Soft Computing (2024) 28:5045–5056
https://doi.org/10.1007/s00500-023-09220-7

APPLICATION OF SOFT COMPUTING

Prediction of service time for home delivery services using machine learning

Jan Wolter[1] · Thomas Hanne[1]

Accepted: 8 September 2023 / Published online: 29 September 2023

Abstract
With the rise of ready-to-assemble furniture, driven by international giants like IKEA, assembly services were increasingly offered by the same retailers. When planning orders with assembly services, the estimation of the service time leads to additional difficulties compared to standard delivery planning. Assembling large wardrobes or kitchens can take hours or even days while assembling a chair can be done in a few minutes. Combined with the usually vast amounts of offered products, a lot of knowledge is required to plan efficient and exact delivery routes. This paper shows how an artificial neural network (ANN) can be used to accurately predict the service time of a delivery based on factors such as the goods to be delivered or the personnel providing the service. The data used include not only deliveries with assembly of furniture, but also deliveries of goods without assembly and delivery of goods requiring electrical installation. The goal is to create a solution that can predict the time needed based on criteria such the type of furniture, the weight of the goods, and the experiences of the service technicians. The findings show that ANNs can be applied to this scenario and outperform more classical approaches, such as multiple linear regression or support vector machines. Still existing problems are largely due to the provided data, e.g., a large difference between the number of short and longer duration orders, which made it harder to accurately predict orders with longer duration.

Keywords Home delivery · Service times · Prediction · Neural networks

1 Introduction

Estimating the time required to assemble furniture at a customer location or to install a washing machine is a complex task that requires in-depth knowledge about the products contained in the delivery, the level of experience of the delivery personnel and the conditions at the place of delivery (i.e., which floor, presence of an elevator, etc.). For most, if not all, delivery companies, this knowledge exists only in unstructured and implicit form in the heads of the dispatcher personnel. This paper will use the term service time or order duration to describe the time a delivery vehicle is standing at a customer location to unload and deliver and possibly to assemble goods. Similar terms seen in the research include service duration, stop time, or time window (in the context of the vehicle routing problem).

Implicit employee knowledge carries the obvious risk of knowledge drain in the event of the resignation, termination, or retirement of an employee. Furthermore, the issue at hand is complex and requires an amount of knowledge that can hardly be documented. The company, whose data will be analyzed in the paper, delivers from a range of roughly 100,000 products. Documenting the know-how for even a subset of these articles is an unsurmountable challenge in a financially competitive market such as logistics.

The underlying problem can be considered as a regression problem and has, therefore, seen a considerable amount of research in the last decades. Predicting a dependent variable by analyzing multiple independent variables has been done in countless ways, from artificial neural networks to machine learning techniques, such as multiple linear regression or support vector machines.

✉ Thomas Hanne
thomas.hanne@fhnw.ch

[1] Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Olten, Switzerland

However, the application of these techniques to the specific logistics problem of predicting service times has not seen as much research. Most research that is similar in nature addresses different real-world issues. The underlying reasons for this lack of research might be the rather narrow field of application, a lack of sophisticated data to analyze, or the small direct financial incentive presented by reducing the risk of knowledge drain.

Predicting accurate service times, even if they are only as good as the estimate of a human, will still give a company a competitive advantage. There are two clearly identifiable issues with estimates by a human:

- The time it takes to review an order, the team that executes the service, and other possible influencing factors are varied and complex, and require a broad human knowledge base of the service technicians, the product catalog, and the issues that could arise during service execution.
- Losing an experienced dispatcher can hinder productivity immediately, and new employees need months or years to acquire the background knowledge they need to do their job efficiently. Furthermore, even small errors can lead to high costs in later stages of the delivery process. When a dispatcher estimates the service time based on the order positions and overlooks the fact that not one, but three wardrobes need to be delivered and assembled, the planned tour cannot be executed as scheduled due to time constraints. Further negative consequences are increased costs and non-monetary damages such as dissatisfied customers and unhappy service technicians.

Training a neural network on historical order data could fix both these issues.

- Calculating the service time can be included in existing automated processes. This would reduce the effective work required by a dispatcher as it is only necessary to check whether the estimated time is reasonable. This step can also become superfluous after an initial adaption phase and increasing trust in the system.

A well-trained neural network does not accidentally "overlook" information from a customer order and is free from careless mistakes. New or unplanned orders could still lead to errors, but the retraining of the neural network to include changes in patterns takes hours or days, instead of months or years.

2 Related work

The literature review focused on finding similar research on optimal estimation of service times in the same or a related business domain. Unfortunately, only related publications in different business domains (i.e., surgery, construction, project management) could be found.

There are examples of problems in a very similar business domain. Coelho et al. (2016) describe a solution to optimize lunch breaks in furniture delivery tours but take the duration of the time windows for assembly as a fixed and predetermined value. Fries et al. (2018) describe a model to plan delivery routes for furniture and home appliances with assembly at the customer location, but again take the duration of the assembly as a given parameter and do not calculate it. Suzuki et al. (2017) analyzed how the furniture assembly duration changes with different numbers of people assembling the furniture. In contrast to the situation at hand, they did not consider the problem in a logistical sense, and the participants were not professional, but lay people building furniture for themselves.

Some notable cases with similar problems but in different business domains are the case of a Swiss in-home patient care service (Kooijman 2019); a case about estimating the duration of surgery procedures based on data about the procedure, the patient and the doctors (Ng et al. 2017); a model to predict the duration of aircraft assembly (Guo and Wei 2011) considering the airplane-type and the number of workers; or the time it takes to resolve a traffic incident on a highway, based on road conditions and speeds (Vlahogianni and Karlaftis 2013).

In the field of logistics, the planning of service times is often only a part of the larger vehicle routing problem (VRP), which is based on the traveling salesman problem (TSP). There exist many different variants of this problem when it comes to how services or deliveries are executed at geographically separated locations. The classic problem of reaching a number of nodes by the shortest possible way was well described by Laporte (1992). A variation of this basic version is the consideration of time windows which specify when the traveler or the vehicle has to arrive at a specific node (Ahn and Shin 1991). This is related to the situation discussed in this paper, as the vehicle routing problem with time windows (VRPTW) considers the duration that a vehicle has to stand at a stop. Calculating the duration of this stop time is generally considered a different problem.

Other noticeable variants of the vehicle routing problem include the consideration of multi-vehicle problems (Beaujon and Turnquist 1991), mixed pickup and delivery

(Golden and Wasil 1988) or nodes which need to be visited in specific orders (Desrosiers et al. 1995).

The problem of having variable-timed stops at each node was described by Vössing (2017) in an attempt to show why many companies still rely on human dispatcher knowledge for tour planning and time estimations. The variations in the required time are complex and require a case-by-case analysis of the underlying reasons and data. Extrapolation from one problem domain to another is often not possible, because the problems have been simplified too much to be able to implement them in the real world (Sörensen et al. 2008).

Collins and Sisley (1994) describe a software solution that can plan and schedule service personnel for repairs and maintenance in a B2B context, based on different criteria such as the identity of the customer, the urgency of the call or issue, and geographic factors such as the current location of service staff. What this solution apparently does not consider is the time that a service technician spends at a given site. Instead, the solution plans in real time and only takes into account service technicians who are not currently working on a repair or maintenance job.

3 Research methodology and used data

In our study, we assume that machine learning can be used to predict the time required to service a customer in the context of home deliveries. The following research questions are addressed:

- Which attributes of a typical furniture delivery are good indicators of the service duration?
- Which architecture of an artificial neural network produces the best predictions based on the data at hand?

In this study, we analyze empirical data supplied by a last mile delivery company from its order planning and telematics software. The data includes roughly 90,000 customer orders including customer locations (address data) and ordered items (either physical or service items) and quantities, master data on the sold items, and information about vehicles, drivers, service technicians and executed tours.

Using an ETL procedure (extract, transfer, load) the required information is extracted from two databases, one containing staff information, the other being the basis for the order planning software. The data is denormalized into a smaller number of tables and further cleaned during several preprocessing steps, e.g., to remove outliers. Outliers were defined by expert decision and included some rare types of deliveries, service times greater than 10 h or deliveries requiring special personnel or vehicles.

For machine learning purposes, item numbers need to be one-hot-encoded, which requires having individual columns for every item number. Due to the large number of physical items only the product category was used as a column description to avoid too many features.

The machine learning steps are programmed in Phyton. Machine learning algorithms are used from the TensorFlow package maintained by Google which provides an API for Machine Learning under an open-source license (Metz 2015). In particular, this software provides the functionalities of an ANN that are relatively easy to implement and efficient to use. The TensorFlow API in the current version 2.0 provides various configuration possibilities of an ANN and contains eight ready-to-use optimizers which can be employed for the learning phase of the ANN.

The software was developed using a Miniconda installation, a software solution that bundles Python and Conda, a package manager for easy installation of new Python frameworks such as Keras 1.0.8, Pandas 0.25.2 and Numpy 1.16.5 which were used during the study. Upon the installation of Miniconda, Jupyter Notebook enables the smooth and comfortable execution of Python procedures in a browser-based solution. All computational results were calculated on a computer with AMD Ryzen 5 2600X CPU overclocked at 4.0GHZ and MSI GeForce RTX 2080 Ti GAMING X TRIO GPU with 64 GB RAM.

4 Data preprocessing and analysis

For each order, over 200 distinct features are collected during the order processing. Most of these factors do not give meaningful input and have no correlation with the service time (i.e., phone number of the customer, the identity of the dispatcher, or unstructured order notes) and have, therefore, been excluded in advance.

For some factors, the catalog of past orders is too small to extrapolate useful information. The expected delivery date could have an influence on service times, as staff productivity could be reduced during the hot summer months or unloading the products could be slower when it is cold and icy outside. Unfortunately, the available data only contains orders from six months, so not all seasons can be considered.

There are multiple features where there may or may not be a correlation or predictive quality. An example of such an unclear feature is the position of the order on the tour. Service technicians could be tired or exhausted during the day, which could affect assembly times.

The following features were included in the ETL procedure. It should be noted that some of these features have no guaranteed influence on service time, but to be sure, need to be evaluated (Table 1).

Table 1 Overview of selected features for training

Feature	Type	Known during order creation
Order ID	ID	Yes
Tour ID	ID	Yes
Planned Service Time	Integer	Yes
Self-Reported Service Time	Integer	Yes
Order Type	Categorical (7)	Yes
Required Qualification 1	Categorical (15)	Yes
Required Qualification 2	Categorical (19)	Yes
Required Qualification 3	Categorical (21)	Yes
Franchise	Categorical (6)	Yes
Customer ZIP	Categorical (2936)	Yes
Exit Hub	Categorical (6)	Yes
Planning office	Categorical (9)	Yes
Distribution Channel (Online or Store)	Categorical (4)	Yes
Delivery region	Categorical (58)	Yes
Order creation date	Date	Yes
Customer language	Categorical (4)	Yes
Order weight	Integer	Yes
Order volume	Integer	Yes
[Physical Articles]		
[Service Articles]		
Driver ID	Categorical (144)	No
Co-Driver ID	Categorical (165)	No
Delivery vehicle ID	Categorical (74)	No
Tour position	Integer	No
Driver experience	Integer	No

The column "Type" gives an indication of the nature of the feature. The last column indicates whether a feature is known at the point in time where an order is created in the dispatcher software. This is relevant in that an initial estimation of the duration of service is currently required when the order is created, even if not all the features that affect the duration are known.

The order and tour ID were included for debugging purposes and can obviously have no predictive qualities for a label. The planned and self-reported service time are the two label candidates that will be evaluated for their usefulness in the following chapters. They differ in that the planned label is the required service time specified by the dispatcher prior to delivery, while the self-reported label is the service-time manually documented by the delivery staff. The available candidates for labels are flawed as the self-reported duration of an order might be biased, because the service technicians may report inaccurate data. At the same time, the planned order duration has the downside of being inaccurate, because it is only a human estimation that might not consider all essential factors influencing order duration.

5 Machine learning approaches and results

To gain a better understanding of good solutions, a categorization and a regression model were built to predict the order durations. Both models were compared to a baseline implementation using a different technique, a support vector machine (SVM) or a multivariate linear regression. Section 5.1 discusses the implementation of the SVM and the ANN and compares the quality of the predictions for categorizations. Section 5.2 compares the implementation and predictions for the Multivariate Linear Regression and the ANN for regressions.

5.1 Categorization

In the original form, both planned and self-reported order durations are present as regular integers rather than categories. The formation of categories from these integers is achieved by dividing them into groups of minutes. By dividing the integers by 10, Category 0 contains all orders from 0 to 9 min, Category 1 all orders between 10 and 19 min, et cetera.

The evaluation of the order durations shows a clear trend towards short durations. As can be seen in Fig. 1, the orders show a very uneven distribution of duration, with most orders having a duration of less than 60 min, and very few long orders recorded.

This can be problematic in categorizations, as some categories have much smaller samples than others. Therefore, the models in the categorization comparison are additionally evaluated with and without oversampling. To apply oversampling to a data set, copies of the rare category are to be added to the data set. This shifts the distribution of the orders into the direction of the rare categories and prompts the ANN to pay more attention to their characteristics (Brownlee 2020). Points were added to each class, so that all classes appear with the same relative frequency in the data set. Note that there are other approaches for addressing the imbalance problem such as undersampling the majority class. However, undersampling is not recommended as a first choice to address the issue (Wongvorachan et al. 2023).

While Sect. 5.1.1 shows the implementation and prediction results for the SVM, Sect. 5.1.2 provides the corresponding information for the ANN implementation. In addition, the section compares prediction accuracy for different group sizes, not only for discrete values. In real world applications, a time window with a high confidence might be more useful than a discrete, lower-confidence prediction of service duration. Especially considering that there are influencing factors that cannot be considered in the model (future weather, sick leaves, etc.), which increase the inherent inaccuracy of a discrete prediction.

5.1.1 Support vector machine

A support vector machine is one of the most common implementations and starting points for categorization problems (Colas and Brazdil 2006). The primary goal of an SVM is to separate classes or categories with a linear separator that has the largest margin possible. In a visual sense, this means that the goal is to create a boundary as large as possible between two data clusters. To increase the usefulness of the SVM, additional separators can be added to distinguish more than two classes. Furthermore, the kernel trick can be used to separate the classes in a non-linear way (Russel and Norvig 2020, p. 757).

As it was experimentally found that training the model takes longer and longer with a larger number of samples in the data set, only 10,000 data items are randomly sampled for training. The complete data set contains around 35,000 samples. The label data has already been divided into classes of 10 min. After the division into classes of ten minutes, 26 categories (one for each 10 min increment between 0 and 250 min of duration) emerge and need to be explicitly declared for the model.

Table 2 shows the results for different models built with SVM. In the beginning, only individual, distinctly separated classes were used. This confirms that the model can achieve good results in simple cases with an accuracy of over 0.9. 20% of the data set was used for testing and 80% of the data set was used for training.

The C parameter in these early and easily separable cases made almost no difference. It tells the optimization how acceptable misclassifications are, in a trade-off against wider-margin classifiers. The kernel defines the method that is used to generate the classifiers between the classes, for example a linear or sigmoid function.

Looking at the prediction quality on the complete data set, some interesting observations can be made:

- The prediction quality for the sigmoid kernel was far worse in the cases where the data set was oversampled.
- For the sigmoid kernel, the C parameter did not noticeably change the prediction quality in any case.
- The RBF kernel was more sensitive to the C parameter, and higher values for C generally lead to better prediction results.

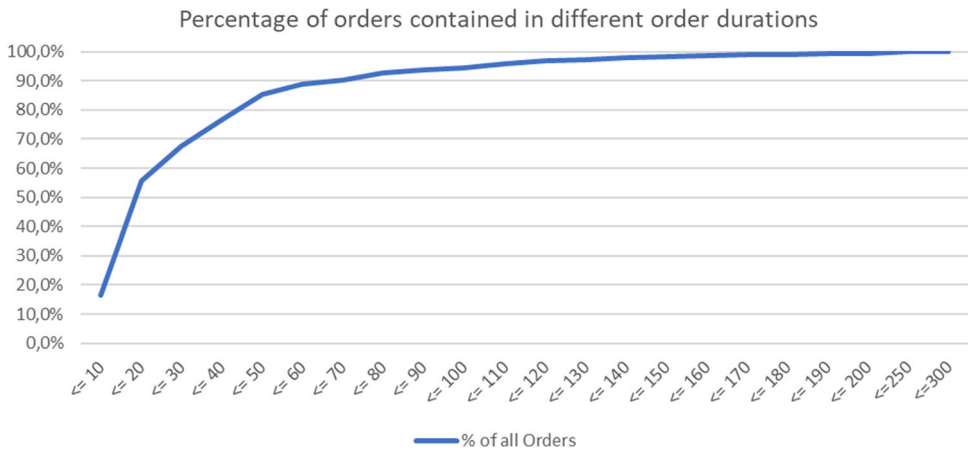

Fig. 1 Percentage of orders contained in different order duration classes

Table 2 Categorization accuracy for different SVM implementations on planned label data

Included classes	Kernel	C	Accuracy	Training duration
1, 7, 14, 21	RBF	1	0.949	15 s
1, 7, 14, 21	RBF	7	0.942	15 s
1, 7, 14, 21	RBF	20	0.938	15 s
1, 7, 14, 21	Sigmoid	1	0.897	15 s
1, 7, 14, 21	Sigmoid	3	0.896	15 s
1, 7, 14, 21	Sigmoid	20	0.930	15 s
All (0–25)	RBF	20	0.545	40 s
All (0–25)	Sigmoid	20	0.312	45 s
All (0–25)	RBF	1	0.519	55 s
All (0–25)	Sigmoid	1	0.327	40 s

- The RBF kernel showed better predictive quality on the data set that was not oversampled, except for the case with a C value of 20. In that case, the oversampled data set produced the overall best results for this kernel.

There are no training results for the polynomial and the linear kernel, as these models did not finish training within 30 min, even on tiny data sets. Table 3 shows the results of several model training runs using the self-reported order duration as the label.

Notable is the lack of improvement when oversampling the orders for training and the much lower accuracy in comparison to the planned label. In contrast to the planned label duration, the models, in this case, showed practically no sensitivity to the C parameter.

5.1.2 Artificial neural network implementation for multi-class classification

Following the results of the SVM implementation, this section will show the equivalent implementation of categorization using an ANN. As in the case of SVM, several different combinations and network designs are compared to find suitable candidates.

TensorFlow implements three out-of-the-box loss functions for use in categorizations, i.e., different possibilities to measure the distance between a given prediction and the corresponding label:

- Sparse categorical cross-entropy,
- Categorical cross-entropy,
- Binary cross-entropy.

While the sparse categorical cross-entropy and the regular categorical cross-entropy share the same loss function, the sparse categorical cross-entropy loss function was implemented to handle label data that is not one-hot-encoded, i.e., one output node containing different integer values indicating the predicted class. In contrast, the regular categorical cross-entropy is intended to handle labels and output nodes that are one-hot-encoded, i.e., where one output neuron indicates the activation of that given class. The Binary Cross-Entropy loss function furthermore is intended for use with only two classes (TensorFlow 2020a, b, c), which is clearly not the case for this scenario. In contrast to the next section on regression, we, therefore, consider only one type of loss function for all trainings, the categorical cross-entropy.

Several architectures of the ANN were experimentally evaluated. The number of hidden layers and activation function is documented in Tables 4 and 5. The number of neurons in the hidden layers is half of the size of the input layer. All networks are designed with an input layer equal to the size of the number of different features and an output layer equal to the number of different categories that are to be separated. Fivefold cross-validation was used for the model with a split of 20% for test and 80% for training.

Table 4 shows the results of training runs of the ANN categorization using the planned order duration as a label. It can be noted that the Softmax activation function generally produced better predictions than the Thanh, Sigmoid and Hard Sigmoid functions. When using the Softmax

Table 3 Categorization accuracy for different SVM implementations on self-reported data

Included classes	Kernel	C	Accuracy with oversampling	Accuracy without oversampling
1, 7, 14, 21	RBF	1	0.944	0.947
1, 7, 14, 21	RBF	7	0.933	0.945
1, 7, 14, 21	RBF	20	0.933	0.945
1, 7, 14, 21	Sigmoid	1	0.936	0.947
1, 7, 14, 21	Sigmoid	3	0.935	0.947
1, 7, 14, 21	Sigmoid	20	0.93	0.942
All (0–25)	RBF	20	0.381	0.38
All (0–25)	Sigmoid	20	0.417	0.38
All (0–25)	RBF	1	0.418	0.38
All (0–25)	Sigmoid	1	0.416	0.38

Table 4 Comparison of different prediction accuracies for classifications on planned label data

Activation function	Optimizer	Layers	Loss/Accuracy/Duration w/o Oversampling	Loss/Accuracy/Duration w. Oversampling
Softmax	Adagrad	2	0.67/0.78/1540 s	0.47/0.77/1400 s
Softmax	RMSProp	2	0.93/0.69/1760 s	0.79/0.70/1800 s
Softmax	Adam	2	7.81/0.67/1100 s	27.43/0.71/1000 s
Softmax	SGD	2	1.11/0.68/660 s	2.01/0.42/540 s
Thanh	Adagrad	2	4.61/0.09/1540 s	9.46/0.39/1200 s
Sigmoid	Adagrad	2	5.92/0.09/1500 s	3.93/0.16/1200 s
Hard Sigmoid	Adagrad	2	4.26/0.09/1540 s	5.16/0.16/1400 s
Softmax	Adagrad	1	0.45/0.76/1260 s	0.48/0.78/1200 s
Softmax	Adagrad	3	0.41/0.77/1720 s	0.49/0.77/1800 s

Table 5 Comparison of different prediction accuracies for classifications on self-reported label data

Activation function	Optimizer	Layers	Loss/Accuracy/Duration w/o Oversampling	Loss/Accuracy/Duration w. Oversampling
Softmax	Adagrad	2	0.40/0.86/1300 s	0.69/0.71/940 s
Softmax	RMSProp	2	0.50/0.79/1430 s	1.07/0.58/1120 s
Softmax	Adam	2	82.34/0.78/900 s	19.25/0.38/650 s
Softmax	SGD	2	0.62/0.77/170 s	1.00/0.58/360 s
Thanh	Adagrad	2	10.26/0.58/1100 s	8.53/0.39/950 s
Sigmoid	Adagrad	2	3.24/0.58/1160 s	4.00/0.09/1010 s
Hard Sigmoid	Adagrad	2	2.28/0.58/1160 s	4.40/0.09/990 s
Softmax	Adagrad	1	0.53/0.86/1040 s	0.74/0.68/820 s
Softmax	Adagrad	3	0.49/0.87/1340 s	0.86/72/1100 s

activation function, the optimizer made almost no difference in prediction accuracy. In addition, there was no significant improvement when oversampling the underrepresented classes for this case.

In contrast to the implementation of the regression in Sect. 5.2, the learning rate for the SGD optimizer had to be decreased from 0.001 to 0.0001 for all training runs. The initial learning rate led to a learning process, where the accuracy only increased very slowly. Increasing the learning rate from 0.001 to 0.01 led to a constant accuracy and no improvements over multiple iterations. Table 5 shows the result of the same model layouts but using the self-reported label for training.

When comparing the results with the planned label, strong similarities emerge. The Softmax activation function produced better results than the other functions, while the optimizer and oversampling had no significant impact on prediction accuracy. Overall, the predictions on the self-reported label are more accurate than on the planned label.

The difference in execution time between the planned and self-reported label and between the models with and without oversampling is based solely on differently sized data sets that were used for training and is not an indication of different speeds between the models.

The baseline for comparison is the case of 10 min category sizes, which were used previously for all training runs. The network layout is identical to the baseline in the previous runs as well, a Softmax activation function, the Adagrad optimizer, and two hidden layers of neurons.

Table 6 shows the prediction results for seven different category sizes, ranging from one extreme of 6 min per

Table 6 Classification results for different sizes of categories

Category size	Number of categories	Loss	Accuracy
6 min	43	0.705	0.745
8 min	32	0.840	0.760
10 min	26	0.471	0.777
12 min	22	0.607	0.797
14 min	19	0.545	0.830
20 min	13	0.493	0.805
30 min	9	0.229	0.885

category to the other extreme with 30 min windows for each category. For this training run, only the planned label was used, as the results of the self-reported label should be similar or identical for this analysis. A 30 min prediction window does not seem useful for the real world. However, it is a good indication that the model is working as intended, as the predictions get better with broader categories, and the loss decreases. Where the border between usefulness and inaccuracy lies, is a decision best made by the business that uses the model afterwards.

The decline in accuracy in the 20 min category sizes is notable. This discrepancy occurred in several runs and cannot be explained with confidence. One possible factor might be a large cluster of fairly identical orders that have durations just at the border of two categories when they are split every 20 min. If there are many similar orders that are between 18 and 22 min, for example, accuracy may decrease when the border is drawn at 20 min. If the category size is 8 min instead, one border will be at 16 min and another at 24 min, nicely including the imaginary group of orders.

5.2 Regression

In the previous section, the predictive performance of an ANN was compared with that of a support vector machine for the case at hand. This section provides a similar comparison, but for a regression model using a multivariate linear regression and a maximum likelihood estimate (MLE) using the ANN.

5.2.1 Linear regression model

To implement a baseline for the regression model, a multivariate linear regression model was built. This technique builds upon the simpler univariate linear regression. The univariate linear regression takes one feature as input and thus predicts the output. It can easily be imagined graphically as drawing a straight line through data plotted on a 2D coordinate system.

Obviously, not all predictions can be made on the basis of a single input variable. To use all the available features for this case, multivariate linear regression is used. The principles stay the same, but instead of fitting a one-dimensional line through a two-dimensional data set, an n-1-dimensional plane is fitted through an n-dimensional space (Russel and Norvig 2020, p. 734 ff.).

Linear regression was chosen as a baseline comparison as it is easy to grasp and implement. In addition, TensorFlow offers an extensive tutorial for the implementation (TensorFlow 2020a, b, c). The code for the model was mainly adapted directly from the provided tutorial.

The model uses MSE as its loss by default, which makes it easy to compare its results to other implementations in the regression. From several runs, the model produced a somewhat consistent MSE loss of around **625** for the planned duration label and **440** for the self-reported label. The training runs finished in under 5 min. The next section will allow the comparison of these results with the loss that is produced from the implementation of the ANN.

5.2.2 Artificial neural network implementation for regression

We now show how a regression works using an artificial neural network. In a fundamental sense, the model is very similar to the ANN for categorization, but instead of several nodes in the last layer, each per category, the regression contains only one node which is activated and indicates the value of the regression.

During initial testing, it was found that the SoftPlus activation function in the first layer improved performance slightly. As the input data in this layer is fully one-hot-encoded and, therefore, binary, the SoftPlus function covers the possible value range nicely.

After defining the layout of the neural network, the optimizer is initialized. Depending on the class that is used, different parameters can be passed to the optimizer. In the case of the RMSProp optimizer, only the learning rate needs to be provided beforehand.

Several tutorials have suggested a good initial learning rate of 0.001. For some models, it made sense to increase the learning rate; for some it has been decreased. To determine whether the learning rate needed to be adjusted, the progress that was made from epoch to epoch during the training progress was evaluated. If the loss or accuracy was not changing at all or did not converge until the last epoch, a change in the learning rate led to useful results in all cases.

How long the model takes to train depends mainly on the size of the network, the complexity of the optimizer and activation functions, and the amount of data that is used for training. An example of how the training can be supervised is shown in Fig. 2. With this feedback, the training parameters can be adjusted if, for example, the loss does not change after the first few epochs (model converges to early) or still makes considerable improvements during the last few epochs (model has not converged in the end).

After the model has finished training, the weights are fixed, and the test data set can be used to evaluate the accuracy of new and unknown data. The complete data set consists of roughly 40,000 samples, which are split into 80% training data and 20% test data. Several different activation functions, optimizers, and numbers of hidden layers were evaluated to find the optimal layout with the

```
Train on 20400 samples
Epoch 1/20
20400/20400 [==============================] - 20s 981us/sample - loss: 195049.0942 - mae: 50.5316 - mse: 195049.1875
Epoch 2/20
20400/20400 [==============================] - 19s 939us/sample - loss: 956.0517 - mae: 19.4152 - mse: 956.0522
Epoch 3/20
20400/20400 [==============================] - 19s 940us/sample - loss: 780.0074 - mae: 17.2512 - mse: 780.0075
Epoch 4/20
20400/20400 [==============================] - 19s 929us/sample - loss: 703.8769 - mae: 16.0755 - mse: 703.8771
Epoch 5/20
20400/20400 [==============================] - 19s 920us/sample - loss: 660.6106 - mae: 15.5295 - mse: 660.6107
Epoch 6/20
20400/20400 [==============================] - 19s 917us/sample - loss: 612.1501 - mae: 14.8455 - mse: 612.1500
Epoch 7/20
20400/20400 [==============================] - 19s 916us/sample - loss: 587.3101 - mae: 14.4393 - mse: 587.3103
```

Fig. 2 Example of training epochs

highest accuracy. The input layer for all variations of the network is equal to the number of features in the data set, while the output layer is just a single neuron to give a prediction in minutes. The number of neurons in the hidden layers is half of the size of the input layer. Fivefold cross-validation was used for the model with a split of 20% for test data and 80% for training data. We report the quality of the models in form of the mean absolute error (MAE) and the mean square error (MSE) used as typical loss functions and suggested by the TensorFlow software.

Table 7 shows the results for several different layouts and iterations for prediction of the planned duration of the orders. Notable is the significant difference between layouts with "good" predictions and those with significant losses. One group of layouts shares very similar results in the range of 8.5–9.5 MAE and 180–250 MSE while another group of layouts shows losses with really large or infinitely high (NaN) losses. Multiple runs with these layouts manifested this image, as the losses for the "good" layouts stayed in the aforementioned ranges, and the "bad" layouts continually produced weak predictions. Interesting is the gain in accuracy and performance in the models that used the Principal Component Analysis (PCA) to reduce the number of features by combining the existing features mathematically into a smaller number of features. The gain in accuracy could be due to reduced complexity and thus a lower probability of finding local minima. The gain in performance is following the reduction in complexity, as a tenfold decrease in the number of features in the models led to a roughly tenfold decrease in training duration.

When training the identical layouts on the self-reported labels, a similar picture emerges. A group of predictions in the area of 9–11 MAE and 200–300 MSE for different layouts was reproducible for several neural network

Table 7 Comparison of different prediction accuracies for regression with planned label data

PCA	Activation function	Optimizer	Layers	MAE	MSE	Training Duration
No	None[a]	RMSProp	2	8.23	252.78	800 s
No	Linear	RMSProp	2	8.80	316.18	800 s
No	Exponential	RMSProp	2	NaN	NaN	800 s
No	ReLu	RMSProp	2	32.06	2177	800 s
No	None	Adam	2	10.22	395.31	500 s
No	None	AdaGrad	2	9.80	361.36	700 s
No	None	SGD	2	NaN	NaN	300 s
No	None	RMSProp	1	9.62	322.84	700 s
No	None	RMSProp	3	8.29	350.66	1000 s
Yes	None	RMSProp	2	**8.70**	**169.79**	90 s
Yes	Linear	RMSProp	2	9.67	229.67	90 s
Yes	Exponential	RMSProp	2	9.58	345.42	100 s
Yes	ReLu	RMSProp	2	**8.29**	**182.05**	90 s
Yes	None	Adam	2	10.10	207.09	80 s
Yes	None	AdaGrad	2	10.23	370.31	80 s
Yes	None	SGD	2	NaN	NaN	90 s
Yes	None	RMSProp	1	**9.03**	**188.37**	80 s
Yes	None	RMSProp	3	10.36	358.14	110 s

[a]I.e. the input equals the output, no function applied

layouts, while some layouts continuously produced bad results (Table 8).

The differences between the range of predictions for the planned and self-reported label data are interesting. The models for the planned labels produced better results for the MAE loss and worse results for the MSE loss when compared to the results of the models trained on the self-reported labels. This seems to indicate that the self-reported label is more difficult to predict on average (higher MAE) but easier to predict for longer duration orders (lower MSE).

The reasons for this difference in quality could, therefore, be twofold. On one hand, the self-reported labels could be influenced more strongly by factors that have not been available as features to the model. This would make predictions more difficult in general and lead to an increase in MAE overall.

On the other hand, the self-reported label could be reported with higher accuracy for longer orders compared to the planned label. Small factors such as the floor to which an order is delivered do not have an enormous impact for furniture that takes 3 h to assemble. This could result in the model having a lower unexplainable variance for orders with longer duration and thus a lower MSE loss.

When comparing the MSE loss of the ANN with the loss of the MLE from the previous section, the higher predictive quality becomes evident. Interestingly, the MLE produced a smaller MSE loss on the self-reported label as well. This is another indicator of structural differences in the quality of the labels or the way the information was collected.

Comparing the predictions of the multivariate linear regression from the previous section with the predictions of the ANN graphically, the better prediction quality also becomes visually noticeable.

Figure 3 shows 6000 orders that were predicted by the MLE, and Fig. 4 shows the same number of orders predicted by the ANN. Each point represents the predicted value for a delivery and the actual time that was reported (label). For perfect accuracy, all points in the scatterplot would follow a straight line from the 0, 0 coordinate at the bottom to the 300, 300 coordinate in the top right. In direct comparison, the difference in predictive quality shown numerically in Table 9 is shown graphically in more details.

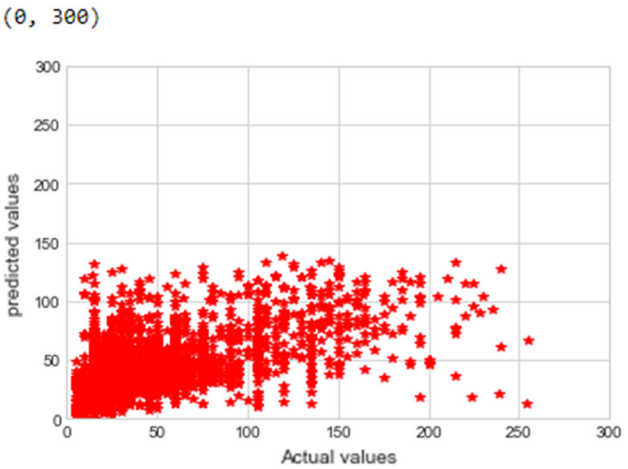

Fig. 3 Graphical representation of MLE predictions and actual values

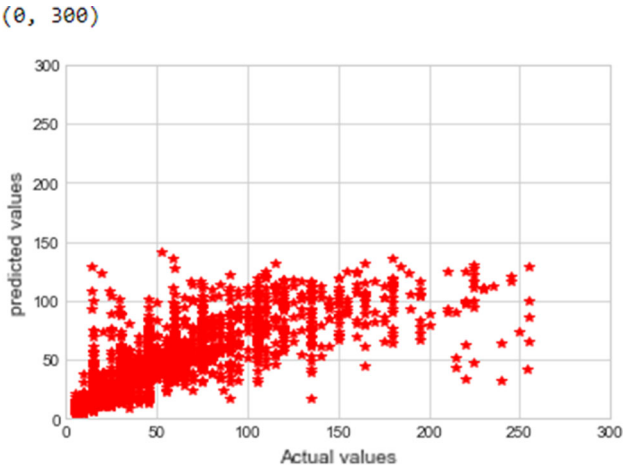

Fig. 4 Graphical representation of ANN predictions and actual values

The points in the plot in Fig. 3 (MLE predictions) spread out in a larger pattern, away from the imaginary line from the bottom left to the top right in the diagram. The

Table 8 Comparison of different prediction accuracies for regression with self-reported label data

Activation function	Optimizer	Layers	MAE	MSE	Training Duration
None	RMSProp	2	9.62	201.26	820 s
Linear	RMSProp	2	9.39	209.68	820 s
Exponential	RMSProp	2	NaN	NaN	800 s
ReLu	RMSProp	2	33.74	2172.19	780 s
None	Adam	2	12.02	285.31	440 s
None	AdaGrad	2	11.55	282.68	680 s
None	SGD	2	NaN	NaN	240 s
None	RMSProp	1	10.24	253.40	660 s
None	RMSProp	3	11.16	301.20	920 s

Table 9 Comparison of prediction results for regression

	MSE	Label
MLE	625	Planned
ANN	180–250	Planned
MLE	440	Self-Reported
ANN	200–300	Self-Reported

points in Fig. 4 (ANN predictions) spread out less and thus indicate a better prediction quality. Interestingly, both models are reluctant to predict orders with longer durations.

6 Conclusions

The main objective of this research paper was the evaluation of machine learning techniques to predict the time required to execute orders and assembly by professionals at customer locations.

The process of this evaluation started with the analysis of related works and the research of various machine learning techniques in the literature review. After finding several cases in different business domains but similar problem domains, it was concluded that there is a research gap for the prediction of these durations in the context of home deliveries.

After collecting additional information on available frameworks and functionalities, the data required for the research was collected, analyzed, and transformed. This allowed the exclusion of outliers in the orders, structural analysis of some individual columns and order types, and the evaluation of which features are required to predict the two label candidates as accurately as possible.

Finally, the problem domain was evaluated as a regression or a categorization problem, with two separate techniques to compare them with each other. In both cases, the artificial neural network provided the most accurate predictions, with a rather small gap when compared with a support vector machine for categorization and a distinctive gap in accuracy when compared with a multivariate linear regression.

Not only do these results indicate that the ANN was useful for predicting, but it also seems that the accuracy of the predictions is high enough to be useful to the company providing the data. Some considerations when implementing the results into a practical application are the quality of the data collected and what the tolerances are for error or impreciseness. Especially in the case of the categorization, the size of the category is an essential factor for accuracy and needs to be weighed against the fuzziness of the predictions.

7 Outlook

The available candidates for labels are flawed in that the self-reported duration of an order could be biased, because the service technicians may report inaccurate data. At the same time, the planned order duration has the obvious disadvantage of being inaccurate, because it is only a human estimation that may not consider all the essential factors that influence the order duration. Fortunately, the company providing the data set has recently implemented digital location and duration tracking via GPS for all service technicians. This has allowed collecting unbiased data directly from an IT system. When the available data set with such durations is large enough, this project will be revisited, and the model will be retrained with the new label.

Author contributions JW conducted the main part of the study and wrote the first draft of the paper. TH supervised the study and revised the paper draft.

Funding Open access funding provided by FHNW University of Applied Sciences and Arts Northwestern Switzerland. No funding was received to assist with the preparation of this manuscript.

Data availability Due to confidentiality reasons, the company-specific data used in the analysis cannot be publicly provided.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose. There is no conflict of interest related to this paper.

Ethical approval Ethics approval is not required for this type of study.

Informed consent Informed consent was not required from the coworkers of the involved company of the study as the analysis used anonymous data only.

References

Ahn B-H, Shin J-Y (1991) Vehicle-routing with time windows and time-varying congestion. J Operational Res Soc 42(5):393–400

Beaujon G, Turnquist M (1991) A model for fleet sizing and vehicle allocation. Transp Sci 25(1):1–97

Brownlee J (2020) Random oversampling and undersampling for imbalanced classification. Accessed 15 June 2020 at Machine Learning Mastery: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Coelho L, Gagliardi J-P, Renaud J, Ruiz A (2016) Solving the vehicle routing problem with lunch break arising in the furniture delivery industry. J Operational Res Soc 67(5):743–751

Colas F, Brazdil P (2006) Comparison of SVM and some older classification algorithms in text classification tasks. Artificial intelligence in theory and practice, p. 169–178. Santiago, Chile: IFIP AI 2006

Collins J, Sisley E (1994) Automated assignment and scheduling of service personnel. IEEE Expert 9(2):33–39

Desrosiers J, Dumas Y, Solomon M, Soumis F (1995) Time constrained routing and scheduling. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Handbooks in operations research and management science, vol. 8, p. 35–139. Elsevier Science B.V., Amsterdam

Fries C, Vinholi L, Vargas D (2018) A routing vehicles model for the delivery of furniture and home appliances of a retail company in southern Brazil. Proceedings of the International Conference on Industrial Engineering and Operations Management (p. 986–993). Pretoria/Johannesburg, South Africa: IFEES

Golden B, Wasil E (1988) Vehicle routing: methods and studies. North-Holland, Amsterdam

Guo W.-d., Wei F.-j. (2011) Application of grey prediction models in final assembly duration of civil aircraft. 2011 IEEE 18th International Conference on Industrial Engineering and Engineering Management. Changchun, China: IEEE

Kooijman C (2019) Personal communication: phone interview. (J. Wolter, Interviewer)

Laporte G (1992) The Traveling Salesman Problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(2):231–247

Metz C (2015) Google Just Open Sourced TensorFlow, Its Artificial Intelligence Engine. Retrieved 07 Dec 2019, from Wired: https://www.wired.com/2015/11/google-open-sources-its-artificialintelligenceengine/

Ng NH, Gabriel R, McAuley J, Elkan C, Lipton Z (2017) Predicting surgery duration with nerual heteroscedastic regression. Proceedings of the 2nd Machine Learning for Healthcare Conference, p. 100–111

Russel S, Norvig P (2020) Artificial intelligence: a modern approach (20 Ausg.). Pearson Education, Upper Saddle River

Sörensen K, Sevaux M, Schittekat P (2008) "Multiple neighbourhood" search in commercial VRP packages: evolving towards self-adaptive methods. Adaptive and multilevel metaheuristics. Springer, Berlin, pp 239–253

Suzuki N, Imashiro M, Sakata M, Yamamoto M (2017) The effects of group size in the furniture assembly task. Human interface and the management of information: supporting learning, decision-making and collaboration, p. 623–632. Springer, Vancouver

TensorFlow (2020a) Build a linear model with estimators. Accessed 24 May 2020a at TensorFlow Core: https://www.tensorflow.org/tutorials/estimator/linear

TensorFlow (2020b) Module: tf.keras.losses. Accessed 06 June 2020b at TensorFlow Core: https://www.tensorflow.org/api_docs/python/tf/keras/losses

TensorFlow (2020c) tf.estimator.LinearRegressor. Accessed 11 June 2020c at TensorFlow API documentation: https://www.tensorflow.org/api_docs/python/tf/estimator/LinearRegressor

Vlahogianni E, Karlaftis M (2013) Fuzzy-entropy neural network freeway incident duration modeling with single and competing uncertainties. Comput-Aided Civil Infrastructure Eng 28(6):420–433

Vössing M (2017) Towards managing complexity and uncertainty in field service technician planning. 2017 IEEE 19th Conference on Business Informatics (CBI). IEEE, Karlsruhe

Wongvorachan T, He S, Bulut O (2023) A comparison of undersampling, oversampling, and SMOTE methods for dealing with imbalanced classification in educational data mining. Information 14(1):54