Les Cahiers du GERAD ISSN: 0711–2440

Learning-based prediction of conditional wait time distributions in multiskill call centers

M. Thiongane, W. Chan, P. L'Ecuyer

G–2020–84

December 2020

Citation suggérée : M. Thiongane, W. Chan, P. L'Ecuyer (Décembre 2020). Learning-based prediction of conditional wait time distributions in multiskill call centers, Rapport technique, Les Cahiers du GERAD G–2020–84, GERAD, HEC Montréal, Canada.

Avant de citer ce rapport technique, veuillez visiter notre site Web (https://www.gerad.ca/fr/papers/G-2020-84) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

Suggested citation: M. Thiongane, W. Chan, P. L'Ecuyer (December 2020). Learning-based prediction of conditional wait time distributions in multiskill call centers, Technical report, Les Cahiers du GERAD G–2020–84, GERAD, HEC Montréal, Canada.

Before citing this technical report, please visit our website (https://www.gerad.ca/en/papers/G-2020-84) to update your reference data, if it has been published in a scientific journal.

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2020
– Bibliothèque et Archives Canada, 2020

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2020
– Library and Archives Canada, 2020

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Learning-based prediction of conditional wait time distributions in multiskill call centers

Mamadou Thiongane [a]

Wyean Chan [b]

Pierre L'Ecuyer [b]

[a] *Department of Mathematics and Computer Science, University Cheikh Anta Diop, Dakar, Sénégal*

[b] *GERAD & Département d'Informatique et de Recherche Opérationnelle (DIRO), Université de Montréal, Montréal (Québec), Canada, H3C 3J7*

mamadou.thiongane@ucad.edu.sn
chanwyea@iro.umontreal.ca
lecuyer@iro.umontreal.ca

December 2020
Les Cahiers du GERAD
G–2020–84

Abstract : Based on data from real call centers, we develop, test, and compare forecasting methods to predict the waiting time of a call upon its arrival to the center, or more generally of a customer arriving to a service system. We are interested not only in estimating the expected waiting time, but also its probability distribution (or density), conditional on the current state of the system (e.g., the current time, queue sizes, set of agents at work, etc.). We do this in a multiskill setting, with different call types, agents with different sets of skills, and arbitrary rules for matching each calls to an agent. Our approach relies on advanced regression and automatic learning techniques such as spline regression, random forests, and artificial neural networks. We also explain how we select the input variables for the predictors.

Keywords: Delay prediction, wait time, distributional forecast, automatic learning, service systems, multiskill call centers

Acknowledgements: This work was supported by grants from NSERC-Canada, Hydro-Québec, and a Canada Research Chair to P. L'Ecuyer. Ger Koole (VU Amsterdam) helped by providing data.

1 Introduction

1.1 How long will I wait?

You make a phone call to reach your airline, bank, utility, or credit card provider, or some government service, and get a too familiar message: "All our agents are currently busy. Your call is very important to us. Please hold and one of our representatives will be with you as soon as possible." or "We currently experience a larger volume of calls than usual." Most often, the message gives no information on how much you have to wait. Sometimes, it provide a forecast of your waiting time, but the forecast can be quite inaccurate and you have no idea of its accuracy, so it may be more misleading than useful.

In a dream situation, you would be told upon arrival the exact time when the service will start. Then you could do some other activity and show up (or pick the phone) only at the right moment. Of course, this is unrealistic. The wait time is usually random and often hard to predict. A more reasonable wish could be to receive an estimate of your *expected* wait time, conditional on the current state of the system upon arrival. This conditional expectation can be computed exactly for very simple models, but in more realistic and complex systems with multiple types of customers, different types of servers, and nontrivial routing and priority rules, it is generally hard to compute and even to approximate.

Telling the customer its conditional expected wait time upon arrival only provides limited and unsatisfactory information. If you are told "your predicted waiting time is 18 minutes" with no additional qualification, and you end up waiting 28 minutes, or you come back after 15 minutes only to find that you have missed your turn, you may conclude that those predictions are not so useful in the end. An improvement can be to provide a *prediction interval* (PI), such as "we predict with 95% confidence that your waiting time will be between 13 and 25 minutes." Or even better, a plot of the density of the waiting time distribution conditional of the current state of the system, with PIs indicated on the plot. The aim of this paper is to propose methods that can compute such *distributional forecasts* and compare their performance on real data taken from a call center.

Figure 1 gives an example of what information could be shown on the phone screen of a customer, for a call made at 13:00:00. The plot gives an estimate of the conditional density of the time at which the call will be answered, with a red line marking the expectation and two green lines indicating a 90% PI (they are at the 5% and 95% quantiles of the predicted distribution). The estimated expected answering time is at 13:05:43, and the PI is (13:03:56, 13:08:17). A key question is: How can we construct such predictors?

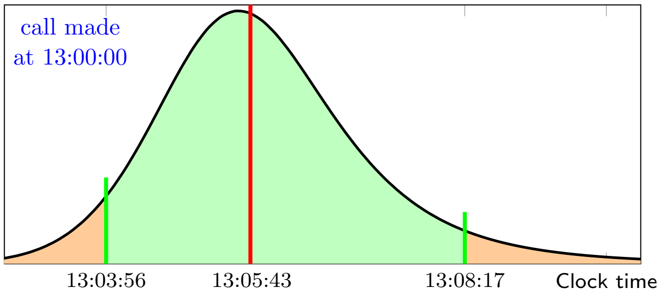

Figure 1: Predicted distribution of the answering time of a call when it arrives, at 13:00:00. The red line marks the expected answering time and the two green lines are the boundaries of a 90% PI. Each tail filled in orange contains 5% of the probability density.

One simple model for which the exact distribution of the wait time can be computed when the call arrives, conditional on the number of waiting calls (the queue length) at that time, is a G/M/s queue [7, 17, 26]. In that model, the arrival process is arbitrary, but there is a single call type, the calls are answered by order of arrival, there are s identical servers, each one handling one call at a time, and the service times are assumed independent with an exponential distribution of rate μ. In that case, if all servers are busy and there are q other calls waiting in queue when a call arrives, then the wait time of this arriving call is the sum of $q+1$ independent exponential random variables with rate μ. This is an Erlang random variable with shape parameter $q+1$ and rate parameter μs. It has mean $(q+1)/(\mu s)$, variance $(q+1)/(\mu s)^2$, and density $f(x) = (\mu s)^{q+1} x^q e^{-\mu s x}/q!$ for $x > 0$. The explanation is that the arriving call must wait for $q+1$ ends of service before getting attention, and the times between the successive ends of service when the s servers are busy are independent and exponential with rate μs. Independent exponential service times are rather unrealistic and, more importantly, in this paper we are interested in multiskill call centers, with different call types and separate groups of agents having different skill sets (each group can handle a different subset of the call types). No analytic formula for the wait time density or expectation is available for this case, and making the conditional predictions is much harder.

Our discussion so far was in terms of phone calls to a call center, but the models and methods studied in this paper apply more generally to "customers" or "users" who have to wait for a service; for example patients arriving to a medical clinic, or people grabbing a numbered ticket on their arrival to a store or at some governmental service such as the passport office. In the following, we use "call" and "customer" interchangeably, an "agent" is the same as a "server", and the "wait time" is also called "delay."

1.2 Brief review of earlier work

Previous research on wait time prediction was mostly for systems with a single type of customers and identical servers. There are two main categories of proposed prediction methods for that case: the *queue-length* (QL) predictors, and the *delay-history* (DH) predictors. QL predictors use the queue length at the customer arrival, together with some system parameters, to predict the wait time. The exact analytic formula given earlier for the G/M/s example is an example of a QL predictor. It can predict not only the expectation, but also the density of the wait time, and it is the best possible predictor in that situation. This type of QL predictor has been studied by [21, 23, 24, 39]. DH predictors use the wait times of the previous customers to predict the wait time of a new arriving customer; see [2, 11, 22, 30, 36]. For example, one can predict the wait time of this customer by the wait time of the most recent customer of the same type who had to wait and already started its service (so we know its wait time), or maybe the average wait time of a few of those (e.g., the three to ten most recent ones). There are other variants. These predictors are generally designed to produce only a point estimate, i.e., only predict the expected wait time. They are further discussed in Section 2.1. For queueing systems with multiple customer types but a single group of identical agents that can handle all types, delay predictors based on QL and DH have beed examined in [32].

For the general multiskill setting with multiple types of customers, where each type has its own queue, each server can handle only a subset of these types, and the matching between customers and servers can be done using complicated rules, predicting the delay is much more difficult. Very little has been done so far for this situation. A simple QL predictor that looks only at the queue length for the type of the arriving customer does not work well, because it neglects too much information; e.g., the lengths of the queues for the other types that the agents serving this type can also serve. If the agents that can serve the arriving type are too busy serving some other types, then the arriving customer can wait much longer. It appears difficult to extend the QL predictors to multiskill settings. For this reason, earlier work used mostly DH predictors. They are easy to apply even for complicated multiskill systems, but unfortunately they often give large prediction errors. See [35, 36] and Section 2.1.

The basic idea of a *point predictor* for the delay is to select a set of input variables that represent the current relevant information (the state of the system), and define a real-valued *predicting function* of these variables that would represent the expected delay given the current information. This is a multivariate function approximation problem. Given available data, finding a good candidate function is a (nonlinear) multivariate regression problem. There are many ways to perform this regression and some have been examined in the literature. In particular, Thiongane et al. [35] proposed and compared such data-based delay predictors for multiskill queueing systems, using *regression splines* (RS) and *artificial neural networks* (ANN) to define (or learn) the predicting function. The input variables were the lengths of selected queues and the wait time of the most recent customer of the same type having started its service. The different proposed methods were compared on simulated models of call centers. In a similar vein, *lasso regression* (RL) was explored in [1] to predict wait times in emergency health-care units. The input variables included the queue lengths for the different priority levels, the overall load, etc. Some of these variables correspond to QL predictors that are not applicable for general multiskill call centers. Nevertheless, RL is also usable for multiskill call centers, with appropriate inputs. In this health-care application, the patients are classified by priority levels (the priority corresponds to the customer type), and the agent groups are defined differently than by subsets of the types; they may correspond to doctors, nurses, assistants, etc.

Thiongane et al. [37] made further comparisons between various types of DH predictors and regression-based (or learning-based) methods, including multilayer feed-forward neural networks, using data taken from a real call center. They also proposed a method to select a set of relevant input variables. The performance of these regression-type predictors depend very much on which inputs variables are considered, and on how much relevant data is available for the learning. If important variables are left out, the performance may degrade significantly. If there is too little data, or if the current setting differs too much from the one in which the data was obtained, then the forecasting error is also likely to be large. Too little data combined with too many input variables also lead to overfitting.

All the methods and papers discussed so far for the multiskill systems are for *point predictions* only, i.e., to predict the conditional expected wait time (a single number), and not to predict its distribution. A few papers examine quantile prediction of the waiting times of patients in health-care emergency departments[10, 34]. The authors predict the median, 10%, 90% or 95% percentiles of the wait times using multiple regression.

1.3 Contribution and outline

The present paper is follow up to [37]. As an important extension, we propose methods to estimate the *conditional density* and *quantiles* of the wait time, as in Figure 1, instead of only the expectation. We also report additional experiments, and use data from a second call center.

The rest of the paper is organized as follows. In the next section, we specify and discuss the DH and regression-based delay predictors considered in our experiments. In Section 3, we explain how we propose to estimate the conditional density of the wait time. In Section 4, we describe the experiment setup, and in Section 5 and 6, we report on numerical experiments with data from two different call centers. The first one is the call center of an Israeli bank, and the second one is from an information technology (IT) company in the Netherlands. For each one, we describe the data, and we compare our different point predictors and density predictors for the delay. Section 7 provides a conclusion.

2 Point predictors for multiskill systems

We now discuss different types of point predictors used and compared in our experiments. These predictors return a single number, which may be interpreted as an estimate of the expected delay. They are all “learning-based” in some sense, although for the DH predictors, the learning is based only on the delays of the very recent customers. We exclude QL predictors, since they are not adapted to multiskill settings.

2.1 Delay-history predictors

In a multiskill call center, a DH predictor estimates the wait time of a new arrival by looking only at the delays of the most recent customers of the same type who started their service. There is no learning based on lots of data to estimate function parameters, so these predictors are simple and easy to implement. The DH predictors discussed here are the best performers according to our previous experiments in [35, 36, 37].

The simplest and most popular DH predictor is the *Last-to-Enter-Service* (LES) predictor. It returns the wait time of the most recent customer of the same type among those who had to wait and have started their service [22].

A generalization often used in practice [11] is to take the N most recent customers of the same type who had to wait and have started their service, for some fixed positive integer N, and average their wait times. This is the *Averaged LES* (Avg-LES).

One variant of the Avg-LES also takes the average of the wait times of past customers of the same type who had to wait and have started their service, but the average is only over the customers who found the same queue length as the current one when they arrived. This predictor was introduced in [36] and was the best performing DH predictor in the experiments made in that paper. It is called the *Average LES Conditional on Queue Length* (AvgC-LES).

Another one is the *Extrapolated LES* (E-LES), defined as follows [36]. For a new arriving customer, it looks at the delay information of all customers of the same type that are *currently waiting in queue*. The final delays of these customers are still unknown, but the (partial) delays elapsed so far are extrapolated to predict the final wait times of these customers. E-LES returns a weighted average of these extrapolated delays, as explained in [36].

The *Proportional Queue LES* (P-LES) predictor starts with the delay d of the LES customer and makes an adjustment to account for the difference in the queue length q_{LES} at the arrival of this LES and the current queue length q when the new customer arrives [20]. The adjusted predictor is

$$D = d(q+1)/(q_{\text{LES}}+1).$$

2.2 Regression-based predictors

Regression-Based Predictors construct a multivariate *predictor function* of selected input variables deemed important. It approximates the conditional expectation of the delay W of an arriving customer of type k, conditional on the current state of the system, which is represented by the vector $\mathbf{x}$ of these input variables. The predictor function for customer type k is $p_{k,\theta(k)}$, where $\theta(k)$ is a vector of parameters which depends on k. It must be estimated (learned) from the data in a training step. The predicted delay when in state $\mathbf{x}$ will be $p_{k,\theta(k)}(\mathbf{x})$. Constructing this type of predictor involves three main parts that are inter-related, for each k: (a) selecting which variables to put in $\mathbf{x}$; (b) selecting the general form of $p_{k,\theta(k)}$; and (c) estimating (learning) the parameter vector $\theta(k)$. In the remainder of this section, we explain how we have implemented these three parts in our experiments. The method used for part (c) depends very much on the choice of predictor function in part (b). For that, we will consider and compare the following three choices: (1) a smoothing (regression) cubic spline additive in the input variables (RS), (2) a lasso (linear) regression (LR), and (3) a deep feedforward multilayer artificial neural network (ANN).

2.2.1 Identifying the important variables

In the G/M/s queuing system discussed earlier, the analytic formula tells us clearly that the only important input variables are the number s of servers and the number q of customers in the queue. Everything else is irrelevant. For more complex multiskill systems, however, identifying the most relevant inputs for the prediction for a given customer type k is not so simple. Leaving out important variables is bad, and keeping too many of them leads to overfitting, as is well-known from regression theory.

For our numerical examples with the call center data, we used the following methodology. We started by putting as candidates all the observable variables that could have a chance of helping the prediction, then we used a feature selection algorithm to perform a screening among them. The candidate input variables were the following: the vector $\mathbf{q}$ of current queue lengths for all call types; the number s of agents that are serving the given call type, the total number n of agents currently working in the system, the current time t (when the call arrives), the wait time of the N most recently served customers of the given call type, and the delay predicted by the DH predictors LES, P-LES, E-LES, Avg-LES, and AvgC-LES.

To screen out and make a selection among these inputs, we used a technique based on the *random forest* (RF) bootstrapping methodology of Breiman [6]. Among the various feature selection algorithms based on this methodology, we picked *Boruta* [27], which was the best performer in empirical comparisons between many selection algorithms in [9]. The general idea of random forests is to generate a forest of decision trees whose nodes correspond to the selection decisions for the input variables. Boruta extends the data by adding copies of all input variables, and reshuffles these variables to reduce their correlations with the response. These reshuffled copies are named the *shadow features*. Boruta then runs a random forest classifier on this extended data set. It makes bootstrap samples on the training set and constructs decision trees from these samples. The importance for each input variable is assessed by measuring the loss of accuracy of the model when the values of this input are permuted randomly across the observations. This measure is called the *mean decrease accuracy*. It is computed separately for all trees of the forest that use the given input variable. The average and standard deviation of the loss of accuracy is then computed for each input, a Z score is obtained by dividing the average loss by its standard deviation, and this score is used as the importance measure. The maximum Z-score among the shadow features (MZSA) is used to select the variables deemed useful to predict the delay. The input variables are then ranked according to these scores, and those with the highest scores are selected. The variables whose Z-scores are significantly lower than MZSA are declared “unimportant”, those whose Z-scores are significantly higher than MZSA are declared “important” [27], and decisions about the other ones are made using other rules.

2.2.2 Measuring the prediction error

The parameter vector θ is estimated by minimizing the *mean squared error* (MSE) of point predictions. That is, if $E = p_{k,\theta(k)}(\mathbf{x})$ is the predicted delay for a given customer of type k who receives service after some realized wait time W, the MSE for type k calls is

$$\mathrm{MSE}_k = \mathbb{E}[(W - E)^2].$$

This expectation cannot be computed exactly, but we can estimate it by its empirical counterpart, the *average squared error* (ASE), defined as

$$\mathrm{ASE}_k = \frac{1}{C_k} \sum_{c=1}^{C_k} (W_{k,c} - E_{k,c})^2 \tag{1}$$

for customer type k, where C_k is the number of customers of type k who had to wait and for which we made a prediction. In the end, we use a normalized version of the ASE, called the *root relative average squared error* (RRASE), defined as the square root of the ASE divided by the average wait time of the C_k served customers, rescaled by a factor of 100:

$$\mathrm{RRASE}_k = \frac{100\,(\mathrm{ASE}_k)^{1/2}}{(1/C_k)\sum_{c=1}^{C_k} W_{k,c}}.$$

We estimate the parameter vector $\theta(k)$ for each k in this way from a learning data set that represent 80% of the collected data. The other 20% of the data is saved to measure and compare the accuracy of these delay predictors.

2.2.3 Regression splines (RS)

Regression splines (RS) are a powerful class of approximation methods for general smooth functions [8, 25, 40]. Here we use smoothing additive cubic splines, for which the parameters are estimated by least-squares regression after adding a penalty term on the function variation to favor more smoothness. If the information vector is written as $\mathbf{x} = (x_1, \ldots, x_D)$, the additive spline predictor can be written as

$$p_{k,\theta(k)}(\mathbf{x}) = \sum_{d=1}^{D} f_d(x_d),$$

where each f_d is a one dimensional cubic spline. The parameters of all these spline functions f_d form the vector θ. We estimated these parameters using the function `gam` from the R package `mgcv` [41].

2.2.4 Lasso Regression (LR)

Lasso Regression is a type of linear regression [13, 25, 38] with a penalty term proportional to the sum of absolute values of the magnitude of coefficients, added to reduce overfitting. The LR predictor is

$$p_{k,\theta(k)}(\mathbf{x}) = \beta_0 + \sum_{d=1}^{D} \beta_d \cdot x_d,$$

where $\mathbf{x} = (x_1, \ldots, x_D)$ is the input vector, as in ordinary linear regression, but the vector of coefficients $\theta(k) = (\beta_0, \beta_1, \ldots, \beta_D)$ is selected to minimize the sum of squares of errors plus the penalty term. To estimate the parameter vector $\theta(k)$, we used the function `glmnet` in the R package `gmlnet` [12].

2.2.5 Artificial neural networks (ANN)

An *artificial neural network* (ANN) is an effective tool to approximate complicated high-dimensional functions [4, 28]. Here we use a deep feedforward ANN, which contains one input layer, one output layer, and a few intermediate (hidden) layers. The outputs from the nodes at layer ℓ are the inputs for all nodes at layer $\ell + 1$. Each node of the input layer corresponds to one element of the input vector $\mathbf{x}$. The output layer has a single node, which returns the predicted delay. At each hidden node, we have a rectifier activation function of the general form $h(\mathbf{z}) = \max(0, b + \mathbf{w} \cdot \mathbf{z})$, where $\mathbf{z}$ is the input vector for this node, whereas b and the vector $\mathbf{w}$ are parameters learned by training [14]. At the output node, to predict the delay, we use a linear activation function of the form $h(\mathbf{z}) = b + \mathbf{w} \cdot \mathbf{z}$ where $\mathbf{z}$ is the vector of outputs from the nodes at the previous hidden layer. Here, the vector θ represents the set of all the parameters b and $\mathbf{w}$, over all the nodes of the ANN. Good parameter values are learned by a back-propagation algorithm that relies on a stochastic gradient descent method. Several hyperparameters used in the training are determined empirically. For a guide on training, see [3, 5, 15, 18]. For this paper, we did the training using the Pylearn2 software [16].

3 Density predictors

We saw that for a G/M/s queue, the delay of a customer who finds q waiting customers in front of him upon arrival has an exact Erlang distribution whose density has an explicit formula. But for more complex multiskill systems, the distribution of the delay conditional on the current state of the system has an unknown density which is likely to be very complicated and is much harder to estimate. Estimating a general univariate density in a non-parametric way, from a given data set coming from this density, is already a difficult problem in statistics. With the best available methods, e.g., kernel density estimators, the error in the density estimate converges at a slower rate than the canonical rate of $\mathcal{O}(n^{-1/2})$ as a function of the number n of observations [31]. Estimating a conditional density which is a multivariate function of several input variables is even more difficult.

After a few initial attempts, we decided not to estimate directly the density of the delay time conditional on $\mathbf{x}$, but we consider alternatives that fit the prediction errors to a simpler parametric model conditional on selected information from $\mathbf{x}$, and kernel density estimators that depend on this limited information. We take our point predictor of the delay (the estimate of the expected wait time), and add to it the estimated density of the prediction error. That is, for each predictor and each call type, we first estimate (or learn) the parameters of the point predictor, then compute the prediction error for each customer. After that, we fit a parametric or non-parametric density to these errors, or we train a learning algorithm to model these errors conditional on $\mathbf{x}$. This provides an estimate of the density of the prediction error. By adding the point predictor to this density, we obtain a predicted density for the delay.

Specifically, let us write the wait time W of a customer as

$$W = p_{k,\theta(k)}(\mathbf{x}) + \epsilon(k, \mathbf{x}), \tag{2}$$

where $\epsilon(k, \mathbf{x})$ is the prediction error. The idea is to estimate the density of $\epsilon(k, \mathbf{x})$ and this will give us the density of W, since $p_{k,\theta(k)}(\mathbf{x})$ is a constant. The density of $\epsilon(k, \mathbf{x})$ certainly depends on k and $\mathbf{x}$, so a key issue is how to model this dependence. A very simple solution would be to ignore the dependence on $\mathbf{x}$ and just pick a density for each k. A more refined solution is to partition the space of values of $\mathbf{x}$ in a small number of subsets and estimate one density for each subset. The number of subsets should not be too large, because it would lead to estimating too many different densities, and eventually to overfitting.

In this paper, we consider two different ways to partition the domain space of $\mathbf{x}$. The first approach uses clearly defined cuts of the space defined by selecting which *features* of $\mathbf{x}$ to emphasize and which ones are to be aggregated. The second method uses a learning-based algorithm, in which the partitions are not explicitly defined. Note that these approaches can be used to estimate the error density of any predictor, and are not limited to the RS, LR, and ANN methods presented in Section 2.

3.1 Parametric model conditional on queue length

To predict the mean wait time, we trained a function of the entire vector $\mathbf{x}$. But predicting an entire density as a function of $\mathbf{x}$ is more difficult. With the approach described here, we will make our prediction as a function of a more limited amount of information. We partition the space of $\mathbf{x}$ according to the most important predictive features for the waiting time, and aggregate the rest of the features. Suppose we are predicting the density for call type k. For our case studies, the Boruta algorithm described in Section 2.2 selected the queue length q_k of call type k as the most important feature. This motivates the idea of fitting a parametric probability model of the prediction errors $\epsilon(k, \mathbf{x})$ conditional only to the queue length q_k. That is, we would fit a different model for each value of q_k. By doing that, a lot of information in $\mathbf{x}$ is ignored, but we hope that much of the variability on $\epsilon(k, \mathbf{x})$ is captured by q_k.

In our numerical studies, we tried to fit several distributions to the realizations of $\epsilon(k, \mathbf{x})$ conditional on q_k, including the gamma, lognormal, log-logistic, and Weibull distributions, and shifted version of them. Shifting these distributions is necessary because $\epsilon(k, \mathbf{x})$ must be allowed to take negative values. That is, before fitting one of the distributions named above, we first shift the errors by adding an positive constant $\tau_k(q)$ large enough so that $\tau_k(q) + \epsilon(k, \mathbf{x}) > 0$ for all $\mathbf{x}$ with $q_k = q$. Then the distribution is fitted to the shifted observations. After that, a negative shift of $-\tau_k(i)$ is applied to obtain the distribution of the prediction error. For each call type k and queue length q_k, we used a grid search to find the best value for $\tau_k(q_k)$. In our case studies, the *shifted log-logistic* gave the best fit in most cases, and the best shift did not depend much on q_k. One explanation for this could be that both the mean and standard deviation of the delay increase approximately linearly in q_k, so the starting point of the distribution does not change much. As an illustration, Figure 2 compares the density functions of the errors fitted with the log-logistic, gamma, lognormal, and Weibull distributions, for some call type k in our data, conditional to $q_k = 2$. For this figure and for all the numerical results reported in the paper, we fitted the parametric models using the R package `fitdistrplus`, and the package `actuar` for the log-logistic distribution. For comparison, the figure also shows a *kernel density estimator* (KDE) conditional on type k and $q_k = 2$. This KDE was obtained as discussed in Section 3.2.

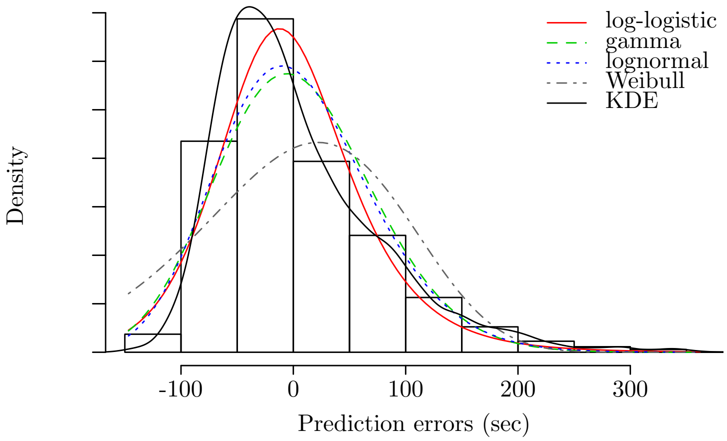

Figure 2: The distribution of the prediction error made by the ANN predictor, conditional to $q_k = 2$, for some call type k, and four parametric densities (log-logistic, gamma, lognormal, and Weibull) fitted to this data. The black curve is a KDE. Among the four parametric densities, the log-logistic gives the best fit.

For each k and each value of q_k until a certain threshold where the amount of training data becomes too small, we fit a different shifted log-logistic density. For queue lengths that have insufficient training data, we can pool together several values of q_k and fit one density for them. In the numerical section, we set the threshold to $q_k = 5$, and all states $\mathbf{x}$ for which $q_k > 5$ are pooled in a single group, for each k. Interestingly, when observing the empirical mean and standard deviation of the prediction error conditional to q_k, we find that we can also fit a simple model for the parameters of the log-logistic as a function of q_k. By design, the mean of the shifted log-logistic density for the prediction error should be zero (or near zero). We have observed that the standard deviation is not far from affine in q_k. Figure 3 gives an illustration. This means that we can fit a linear regression model for the standard deviation as a function of q_k, for each k. The scale and shape parameters of the log-logistic distribution can then be determined by the mean and standard deviation.

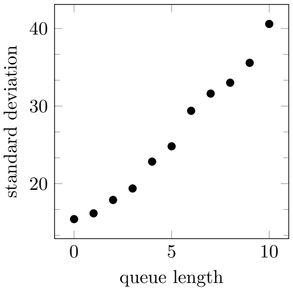

Figure 3: Standard deviation (in seconds) of the prediction error made by the ANN predictor conditional on the queue length, for one call type in the test data set.

According to Boruta, the second most important variable for the prediction after q_k is t, the arrival time of the call during the day. Therefore, to further refine the partition of the space of values of $\mathbf{x}$, we could model the density of the prediction error as a function of both q_k and t. We did not do that for the numerical experiments reported here, but this could be explored in the future.

3.2 A non parametric option: Using a Kernel density estimator

Parametric distributions are attractive because the entire density is defined by just a few parameters (the scale, shape, and location parameters for the shifted log-logistic). However, the true density function of the prediction error may have a shape that hardly matches any of the common families of distributions. A *kernel density estimator* (KDE) provides a much more flexible non-parametric solution. The KDE can be used directly to estimate the density of the prediction error. Figure 4 compares the KDE with the best fitted shifted log-logistic. We see a significant gap between the two densities for small queue length q_k. For this figure and for all the numerical results reported in this paper, the KDE was computed using the Epanechnikov kernel and the bandwidth was selected using the heuristic of Silverman [19, 33].

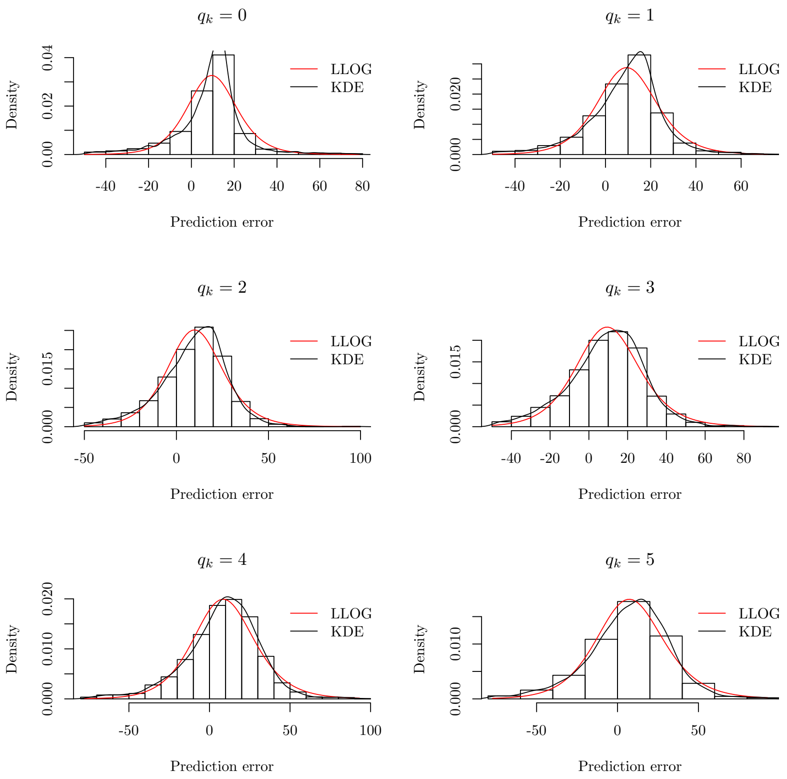

Figure 4: Log-logistic density (red) and KDE (black) for the prediction error (in seconds) made by an ANN predictor, conditional to queue lengths q_k from 0 to 5.

One drawback of the KDE is that it performs poorly when there are too few observations, and it is also difficult to extrapolate the density obtained from the KDE for frequent values of q_k to rare (larger) values, as we did with the parametric log-logistic density. For this reason, we may want to use a combined model: construct the KDE for queue lengths q_k that have a large number of observations, and use a parametric model to extrapolate the density for less frequent data points. Another possibility could be to use the KDE with a scaling factor proportional to q_k, but we did not do that.

3.3 Quantile regression with random forest

In the two previous subsections, we partitioned the space of values of $\mathbf{x}$ manually by reducing the number of dimensions on $\mathbf{x}$, and fitted a different distribution over each piece of the partition, for each k. This makes a lot of distributions and it can hardly be used unless we select only a few important features from $\mathbf{x}$, say one or two. In this section, we take a different approach. We do not partition explicitly the space of values of $\mathbf{x}$, but we rather leave this task to a learning-based algorithm. More specifically, we use a *quantile random forest* (QRF) algorithm that takes as input the entire vector $\mathbf{x}$, trains an ensemble of decision trees, and outputs the desired quantile and density functions. The partition rules learned by QRF can be very complex, much more than simply aggregating $\mathbf{x}$ conditionally to the queue length as in Section 3.1.

The QRF is implemented and trained identically to an ordinary RF, except that each leaf of a tree retains in memory the list of errors $\epsilon(k, \mathbf{x})$ for all training predictors $\mathbf{x}$ that ended in this leaf, instead of only the average $\epsilon(k, \mathbf{x})$ as in a RF. That is, QRF is not trained specifically to predict the quantiles or the density function, but it is a by-product of an RF trained to predict the expectation of $\epsilon(k, \mathbf{x})$. Each decision tree v of the QRF defines a complete partition of the space of $\mathbf{x}$, and consequently of the training data set, say $\hat{\mathbf{X}} = \cup_{l=1}^{L_v} \hat{\mathbf{X}}_l^v$, where L_v is the number of leaves in tree v. Identical values of $\mathbf{x}$ always fall in the same leaf of the tree v. However a leaf can be the end node for a large number of different states $\mathbf{x}$, with different prediction errors $\epsilon(k, \mathbf{x})$. The collection of these $\epsilon(k, \mathbf{x})$ is then used to estimate the probability distribution of $\epsilon(k, \mathbf{x})$ for the $\mathbf{x}$ that belong to that leaf.

After completing the training stage, QRF estimates the α-th quantile of the error $\epsilon(k, \mathbf{x}')$ for a given input $\mathbf{x}'$ as follows. For each decision tree v, it identifies the terminal leaf $l_v(\mathbf{x}')$ to which $\mathbf{x}'$ belongs, after traversing through the branches of the tree. Next, it computes the empirical α-th quantile from the saved list of all values of $\epsilon(k, \mathbf{x})$ for $\mathbf{x} \in \hat{\mathbf{X}}_{l_v(\mathbf{x}')}^v$ in the training set. These steps are repeated for every decision tree in QRF, and a weighted average of the α-th quantiles is finally returned.

To estimate the density conditional on some state $\mathbf{x}'$ with the QRF, we take the saved list of all values of $\epsilon(k, \mathbf{x})$ for $\mathbf{x} \in \hat{\mathbf{X}}_{l_v(\mathbf{x}')}^v$ in the training set, and merge these lists for all trees v into a single list of observations. Then a KDE is constructed by using these observations, to obtain a predicted density that depends on $\mathbf{x}'$. Figure 5 shows an example with five density functions estimated via QRF (in black) for five different vectors $\mathbf{x}$ with the same queue length $q_k = 2$. These densities are compared with the direct KDEs (in red) obtained as explained in Section 3.2, by using all the values of $\mathbf{x}$ for which $q_k = 2$. The QRF-based density estimator depends on more than just k and q_k.

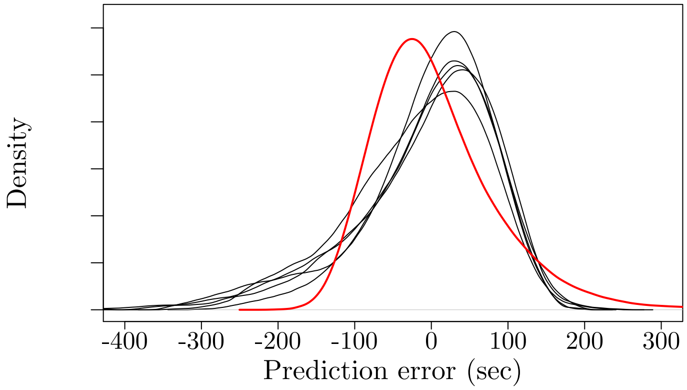

Figure 5: Densities predicted by QRF (in black) for five different input vectors $\mathbf{x}$, lying in different tree leaves but having the same queue size $q_k = 2$, for some call type k. The KDEs obtained for all $\mathbf{x}$ with $q_k = 2$ are shown in red, for comparison.

Training a QRF model requires substantially more computing power and CPU time than fitting a parametric distribution. It also involves many more parameters, which need to be saved after the training. On the other hand, we only need to train one QRF for each call type, since the quantile regression is conditional on the entire vector $\mathbf{x}$. QRF has likely a higher learning capacity than a parametric model, but it also faces a larger risk of overfitting. In Section 5, we compare it empirically to the more classical KDE and to parametric methods, with the call center data. In all our experiments, we use the QRF implemented in the R package `quantregForest` [29]. The learning capacity can be adjusted by changing the number and the depth of the decision trees, but the results reported in this paper were obtained with the default values.

4 Numerical experiment setup

Before presenting our numerical results, we describe our methodology for comparing the prediction algorithms for the expected delay times and the density functions. We have data for two call centers.

The first is the call center of a small bank in Israel and the second is from a large information technology (IT) company located in the Netherlands. We will refer to them as the Bank and IT call centers, respectively. The data set for each call center contains exactly one year of observations, from January 1 to December 31 (but for different year). We partition each data set chronologically as follows: the first 80% of the data is the *training set* and the remaining 20% is the *test set*.

First, we identify the important predictive features of the delay time by using the Boruta algorithm, as explained in Section 2.2. For both the Bank and IT call centers, the features vector was chosen as $\mathbf{x} = (\mathbf{q}, a, l, s, n, t)$ where $\mathbf{q}$ is the vector of queue sizes for all call types, a and l are the delay prediction by AvgC-LES and by LES, respectively, s is the number of agents that can serve this call, n is the total number of agents, and t is the time of arrival of the call.

We train the regression-based predictors (RS, LR, ANN) to predict the mean waiting time, using the training set. We compare their accuracy to those of the (simpler) DH predictors, which do not require any training, based on the RRASE score, computed over the test set.

To compare the density predictors, we select ANN as the point predictor of the mean delay time, because it often displays the best accuracy in our experiments. The training set for the density predictors is the set of prediction errors obtained by the ANN over its training set. Then, we fit or train a parametric log-logistic distribution (LLOG), a KDE, and a QRF.

Because the true density function of the prediction error is unknown, we compare the coverages of 90% PIs instead, as well as the coverages of the tails. That is, we use the estimated conditional density to compute a 90% PI on the answering time (or equivalently the wait time) of each call. Once we know the true answering time, we can check where it lies among the three following possibilities (1) to the left of the PI; (2) inside the PI; (3) to the right of the PI. Then we can compute the proportion of calls falling in each of the three categories. Ideally, there should be 5% in each of categories (1) and (3), and 90% in category (2). In our numerical experiments, we compare the observed proportions to these ideal ones. We also do the same with a 80% PI.

5 Experiments with data from a bank call center

5.1 The call center and available data

Our first data set is from the small call center of a bank in Israel, recorded over the entire year of 1999. This center operates from 7:00 to midnight on weekdays (Sunday to Thursday), and on weekends (Friday to Saturday) it closes at 14:00 on Friday and reopens at around 20:00 on Saturday. There are five inbound call types, one outbound type, and the center has eight working agents on average.

About 65% of the calls are served by the Interactive Voice Response (IVR) unit and leave without interacting with an agent. For the other 35%, the customer wants to speak to an agent. If no idle agent is available, the customer is placed in a queue and receives information on the queue size and the waiting time of the customer at the head of queue (the HOL predictor). The customers are served in first-come-first-served (FCFS) order. Table 1 gives a statistical summary of the arrival counts and wait times during the year. Type 1 has by far the largest volume.

Table 1: Statistical summary of arrival counts and waits for the Bank call center.

Call type	1	2	3	4	5
Total number calls	302 522	67 728	39 342	20 732	12 295
Served, no wait	42%	34%	36%	30%	44%
Served, waited	46%	36%	56%	51%	46%
Abandon	11%	30%	8%	19%	10%
Avg wait time (sec)	99	145	121	167	138
Avg service time (sec)	187	124	263	369	274
Avg queue length	6.3	2.5	2.0	1.7	0.9

5.2 Experimental results on predictions

We first report in Table 2 the RRASE for six predictors of the mean delay. We see that all the learning-based predictors are more accurate than the DH predictors. RS and ANN are the most accurate. Among the DH predictors, AvgC-LES gives the best performance and is not too far from RS and ANN for call type 1.

Table 2: The Bank call center: RRASE of the four largest-volume call types (lower is better). The RS and ANN are the most accurate predictors here.

	DH predictors			Learning-based predictors		
Type	Avg-LES	LES	AvgC-LES	RS	LR	ANN
1	0.925	0.941	0.765	0.731	0.737	**0.701**
2	0.980	0.990	0.933	0.762	0.849	**0.751**
3	0.922	0.941	0.853	**0.725**	0.750	0.737
4	1.348	1.541	1.320	**1.178**	1.205	1.185

Tables 3 and 4 compare the PI coverages for the error on the delay prediction for call type 1 by ANN, when the density of the prediction error is estimated by the QRF, the fitted LLOG, and the KDE, as explained in Section 3. We tried both 90% and 80% PIs. Recall that QRF is trained only once with all the vectors $\mathbf{x}$ in the training data set, whereas LLOG and KDE are fitted separately for each queue size q_k.

The coverages were computed as explained in the last paragraph of Section 4. For each call (represented by $\mathbf{x}$) in the test set, the expected delay was estimated using the ANN predictor, and the estimated 5%, 10%, 90%, and 95% quantiles of the prediction error were also computed using each of the three methods: QRF, LLOG, and KDE. Because there are few data for large queue size, all $\mathbf{x}$ for which $q_1 > 5$ were grouped together. We computed the percentages of calls falling in each of the three categories (1) to (3), for both the 90% and the 80% PI's. The percentages are reported in the tables. For call type 1, QRF clearly has a much better coverage accuracy than LLOG and KDE, with values that are fairly close to the desired coverage levels.

Table 3: The Bank call center, call type 1: Coverage percentage of a 90% PI and of the 5% tail on each side for the test data set, for different queue sizes.

Queue	$< 5\%$			$[5\%, 95\%]$			$> 95\%$			Total
size	QRF	LLOG	KDE	QRF	LLOG	KDE	QRF	LLOG	KDE	obs
0	5.05	10.25	2.76	92.02	89.21	96.88	2.94	0.53	0.35	6222
1	5.06	11.06	5.33	90.27	88.58	93.62	4.66	0.35	1.03	4738
2	5.10	9.42	13.36	89.28	90.26	85.97	5.60	0.31	0.66	3173
3	5.57	9.91	11.29	89.08	89.41	87.62	5.33	0.66	1.08	2117
4	4.47	8.95	9.84	91.34	90.00	87.99	4.17	1.04	2.16	1340
5	4.58	9.27	9.05	89.60	89.38	87.70	5.81	1.34	3.24	895
≥ 6	5.64	13.14	10.26	89.95	85.24	87.71	5.41	1.62	2.01	1294

Table 4: The Bank call center, call type 1: Coverage percentage of a 80% PI and of the 10% tail on each side for the test data set, for different queue sizes.

Queue	$< 10\%$			$[10\%, 90\%]$			$> 90\%$			Total
size	QRF	LLOG	KDE	QRF	LLOG	KDE	QRF	LLOG	KDE	obs
0	9.77	14.15	17.76	81.28	82.29	81.71	8.95	3.55	0.53	6222
1	10.13	16.36	18.80	78.75	81.17	80.77	11.12	2.47	0.42	4738
2	9.90	15,73	16.75	79.55	81.31	81.54	10.56	2.96	1.71	3173
3	9.49	15.54	15.78	79.88	82.38	82.05	10.63	2.08	2.17	2117
4	7.90	14.62	14.54	83.00	82.48	81.73	9.10	2.91	3.73	1340
5	8.04	16.42	15.53	81.01	81.56	80.34	10.95	2.01	4.13	895
≥ 6	10.20	17.77	15.77	80.14	80.76	78.83	9.66	1.47	5.41	1294

Table 5 shows some details on the values of the 10% and 90% quantiles that delimit a 80% PI that would be announced to a customer in the test set, aggregated by the queue size. For a fixed queue size, LLOG and KDE return constant estimates for the quantiles, because they estimate the density as a function of the queue size only. QRF, on the other hand, returns quantile estimates that depend on $\mathbf{x}$, so it can return very different values for the same queue length q_k. In the table, we give the mean and standard deviation of these values for each q_k. As expected, the width of the 80% PI increases with the queue size for all three predictors. Figure 6 compares the density functions obtained by LLOG fitting and by a KDE. LLOG has better fit when the queue is longer, but it has significant fitting error when the queue is short (sizes 0, 1 and 2),

Table 5: The Bank call center, call type 1: the mean and standard deviation of the 10% and 90% quantiles on the predictor error for QRF, and the actual quantiles for LLOG and KDE, conditional on the queue size.

	10%				90%			
Queue	QRF		LLOG	KDE	QRF		LLOG	KDE
size	mean	std dev.			mean	std dev		
0	-64.85	27.97	-40.70	-30.62	31.89	7.92	39.99	55.73
1	-92.59	34.26	-58.26	-49.17	49.63	11.73	67.32	88.67
2	-109.12	36.57	-75.14	-71.16	69.72	13.93	90.13	101.84
3	-126.86	42.03	-85.79	-84.67	85.92	18.63	120.13	118.55
4	-140.02	44.92	-90.10	-91.48	98.12	22.80	126.39	120.07
5	-152.08	46.95	-97.73	-101.33	105.77	28.96	152.54	129.03
≥ 6	-176.40	62.09	-107.98	-118.73	121.04	36.89	214.05	161.03

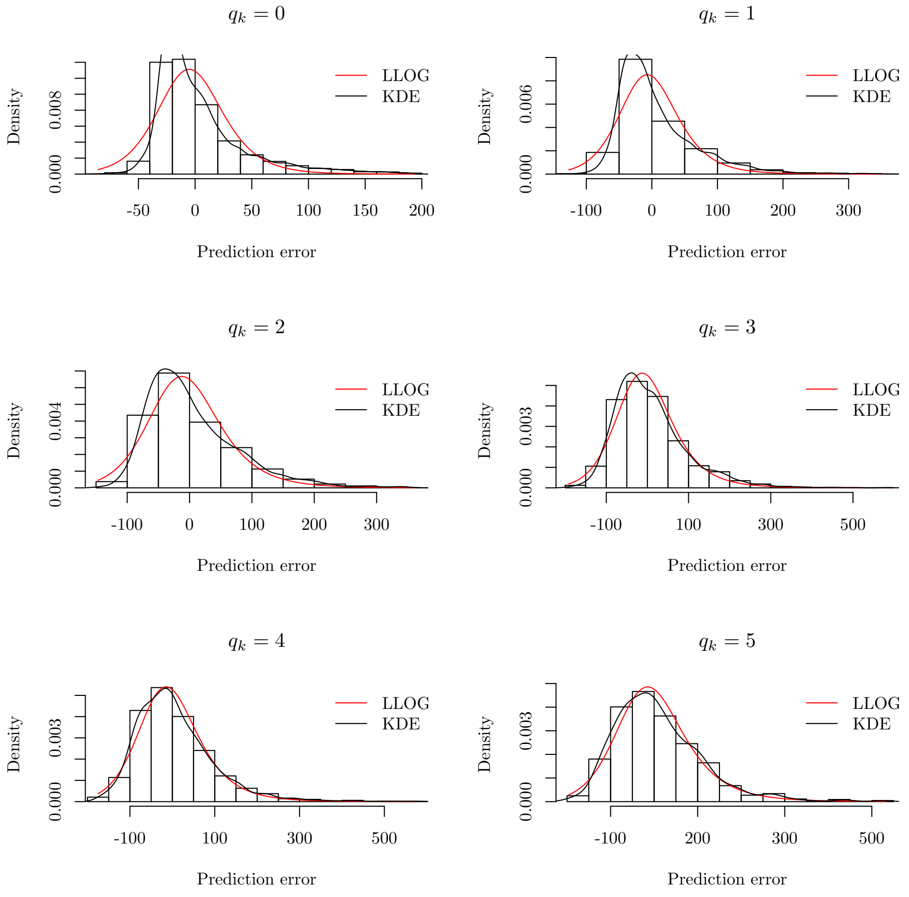

Figure 6: The Bank call center, call type 1: Estimated density of the prediction error (in seconds) for fitted LLOG and the KDE, conditional on the queue size.

6 Experiments with data from an IT call center

6.1 The available data

This is a call center of an information technology (IT) company located in The Netherlands. The data was collected over the entire year of 2014, and contains a total of 1,543,164 call logs. The center operated from 8:00 to 20:00 on weekdays (Monday to Friday), and served 27 call types with approximately 312 agents. About 56% of the calls have received service immediately, 38% have waited before getting a response, and 6% have abandoned. In this study, we consider only the five call types (type 1 to 5) that represent the largest volume of incoming calls (more than 90% of the total volume). Table 6 gives a statistical summary of the arrival counts, wait times, and service times, for the five call types.

Table 6: The IT call center: a statistical summary of arrivals counts and waits during the year. Adapted from [37].

	Type 1	Type 2	Type 3	Type 4	Type 5
Total number calls	568 554	270 675	311 523	112 711	25 839
Served, no wait	61%	52%	55%	45%	34%
Served, waited	35%	40%	40%	46%	54%
Abandon	4%	7%	5%	8%	12%
Avg wait time (sec)	77	91	83	85	110
Avg service time (sec)	350	308	281	411	311
Avg queue length	8.2	3.3	4.4	4.3	0.9

We partition the week days in two categories, according to the arrival patterns and volumes. Monday forms its own category, while the four others days (Tuesday to Friday) form the second category. For each call type k, we built two sets of prediction functions, one for each category. In this paper, we report prediction results for the second category, for which we have more data. Results on the prediction of the expected wait time were already presented in [37], but that paper did not consider quantile and density prediction. We summarize the wait time prediction results here for the sake of completeness.

6.2 Experimental results on predictions

Table 7 compares the RRASE values of six different predictors of the mean delay time, namely three DH predictors and three learning-based predictors. ANN is the clear winner among those six.

Table 7: The IT call center: the RRASE for the five call types. The ANN has the best accuracy, followed by RS.

	DH predictors			Learning-based predictors		
Type	Avg.LES	LES	AvgC-LES	RS	RL	ANN
Type 1	0.489	0.443	0.443	0.396	0.415	**0.361**
Type 2	0.610	0.565	0.577	0.492	0.515	**0.462**
Type 3	0.567	0.516	0.518	0.455	0.471	**0.448**
Type 4	0.487	0.424	0.445	0.395	0.385	**0.377**
Type 5	0.697	0.661	0.624	0.501	0.517	**0.487**

Tables 8 and 9 compare the PI coverages exactly as in Tables 3 and 4, for call type 1 of the IT call center. In contrast with the Bank call center, here the KDE gives the coverage closest to the target for the 95% and 90% quantiles (the right tail). For the 5% and 10% quantiles (the left tail), QRF does better when the queue size is small whereas the KDE is more accurate for longer queue sizes. LLOG lags behind.

Table 8: The IT call center, call type 1: Coverage percentage of a 90% confidence predictor and of the two 5% tails for the test data set, for different queue sizes.

Queue	$< 5\%$			$[5\%, 95\%]$			$> 95\%$			Total
size	QRF	LLOG	KDE	QRF	LLOG	KDE	QRF	LLOG	KDE	obs
0	4.79	10.34	8.15	93.93	85.36	86.28	1.27	4.29	5.55	3778
1	5.93	9.60	7.60	90.54	86.81	87.13	3.52	3.57	5.25	3693
2	5.48	9.29	7.07	90.37	87.86	88.17	4.13	2.83	4.75	3407
3	6.22	9.38	7.32	89.92	87.57	87.78	3.85	3.04	4.88	2907
4	6.34	8.55	6.26	89.45	88.59	88.73	4.20	2.85	5.00	2380
5	6.94	8.44	6.25	89.14	88.17	88.95	3.91	3.37	4.78	2046
≥ 6	8.36	6.54	5.32	86.54	90.77	89.69	5.11	2.69	4.99	10837

Table 9: The IT call center, call type 1: Coverage percentage of a 80% PI and for the two 10% tails, for the test data set, for different queue sizes.

Queue	$< 10\%$			$[10\%, 90\%]$			$> 90\%$			Total
size	QRF	LLOG	KDE	QRF	LLOG	KDE	QRF	LLOG	KDE	obs
0	10.08	14.74	12.73	84.20	78.22	76.83	5.72	7.04	10.45	3778
1	11.40	14.32	12.48	79.53	78.45	77.23	9.07	7.23	10.29	3693
2	11.24	14.31	12.15	79.63	77.92	77.90	9.13	7.77	9.95	3407
3	11.35	14.62	12.32	80.29	77.85	77.74	8.36	7.53	9.94	2907
4	11.64	13.53	11.39	79.87	79.12	78.61	8.49	7.35	10.00	2380
5	12.70	13.10	11.32	79.46	79.52	78.59	7.84	7.38	10.09	2046
≥ 6	13.89	13.37	10.36	77.14	78.54	80.19	8.97	8.09	9.45	10774

Tables 10 and 11 show a different story for call type 2. QRF clearly dominates in the left tail, whereas in the right tail, LLOG wins for small queue sizes and QRF catches up for larger queue sizes. Figure 7 compares the density functions given by LLOG and KDE from the training data set. There is significant fitting error when the queue size is short (0 to 2), but it improves when the queue is larger. Overall, the coverage accuracies from LLOG and KDE are relatively similar.

Table 10: The IT call center, call type 2: Coverage percentage of a 90% PI and of the 5% tail on each side for the test data set, for different queue sizes.

Queue	$< 5\%$			$[5\%, 95\%]$			$> 95\%$			Total
size	QRF	LLOG	KDE	QRF	LLOG	KDE	QRF	LLOG	KDE	obs
0	5.37	9.39	11.95	93.21	85.30	85.19	1.41	5.30	2.85	4450
1	6.45	11.46	13.77	90.65	83.82	83.30	2.88	4.71	2.91	3672
2	7.05	13.19	16.53	89.53	83.01	81.03	3.41	3.79	2.42	2637
3	8.28	15.96	18.78	88.01	80.00	78.50	3.70	4.03	2.70	1810
4	7.11	12.76	13.53	88.07	81.95	82.26	4.81	5.27	4.20	1308
5	6.92	14.18	14.18	87.48	81.67	82.12	5.58	4.13	3.68	895
≥ 6	7.65	13.21	11.67	87.61	84.29	85.72	4.74	2.50	2.60	1961

Table 11: The IT call center, call type 2: Coverage percentage of a 80% PI and of the 10% tail on each side for the test data set, for different queue sizes.

Queue	$< 10\%$			$[10\%, 90\%]$			$> 90\%$			Total
size	QRF	LLOG	KDE	QRF	LLOG	KDE	QRF	LLOG	KDE	obs
0	11.98	12.90	13.65	82.22	77.51	67.59	5.80	9.60	18.76	4450
1	12.23	16.04	19.20	80.28	75.52	74.65	7.49	8.44	6.15	3672
2	14,26	18.54	21.05	78.27	74.59	74.52	7.47	6.86	4.44	2637
3	16.08	21.22	23.37	76.08	71.22	71.16	7.85	7.57	5.47	1810
4	13.23	17.43	17.97	76.76	73.39	74.77	10.02	9.17	7.26	1308
5	12.74	19.89	19.89	78.10	71.51	73.30	9.16	8.60	6.82	895
≥ 6	14.13	21.62	20.49	77.72	72.97	74.29	8.28	5.41	5.28	1961

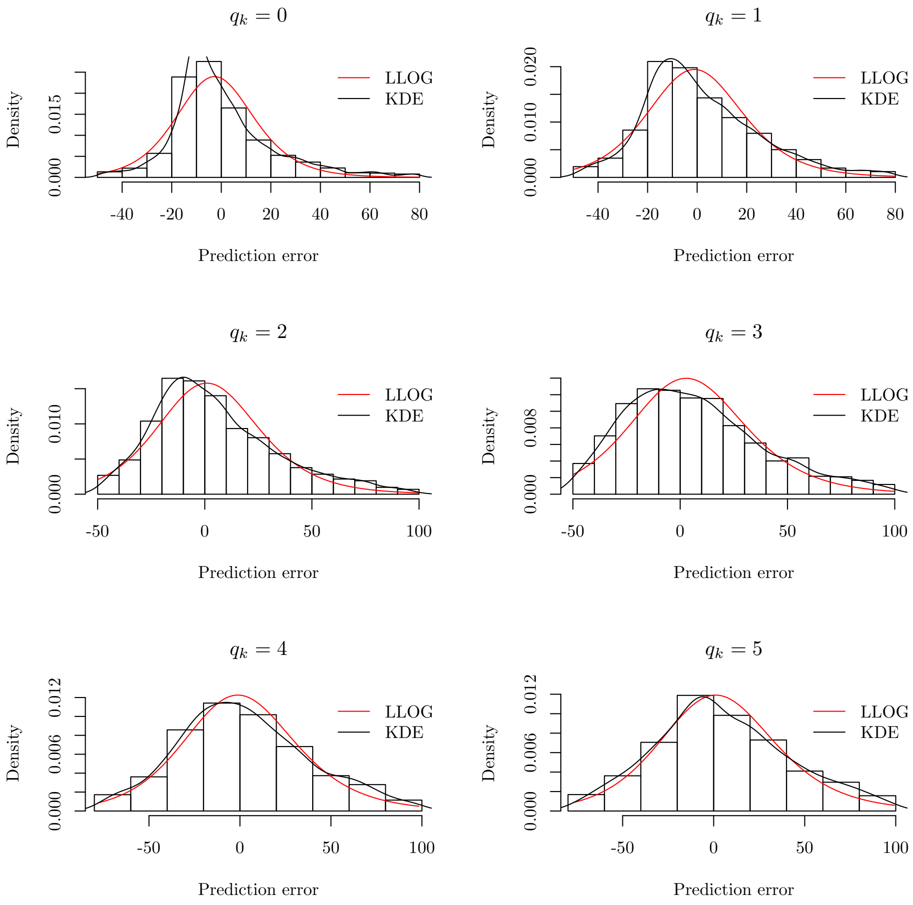

Figure 7: The IT call center, call type 2: Density function of the prediction errors for LLOG and KDE, conditional on the queue size.

Figure 8 shows Q-Q plot of the empirical distributions from the training data set and the test data set, for queue size from 2 to 5. These distribution of the training set and test set are different in the tails, especially when the error is negative (i.e., the delay is higher than predicted). The significant difference for $q_k = 3$ can be seen from the coverage values in Tables 10 and 11.

7 Conclusion

We discussed and compared empirically different predictors for the waiting time of a call (or customer) when this call arrives at a multi-skill call center. The more accurate predictors among those examined are the predictors based on regression methods and automatic learning, and more specifically the predictors defined by deep multilayer neural networks, at least when enough data is available. We also examined different ways of modeling the distribution of the prediction error and estimating its density and its quantiles. We tried fitting known parametric distributions as well as a KDE to the observations of this prediction error conditional on the queue length, and the KDE usually gives a better fit even for the independent test data. But another non-parametric method named QRF, based on the random forest methodology, gave the best results in the majority of cases and performed reasonably well in general. On the flip side, this method is computationally more intensive. Suggestions for follow-up work include studying large call centers in which the waiting times are much longer, studying wait time predictions in other types of service systems (e.g., healthcare clinics), and trying to find better methods to estimate the density of the wait times.

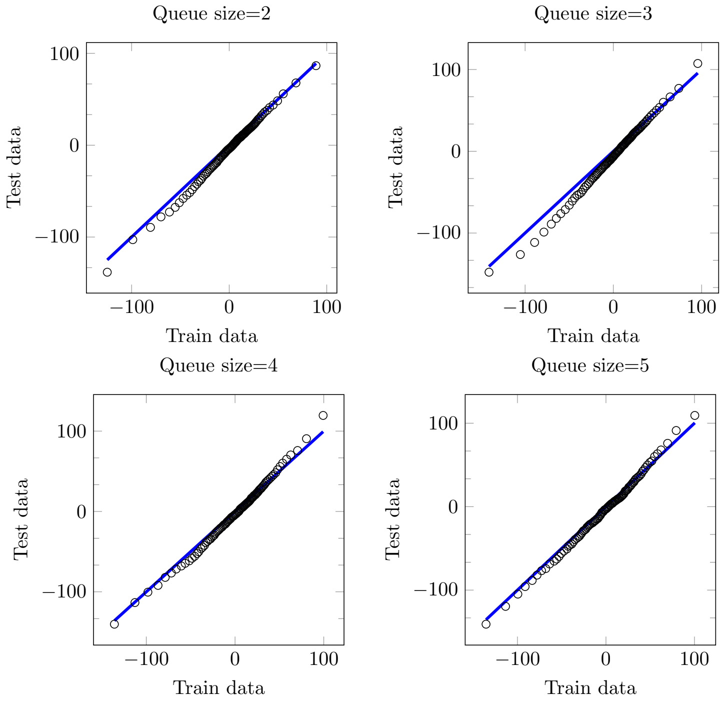

Figure 8: The IT call center, call type 2: Q-Q plot of the empirical distributions between the training data set and the test data set.

References

[1] Ang, E., Kwasnick, S., Bayati, M., Plambeck, E., Aratow, M.: Accurate emergency department wait time prediction. Manufacturing & Service Operations Management 18(1), 141–156 (2016)

[2] Armony, M., Shimkin, N., Whitt, W.: The impact of delay announcements in many-server queues with abandonments. Operations Research 57, 66–81 (2009)

[3] Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. Neural Networks: Tricks of the Trade 7700, 437–478 (2012)

[4] Bengio, Y., Courville, A.C., Vincent, P.: Unsupervised feature learning and deep learning: A review and new perspectives (2012), http://arxiv.org/abs/1206.5538

[5] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012)

[6] Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)

[7] Cooper, R.B.: Introduction to Queueing Theory. North-Holland, New York, NY, second edn. (1981)

[8] de Boor, C.: A Practical Guide to Splines. No. 27 in Applied Mathematical Sciences Series, Springer-Verlag, New York (1978)

[9] Degenhardt, F., Seifert, S., Szymczak, S.: Evaluation of variable selection methods for random forests and omics data sets. Briefings in Bioinformatics 20(2), 492–503 (2017)

[10] Ding, R., McCarthy, M.L., Desmond, J.S., Lee, J.S., Aronsky, D., Zeger, S.L.: Characterizing waiting room time, treatment time, and boarding time in the emergency department using quantile regression. Academic Emergency Medicine 17(8), 813–823 (2010)

[11] Dong, J., Yom Tov, E., Yom Tov, G.: The impact of delay announcements on hospital network coordination and waiting times. Management Science 65(5), 1949–2443 (2018)

[12] Friedman, J., Hastie, T., Tibshirani, R., Narasimhan, B., Simon, N., Qian, J.: R Package glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models (2019), https://CRAN.R-project.org/package=glmnet

[13] Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(1), 1–22 (2010)

[14] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G., Dunson, D., Miroslav (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 315–323 (2011)

[15] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org

[16] Goodfellow, I., Warde-Farley, D., Lamblin, P., Dumoulin, V., Mirza, M., Pascanu, R., Bergstra, J., Bastien, F., Bengio, Y.: Pylearn2: A machine learning research library (08 2013)

[17] Gross, D., Harris, C.M.: Fundamentals of Queueing Theory. Wiley, New York, NY, third edn. (1998)

[18] Gulcehre, C., Bengio, Y.: Knowledge matters: Importance of prior information for optimization. Journal of Machine Learning Research 17, 1–32 (2016)

[19] Hörmann, W., Leydold, J., Derflinger, G.: Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin (2004)

[20] Ibrahim, R., L'Ecuyer, P., Shen, H., Thiongane, M.: Inter-dependent, heterogeneous, and time-varying service-time distributions in call centers. European Journal of Operational Research 250, 480–492 (2016)

[21] Ibrahim, R., Whitt, W.: Real-time delay estimation based on delay history. Manufacturing and Services Operations Management 11, 397–415 (2009)

[22] Ibrahim, R., Whitt, W.: Real-time delay estimation in overloaded multiserver queues with abandonments. Management Science 55(10), 1729–1742 (2009)

[23] Ibrahim, R., Whitt, W.: Delay predictors for customer service systems with time-varying parameters. In: Proceedings of the 2010 Winter Simulation Conference. pp. 2375–2386. IEEE Press (2010)

[24] Ibrahim, R., Whitt, W.: Real-time delay estimation based on delay history in many-server service systems with time-varying arrivals. Production and Operations Management 20(5), 654–667 (2011)

[25] James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning, with Applications in R. Springer-Verlag, New York (2013)

[26] Kleinrock, L.: Queueing Systems, Vol. 1. Wiley, New York, NY (1975)

[27] Kursa, M.B., Rudnicki, W.R.: Feature selection with the boruta package. Journal of Statistical Software 36, 1–13 (2010)

[28] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521, 436–444 (2015)

[29] Meinshausen, N.: Quantile regression forests. J. Mach. Learn. Res. 7, 983–999 (2006)

[30] Nakibly, E.: Predicting Waiting Times in Telephone Service Systems. Master's thesis, Technion, Haifa, Israel (2002)

[31] Scott, D.W.: Multivariate Density Estimation. Wiley, second edn. (2015)

[32] Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining for delay prediction in multi-class service processes. Information Systems 53, 278–295 (2015)

[33] Silverman, B.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)

[34] Sun, Y., Teow, K.L., Heng, B.H., Ooi, C.K., Tay, S.Y.: Real-time prediction of waiting time in the emergency department, using quantile regression. Annals of Emergency Medicine 60(3), 299–308 (2012)

[35] Thiongane, M., Chan, W., L'Ecuyer, P.: Waiting time predictors for multiskill call centers. In: Proceedings of the 2015 Winter Simulation Conference. pp. 3073–3084. IEEE Press (2015)

[36] Thiongane, M., Chan, W., L'Ecuyer, P.: New history-based delay predictors for service systems. In: Proceedings of the 2016 Winter Simulation Conference. pp. 425–436. IEEE Press (2016)

[37] Thiongane, M., Chan, W., L'Ecuyer, P.: Delay predictors in multi-skill call centers: An empirical comparison with real data. In: Proceedings of the International Conference on Operations Research and Enterprise Systems (ICORES). pp. 100–108. SciTePress (2020), https://www.scitepress.org/PublicationsDetail.aspx?ID=QknUuVhZF/c=&t=1

[38] Tibshirani, R.: Regression shrinkage and selection via the LASSO. Journal of the Royal Statistical Society, Series B (Methodological) pp. 267–288 (1996)

[39] Whitt, W.: Predicting queueing delays. Management Science 45(6), 870–888 (1999)

[40] Wood, S.N.: Generalized Additive Models: An Introduction with R. Chapman and Hall / CRC Press, Boca Raton, FL, second edn. (2017)

[41] Wood, S.N.: R Package mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation (2019), https://CRAN.R-project.org/package=mgcv