Improving Operation Efficiency through Predicting Credit Card Application Turnaround Time with Index-based Encoding

Jing Xiang Toh
Analytics Centre of Excellence, DBS Bank Singapore, joshuatohjx@dbs.com

Kay Jan Wong
Analytics Centre of Excellence, DBS Bank Singapore, kayjanwong@dbs.com

Samarth Agarwal
Analytics Centre of Excellence, DBS Bank, Singapore, samarthagarwal@dbs.com

Xuejie Zhang*
Analytics Centre of Excellence, DBS Bank, Singapore, xuejiezhang@dbs.com

John Jianan Lu
Analytics Centre of Excellence, DBS Bank, Singapore, johnlu@dbs.com

Abstract

This paper presents the successful use of index encoding and machine learning to predict the turnaround time of a complex business process – the credit card application process. Predictions are made on in-progress processes and refreshed when new information is available. The business process is complex, with each individual instance having different steps, sequence, and length. For instances predicted to have higher than normal turnaround time, model explain-ability is employed to identify the top reasons. This allows for intervention in the process to potentially reduce turnaround time before completion.

CCS CONCEPTS •Applied computing~Enterprise computing~Business process management~Business process modeling •Computing methodologies~Machine learning~Learning paradigms~Supervised learning~Supervised learning by regression

Additional Keywords and Phrases: Complex Business Process, Index Encoding, Machine Learning, Model Explain-ability, Consumer Banking, Credit Card Application

ACM Reference Format: Jing Xiang Toh, Kay Jan Wong, Samarth Agarwal, Xuejie Zhang, John Jianan Lu. 2022. Improving Operation Efficiency through Predicting Credit Card Application Turnaround Time with Index-based Encoding. In The Web Conference 2022, FinWeb: The 2nd Workshop on Financial Technology on the Web, April 25-26, 2022, Lyon, France.

1 Introduction & Background

Business processes in the banking and finance industry can be complex by nature in practice. In credit card applications, complexity arises from product categories, credit ratings of customers, channels from which applications are submitted and the completeness of supporting documents etc. As such, credit card applications can differ in terms of processing steps, sequence of steps and total number of steps.

* Denotes corresponding author

It is in the interest of the bank to shorten the turnaround time of credit card applications. Machine learning can be used to predict turnaround time of on-going applications and identify applications that are predicted to take longer than normal to complete while model explain-ability can identify the variables that contributed to the prediction. This will allow the bank, especially the operation team, to proactively intervene and possibly reduced the actual turnaround time.

Index encoding can be used to achieve the above. The credit card application process needs to be encoded so that the complexity of the process, including differing steps, sequence and number of steps can be captured in a machine learning model. Then predictions can be made at any point throughout the on-going application process. Model explain-ability, apart from being used to identify the top variables, can be built on to provide actionable follow ups and to improve the business process. For example, suppose an application is predicted to take longer than normal due to inconsistent information submitted, the customer can be reached out to earlier.

This paper demonstrates how turnaround time of a complex business process can be modelled with index-based encoding and how it can be reduced by using model explain-ability to identify and mitigate variables that correlates with high turnaround time.

2 Literature Review & Related work

Predictive monitoring of business processes falls under the category of process mining [1] that aims to predict the outcome of on-going processes. Outcomes can include whether a medical treatment process is successful or if a purchase order will be fulfilled by potential customers.

Earlier methods of process mining include sequence classification [2, 3], where initial simple symbolic sequences of processes are extracted from past historical data and mapped against on-going processes to determine the outcome. These methods take the latest snapshot of the variables but not how the variables changed and evolved to the current state.

Later approaches extract complex symbolic sequences that retain information on the evolution of the variables [4]. In particular, index-based encoding and a combination of index-based encoding and encoding based on Hidden Markov [5] Models (HMMs) have been shown to reliably predict outcomes by a substantial margin when compared to traditional Boolean, frequency-based or last state encoding. This is so even at initial stages of the business process.

This paper focus on index-based encoding instead of the combination mentioned above as the latter requires additional compute resources but do not provide significant or consistent improvement in performance.

3 Modelling Turnaround Time

All numbers in this paper were altered due to sensitivity of information but the altered numbers do not change the conclusions or insights derived.

3.1 Label – Turnaround Time (TAT)

The objective of the model is to predict the turnaround time of credit card applications. Specifically, turnaround time is the duration from when an application is received in the system to when a decision has been made to approve or reject the application. Majority of the applications are completed in a few days. For applications that take longer than normal, common reasons encompass insufficient or inconsistent documents submitted or issues with credit worthiness.

3.2 Static and Dynamic Variables

Variables in the credit card application process can be categorized into static and dynamic variables. The latter category of variables are variables that will be index encoded.

Static variables are variables that are processed from the application or engineered. These variables are available at the start of the application, and they remain constant throughout the entire process. In total, there are 14 processed static variables and 4 engineered static variables. Examples of processed static variables include whether information submitted for the application is clean, channel from which the application is submitted and nationality of the applicant etc. Engineered static variables include the day of week that an application is submitted, total number of applications submitted on the same day and so on.

Dynamic variables are variables that are updated at every step of the application process. The number of steps for each application is capped at 100. Distribution of the number of steps for the applications is long tailed – capping at 100 steps reduces computing resources and 98% of applications have less than 100 steps. Capturing how the dynamic variables evolve throughout the process enables better prediction of turnaround time.

Within each step, there are 6 processed dynamic variables and 5 engineered dynamic variables. Examples of processed dynamic variables include the step description, step status, discrepancies raised if any and step time taken etc. Engineered dynamic variables include the total time taken since the start of the application and number of steps taken so far.

3.3 Last State Variables

Last state variables are essentially dynamic variables from the last known step of the business process and are added to improve the predictive power of the model. These variables allow the model to capture the progress of each application relative to the progress of other applications.

3.4 Index – Based Encoding

Index based encoding enables applications with different steps and length to be modelled such that the complex symbolic sequences are captured and that a prediction can be made at any point of the process.

Table 1 shows index-based encoding being used with the static, dynamic and last state variables mentioned in earlier sections. Due to the large number of variables, only two static variables, two last state variables, one dynamic variable and the first nine steps are shown.

In Table 1, two applications from two different customers are shown and this is reflected in the 'Customer' column. For each of the application, updates throughout the process at different timing are appended in new rows and this is reflected in the 'Updates' column. The corresponding time taken for each of the application is reflected in the 'Turnaround Time' column.

Using application 1 as an example, static variables remain constant throughout the process while the dynamic variables are updated. Last state variables reflect the last known step of the corresponding dynamic variables. A single update has at least one new step of dynamic variables being added. When the updates across the two different applications are compared, it is seen that the complex sequence unique to each application is captured in the encoding.

Prediction of turnaround time can be made at the first update and refreshed with every subsequent update. As the number of updates increases, new information is added, and the prediction becomes more accurate. It is important that the prediction is accurate with as few updates as possible for the prediction to be useful.

Table 1: **Example of Index Encoding used with Static, Dynamic and Last State Variables**

Customer	Updates	Turnaround Time	Remaining TAT	Static Variables (only 2 shown)		Last State Variables (only 2 shown)		Dynamic Variable (only step description is shown and only first 9 steps are shown)								
#	#	Days	Days	Clean	Channel	Step Desc	Number Of Steps	Step 1 Desc	Step 2 Desc	Step 3 Desc	Step 4 Desc	Step 5 Desc	Step 6 Desc	Step 7 Desc	Step 8 Desc	Step 9 Desc
1	1	3	3	Y	Online	bureau out	6	edit	bureau in	cross check	pre val	app score	bureau out	NA	NA	NA
1	2	3	2	Y	Online	judgement	7	edit	bureau in	cross check	pre val	app score	bureau out	judgement	NA	NA
1	3	3	1	Y	Online	decision	9	edit	bureau in	cross check	pre val	app score	bureau out	judgement	setup	decision
2	1	5	5	N	Branch	cross check	3	edit	bureau in	cross check	NA	NA	NA	NA	NA	NA
2	2	5	5	N	Branch	maker rework	4	edit	bureau in	cross check	maker rework	NA	NA	NA	NA	NA
2	3	5	4	N	Branch	ca 1st follow	5	edit	bureau in	cross check	Maker rework	ca 1st follow	NA	NA	NA	NA
2	4	5	2	N	Branch	pre val	8	edit	bureau in	cross check	maker rework	ca 1st follow	maker complied	checker complied	pre val	NA

3.5 One hot and Label Encoding

Further encoding is required post index-based encoding for the data to be 'readable' by a machine learning model. Static variables are one-hot encoded while last state and dynamic variables are label encoded due to the high count of variable. Post one-hot and label encoding, there are close to 100 columns for static variables and 600 plus columns for dynamic variables.

3.6 Data Processing

Raw data is sourced from multiple business units that includes Operation, Credit and Risk. The raw data is then extracted, transformed, and loaded in a Hadoop distributed file system as parquet files using Python-Spark. Next, data processing, merging of tables, feature engineering and index-based encoding are done before the processed data is sufficiently small enough to be converted to a Pandas Data Frame. The data is then one-hot and label encoded.

3.7 Train, Test and Production Data Sets

Post processing, the data is split into train, test and production data sets.

Train data set consists of a full year of data and is used to train and cross validate the machine learning model. It follows that applications must be completed so turnaround time is available for training.

Test data set consist of two months of data and is used as an out of sample test set to track the performance of the machine learning model. As such, applications in the test data set are completed as well.

Production data set consist of in progress applications only. Prediction for the turnaround time are made for every update of each application until the application is decisioned. Hence, the performance of the model on these applications can only be tracked after the applications are completed.

3.8 Predicted Remaining Time

Predicted remaining time is derived by subtracting time taken so far (engineered dynamic variable) from the predicted Turnaround time. This is calculated for in progress applications in the production data set to make it possible to proactively intervene on applications that are predicted to take longer than normal.

3.9 Predefined Split for Cross Validation

A consideration that arises from using index-based encoding with machine learning is the possibility of biasness when an update is cross validated against another update from the same application. Thus, a predefined split for cross validation is used to ensure that updates from the same application are assigned to the same cross validation fold. The algorithm is as follows:

1) Randomly assign updates from the same application an integer from 1 to N, where N is the number of cross validation folds
2) Sort updates into N folds for cross validation when training machine learning model

3.10 Sample Weight for Trend of Average Turnaround Time

A critical factor to consider and mitigate is the trend of the average turnaround time. For example, turnaround time may edge higher towards the end of the year due to seasonality or trend lower in recent months due to improvement in the business process. Figure 1 shows the trend of average turnaround time with unit in days.

It follows that heavier sample weights should be given to more recent applications in the training of the machine learning model. The sample weights can be optimized through experiments. The algorithm is as follows:

1) Assign sample weight w1 (default value of 5) to applications from the most recent month.
2) Assign sample weight w2 (default value of 3) to applications from the second most recent month.
3) Assign sample weight w3 (default value of 1) to all other applications
4) Vary w1 and w2 with integer values from 1 to 10 with the condition that w1 >= w2 >= w3 to obtain the best performance for prediction

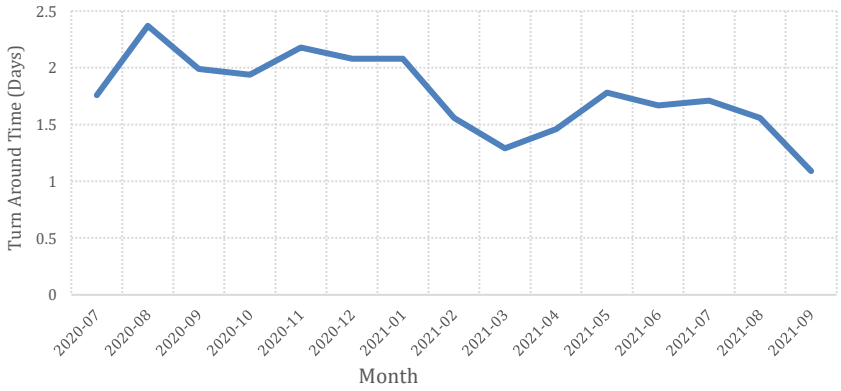

Figure 1: Trend of Average Turnaround time (days) with a variance of about 0.33 days.

3.11 Machine Learning Model

Extreme Gradient Boosting (XGBoost) [6] is used for the machine learning regression model. It is a supervised machine learning technique that is tree-based and open sourced. XGBoost model is trained on the train data set with completed applications. Grid search, with cross validation using the predefined split mentioned in an earlier section, is employed to search for the best parameters. Parameters in the grid search include 'learning rate', 'n

estimators', 'min child weight', 'gamma', 'subsample', 'column sample by tree' and 'max depth'. Sample weights for data that are more recent are then optimized to account for any shift in the trend of turnaround time.
Performance of the train and test data sets are compared to ensure that there is minimal over fitting of the model. The trained XGBoost model is then used to predict turnaround time of in progress applications based on the latest available update. Figure 2 shows the overall flow of the data pipeline. Parameters of the final XGBoost model trained are: learning rate: 0.1, N estimators: 400, Min child rate: 1, Gamma: 0, Subsample: 1, Col sample by tree: 0.9, Max depth: 7.

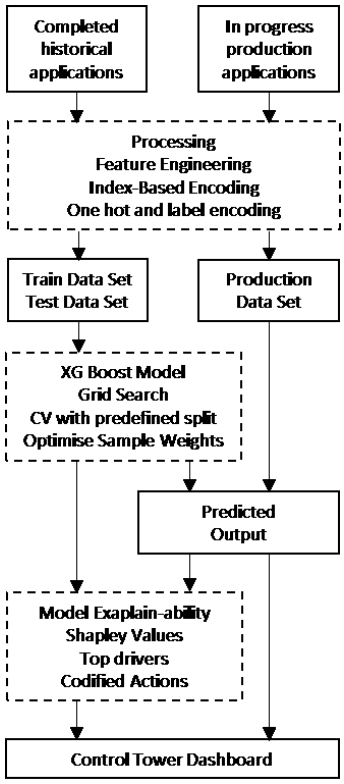

Figure 2: Flow chart of the data pipeline

4 Model Explain-ability

4.1 Feature Importance

The most important features from the XGBoost model is:

1) Step time taken for critical steps
2) Count of discrepancies in application
3) Whether the application is 'clean'
4) Channel from which applications are submitted
5) Step status of applications.

4.2 Shapley Values

Shapley values [7] are used to make sense of the model prediction. It originated from cooperative game theory where it is used for fair distribution for rewards based on contributions made. Here, contribution of each variable to the prediction can be quantified relative to the rest of the variables at the model level, sub population level and application level. Furthermore, a variable can be determined to have made a positive or negative contribution. In contrast, conventional feature importance in machine learning only provides the magnitude of variable importance at the model level.

4.3 Top drivers

At the model level, top drivers or variables for predicting turnaround time include duration of critical early steps, number of discrepancies in applications and whether the applications are clean.

Certain variables that are relatively significant present opportunities to improve the application process. For example, it is found that applications submitted on Fridays take longer to complete as they are only processed on Mondays along with applications from the weekends. Another example is that applications submitted through physical bank branches on paper forms take longer than digital submissions. Both findings correspond with knowledge on the ground and their relative significance provides incentive to finetune the business process.

4.4 Codified Actions

At the sub population and application level, in progress applications that are predicted to take longer than normal to complete can possibly be intervened proactively to potentially lower the turnaround time.

This involves identifying variables that are relatively significant and can be acted upon. Specific follow-up actions are then prescribed, tagged to the variables, and codified into the application process. As an example, suppose long duration at credit step is among the top drivers for the current in progress batch of applications. The owner of the business process can be prompted to assign more manpower and to investigate if there are any other root causes.

5 Results and Discussion

5.1 Performance Metric

Mean absolute error (MAE) is used as the performance metric for the prediction of turnaround time. Note that turnaround time is recorded in minutes, but it is presented in days for brevity.

5.2 Experiment Results

Four sets of experiments were conducted and the MAE for the train and out of sample test data sets are shown in Table 2.

Table 2: Experiment Results - Mean Absolute Error (days)

Experiment	Train	Test	Note
1	0.75	0.97	Best performance
2	1.57	2.37	Without index encoding
3	0.77	1.17	Without last state encoding
4	0.75	1.05	Without sample weight

In experiment 1, all variables, encodings and methods described in section 3 are used. The train and test MAE are 0.75 days and 0.97 days respectively. Model parameters are optimized using grid search and predefined cross validation. Experiment 1 has the best performance.

In experiment 2, index encoding is not used in order to quantify its impact on the model performance. Dynamic variables are dropped, and the model is trained only on static and last state variables. Test MAE is significantly higher at 2.37 days. Hence, index encoding has a significant impact on model performance.

In experiment 3, last state encoding is not used in order to quantify its impact on the model performance. Last state variables are dropped, and the model is trained only on static and dynamic variables. Test MAE is higher at 1.17 days. Thus, last state encoding has an impact on model performance as well.

In experiment 4, sample weight is not used in order to quantify its impact on the model performance. Test MAE is marginally higher at 1.38 days. However, impact of sample weight on model performance will likely depend on the trend of Turnaround time as mentioned earlier in section 3.10.

5.3 Median Turnaround Time

This section looks at the median absolute error of experiment 1, which has the optimal mean absolute error. The median absolute error for the train and test data sets are 0.23 days and 0.35 days respectively, which is significantly lower than the corresponding mean absolute error. This is shown in Table 3.

Table 3: Comparison of Mean and Median Absolute Error for Experiment 1

Experiment 1	Mean Abs Err (days)	Median Abs Err (days)
Train	0.76	0.23
Test	1.05	0.35

Distribution of the absolute error, which has a long tail, is shown in Figure 3. The considerable difference between the mean and median absolute error corresponds to the observation of the long tail. Investigation of applications that falls in the long tail is done and the commonality with the highest frequency is that these applications required communication from the customer at one point in time. It is inferred that the high prediction error can be attributed to the inability of the model to predict with accuracy when a particular customer will reply the bank should the need arise. Given that the majority of the applications do not fall into the long tail, it is more useful to use the median absolute error to gauge and communicate the performance of the machine learning model.

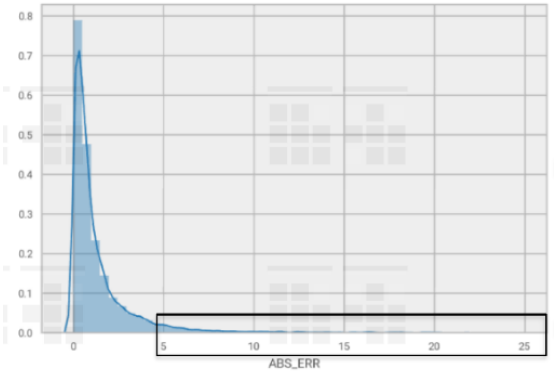

Figure 3: Long tail distribution of Absolute Error

5.4 Validating Index Encoding

A characteristic of index encoding is that a fresh prediction can be made with every new update of the application. It follows naturally that the prediction error of subsequent predictions will be lower as more information is added. This hypothesis is proven in Figure 4, which shows a plot of prediction error against the number of updates with a clear downwards trend.

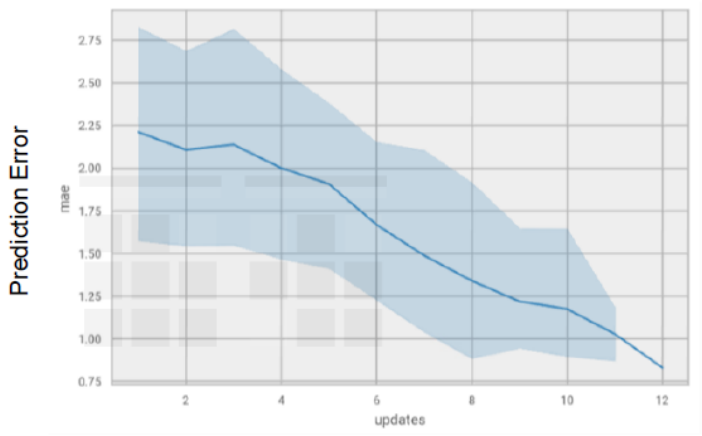

Figure 4: Plot of Prediction Error against number of updates

6 Conclusions and Future Work

This paper shows that index encoding and machine learning can be used to predict the turnaround time of a complex business process. Prediction can be made on in progress processes and refreshed when new information is available. Experiments results proved that the use of index encoding, last state encoding and sample weight improves the predictive power of the machine learning model. For applications that are predicted to have longer than normal turnaround time, model explain ability can be used to identify the top contributing variables. This allows for intervention in the process to potentially reduce turnaround time before completion.

Predicting the next step in a business process is possible [8] and useful. This will enable better allocation of resources in anticipation of any surge in volume for a particular step. Possible future work is the use of recurrent neural networks with Long Short Term Memory (LSTM) [9] architectures for predicting turnaround time and next step. LSTM has been shown to successfully predict next step and turnaround or remaining time for business process with considerable success [10].

Acknowledgement

We would like to thank our sponsors and collaborators from DBS Bank for supporting our work: Sameer Gupta, Gautam Gorki, Varun Gupta, Raju Nair, Hazel Seow, Geraldine Shu Fen Low.

REFERENCES

<bib id="bib1"><number>[1]</number>Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C. 2014. Predictive monitoring of business processes. In: Proc. Of CAiSE. Springer, 457-472.</bib>
<bib id="bib2"><number>[2]</number>Xing, Z., Pei, J., Keogh, E.J.. 2010. A brief survey on sequence classification. SIGKDD Explorations, 12, I. 40-48</bib>
<bib id="bib3"><number>[3]</number>Van der Aalst, W.M.P., Pesic, M., Song, M. 2010. Beyond process mining: From the past to present and future. In: Proc. Of CAiSE. 38-52.</bib>

<bib id="bib4"><number>[4]</number>Anna Leontjeva, Raffaele Conforti, Chiara Di Francescomarino, Marlon Dumas, Fabrizio Maria Maggi. 2015. Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes. In International Conference on Business Process Management. 297.</bib>
<bib id="bib5"><number>[5]</number>Rabiner, L. 1989. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77, 2. 254-286.</bib>
<bib id="bib6"><number>[6]</number>Chen, Tianqi, and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 785-794.</bib>
<bib id="bib7"><number>[7]</number>Scott Lundberg, Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. arXiv: 1705.07874.</bib>
<bib id="bib8"><number>[8]</number>Becker, J., Breuker, D., Delfmann, P., Matzner, M. 2014. Designing and implementing a frame work for event-based predictive modelling of business processes. In: Proceedings of the 6th International Workshop on Enterprise Modelling and Information Systems Architectures. 71-84.</bib>
<bib id="bib9"><number>[9]</number>Hochreiter, S., Schmidhuber, J.. 1997. Long short term memory. Neural Computation, 9, 8. 1735-1780.</bib>
<bib id="bib10"><number>[10]</number>Niek Tax, Ilya Verenich, Marcello La Rosa, Marlon Dumas. 2017. Predictive Business Process Monitoring with LSTM Neural Networks. arXiv:1612.02130v2</bib>