Detecting Unusual Expense Categories for Financial Advice Apps

Axel Brando
BBVA Data & Analytics | Universitat de Barcelona
axel.brando@bbvadata.com | axelbrando@ub.edu

Jose A. Rodríguez-Serrano
BBVA Data & Analytics
Spain
names.surname1.surname2@bbvadata.com

Jordi Vitrià
Departament de Matemàtiques i Informàtica
Universitat de Barcelona
jordi.vitria@ub.edu

KDD Workshop on Anomaly Detection in Finance, August 2019, Anchorage, USA

ABSTRACT

The abundance of data about user transactions and interactions has fostered the use of machine learning techniques in financial institutions and startups. Going beyond classical application areas, such as campaign management, these methods can be used to personalize services at different levels. In this paper we explore a use case, related to mobile banking apps, to forecast unusual expenses. Building a system to forecast customer expenses involves large amounts of data and requires the use of high capacity machine learning models. Deep neural networks are clear candidates to this end, but the mainstream application of this technology is still based on producing point estimates. This is a clear limitation in scenarios where being aware of the uncertainty in prediction is crucial. This paper proposes the use of neural networks that yield distributions rather than point estimates. This extension can be applied to any neural network by considering a loss function corresponding to the log-likelihood of a Laplacian distribution of the output. We show with a large transaction dataset that it is possible to leverage uncertainty information in order to **robustly detect outliers, corresponding to unusual expense categories** and that this information could be effectively translated into notifications to users in the app, which would allow them to review and better understand their expenses.

KEYWORDS

Heteroscedastic uncertainty, deep learning, financial applications.

ACM Reference Format:
Axel Brando, Jose A. Rodríguez-Serrano, and Jordi Vitrià. 2019. Detecting Unusual Expense Categories for Financial Advice Apps. In *Proceedings of KDD Workshop on Anomaly Detection in Finance.* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

Banks, other financial institutions and startups have started offering personalized money management and advice tools to their users. As a recent benchmark shows[15], there is increasing expectation for financial tools to offer services such as predictive insights, personalized offers or automation. Not surprisingly, this domain has witnessed progress in machine learning algorithms, such as recommenders [2, 13], to power such functionalities, or to optimize aspects such as customer retention [10].

As an example, the BBVA mobile app provides more than 5M customers with advice services such as categorization of expenses, aggregation of other bank accounts, forecasting expenses, personalized alerts or estimation of real estate prizes, among others.

Use Case: Unusual Expense Forecasting. One of the BBVA mobile app functionalities, 'Forecasted Expenses', tries to estimate the monetary amount that a user will spend *next month* on a given category of interest[1] . Here, the term 'categories' denote expense codes grouped to a certain level of semantics, e.g. "groceries" or "electricity bill". This is typically the output of a categorizer, which nowadays is a basic building block in Personal Finance Management (PFM) Tools [15].

In the app, a forecasting model is employed to estimate expenses in month number $T + 1$ from the historical series of T months. The app then displays a list of such anticipated expenses. In some particular cases (e.g. a recurring bill), the expense is due to a single transaction and a model tries to estimate the arrival date; in such a case, the upcoming transaction is displayed in a so-called Financial Calendar. Fig. 1 and 2 show screenshots of the Calendar View and the Categories view for a real sample of Forecasted Expenses.

We notice that human expenses can be intermittent, very variable and even inconsistent or erratic. Therefore, forecasting expenses is a difficult task, and in many cases it is possible that there is little signal that can be used to anticipate the next month. While there are still a set of situations where we observe signal, such as bills, or recurrent expenses, many forecasts will be meaningless – or in practice reduce to the mean of the past expenses. In many cases, users will not be that interested in knowing the exact forecast, but more interested in *knowing whether the actual expense they made significantly deviates from the forecast.* This corresponds to (positive and negative) situations such as spending higher than a certain usual budget, an expected income that did not arrive, achievements in savings or changes in spending behavior. These are day-to-day money management situations that can go unnoticed and that we might be interested in detecting and signaling them to customers, e.g. through notifications.

[1] In this text, we will always talk about 'expenses' to simplify, but note that, without loss of generality, the method can also be applied to detect unusual income. From now on we treat income as a "negative" expense, and hence the same framework is applicable.

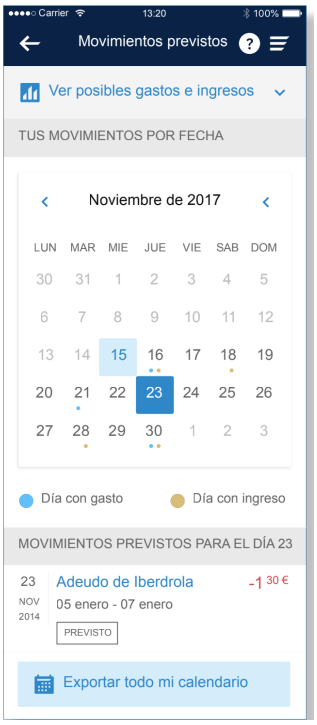

Figure 1: Forecasted Expenses in the BBVA App: Calendar View

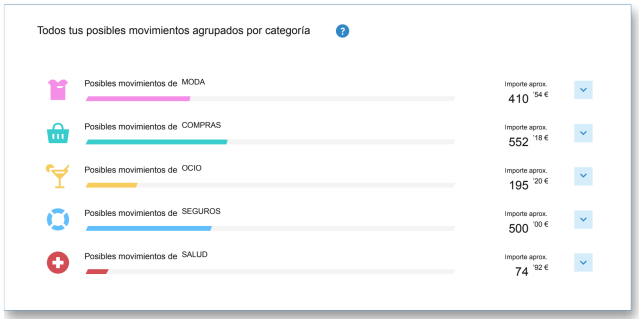

Figure 2: Forecasted Expenses in the BBVA App: Categories View

In this work, we describe part of an exploratory study where the goal is to detect unusual expenses using real data from the aforementioned application, and deep neural network models.

Next, we explain how mainstream implementations of deep networks fail at outputting a distribution and our envisaged solution for unusual expense detection.

2 MODEL: FORECASTING AND UNUSUAL EXPENSE DETECTION

2.1 Background on forecasting expenses

For each user and each category, we can construct a series of T historical values of past expenses, which we denote $\mathbf{y} = \{y_1, \ldots, y_T\}$. The task of the forecasting module is to estimate a function

$$\hat{y}_{T+1} = \phi(\mathbf{y})$$

such that $\hat{y}_{T+1}$ is the estimated value of the expense in the next month, for the given user and category.

There are many forecasting (and regression) models that can be applied to fit a dataset of windows of the form $(\{\mathbf{y}, y_{T+1}\})$ and approximate $\phi(\cdot)$ [9]. A common choice are methods based on random forests [4, 12] or gradient boosting [7]. Another common choice would be to use deep learning methods. We have confirmed in a previous study [6] that deep learning yields more accurate *point estimates*; i.e estimates of y_{T+1} achieving lower error with respect to the true y's, when evaluated on common regression error metrics such as mean absolute error or rooted mean-squared error [9].

However, one problem of deep learning models for regression is that they provide *point estimates* of y_{T+1}, but not information related to the uncertainty or the distribution of the of the estimate. This is because the straightforward version of a neural network for regression contains a single output layer corresponding to the response variable y_{T+1}. Under these circumstances, detecting an unusual expense can be done by e.g. measuring the absolute difference between the observed expense and the forecasted one, $|y_{T+1} - y_{obs}|$. But this ignores the variance of the estimate (a large difference can be due to a bad estimate or a large variance). Therefore, *we seek a more principled method to detect unusual expenses using deep regression networks.*

2.2 Capturing uncertainty with neural networks

To be capable of detecting unusual expenses, we need the neural network to output a distribution, instead of a point estimate.

Network architecture. Inspired by the literature in neural networks for heteroscedastic data [3, 5, 14], we propose a network architecture that produces an output of the form:

$$\hat{\mu}_{T+1} = \phi(\mathbf{y})$$

$$\hat{\sigma}_{T+1} = \psi(\mathbf{y})$$

Here, $\phi : \mathbb{R}^T \to \mathbb{R}$ and $\psi : \mathbb{R}^T \to \mathbb{R}$ denote the output of neural networks which take $\mathbf{y}$ as input, and each output a single scalar. Those scalars represent the location and scale parameters of a unimodal distribution of choice (e.g. Normal or Laplacian), which we call $g(y_{T+1}|\mu, \sigma)$ for generality.

One can interpret this setting as inputting the vector of historical values $\mathbf{y}$ and outputting a distribution.

For clarity, an example of such a network is illustrated in Fig. 3, taking the exemplary architecture of a multi-layer perceptron, where the input $\mathbf{y}$ and outputs $\hat{\mu}$ and $\hat{\sigma}$ are linked by fully-connected layers.

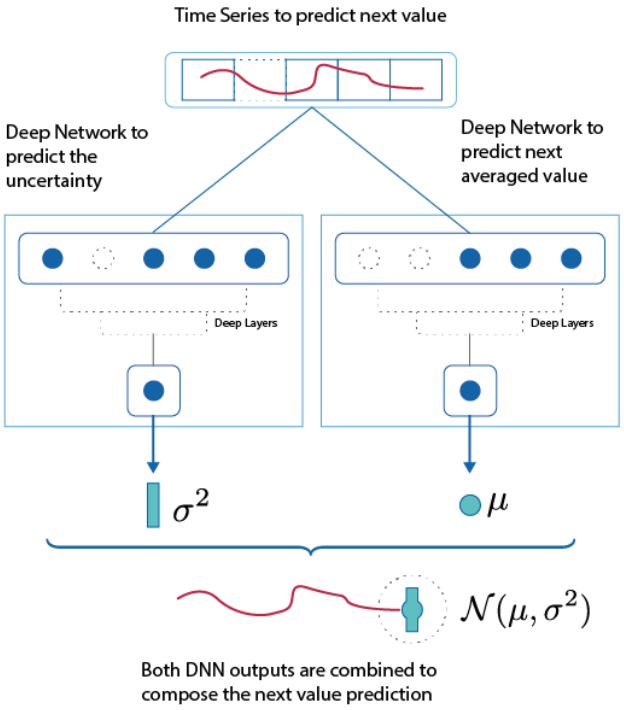

Figure 3: Network architecture to predict a distribution.

Note that this is for illustration purposes, and that the formulation used in this paper is independent from the specific network architecture (as long as the loss function is differentiable). Therefore, we note other choices are possible and the only requirement is outputting two scalars. Therefore, the network could also contain convolutional layers or LSTM layers [11] – in fact, the latter are used in experiments.

Differently from "vanilla" neural network architectures, which would have a single output and could be minimized by optimizing e.g. a loss capturing a sum-of-squared errors, now we have two outputs $\hat{\mu}$ and $\hat{\sigma}$ that represent a distribution. Therefore, in order to estimate the parameters of such a network, we minimize a loss suitable for distrubutions, such as the log-likelihood.

Regardless of the distribution, one advantage is that the outputs of the network, ϕ and ψ, can be interpreted directly as the expected value y_{T+1} and its associated "uncertainty".

Training. As in any probabilistic model, an objective function that is suitable to fit parameters so that the distributions $g(\cdot)$ best fit the observed data is the log-likelihood:

$$log\mathcal{L}(\mathbf{W}, \mathbf{V}) = \sum_{n=1}^{N} \log g(y_n|\phi(\mathbf{y}_n; \mathbf{W}), \psi(\mathbf{y}_n; \mathbf{V})) \quad (1)$$

Here, $\mathbf{W}$ denotes all the parameters (weights) of the network responsible for predicting ϕ, while $\mathbf{V}$ denotes the weights of the network responsible for predicting ψ. As noted in Eq. 1, we have made explicit that ϕ and ψ depend on $\mathbf{W}$ and $\mathbf{V}$. Additionally, $\mathbf{y}_n$ and y_n is the dataset of N samples, consisting of pairs of historical value and the true value to predict.

In order to find the weights $\mathbf{W}$ and $\mathbf{V}$ that maximize the objective function in Eq. 1, we typically use stochastic gradient ascent methods. With current tools such as `TensorFlow` [1], which allow automatic differentiation of arbitrary computation graphs, we can input that objective function and optimize the weights against a training set. The particular distribution chosen and specific loss function will be discussed in the experimental section.

2.3 Outlier detection

Once the network is trained, it can be used to forecast the distribution of the next expense value in a certain category for a certain user.

For a user and category with a historical series $\mathbf{y}$, who observes a value y_{obs} as the next expense, one can compare this observation to the predicted distribution $g(y_{T+1}|\phi(\mathbf{y}), \psi(\mathbf{y}))$. This allows checking whether the new observation is likely according to the predicted distribution.

In fact, the likelihood $g(y_{obs}|\phi(\mathbf{y}), \psi(\mathbf{y}))$ could be a direct indicator of 'normality' which takes into account *both the absolute error and the variance of the prediction.* Perhaps in order to be more interpretable, one could compute the probability, under g, that an expense is higher than y_{obs}.

$$P(Y > y_{obs}) = 1 - G(y_{obs}) \quad (2)$$

where G denotes the cumulative distribution, which has known closed-form for cases such as those of the Normal or Laplacian distributions.

Therefore, we would flag an expense as unusual if the corresponding indicator is below a pre-specified threshold.

3 EXPERIMENTAL VALIDATION

We have carried out a proof of concept of the proposed method. For a quantitative evaluation, we used an anonymized and aggregated dataset of expenses containing 2M pairs of samples ($\mathbf{y}$, y_{T+1}) for training and 1M for testing, corresponding to real expenses in certain categories of users with explicit consent.

We highlight that in this paper we evaluate the capability of the heteroscedastic model to detect outliers. An exhaustive evaluation on how the model performs in terms of forecasting accuracy and uncertainty modeling (without the outliers use case) can be found in [5], as well as comparison to other models.

Below we specify some details of the implementation and evaluation.

Output distribution. As the specific choice of output distribution $g(\cdot)$, we took a Laplacian distribution. In preliminary experiments, we noticed that a Laplacian converges faster and yields better results than a Normal distribution. We attribute this observation to the fact that our data contains many (and strong) outliers.

When plugging in the specific equation of the Laplacian distribution into the loss function 1, one obtains:

$$\mathcal{L}\left(\boldsymbol{W}, \boldsymbol{V}; \{(\boldsymbol{y}_i, y_i)\}_{i=1}^{N}\right) = -\sum_{i=1}^{N}\left[-\log(\Psi(\boldsymbol{y}_i)) - \frac{1}{\Psi(\boldsymbol{y}_i)} \mid y_i - \phi(\boldsymbol{y}_i) \mid\right] \tag{3}$$

Details of the Network architecture. After a refinement process, the neural network architecture used was 2 LSTM layers [8] with 128 output neurons each one concatenated with two dense layers of 128 and 2 output neurons, respectively.

Train/Test split. The test series consist of a history of T months, and the target is the expense in month $T+1$. For training, we use series which have *the same month of the previous year as target*, and a history of the same length. In total we use 2M series for training and 1M for testing.

3.1 Qualitative results

Fig. 4 shows some examples of series and compares the next true value with the predicted distribution. In order to visualize the predicted distribution, we plot the location and scale parameters; location $\Phi(\boldsymbol{y}_i)$ as a point, and the scale parameter as an error bar of lengh $\Psi(\boldsymbol{y}_i)$ centered in $\Phi(\boldsymbol{y}_i)$.

In Fig. 4 we show three types of examples. The first row shows examples of correct forecast, where the observed target expense (cross) is near the prediction (red dot), and the scale parameter of the distribution is small, indicating a sharp distribution, such as a periodic trimonthly bill (first row, right). The second row shows examples where the absolute difference between the observed target expense and the prediction is far away in absolute terms, but still likely when taking into account the distribution. Finally, the last row contains examples where the prediction is far off the observed value, both in absolute terms and taking into account the distribution. Here, oth columns show an expense many times larger the maximum amount of the previous expense. *These cases are what we would label as unexpected expenses, whe ones that we would like to signal to users.*

What is clear from Fig. 4 is that, in order to perform outlier detection, it is crucial to predict a distribution, rather than a single point estimate. In the cases where the error bar depicting the scale is large, a big absolute error is not necessarily an indicator of "unusual".

3.2 Quantitative evaluation

To evaluate the performance of outlier detection, we labelled some series as "outliers" and evaluate if the outliers lie in the last positions when ranking the test examples by decreasing likelihood.

To label the bins as outliers, we took a test set with samples of the form $(\mathrm{y}, y_{T+1}, y_{obs})$. We define the error as $|y_{T+1} - y_{obs}|$, sorted by increasing error and discretized into 10 bins, which we call $Q10, \ldots, Q100$. We assume that the bins (Q80, Q90, Q100) correspond to samples with very large errors and, regardless of the response distributions, will contain a huge fraction of outliers. We verify this by visual inspection. All in all, Q80, Q90 and Q100 are classes of samples with (increasingly) high presence of outliers.

Then, we will sort all the samples of the test set by likelihood. For each likelihood value, we compute:

- x-axis: The total % of samples above that likelihood value
- y-axis: The % of samples of a certain outlier class (Q80, Q90 or Q100) with likelihood above that value.

With that definition, Fig. 5 shows the reject curves for Q80, Q90 or Q100. The result is satisfactory: the samples with high likelihoods (small % values of the x-axis) contain a small number of outliers, and outlier presence signigicanly increases for the samples with lowest likelihoods (i.e. high % values the x-axis). That the increase occurs further in the x-axis, depending on whether we take Q80, Q90 and Q100, which are the outlier classes sorted by 'severity'.

This indicates that the the current model is effectively assigning low likelihoods to outliers, and consistently more to stronger outliers. These outlier detections could translate into notifications to users in the app so that they can review and better understand their expenses.

4 CONCLUSION

Motivated by the use of a deep neural network model in a financial application, we study a baseline method to extend the output of neural networks to yield distributions rather than point estimates. We show that this extension can be easily applied to any deep regression network by considering a probabilistic loss function. The produced estimators shows promising results in a real scenario that involves millions of bank customers.

Further extensions could be defined in a number of directions. One of the most promising directions is the extension of this model to deal with heterogeneous distributions. The use of simple probabilistic models to characterize output uncertainty, mainly the Normal and Laplacian models, presents some clear limitations if we are interested in managing outliers based on their uncertainty. For example, uncertain outcomes related to multimodal outputs could require a treatment that is significantly different for unimodal outputs with large variance. This limitation can be overcome by designing neural networks that can output an arbitrary, heterogeneous distribution for every input.

5 ACKNOWLEDGMENTS

We gratefully acknowledge the Industrial PhD Plan of Generalitat de Catalunya with BBVA Data and Analytics for funding this research. The work of Jordi Vitria was partially funded by TIN2015-66951-C2, RTI2018-095232-B-C21 and SGR 1219. We would like to thank Cesar de Pablo and Damia Torres for fruitful discussions and Jairo Mejia and Joan Llop for producing the designs of some of the figures.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Leonardo Baldassini and Jose Antonio Rodríguez-Serrano. 2018. client2vec: Towards Systematic Baselines for Banking Applications. abs/1802.04198 (2018). http://arxiv.org/abs/1802.04198

[3] Christopher M Bishop. 1994. Mixture Density Networks. (1994).

[4] Ash Booth, Enrico Gerding, and Frank Mcgroarty. 2014. Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications* 41, 8 (2014), 3651–3661.

[5] Axel Brando, Jose A Rodríguez-Serrano, Mauricio Ciprian, Roberto Maestre, and Jordi Vitrià. 2018. Uncertainty Modelling in Deep Networks: Forecasting Short and Noisy Series. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 325–340.

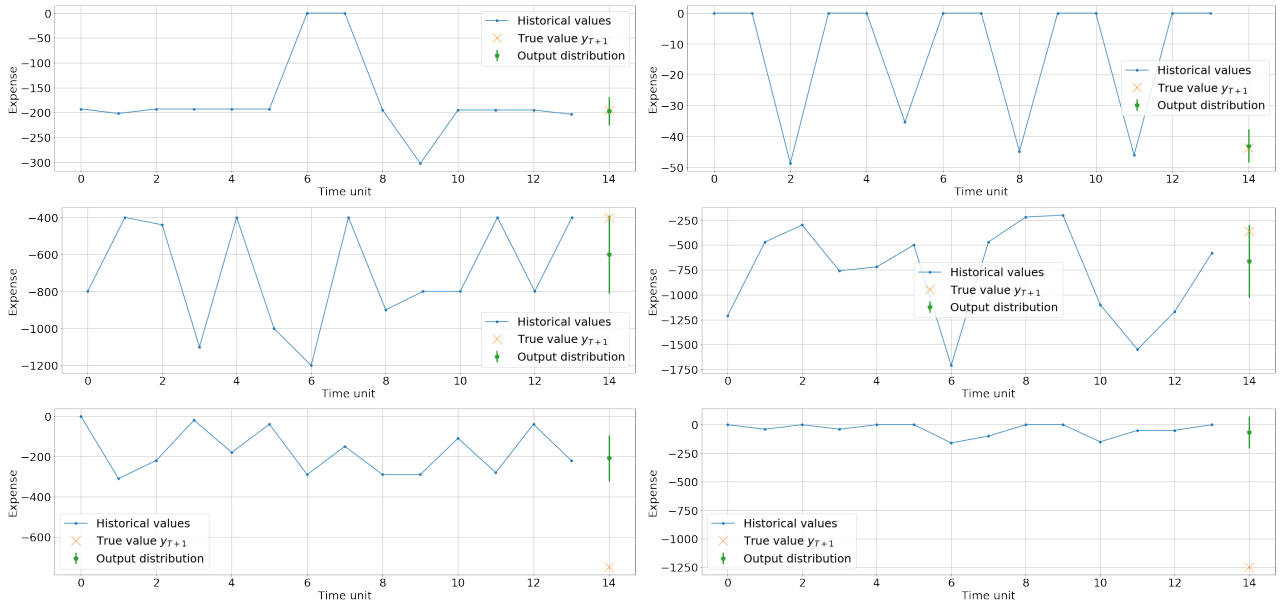

Figure 4: Examples of expense series (solid blue), their observed target expense (cross) and their prediction, showing the parameters of the predicted distribution: predicted location (red dot) and scale (error bar). 1st row: Examples of accurate forecasts. 2nd row: Examples of forecast with significant absolute deviation, but within the "normality", according to the distribution. 3rd row: Outliers, i.e. forecasts with large absolute deviation and low likelihood in the output distribution.

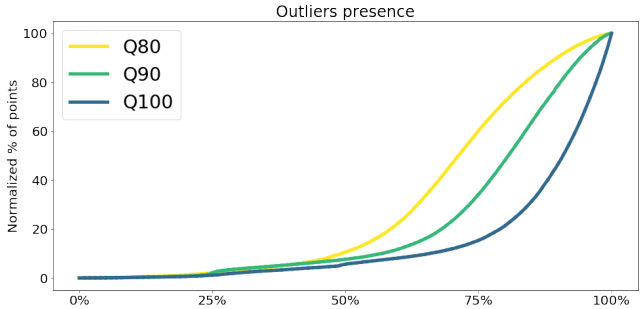

Figure 5: Outlier (unexpected expense) detection performance.

[6] M. Ciprian, L. Baldassini, L. Peinado, T. Correas, R. Maestre, J.A. Rodriguez-Serrano, O. Pujol, and J. Vitrià. 2016. Evaluating uncertainty scores for deep regression networks in financial short time series forecasting. *Workshop on Machine Learning for Spatiotemporal Forecasting, in NIPS* (2016).

[7] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice*. OTexts.

[10] Dudyala Anil Kumar, Vadlamani Ravi, et al. 2008. Predicting credit card customer churn in banks using data mining. *International Journal of Data Analysis Techniques and Strategies* 1, 1 (2008), 4–28.

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.

[12] Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[13] Sandra Mitrović and Gaurav Singh. 2016. Predicting Branch Visits and Credit Card Up-selling using Temporal Banking Data. *ECML/PKDD* (2016).

[14] James W. Taylor. 2000. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. *Journal of Forecasting* 19 (2000), 299?311. Issue 4.

[15] Minna Technologies. [n. d.]. The 2019 Retail Banking PFM Benchmark. https://minnatechnologies.com/personal-finance-management/.