Direction of the Difference Between Bayesian Model Averaging and the Best-Fit Model on Scarce-Data Low-Correlation Churn Prediction

Paul J. Darwen(✉)

James Cook University, 349 Queen Street, Brisbane, QLD, Australia
paul.darwen@jcu.edu.au

Abstract. On a scarce-data customer churn prediction problem, using the tiny differences between the predictions of (1) the single-best model and (2) the ensemble from Bayesian model averaging, gives greater accuracy than state-of-the-art approaches such as XGBoost. The proposed approach reflects the cost-benefit aspect of many such problems: for customer churn, incentives to stay are expensive, so what's needed is a short list of customers with a high probability of churning. It works even though in every test case, the predicted outcome is always the same from both the best-fit model and Bayesian model averaging. The approach suits many scarce-data prediction problems in commerce and medicine.

Keywords: ensembles · customer churn · small data · scarce data · Bayesian model averaging · committee of committees · cold start problem

1 Introduction and Motivation

Many prediction problems suffer from scarce data: few rows of data, and the few input columns have poor discriminant value, giving low correlation. Such small-data low-dimensional problems include medical triage prediction [10], the diagnosis of rare cancers [8], and the hospital readmission problem [9,12,14].

As a typical example of such scarce-data problems, here the task is to predict which e-commerce customers will churn (change to a different provider). Churn affects mobile phone providers [1], online games [15], e-commerce web sites [18, 22], and any service where the customer has a low cost of switching providers.

Predicting customer behaviour is still not a solved problem, even for big-data problems like mobile phone customer churn [19]. There are many more businesses with low-dimensional small-data prediction problems than businesses with the luxury of big data sets. This includes fields with rapid changes (such as fashion and phones) in which data more than a few months old is irrelevant.

1.1 Previous Work and Current Contribution

The class of problems studied here are small-data, low-correlation prediction tasks. Ensemble approaches have been used for other problems or in other ways [6,24], such as high-correlation (instead of low-correlation) big data [4], or only to do feature selection [17], or with small ensemble sizes like eight models [16] instead of thousands, or to track changes over time [2,21]. And of course customer churn prediction has been attempted in many ways [15,19,22], usually presuming that huge data sets are available [1].

Predicting from small-data low-correlation data has much in common with the cold-start problem in recommender systems. The approach presented here relies heavily on selecting suitable inputs/attributes, as do some studies of the cold-start problem [3,25].

The approach presented here has been used on predicting river floods [7] and cancer diagnosis [8], but this study is the first to apply the approach to an e-commerce prediction problem. The results will show that for a customer churn prediction problem typical for millions of retailers, the proposed approach gives better accuracy than any other algorithm in the SciKit library for Python.

This paper's approach uses the tiny differences between the predictions of (1) the single-best model, and (2) the ensemble from Bayesian model averaging. The proposed approach is not merely comparing an ensemble method with a single model because both predict the same result, in almost all cases. Section 4 will show that the proposed method does better than XGBoost, which was the best in a recent study about mobile phone churn prediction [1].

2 Background

2.1 A Scarce-Data Low-Correlation Customer Churn Problem

The churn prediction problem comes from an e-commerce site that sells a single product. The data set[1] only covers the browsing history of each customer. There is no demographic or other personal information.

The business case is to identify a short list of customers who are most likely to churn, in order to persuade them to stay, using discounts or similar inducements. The list must be short, because there is a tight budget for these expensive offers. Naturally the seller is reluctant to offer such inducements to a customer who wasn't planning to churn after all. The profit margin is thin, so it only makes sense to offer the inducement if the prediction is correct around 90% of the time.

What measure of accuracy to use? If the classifier predicts the customer will not churn, then no inducement will be offered, so true negative (TN) and false negative (FN) can be ignored. If the classifier predicts a customer will churn, and we offer the inducement, then it would be an expensive waste if the customer had no intention of churning (a false positive, FP). So the chosen measure of accuracy for this problem will be precision = TP/(TP + FP).

[1] See https://www.kaggle.com/huzaiftila/customer-churn-prediction-analysis.

2.2 Description of the Data, and Data Cleaning

Big spenders or "whales" are the most desirable customers. Whales who regularly view the web site tend to keep visiting, and rarely churn. The cases of interest are the big spenders who rarely visit the web site. From the original 91,698 visitors over 5 weeks, the client is only interested in big-spending but rarely-visiting customers: those who spent 2,076 rupees or more in their first 5 weeks, but only had 7 or fewer visits. This gives a small data set of 510 rarely-visiting, big-spending customers. It's close to balanced, with 51.6% of customers churning and 48.4% staying.

Some redundant columns are deleted. It has the sales amounts for each week, so binary "week 1" through "week 4" (to say if they bought anything) are redundant. Similarly, the last three columns (binary for rare, frequent, or regular visitors) are redundant because the column "Visitors Type" gives the same information (and we convert it to 0, 1, and 2 to make it numeric). Similarly deleted are the three columns for high, regular, and low value which merely repeat the column "Customer Value".

The values of each input column are rank normalized from 0 to +1: an input's smallest value becomes 0, the median becomes 0.5, and the biggest becomes 1.

In this paper, accuracy is calculated from leave-one-out cross-validation [13, page 242]. From the data set of 510 rows, we create an equal number of training data sets, each one having 509 rows, with the left-out row becoming the single point of test data for the model built with that training set. This gives an average accuracy without random sampling.

Interaction variables are new inputs created by multiplying the values in existing input columns[2]. For each of the 510 training sets created by leave-one-out cross validation, each pair of inputs is multiplied to create every possible 2-way interaction variable.

Feature selection uses brute-force sampling, generating many combinations of inputs, and looking for the best feature set from the many sets thus generated. In practice, the feature selection only picks less than 10 inputs, from the many on offer: most interaction variables are no better than random.

2.3 Bayesian Model Averaging: Same Prediction as Best Model

Bayesian model averaging is an ensemble approach. It enumerates all possible models, and each model's prediction is weighted by the model's posterior probability of being correct, given the data [11, page 8] [8, section II.C]. The weighted vote from all the models gives the prediction.

Unfortunately, for the problem studied here and for many other scarce-data problems [8], Bayesian model averaging always predicts the same outcome as the single best-fit model. In contrast, the proposed approach makes use of the tiny differences in the predicted likelihoods: the following Sects. 2.4 and 2.5 describe Bayesian model averaging as used here.

[2] Some software packages do this automatically, such as SPSS and Unistat.

2.4 A Likelihood Function for Bayesian Model Averaging

To calculate the probability of a model's correctness, each model needs to a likelihood function. The one used here follows earlier work on predicting floods [7] and lung cancers [8]. Each model includes a linear regression model, with a slope coefficient for each input, plus a coefficient for an intercept. To put a pipe on top of that line (or rather, hyperplane), one more model coefficient is the standard deviation of a bell curve cross-section, to make the model into a likelihood function, as needed for Bayesian model averaging.

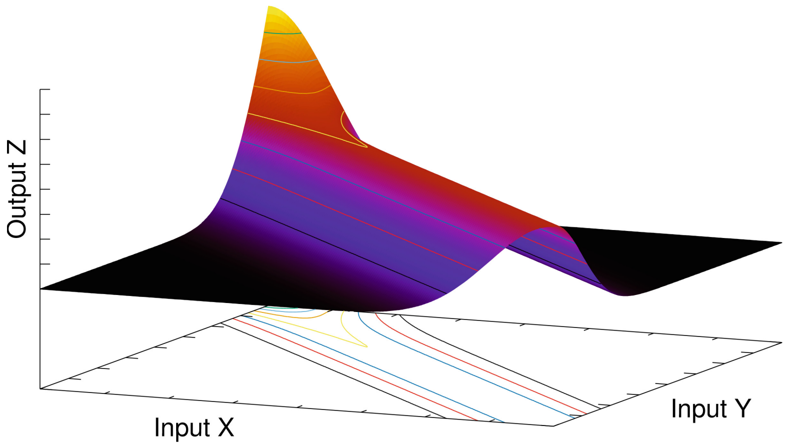

Fig. 1. In Bayesian model averaging, each model calculates likelihoods. This stylized view shows one half of the "buried trombone" likelihood function used here. A model is a line (or hyperplane) from linear regression, covered by a bell curve that inflates at both ends. Each input is ranked, so values go from zero to one, and the bell curve is truncated below zero or above one, requiring the untruncated remainder to get bigger, to guarantee that the integral of each cross-section will always sum to one.

The likelihood predicted by a model is the vertical axis. Back in Sect. 2.2, preprocessing ranked each input's values from 0 to +1 (0 being the smallest, 1 being the biggest, and 0.5 being the median), so a model's pipe-shaped likelihood function must be truncated at 0 and 1. As it approaches a truncation, the each cross-section of the bell curve must integrate to one, so the trombone-shaped blow-up in Fig. 1 is needed.

2.5 Two Approximations to Bayesian Model Averaging

Bayesian model averaging wants to *integrate* over *all* possible models, but this integral can't be done symbolically with a software model. This section describes two approximations to get around those two requirements.

Firstly, for each set of n input features, the continuous integration is approximated with an n-dimensional grid over the space of model parameters. The integral thus becomes a sum over thousands of models.

Secondly, even with a discretized grid, it's too many to enumerate *all* models with non-zero posterior probability of being correct. Ignoring models with a near-zero posterior probability should only give a small error. So in the proposed method in Sect. 3, the enumeration of models in that grid is terminated after it has found a large number of models: 40,000 for 4 inputs, 50,000 for 5, 60,000 for 6, and 75,000 for 7 inputs. Note that number of inputs chosen by feature selection can be different on different data sets, because feature selection is run independently on each data set from the leave-one-out cross-validation.

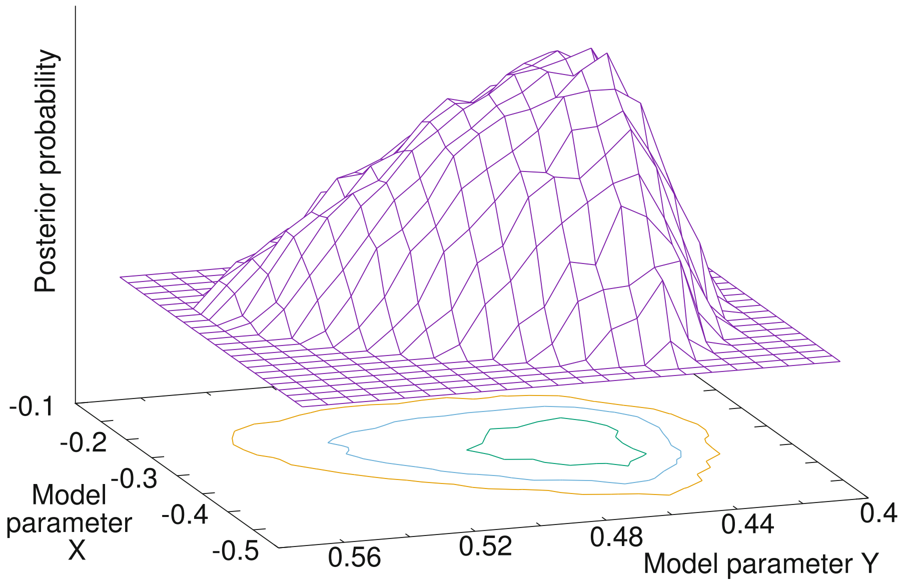

Fig. 2. For the leave-one-out data set created by leaving out customer 144, this shows a posterior landscape: for models whose first two parameters are x and y, the z-axis is the posterior probability that the model is correct, given the data. The distribution is skewed, so the mode will be different from the mean, i.e., the model with the highest probability of being correct (the best-fit model is the peak) will predict differently from the weighted average vote of all models. This happens with other prediction problems, such as for lung cancer [8, Figure 2] and floods [7, Figure 4].

For example, Fig. 2 shows a posterior landscape from a leave-one-out data set with 5 inputs, and the search ended at 50,219 models. This approximation only considers the mountain, but ignores the lower area around it.

For one of the training sets created by leave-one-out-cross validation, Fig. 2 shows a 2-dimensional cross-section of the posterior distribution of models. This data set leaves out customer 144, who becomes the single point of test data. The x and y axes are two model parameters, and take their values from a discretized grid, to give a range of models of varying fit to the data. For each model in x-y space, z is that model's posterior probability of being correct, given the training data. The best-fit model has the highest probability of being correct, so it's the model at the peak. Sloping down from that peak are thousands of models with a worse fit to the data, and thus a lower posterior probability of being correct.

The posterior distribution in Fig. 2 is skewed: this causes the prediction of the best-fit model to be different from the prediction of the weighted average vote of all the models in the distribution. Those two predictions are indeed different in Fig. 3, but only by a tiny amount: the solid line shows the prediction of the most-probable, best-fit model (it's higher at $x = 1$ predicting that churn is more likely) and the dotted line shows the weighted average vote of all models in the landscape of Fig. 2 (which also says that churn is more likely). The punchline is that the difference points to the right answer: the vote's prediction is both lower at $x = 0$ and higher at $x = 1$ than the best-fit model.

The y-axis in Fig. 3 is the probability density of different outcomes, and the x-axis shows the two outcomes: 0 means no churn, 1 is churn, and x-values in between can be interpreted as probabilities [5]. Probability density can exceed 1, but it can never be negative, and must integrate to exactly 1.

The skewness in Fig. 2 is mild enough that both methods predicted that churn is the more likely outcome, and in this example they were both right, this particular customer did indeed churn. In fact, for this customer churn problem, Bayesian model averaging always predicts the same outcome as the best fit model, just like in Fig. 3. So it's no more accurate, and has the disadvantage of using more computer time.

Instead of condemning Bayesian model averaging, Sect. 3 will use it merely to find the direction of the tiny differences in predictions, as seen in Fig. 3. Increasing how large the difference needs to be, can give better accuracy.

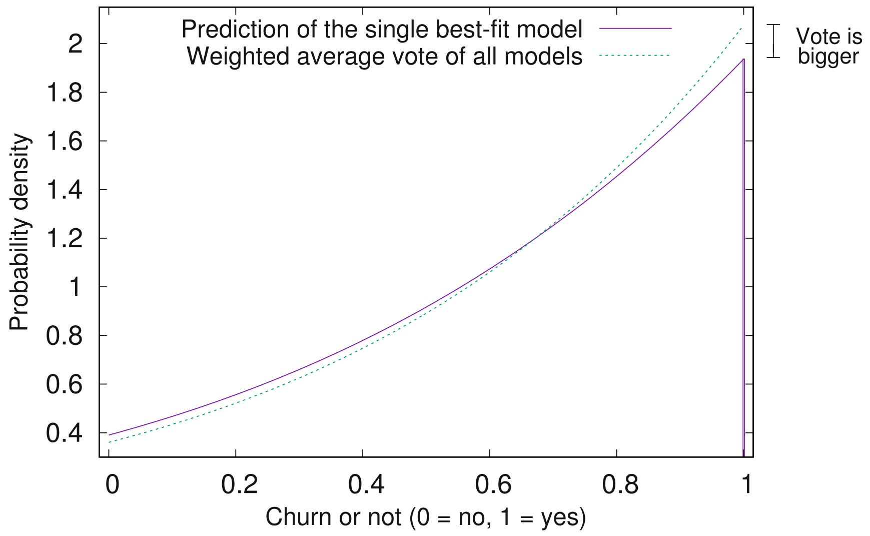

Fig. 3. The skewed posterior landscape in Fig. 2 means the prediction of the best-fit model will be different from the prediction of the weighted vote of all the models. But that difference is small. Both predict this customer is more likely to churn ($x = 1$) than not churn ($x = 0$). This particular customer did indeed churn. Intriguingly, the direction of the difference is leaning the right way: the weighted average vote is both higher at $x = 1$ and lower at $x = 0$. The proposed approach uses those tiny differences.

3 The Proposed Prediction Method

3.1 Feature Selection Difficulties in Scarce-Data Problems

Low-dimensional low-correlation small-data problems have few inputs (i.e., few columns or features), and the data set is short (i.e., few rows or individuals), often just a multiple of 10 or 20 of the small number of inputs.

Any two points in two-dimensional space can have a line that fits them perfectly, and any three points in three-dimensional space can have a plane that fits them perfectly, and so on. So a data set with few rows will have are many combinations of input columns that will fit the data moderately well, purely due to bad luck. That is, feature selection becomes problematic because fewer rows of data increase the probability that an uninformative or random input appears to have a statistical relationship with the output.

To average out the inevitable junk inputs, the proposed approach has a second layer of variation: instead of choosing just a single feature set of inputs and generating a single posterior landscape, it does so for the best 30 feature sets[3]. That's not 30 features, but 30 feature *sets* for each prediction. This will be done for every one of the 510 training sets created by leave-one-out cross validation on the data set of 510 customers, in the following two steps.

Firstly, leave-one-out cross-validation generates 510 data sets (one for each of the 510 rows of data), and then creates all possible 2-way interaction variables, to give thousands of inputs to choose from. For each data set, brute-force sampling generates a very large number of random combinations of inputs, and keeps the best 30 of these feature sets. These 30 feature sets per data set will usually have different inputs, and different numbers of inputs. From the best 30 feature sets, we generate 30 different pairs of predictions, causing the inevitable junk input features to cancel out to some extent.

Secondly, for each of those 30 best feature sets, Bayesian model averaging varies the model's parameters over a discretized grid, to find the best 100,000 or so models. This gives 30 posterior landscapes, like in Fig. 2. Each landscape's highest point represents the best-fit model. The whole landscape shows all the voters in Bayesian model averaging.

In every one of these 30 posterior landscapes, the weighted vote gives slightly different predictions, but always agrees with the best-fit model about which outcome has the higher probability, just like in Fig. 3. The next section makes use of those slight differences, by using only the direction of the difference in each pair of predictions.

This committee-of-committees approach is different from conventional Bayesian model averaging, which only varies model parameters for a single best set of input features. Using 30 different committees, each with thousands of models, doesn't aim to reduce CPU time, but to average out random errors from inputs that are not informative, just like in real elections [20].

[3] Why 30 feature sets? Kreft's 30-30 rule says that 30 groups of at least 30 each is reasonable. Here each group has thousands of models, so 30 of those groups should be enough. More than 30 would be desirable, but the CPU cost is already large.

3.2 Predict from the Direction of the Differences

In Fig. 3, the likelihood of churn is highest at $x = 1$. This indicates that churn is predicted by both the single best model, and by the weighted vote from Bayesian model averaging. The difference in predicted likelihood is tiny.

Consider the following two possibilities. One possibility: if all 30 pairs of predictions give essentially the *same* likelihood for churn, the standard deviation of the 30 differences will be small. This indicates the data in the 30 feature sets are in agreement about which outcome is more likely.

The other possibility: if enough of the 30 pairs disagree, and give sufficiently *different* likelihoods for churn, then the standard deviation of the 30 differences will be bigger. This suggests the 30 feature sets disagree, with some predicting one outcome, and some predicting the other outcome: that's perfectly reasonable, because all will have a few junk inputs.

So consider this proposal for a prediction algorithm. Leave-one-out cross-validation gives 510 different data sets, and each of the 510 data sets has its 30 best feature sets, and each feature set gives both (1) a single best-fit model, which is the highest point in (2) a posterior landscape of thousands of models.

– If the standard deviation of the 30 differences in the likelihood of churn is too small, no prediction is made.
– If the standard deviation of the 30 differences is big enough, then the prediction goes by the average (not the standard deviation) of the differences:
    • If the Bayesian approach (on average) predicts a higher likelihood for output 1 (i.e., the average difference is positive), then predict 1 (churn).
    • If the average difference is negative or zero, then no prediction.

The general idea of the proposed approach is that if the mass of voters have a small but consistent disagreement with the well-informed expert, then take the direction of those differences to get the prediction. It's like a country of 30 different states, where each state's voters cast their choice, and then the vote tally is compared with the percentage given by the best-informed voter for that state. When the votes disagree with the best-informed voter by more than some pre-set percentage (even if they both favor the same candidate), then whichever way the voter tally was different, that difference goes towards that candidate. Imagine if in all 30 states, both the vote tally and the best-informed voter favor the same candidate, but the standard deviation of the differences was big enough; then the direction of the tiny differences is what chooses the winning candidate.

4 Results: Raising a Threshold for Fewer Predictions

The comparison algorithms are all from the Python library SciKit. Each algorithm has a *threshold*, i.e., a parameter which makes it more likely to choose yes or no. In Fig. 4, logistic regression with a threshold halfway between yes = 1 and no = 0 would not favor either outcome. A threshold closer to "yes" would result fewer "yes" predictions, but which are more likely to be "yes", because that y-axis is often interpreted to be a probability.

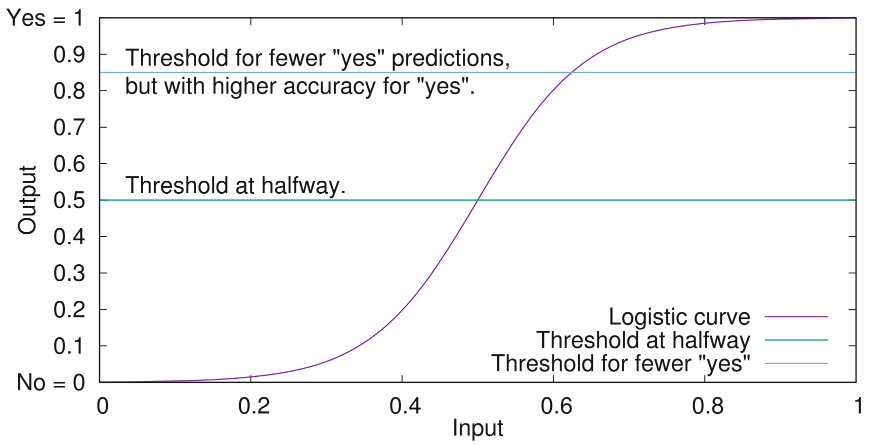

Fig. 4. In this stylized view of logistic regression, the default threshold is in the middle, with no bias towards "yes" or "no". Setting the threshold higher will cause it to make fewer "yes" predictions, for those predictions should be more likely to be correct for "yes". The results below will similarly vary each algorithm's threshold, to go from many "yes" predictions, to fewer predictions but with higher accuracy for "yes".

In the results below, each algorithm's threshold is varied from the neutral middle value (where it is equally likely to predict churn or not) upwards, so that it makes fewer "yes" predictions but are more likely to be correct, until it gets so high that it makes few or no "yes" predictions. For the proposed method, that threshold parameter is the required standard deviation of the difference in the likelihoods of (1) the single best model and (2) the whole ensemble.

4.1 The Bigger the Difference, the Higher the Precision

Figure 5 shows that increasing the required standard deviation of the difference gives better accuracy. The error bars are the 95% confidence intervals for binomial proportions from the Clopper-Pearson method[4]. In Fig. 5, the x-axis says how big must be the standard deviation of the difference between the likelihoods for the two methods, to dare make a prediction. The further to the right, the greater the precision, but for fewer predictions, so the confidence intervals get wider. It gives a short list of customers most likely to churn.

From Fig. 5, by increasing the required direction of the difference, that precision for churn can be raised above 90%, but for a short list of customers. Taking Fig. 5 and using the nonlinear least-squares Marquardt-Levenberg (and using as uncertainties the Clopper-Pearson confidence intervals of the binomial proportions), gives a p-value below 0.01, indicating a relationship exists.

[4] Clopper-Pearson does not approximate a binomial distribution with a normal distribution, instead it gives exact results for small samples.

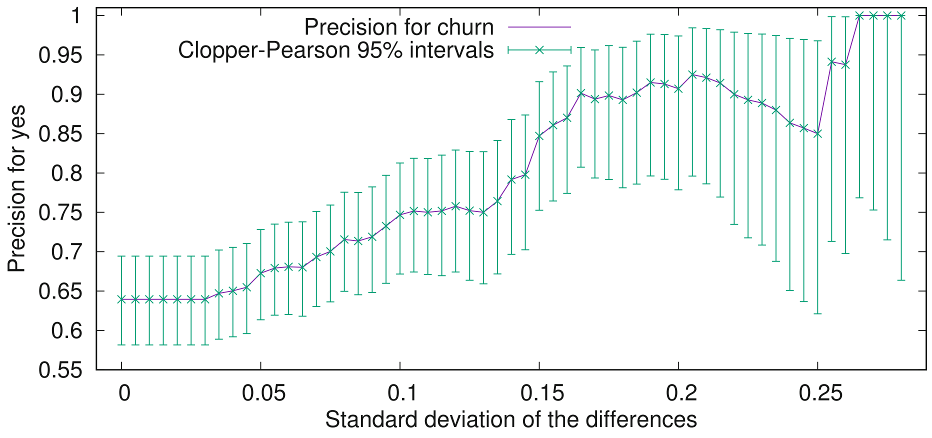

Fig. 5. The x-axis shows how big the standard deviation of the difference between 30 pairs of likelihoods needs to be, to venture a prediction. Moving right by increasing how different the two likelihoods need to be, raises precision but for fewer customers.

4.2 Comparison with XGBoost, Deep Learning, and Others

The proposed approach compares favourably with popular algorithms such as XGBoost and deep learning. Varying each algorithm's threshold (to give fewer but more confident predictions) gives Fig. 6, where x is the size of the short list produced, so that as the threshold becomes more demanding, the short list gets shorter. The y-axis shows how many customers on that short list do churn, representing the precision $= TP \div (TP + FP)$. All algorithms used leave-one-out cross-validation to select the training and test data, so there is no sampling error.

The business case is that around half of the 510 customers will churn, and the profit margin is such that it's only worth offering an incentive to a customer if the chance of them churning is close to 90%. The marketing budget can give an incentive to about 60 customers.

Given these requirements, Table 1 picks the threshold of each algorithm, such that the short list[5] is the smallest size that is at least 60. In Table 1, the proposed method has the highest precision: the threshold that gives the smallest short list of size at least 60 covers 63 customers, of whom 88.89% go on to churn.

XGBoost is slightly behind with precision of 84.13%, and the third-best result is 76.67% from a shallow neural network with two hidden layers, of 15 and 10 neurons[6] Trying many different numbers of layers (and neurons in each layer) suggests that adding more layers or neurons tends to reduce accuracy, perhaps

[5] For some algorithms, a tiny increase in the threshold gives more than one extra customer, so the short list is not always exactly 60.

[6] Other MLP learning parameters were very ordinary: the optimizer was `RMSpro`, the activation function was `relu` for the hidden layers and `sigmoid` for the output, the initializer was `GlorotUniform`.

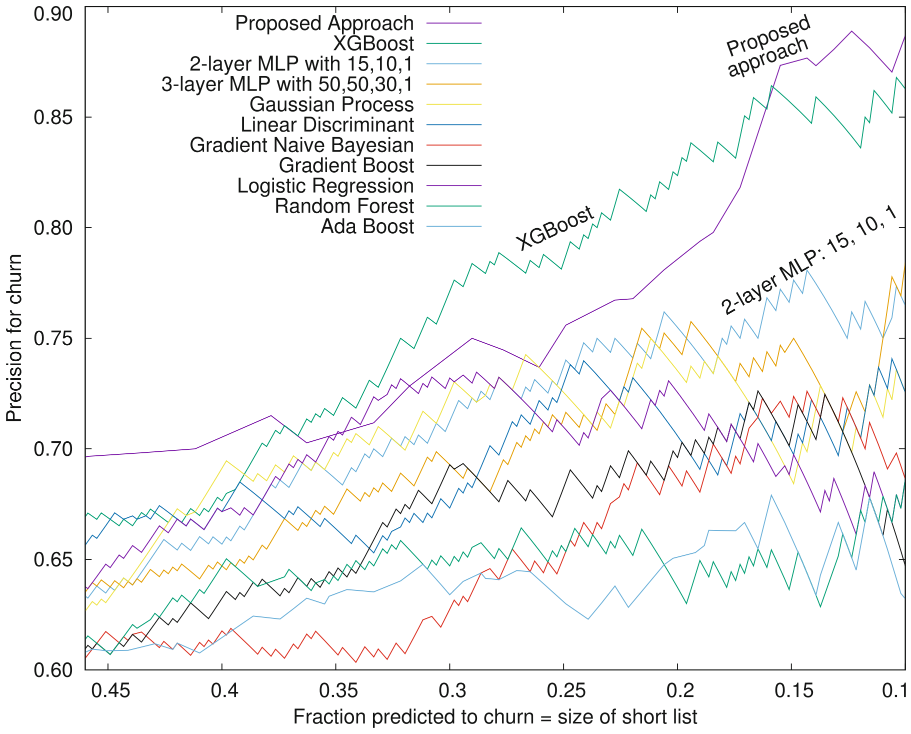

Fig. 6. The client wants a short list of at least 60 of the 510 customers, such that about 90% of them do churn. The y-axis shows how many are in the short list, which gets shorter by adjusting the algorithm's threshold. The x-axis shows how many customers on that short list did in fact churn, being precision = TP/(TP + FP). The proposed method gives a short list from which 88.89% did churn. Second-best is XGBoost, and third-best is a neural network with 15 and 10 neurons in the two hidden layers.

because the data set is so small that there just isn't enough signal to tune a large number of parameters: the next-best of all these attempts was three hidden layers with 50, 50, and 30 neurons, which did slightly worse than SciKit's default neural network algorithm at 73.33%. This is comparable to another study that used deep learning for e-commerce customer churn prediction [18] which achieved precision of 78%.

The one-tailed 2-sample Z-test for proportions on the difference between the proposed method and XGBoost in Table 1 does not show a significant difference, with a p-value of 0.284, but the client still likes 88.9% more than 84.1%.

Comparing the proposed method with the next-best algorithm (a shallow neural network), the Z-test gives a p-value less than 0.05 which suggests a significant improvement. It's possible that more tinkering might help, but adding more layers to the neural network tends to give lower accuracy in Table 1. Random

Table 1. The proposed method narrowly out-performs XGBoost, and is significantly better ($p < 0.05$) than deep learning with either small or large neural networks.

Algorithm	Precision	TP	FP	Short list
Proposed Approach	0.8889	56	7	63
XGBoost	0.8413	53	10	62
2-layer neural network: 15,10,1 nodes	0.7667	46	14	60
3-layer neural network: 50,50,30,1 nodes	0.7167	43	17	60
Gaussian Process	0.7167	43	17	60
Linear Discriminant	0.7167	43	17	60
Gaussian NB	0.7000	42	18	60
Gradient Boost	0.6833	41	19	60
Logistic Regression	0.6833	41	19	60
Random Forest	0.6667	40	20	60
4-layer neural network: 15,12,10,9,8,1 nodes	0.6667	40	20	60
Ada Boost	0.6452	40	22	62

Forest and Gradient Boost work impressively on many other problems, but in this scarce-data task they are at 66.67% and 68.33% respectively.

As mentioned in Sect. 2.2, the data set used here is close to balanced, with 51.6% of customers churning and 48.4% staying, so no claims are made for how well it might cope with unbalanced data sets.

5 Discussion and Conclusion

The proposed approach achieves more accuracy than state-of-the-art algorithms including XGBoost and deep learning, on a real small-data low-dimensional prediction problem whose inputs are weak discriminants, the kind of problem faced by millions of organizations of all sizes.

The result here is broadly in agreement with a previous study about predicting who has cancer [8]. There are many other scarce-data prediction problems, with inputs that are poor discriminants: diagnosing unusual diseases, medical triage prediction [10], and hospital readmission [9,12,14], to name a few.

Like the customer churn problem covered here, many medical prediction problems also have a cost-benefit aspect. For example, a hospital needs a short list of patients who are most in need of care (to match the limited number of available beds), so the accuracy of that short list matters, not accuracy overall.

For most hospitals, these are scarce-data problems: while common diseases like influenza infect millions of people each year, those diseases behave differently from place to place and year to year, so 5-year-old data on influenza in Norway's sub-arctic [23] could be misleading for influenza in Australia's dry outback, due to the evolution of the virus, and differences in weather, housing, and immunities.

Acknowledgments. The author thanks Lachlan Butler, Harrison Burrows, and Juan Moredo for technical support. The author also thanks the anonymous peer reviewers for critically reading the manuscript and suggesting substantial improvements.

References

1. Ahmad, A.K., Jafar, A., Aljoumaa, K.: Customer churn prediction in telecom using machine learning in big data platform. J. Big Data **6**(1), 1–24 (2019). https://doi.org/10.1186/s40537-019-0191-6
2. Albuquerque, R.A.S., Costa, A.F.J., dos Santos, E.M.: A decision-based dynamic ensemble selection method for concept drift. In: 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, pp. 1132–1139. IEEE Computer Society, November 2019. https://doi.org/10.1109/ICTAI.2019.00158
3. Berger, P., Kompan, M.: User modeling for churn prediction in E-commerce. IEEE Intell. Syst. **34**(2), 44–52 (2019). https://doi.org/10.1109/MIS.2019.2895788
4. Braytee, A., Anaissi, A., Kennedy, P.J.: Sparse feature learning using ensemble model for highly-correlated high-dimensional data. In: Cheng, L., Leung, A.C.S., Ozawa, S. (eds.) ICONIP 2018. LNCS, vol. 11303, pp. 423–434. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04182-3_37
5. Caves, C.M., Fuchs, C.A., Schack, R.: Quantum probabilities as Bayesian probabilities. Phys. Rev. A **65**(2), 22305 (2002). https://doi.org/10.1103/PhysRevA.65.022305
6. Cerqueira, V., Torgo, L., Pinto, F., Soares, C.: Arbitrage of forecasting experts. Mach. Learn. **108**(6), 913–944 (2019). https://doi.org/10.1007/s10994-018-05774-y
7. Darwen, P.J.: The varying success of Bayesian model averaging: an empirical study of flood prediction. In: Sundaram, S. (ed.) 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1764–1771. IEEE, November 2018. https://doi.org/10.1109/SSCI.2018.8628939
8. Darwen, P.J.: Cost-effective prediction in medicine and marketing: only the difference between Bayesian model averaging and the single best-fit model. In: 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, pp. 1274–1279. IEEE Computer Society, November 2019. https://doi.org/10.1109/ICTAI.2019.00178
9. Garmendia-Mujika, A., Graña, M., Lopez-Guede, J.M., Rios, S.: Neural and statistical predictors for time to readmission in emergency departments: a case study. Neurocomputing **354**, 3–9 (2019). https://doi.org/10.1016/j.neucom.2018.05.135
10. Garmendia-Mujika, A., Rios, S.A., Lopez-Guede, J.M., Graña, M.: Triage prediction in pediatric patients with respiratory problems. Neurocomputing **326**, 161–167 (2019). https://doi.org/10.1016/j.neucom.2017.01.122
11. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis. Texts in Statistical Science Series, 2 edn. Chapman-Hall, Boca Raton (2004). https://doi.org/10.1201/9780429258411
12. Hammoudeh, A., Al-Naymat, G., Ghannam, I., Obied, N.: Predicting hospital readmission among diabetics using deep learning. Procedia Comput. Sci. **141**, 484–489 (2018). https://doi.org/10.1016/j.procs.2018.10.138
13. Hastie, T., Tibshirani, R., Friedman, J., Franklin, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7

14. Hooijenga, D., Phan, R., Augusto, V., Xie, X., Redjaline, A.: Discriminant analysis and feature selection for emergency department readmission prediction. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 836–842. IEEE, November 2018. https://doi.org/10.1109/SSCI.2018.8628938
15. Liu, X., Xie, M., Wen, X., Chen, R., Ge, Y., Duffield, N., Wang, N.: A semi-supervised and inductive embedding model for churn prediction of large-scale mobile games. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 277–286. IEEE Computer Society (2018). https://doi.org/10.1109/ICDM.2018.00043
16. Najafi, M., Moradkhani, H., Jung, I.: Assessing the uncertainties of hydrologic model selection in climate change impact studies. Hydrol. Process. **25**(18), 2814–2826 (2011). https://doi.org/10.1002/hyp.8043
17. de O. Nunes, R., Dantas, C.A., Canuto, A.P., Xavier, J.C.: Dynamic feature selection for classifier ensembles. In: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), pp. 468–473. IEEE Computer Society, October 2018. https://doi.org/10.1109/BRACIS.2018.00087
18. Pondel, M., Wuczyński, M., Gryncewicz, W., Łysik, Ł., Hernes, M., Rot, A., Kozina, A.: Deep learning for customer churn prediction in e-commerce decision support. In: Proceedings of the 24th International Conference on Business Information Systems, pp. 3–12. TIB Open Publishing (2021). https://doi.org/10.52825/bis.v1i.42
19. Praseeda, C., Shivakumar, B.: Fuzzy particle swarm optimization (FPSO) based feature selection and hybrid kernel distance based possibilistic fuzzy local information C-means (HKD-PFLICM) clustering for churn prediction in telecom industry. SN Appl. Sci. **3**(6), 1–18 (2021). https://doi.org/10.1007/s42452-021-04576-7
20. Rapeli, L.: Does sophistication affect electoral outcomes? Gov. Oppos. **53**(2), 181–204 (2018). https://doi.org/10.1017/gov.2016.23
21. Saadallah, A., Priebe, F., Morik, K.: A drift-based dynamic ensemble members selection using clustering for time series forecasting. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) ECML PKDD 2019. LNCS (LNAI), vol. 11906, pp. 678–694. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46150-8_40
22. Subramanya, K.B., Somani, A.: Enhanced feature mining and classifier models to predict customer churn for an E-retailer. In: Confluence 2017: 7th International Conference on Cloud Computing, Data Science and Engineering, pp. 531–536. IEEE (2017). https://doi.org/10.1109/CONFLUENCE.2017.7943208
23. Tazmini, K., Nymo, S.H., Louch, W.E., Ranhoff, A.H., Øie, E.: Electrolyte imbalances in an unselected population in an emergency department: a retrospective cohort study. PLoS ONE **14**(4), e0215673 (2019). https://doi.org/10.1371/journal.pone.0215673
24. Wurl, A., Falkner, A.A., Haselböck, A., Mazak, A., Sperl, S.: Combining prediction methods for hardware asset management. In: Proceedings of the 7th International Conference on Data Science, Technology and Applications - DATA 2018, pp. 13–23. SciTePress (2018). https://doi.org/10.5220/0006859100130023
25. Zhu, Y., et al.: Addressing the item cold-start problem by attribute-driven active learning. IEEE Trans. Knowl. Data Eng. **32**(4), 631–644 (2020). https://doi.org/10.1109/TKDE.2019.2891530