are also the ones which are more computationally expensive to calculate. This is the case of the cheapest insertion including the depot, since the heuristic itself already has an exponential runtime complexity, and the procedure has to be repeated for all carriers, due to the depot information changing each time. The second main reason for the higher computational time must be found in the nature of a regression model itself. A machine learning model learns from historical data and predicts a target variable, if this newly generated data has to be used for further applications, the data has to be further processed. In contrast with the Genetic Algorithm, which is able to directly generate attractive feasible bundles, it is possible to almost immediately state the attractiveness of the bundles according to the predicted price per customer, but a regression model is not able to check the feasibility of the solutions according to the needs of the bundle generation problem. Therefore, a feasibility check must be undertaken before the attractive bundles are offered to the carriers in the auction phase. It must be therefore acknowledged, that the regression model approach will always require a step more than the Genetic Algorithm for the solution of the bundle generation problem. A third issue with the proposed regression model approach arises with the definition of best bundles after that bidding prices have been predicted. As described above, the predicted bidding price per customer is calculated, the dataset is sorted ascendingly according to this value, and the best 500 bundles per carrier are selected. Since the considered instances always consist of three carriers, the algorithm constantly processes up to 1500 bundles for the feasibility check, it may be that the same bundle is in the best 500 for more than one carrier, in this case it is considered only once. Nevertheless, the number of bundles which form feasible set is unknown before the feasibility check and varies among the instances, whereas the genetic algorithm is designed in a way that always provides only 100 bundles for the auction phase, which are able to build feasible solution sets, as shown in table 17.

The table clearly shows the volatility of the bundle pool size of the regression model approach compared with the Genetic Algorithm; on average, the regression model approach provides a bundle pool of 311,4 bundles, which is more than three times higher than the Genetic Algorithm. Nonetheless, the reported values for the regression model approach confirm the absolute necessity of the feasibility check after the best bundles have been selected according to the predicted price per customer. Since the original best bundles pool consists of 1500 best bundles, an average feasible best bundles pool of 311,4 confirms that the majority of the originally found best bundles do not build feasible solutions, and hence can directly be discarded before the auction takes place. In fact, on average only 20,76% of the originally considered best bundles can be considered as feasible best bundles. Lastly, it must be acknowledged that although the final number of feasible best bundles does not have a significant impact on the running time of the bidding procedure, the more bundles must undergo the feasibility check, the slower the latter is.

6.3 Evaluation of the two models and managerial implications

Both presented methods achieve the purpose of the bundle generation problem, they are able to process a set of single requests and generate bundles which form feasible solutions, drastically reducing the problem size. The way how this is done however significantly differs between the two approaches.

The Genetic Algorithm is a metaheuristic, which every time produces a set of solutions which is manipulated by different operators [18]. The quality of a candidate solution is evaluated by a fitness function, which numerically describes the created bundles through some quality criteria. The proposed framework is conceptualized in a way so that it produces only a limited number of bundles in the final bundles pool, which leads to feasible solution sets.

The alternative method proposed in this thesis works in the opposite way. Since a regression model has to be trained, the bundles first have to be numerically described by some quality criteria, then the candidate solutions can be generated. Since the numerical features are calculated on the complete set of bundles, all of them are equally evaluated before the regression model is applied. Both frameworks work under incomplete information of the carriers, hence the features calculated process exclusively pieces of information regarding the requests in the pool and the carriers' depot. The number of features calculated for the new alternative approach does not significantly differ from the ones used for the fitness function of the Genetic Algorithm; in fact, the latter uses only 5 features, whereas the regression model in the end processes 7 features. These have been selected out of a feature pool consisting of different measures; the final feature choice has been made by letting a Sequential Feature Selector pick only the features that more often bring improvements to the prediction performances of the chosen model. Although Gansterer & Hartl (2018) [18] states that building bundles with clustering algorithms such as the k-means approach did not bring to the desired results, evaluating bundles using its objective function, namely the sum of distances from the centroid, revealed to have a positive influence on the prediction performance on the chosen regression model, so that this feature has been included in the final feature set. In general, both the fitness function of the Genetic Algorithm and the features calculated for the regression model approach take bundle compactness, density, and tour length into consideration. How this is done, however, can be differently and freely interpreted. For example, measuring the compactness taking the idea from a statistical measure such as the standard deviation revealed to have a positive influence on the regression model predicting performances, while this is not considered by the Genetic Algorithm framework. Other features such as the density have been calculated differently in the two approaches, Gansterer & Hartl (2018) [18] describes the density as the average direct travel distance to all requests in a bundle divided by the furthest distance of a request from the bundle's centroid. The alternative approach considers the density taking the idea from how population density of an area is calculated, namely the number of inhabitants divided by area, which in this case is computed as the area of the circle whose radius is the furthest distance of one request from the bundle's centroid. This approach seems to work well for the chosen regression model, so that is has been included in the final feature set. The difference of the two approaches can be also seen in the discrepancy of the effects that the same features have on the final models; this is the case of what in [18] is presented as *radius*, which has been originally implemented in the exact same way in the new approach, nevertheless, the feature revealed to be relatively useless in helping the regression model improve the prediction performances, so that the feature has been finally discarded. Finally, both methodologies introduce a feature which describes the bundles in terms of tour length of the TSP, the Genetic Algorithm constructs an initial solution using a double insertion heuristic and optimises the solution with a local search method, while for the method proposed in this thesis, between a regression-based approach and a cheapest insertion heuristic, the latter revealed to bring more benefits to the final model, despite higher computational time. Finally, leveraging from depot information when calculating bundle tour lengths revealed to be the most powerful feature of the whole created dataset.

In contrast with the fitness function of the Genetic Algorithm, the engineered features for the regression model do not directly assess the attractiveness of a bundle, they are only used to predict bidding prices, and the attractiveness must be assessed with further steps. Since the bundles' bidding prices of the considered instances are constructed on the cost of serving a determined bundle, the prices are among others correlated with the number of customers in a bundle. Since the objective of the tested instances is minimising total transportation costs, the lower the bidding price, the better. If taking the bidding prices into consideration, due to the reason stated above, bundles with less customers will tend to have lower prices that bigger bundles. Since this may be misleading and considering the main reason why it may be beneficial to put customers into bundles in the first place, namely that the level of attractiveness of a single request varies if combined with other requests, the predicted price per customer is calculated dividing the predicted bidding price by the given number of requests in each bundle. Normalising the bidding price by the number of customers provides a unified quality measure, which can be used as measure for assessing attractiveness for future steps. In contrast with the Genetic Algorithm, the regression model approach does not generate any new population, all bundles can be directly picked to form candidate solutions. The bundles are hence sorted ascendingly according to the proposed attractiveness measure and only a small set of the top part of the sorted dataset per carrier, where the predicted most attractive bundles are located, is used for further processing. This initial best bundle pool is then used for assessing which bundles build feasible solution sets if combined with each other and which ones do not; the latter are directly discarded and not processed any further. The remaining bundles form the final attractive feasible bundles pool. The dimension of this portion of the whole dataset, namely how many n best bundles should be chosen per carrier for the feasibility check has been discussed in section 6.1. Since the feasibility of the solution must be assured, the procedure needs a consistent number of bundles, so that all initial requests appear in the set exactly once. Nevertheless, this may be overcome by assuming that the requests initially placed in the customer pool by each carrier are grouped into bundles according to which carrier they used to belong. This assures a worst-case feasible solution set, and in case no further better feasible sets can be found, the requests are returned to the carriers as they originally were.

The empirical study has shown that the more bundles are placed in the initial attractive bundles pool, the higher the chances that a feasible solution other than the original one can be found, and that the solution quality generally improves; nevertheless, this comes with the cost of slightly increasing computational time due to the increasing size of the attractive feasible bundles pool. The procedure has been set, so that 500 best bundles are selected after the predicted price per customer for each carrier is calculated. The performances of the proposed method in terms of solution quality and computational time of the bidding phase have been compared taking 50, 100, 200, 500, 750 and 1000 best bundles per carrier after the application of the regression model. A curve shows how the average solution quality of the instances improves as the number of considered best bundles increase, nevertheless, the curve flattens when considering $n > 500$ best bundles. Considering that the feasible-best bundles pool size increases when increasing n, but the running time of the bidding phase is in absolute value very low, $n = 500$ seemed to be the best trade-off for the purpose of this model.

As far as the solution quality is concerned, choosing $n = 500$ best bundles per carrier to form feasible sets showed that 15 out of 20 tested instances could be solved with equal or better solutions than the Genetic Algorithm. This shows that it is possible to implement an alternative Business Analytics-based approach for the bundle generation problem, and that this methodology reports equal or often improved solutions compared to the benchmark.
As far as the computational time of the two approaches is concerned however, the Genetic Algorithm outperforms the newer method. The computational study showed how the 20 considered instances could be solved in less than 30 seconds by the Genetic Algorithm, while on average it took roughly 6 minutes for the regression model approach to come up with a feasible solution with the set number of n best bundles, and this running time comparison does not even take the training time of the Elastic Net regressor into consideration. It has been shown how the drastic difference in the running time does not significantly depend on the feasible-best bundle pool size, since the bidding phase is efficient for both proposed methods. The driving factor which makes the running time of the regression model approach skyrocket compared to the Genetic Algorithm is the feature calculation. It has been shown how the cheapest insertion heuristic brings the most benefits in terms of improvement of prediction quality of the regression model, but this is the feature with the slowest computational time among the ones chosen. On top of that, since the pieces of information concerning the depots of each carrier are considered singularly, the calculation of the TSP using the cheapest insertion heuristic has to be repeated as many times as the number of carriers in the network, hence three times for the considered instances. This not only slows down the running time of the whole procedure significantly but may represent an issue when considering the scalability of this methodology, in terms of number of bundles to process

and, above all, in terms of increasing number of carriers in a network. A second slowing factor of the regression model approach is represented by the transformation of the best bundle pool of 1500 bundles into the feasible-best bundle pool, since in contrast with the Genetic Algorithm, a feasibility check has to be undertaken before the bundles can be offered to the carriers, and the greater n is, the more bundles must be processed for the feasibility check, the slower the procedure gets. Lastly, as for the Genetic Algorithm, the proposed methodology is able to drastically reduce the problem size compared to the original complete bundle enumeration. It must be acknowledged, that despite the fact that the regression model approach produces more than three times best bundles compared to the Genetic Algorithm, these represent on average the 0,95% of the 32.767 original bundles whose price has been predicted. This approach is hence able to reduce the problem size by more than 99%. The 100 best feasible bundles found by the Genetic Algorithm represent the 0,3% of the original complete bundle enumeration, the difference between the two approaches is therefore only of 0,65%.

As far as the managerial implications are concerned, Gansterer & Hartl (2018) [18] claims that due to the drastic problem size reduction and efficiency of their approach, they were able to solve real-world size instances with 210 requests, 45 or eventually 90 of which landed in the initial customer pool. Results show that the Genetic Algorithm was able to solve the instance of 45 requests in roughly 45 minutes, whereas for 90 requests it took around 71 minutes. These instances were not compared to the proposed alternative approach, however, due to the limitations stated above, it may be concluded that the Genetic Algorithm is more efficient in solving larger instances as well. It must be pointed out however, that the authors tested their algorithm increasing the number of requests per carrier, but not the number of carriers themselves, whose potential increasing number could represent the major downside of the machine learning approach.
Regarding the considered problem size and the tested instances, the alternative method proposed in this thesis could be implemented for the management of a real small network, since it has been shown that it partially outperforms the solution quality of the Genetic Algorithm, with the trade-off of the higher computational time. Nevertheless, it must be also acknowledged, that although the regression-based approach is by far computationally slower than the Genetic Algorithm, in absolute values the instances can still be solved in a reasonable computational time, namely within few minutes.

Both methodologies achieve their goal, the choice of one method instead of the other for the management of a real small network depends on the major needs and desire of the network members. If solution quality and derived routing cost reduction is the major scope of the network, then the regression model approach could be preferred, since it has been shown that it returns better or equal results than the Genetic Algorithm in 75% of the cases. If on the other hands the carriers have the need of quickly adjusting their current routes and acquire the new assigned bundles, then the Genetic Algorithm is the method to prefer.

7 Conclusion

This thesis presents an alternative, business analytics-based method for efficient request bundle generation in auction-based transportation collaborations. Until now, all proposed methods have been based on heuristics; hence, this thesis represents a novelty in the research in this field. Due to the uniqueness of this approach in the bundle generation literature, this thesis takes the Genetic Algorithm of Gansterer & Hartl (2018) [18] as benchmark for the computational study.

This thesis shows that it is possible to leverage historical data of customer bundles and respective bidding prices for the generation of future attractive ones given the problem presented and that the results can compete with the benchmark values provided by the literature. It is shown how with non-strategic pieces of information, features that are highly correlated with the bidding prices can be generated, and a quick linear regression model can effectively predict bidding prices for future auctions. The prediction model itself is not able to generate attractive-feasible bundles, hence data has to be manipulated for the presented scope. A unique measurement for attractiveness and an upper bound which defines the area of the attractive bundles have to be defined. The latter is represented by the choice of the n best bundles. Results show that the presented alternative approach can effortlessly compete with the benchmark methodology as far as the quality of the produced results is concerned, nonetheless, the running time aspect could be improved.

The major limitation of this approach is the computation of some of the chosen features to train the regression model, since, due to performance reason, it has been chosen to include depot information in the features calculation. This makes the features tailor-made for each carrier, the obvious downside is that a feature must be calculated for each member of the network, significantly compromising the scalability of this approach.

It must be also pointed out that the code created for feature calculation has not been subject to significant optimization checks on the programming techniques and functions used.
This thesis may serve as starting point for future research in business-analytics techniques for the presented problem. This study shows that such a technique can compete and partially outperform an existing framework in terms of solution quality, and some aspects of the new developed framework could be further optimised. After that the most suitable n best bundles have been chosen according to the conducted study, this number remained fixed throughout this work. Results show that sometimes a lower n was sufficient to achieve the desired result, and sometimes the algorithm would have needed more bundles to find better solutions than the benchmark. The development of technique which dynamically changes n could represent an opportunity for future development build on this thesis.
Testing the algorithm with real-world data of a small transportation network would be the way forward towards the implementation of this bundle generation tool in a real-world setting. Real data would allow the calculation of new features which could potentially describe the bundles even more powerfully for a more effective attractiveness assessment, leveraging for example from traffic data. Another advantage of such an approach for the bundle generation problem is the adaptability of the machine learning model; in a real-world setting, the model could be trained with more and more data as soon as it becomes available, so that it can learn new patterns and make up-to-date predictions. Future research should hence focus on the implementation of a machine learning-based bundle generation tool in a real-world setting, given the promising results of the developed framework in the considered artificially generated instances.

References

[1] Aloui, A., Hamani, N., Derrouiche, R., & Delahoche, L. (2021). Assessing the benefits of horizontal collaboration using an integrated planning model for two-echelon energy efficiency-oriented logistics networks design. International Journal of Systems Science: Operations & Logistics, 1-22.

[2] Berthold, Borgelt, Höppner, Klawonn, Silipo, Borgelt, Christian, Höppner, Frank, Klawonn, Frank, & Silipo, Rosaria. (2020). Guide to Intelligent Data Science : How to Intelligently Make Use of Real Data (2nd ed. 2020.). Springer International Publishing Imprint: Springer.

[3] Birkes, Dodge, Y., & Dodge, D. Y. (1993). Alternative Methods of Regression. John Wiley & Sons, Incorporated.

[4] Bishop. (2006). Pattern recognition and machine learning. Springer.

[5] Berger, S., & Bierwirth, C. (2010). Solutions to the request reassignment problem in collaborative carrier networks. Transportation Research Part E: Logistics and Transportation Review, 46(5), 627-638.

[6] B. M. Golam Kibria, A. K. Md. Ehsanes Saleh, S., & Mohammad Arashi, A. (2019). Theory of ridge regression estimators with applications. Wiley-Blackwell.

[7] Cavdar, B., & Sokol, J. (2015). A distribution-free TSP tour length estimation model for random graphs. European Journal of Operational Research, 243(2), 588-598.

[8] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[9] Cleophas, C., Cottrill, C., Ehmke, J. F., & Tierney, K. (2019). Collaborative urban transportation: Recent advances in theory and practice. European Journal of Operational Research, 273(3), 801-816.

[10] Dahl, S., & Derigs, U. (2011). Cooperative planning in express carrier networks—An empirical study on the effectiveness of a real-time Decision Support System. *Decision Support Systems*, *51*(3), 620-626.

[11] De Vries, S., & Vohra, R. V. (2003). Combinatorial auctions: A survey. INFORMS Journal on computing, 15(3), 284-309.

[12] Debbage, K. G. (1994). The international airline industry: globalization, regulation and strategic alliances. *Journal of Transport Geography*, *2*(3), 190-203.

[13] Denis. (2021). Applied Univariate, Bivariate, and Multivariate Statistics Using Python.

[14] Dulhare, Ahmad, K., & Bin Ahmad, K. A. (2020). Machine Learning and Big Data. John Wiley & Sons, Incorporated.

[15] Ergun, O., Kuyzu, G., & Savelsbergh, M. (2007). Reducing truckload transportation costs through collaboration. *Transportation science*, *41*(2), 206-221

[16] Ergun, Ö., Kuyzu, G., & Savelsbergh, M. (2007). Shipper collaboration. *Computers & Operations Research*, *34*(6), 1551-1560.

[17] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[18] Gansterer, M., & Hartl, R. F. (2018). Centralized bundle generation in auction-based collaborative transportation. Or Spectrum, 40(3), 613-635.

[19] Gansterer, M., & Hartl, R. F. (2018). Collaborative vehicle routing: a survey. European Journal of Operational Research, 268(1), 1-12.

[20] Gelman, A., Goodrich, B., Gabry, J., & Vehtari, A. (2019). R-squared for Bayesian regression models. *The American Statistician*.

[21] Gelman. (1995). Bayesian data analysis (1. ed..). Chapman & Hall.

[22] Harbison, J. R., & Pekar, P. (1997). Cross-border alliances in the age of collaboration. Booz Allen & Hamilton, 39-51

[23] Hedrih, & Hedrih, A. (2022). Interpreting Statistics for Beginners. Taylor and Francis.

[24] Houghtalen, L., Ergun, Ö., & Sokol, J. (2011). Designing mechanisms for the management of carrier alliances. Transportation Science, 45(4), 465-482.

[25] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern recognition letters, 31(8), 651-666.

[26] Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. Handbooks in operations research and management science, 7, 225-330

[27] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

[28] Kyriakides, & Margaritis, K. G. (2019). Hands-on ensemble learning with Python : build highly optimized ensemble machine learning models using scikit-learn and Keras /. Packt Publishing.

[29] Laporte, G., Ropke, S., & Zaccour, G. (2007). Horizontal Cooperation Among Freight Carriers: Request Allocation and Profit Sharing MA Krajewska, H. Kopfer. *Les Cahiers du GERAD ISSN*, *711*, 2440.

[30] Lenstra, J. K., & Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. Networks, 11(2), 221-227.

[31] Leskovec, Rajaraman, A., & Ullman, J. D. (2014). Mining of Massive Datasets (Second edition.). Cambridge University Press.

[32] Los, J., Schulte, F., Gansterer, M., Hartl, R. F., Spaan, M. T., & Negenborn, R. R. (2020, September). Decentralized combinatorial auctions for dynamic and large-scale collaborative vehicle routing. In *International Conference on Computational Logistics* (pp. 215-230). Springer, Cham.

[33] MacKay, D. J. (1992). Bayesian interpolation. Neural computation, 4(3), 415-447.

[34] Mancini, S., & Gansterer, M. (2022). Bundle generation for last-mile delivery with occasional drivers. Omega, 108, 102582.

[35] Martello, S., & Toth, P. (1987). Linear assignment problems. In *North-Holland Mathematics Studies* (Vol. 132, pp. 259-282). North-Holland.

[36] Matlock, Welch, J. ., & Parker, F. . (1996). Estimating population density per unit area from mark, release, recapture data. Ecological Applications, 6(4), 1241–1253.

[37] Provost, & Fawcett, T. (2013). Data Science for Business. O'Reilly Media, Incorporated.

[38] Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation science, 40(4), 455-472.

[39] Rothkopf, M. H., Pekeč, A., & Harstad, R. M. (1998). Computationally manageable combinational auctions. Management science, 44(8), 1131-1147.

[40] Tan, Steinbach, M., Kumar, V., & Karpatne, A. (2019). Introduction to Data Mining. Pearson Education, Limited.

[41] Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. Journal of machine learning research, 1(Jun), 211-244.

[42] Triki, C., Oprea, S., Beraldi, P., & Crainic, T. G. (2014). The stochastic bid generation problem in combinatorial transportation auctions. European Journal of Operational Research, 236(3), 991-999

[43] Wang, X., Kopfer, H., & Gendreau, M. (2014). Operational transportation planning of freight forwarding companies in horizontal coalitions. European Journal of Operational Research, 237(3), 1133-1141.

[44] Whitenack. (2017). Machine learning with Go : implement regression, classification, clustering, time-series models, neural networks, and more using the Go programming language /. Packt Publishing

[45] Yan, X., & Su, X. (2009). *Linear regression analysis: theory and computing*. world scientific.

[46] DB wird erster intermodaler Partner der Star Alliance. (2022, July 4). [Press release]. https://www.lufthansagroup.com/media/newsroom/group/2022/q3/22-07-04_PI_Star_Alliance_DE_FINAL.pdf [access date: 05.12.2022]

[47] Koehrsen, W. (2018, July 4). Bayesian Linear Regression in Python: Using Machine Learning to Predict Student Grades Part 2. Medium. https://towardsdatascience.com/bayesian-linear-regression-in-python-using-machine-learning-to-predict-student-grades-part-2-b72059a8ac7e [access date: 16.12.2022]

[48] *sklearn.linear_model.BayesianRidge*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.BayesianRidge.html [access date: 16.12.2022]

[49] *1.1 Linear Models. Bayesian Ridge Regression* (n.d.). https://scikit-learn.org/stable/modules/linear_model.html#bayesian-ridge-regression [access date: 16.12.2022]

[50] Burn, R. (2022, February 26). How to Build a Bayesian Ridge Regression Model with Full Hyperparameter Integration. Medium. https://towardsdatascience.com/how-to-build-a-bayesian-ridge-regression-model-with-full-hyperparameter-integration-f4ac2bdaf329 [access date: 16.12.2022]

[51] Koehrsen, W. (2018b, July 5). Introduction to Bayesian Linear Regression - Towards Data Science. Medium. https://towardsdatascience.com/introduction-to-bayesian-linear-regression-e66e60791ea7 [access date: 16.12.2022]

[52] *sklearn.linear_model.LinearRegression*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html [access date: 16.12.2022]

[53] Bhattacharyya, S. (2020, September 28). Ridge and Lasso Regression: L1 and L2 Regularization. Medium. https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b [access date: 17.12.2022]

[54] *sklearn.linear_model.Ridge.* (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html [access date: 17.12.2022]

[55] *1.1 Linear Models. Ridge Regression* (n.d.). https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression [access date: 17.12.2022]

[56] *1.1 Linear Models. Bayesian Lasso Regression* (n.d.). https://scikit-learn.org/stable/modules/linear_model.html#lasso [access date: 17.12.2022]

[57] *1.1 Linear Models. Bayesian Elastic Net* (n.d.). https://scikit-learn.org/stable/modules/linear_model.html#elastic-net [access date: 17.12.2022]

[58] *sklearn.linear_model.ElasticNet.* (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html [access date: 17.12.2022]

[59] Heeswijk, W. van, PhD. (2022, July 8). Regulate Your Regression Model With Ridge, LASSO and ElasticNet. Medium. https://towardsdatascience.com/regulate-your-regression-model-with-ridge-lasso-and-elasticnet-92735e192e34 [access date: 17.12.2022]

[60] Masui, T. (2022, February 12). All You Need to Know about Gradient Boosting Algorithm − Part 1. Regression. Medium. https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502 [access date: 18.12.2022]

[61] *sklearn.ensemble.GradientBoostingRegressor.* (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html [access date: 18.12.2022]

[62] Python API Reference — xgboost 1.7.2 documentation. (n.d.). https://xgboost.readthedocs.io/en/stable/python/python_api.html [access date: 20.12.2022]

[63] Microsoft. (2021, November 21). LightGBM. Microsoft Research. https://www.microsoft.com/en-us/research/project/lightgbm/ [access date: 27.12.2022]

[64] *lightgbm.LGBMRegressor — LightGBM 3.3.3.99 documentation.* (n.d.). https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html [access date: 27.12.2022]

[65] *scipy.optimize.linear_sum_assignment — SciPy v0.18.1 Reference Guide*. (n.d.). https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.linear_sum_assignment.html [access date: 28.12.2022]

[66] *sklearn.feature_selection.SequentialFeatureSelector*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html [access date: 28.12.2022]

[67] *1.13. Feature selection*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/feature_selection.html [access date: 28.12.2022]

Appendix

Table 24 Sequential Feature Selector Linear Models – Carrier 0

Linear Regression	Bayesian Ridge	Elastic Net
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert_depot_0 • 0,72306163284865	• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,723438091721442	• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • max_distance • standard_dev • cheapest_insert_depot_0 • 0,6761067603178099
• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • radius • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,7284509788930814	• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • radius • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,7285642586941504	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • max_distance • bundle_density • 'standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,4474820588882646
• Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,5716648827886674	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,5716366618020549	• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • radius • bundle_density • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,5293611906331234
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • Tour_length_approx_with_depot_0 • cheapest_insert_depot_0 • 0,7621454846746601	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • max_distance • bundle_density • Tour_length_approx_with_depot_0 • cheapest_insert_depot_0 • 0,7655595245204829	• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert_depot_0 • 0,561630342443003
• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • max_distance • cheapest_insert_depot_0 • 0,6483459767719564	• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • cheapest_insert_depot_0 • 0,6482804067453423	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • max_distance • bundle_density • standard_dev • cheapest_insert_depot_0 • 0,6311213498879554
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1	• number_of_customers • Tour_length_approx • dist_from_depot_1 • max_distance

• radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_0 • 0,748543049820739	• radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_0 • 0,7485591956088661	• bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_0 • 0,6844676429573227
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_0 • 0,5992644143824694	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_0 • 0,599276885323465	• number_of_customers • distance_sum_[Km] • dist_from_depot_1 • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,532177575328677

Table 25 Sequential Feature Selector Ensemble Models – Carrier 0

XGB Regressor	LGBM Regressor	Gradient Boosting Regressor
• Tour_length_approxdist_from_depot_1 • Radius • max_distance cheapest_insert • cheapest_insert_depot_0 • 0,730002177025398	• dist_from_depot_1 • radius • max_distance • cheapest_insert • cheapest_insert_depot_0 • 0,7558361736869577	• distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • standard_dev • cheapest_insert • cheapest_insert_depot_0 • 0,7334917648453992
• distance_sum_[Km] • dist_from_depot_1 • radius • cheapest_insert • cheapest_insert_depot_0 • 0,6890057294126217	• distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_0 • 0,7329054995095626	• distance_sum_[Km] • dist_from_depot_1 • radius • cheapest_insert • cheapest_insert_depot_0 • 0,6900347753657707
• distance_sum_[Km] • dist_from_depot_1 • cheapest_insert_depot_0 • 0,539153907409619	• distance_sum_[Km] • dist_from_depot_1 • cheapest_insert • cheapest_insert_depot_0 • 0,532337515538671	• distance_sum_[Km] • dist_from_depot_1 • cheapest_insert_depot_0 • 0,5429993412195316
• number_of_customers • dist_from_depot_1 • radius • max_distance • cheapest_insert • cheapest_insert_depot_0 • 0,6558064707258047	• dist_from_depot_1 • radius • max_distance • cheapest_insert • cheapest_insert_depot_0 • 0,714083643429561	• number_of_customers • dist_from_depot_1 • radius • max_distance • standard_dev • cheapest_insert • cheapest_insert_depot_0 • 0,6545421614182398
• Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0	• distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_0 • 0,6826311716245141	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_0

• 0,657874011878507		• cheapest_insert • cheapest_insert_depot_0 • 0,656474876275875
• distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_0 • 0,6741333089702367	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_0 • cheapest_insert • cheapest_insert_depot_0 • 0,7032981566971952	• distance_sum_[Km] • dist_from_depot_1 • radius • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_0 • 0,6744991311416404
• Tour_length_approx • dist_from_depot_1 • radius • max_distance • cheapest_insert • cheapest_insert_depot_0 • 0,5945145430038539	• dist_from_depot_1 • radius • max_distancecheapest_insert • cheapest_insert_depot_0 • 0,6286538717123574	• Tour_length_approx • dist_from_depot_1 • radius • max_distance • cheapest_insert • cheapest_insert_depot_0 • 0,5986237930750951

Table 26 Sequential Feature Selector Linear Models – Carrier 1

Linear Regression	Bayesian Ridge	Elastic Net
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,8783475418632009	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,8784875656045136	• number_of_customers • Tour_length_approx • dist_from_depot_2 • radius • max_distance • bundle_density • standard_dev • cheapest_insert_depot_1 • 0,6738880221615816
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • bundle_density • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,8450186128940341	• number_of_customers • Tour_length_approx • dist_from_depot_2 • radius • max_distance • bundle_density • standard_dev • cheapest_insert_depot_1 • 0,8462207936953237	• number_of_customers • distance_sum_[Km] • dist_from_depot_2 • max_distance • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,815227797217535
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance • bundle_density • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,7517875160563449	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,666588503881068

• 0,7506037949485354		
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • bundle_density • Tour_length_approx_with_depot_1 • cheapest_insert_depot_1 • 0,9235098499533857	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance • bundle_density • cheapest_insert_depot_1 • 0,9235701770270474	• distance_sum_[Km] • dist_from_depot_2 • cheapest_insert_depot_1 • 0,9055033013650589
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,8266027206982169	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,8265285365698724	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,7767414618037446
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,751541085184877	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,7515627699926533	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert_depot_1 • 0,6995794517297733
• Tour_length_approx • distance_sum_[Km] • radius • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert_depot_1 • 0,8340728019809056	• number_of_customers • Tour_length_approx • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert_depot_1 • 0,8334624898703972	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,8149423974986194

Table 27 Sequential Feature Selector Ensemble Models – Carrier 1

XGB Regressor	LGBM Regressor	Gradient Boosting Regressor
• max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,8030317569182681	• dist_from_depot_2 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_1 • 0,8232311131555463	• dist_from_depot_2 • radius • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,8084205933073884
• number_of_customers • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance	• dist_from_depot_2 • radius • bundle_density • Tour_length_approx_with_depot_1 • cheapest_insert	• number_of_customers • dist_from_depot_2 • max_distance • standard_dev • cheapest_insert_depot_1

• bundle_density • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,8381465442470166	• cheapest_insert_depot_1 • 0,8394157515298654	• 0,8341971700394716
• Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • bundle_densitycheapest_insert • cheapest_insert_depot_1 • 0,6904176617859078	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_1 • 0,710297874823095	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_2' • radius • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,6917989701065188
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,9222479976965747	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_2' • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_1 • 0,9267693335392997	• number_of_customers • Tour_length_approx • dist_from_depot_2 • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,9217953681809247
• Tour_length_approx • dist_from_depot_2 • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,796851528830359	• Tour_length_approx • dist_from_depot_2 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_1 • 0,8081423519132793	• Tour_length_approx • dist_from_depot_2 • radius • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_1 • 0,7980361782219921
• Tour_length_approx • dist_from_depot_2 • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,6964979793784836	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,7107182486503038	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_2 • radius • max_distance • standard_dev • Tour_length_approx_with_depot_1 • cheapest_insert • cheapest_insert_depot_1 • 0,6949519814540236
• Tour_length_approx • dist_from_depot_2 • radius • max_distance • cheapest_insert_depot_1 • 0,823970428359868	• Tour_length_approx • dist_from_depot_2 • radius • max_distance • cheapest_insert • cheapest_insert_depot_1 • 0,8226788055189589	• Tour_length_approx • dist_from_depot_2 • radius • max_distance • standard_dev • cheapest_insert_depot_1 • 0,8257821970710889

Table 28 Sequential Feature Selector Linear Models – Carrier 2

Linear Regression	Bayesian Ridge	Elastic Net
• Tour_length_approx • dist_from_depot_3 • max_distance	• Tour_length_approx • dist_from_depot_3 • max_distance	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_3

• standard_dev • cheapest_insert • cheapest_insert_depot_0 • 0,8450230914382377	• standard_dev • cheapest_insert • cheapest_insert_depot_2 • 0,8450236105268221	• max_distance • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert_depot_2 • 0,8334542410461842
• number_of_customers • Tour_length_approx • radius • max_distance • bundle_density • Tour_length_approx_with_depot_2 • cheapest_insert_depot_2 • 0,8584857811161595	• number_of_customers • Tour_length_approx • radius • max_distance • bundle_density • Tour_length_approx_with_depot_2 • cheapest_insert_depot_2 • 0,8585315525329008	• number_of_customers • distance_sum_[Km] • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8503203625495775
• number_of_customers • distance_sum_[Km] • dist_from_depot_3 • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert_depot_2 • 0,8187969515535725	• number_of_customers • distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert_depot_2 • 0,8197284554079483	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • max_distance • standard_dev • cheapest_insert • cheapest_insert_depot_2 • 0,8085187793883432
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • radius • bundle_density • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8945771155863792	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • radius • bundle_density • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,894777785644042	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • max_distance • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8561767945746892
• Tour_length_approx • dist_from_depot_3 • radius • max_distance • bundle_density • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8723017157117662	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • bundle_density • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8718563270412625	• Tour_length_approx • distance_sum_[Km] • radius • max_distance • bundle_density • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,85102630545572
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • max_distance • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8871609413434381	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • max_distance • cheapest_insert • cheapest_insert_depot_2 • 0,8874456140844227	• number_of_customers • Tour_length_approx • dist_from_depot_3 • bundle_density • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8816486581065416
• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • standard_dev • cheapest_insert_depot_2	• number_of_customers • distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • standard_dev	• number_of_customers • Tour_length_approx • dist_from_depot_3 • radius • max_distance • bundle_density

• 0,7374653329183902	• cheapest_insert_depot_2 • 0,739513634428336	• standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert_depot_2 • 0,717076631212314

Table 29 Sequential Feature Selector Ensemble Models – Carrier 2

XGB Regressor	LGBM Regressor	Gradient Boosting Regressor
• Tour_length_approx • dist_from_depot_3 • radius • cheapest_insert • cheapest_insert_depot_2 • 0,8476877534590358	• dist_from_depot_3 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_2 • 0,8500941264786748	• number_of_customers • dist_from_depot_3 • radius • standard_dev • cheapest_insert • cheapest_insert_depot_2 • 0,847329968445867
• number_of_customers • distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • cheapest_insert_depot_2 • 0,8546778881547743	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • radius • cheapest_insert • cheapest_insert_depot_2 • 0,8527049549376216	• number_of_customers • distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • standard_dev • cheapest_insert_depot_2 • 0,8550469022492431
• Tour_length_approx • dist_from_depot_3 • max_distance • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8196533964088415	• distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_2 • 0,8275546839005784	• number_of_customers • Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8205786547567296
• number_of_customer • dist_from_depot_3 • radius • bundle_density • standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8808294493406645	• number_of_customers • dist_from_depot_3 • radius • max_distance • bundle_density • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8905434405998994	• distance_sum_[Km] • dist_from_depot_3 • radius • standard_dev • cheapest_insert • cheapest_insert_depot_2 • 0,8796283410819268
• Tour_length_approx • dist_from_depot_3 • radius • max_distance • cheapest_insert • cheapest_insert_depot_2 • 0,8718373893627709	• dist_from_depot_3 • radius • max_distance • cheapest_insert • cheapest_insert_depot_2 • 0,8694197491669714	• Tour_length_approx • dist_from_depot_3 • radius • max_distance • bundle_density • standard_dev • cheapest_insert • cheapest_insert_depot_2 • 0,8714859350650445
• distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • bundle_density • standard_dev	• distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • bundle_density • cheapest_insert	• Tour_length_approx • distance_sum_[Km] • dist_from_depot_3 • radius • max_distance • bundle_density

• Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8806898693743662)	• cheapest_insert_depot_2 • 0,8849642457405691	• standard_dev • Tour_length_approx_with_depot_2 • cheapest_insert • cheapest_insert_depot_2 • 0,8809564710613056
• Tour_length_approx • dist_from_depot_3 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_2 • 0,7081210772932145	• dist_from_depot_3 • radius • max_distance • bundle_density • cheapest_insert • cheapest_insert_depot_2 • 0,7217651125818345	• number_of_customers • Tour_length_approx • Radius • max_distance • cheapest_insert • cheapest_insert_depot_2 • 0,7048834136502158