Evaluating the Reliability of Potential Credit Card Holders

Brian Eager and Ryan Meston

Bellevue University

Executive Summary

The purpose of this project is to create a method for evaluating potential client's reliability in making credit card payments based on their payment status history and credit card application. This information will assist in determining how likely a client will be to pay back their credit card if an offer is extended to them. The resulting *reliability measure* must be consistent across three different groups of people: (1) those for whom we have only payment status history but no application, (2) those for whom we have both a payment status history and an application, and (3) those for whom we have an application but no payment status history. This measure can help a credit card company determine how reliably a client is likely to be in making future credit card payments and thus whether to approve the application or extend an offer of credit.

In order for the *reliability measure* to be consistent, we derived it directly from the payment status history when available. To use the same measure to predict the likely reliability of a person who has submitted an application but has no credit history, we trained several predictive models using the data from people for whom we have both an application and a payment history. The best model was a predictive clustering tree, which clusters applications into groups that have similar responses for the application fields such that each group of applications has a different pattern of payment status history. When a new application with no payment status history is received, the model can choose which cluster of applications it most closely resembles based on the responses to the application fields and estimate the likely payment status history for the new application by using the payment status history of the other applications in that cluster. This predictive clustering tree improves accuracy in predicting how frequently a person will be on time and how frequently they will be behind on their payments.

Introduction and Background

A commonly used metric for the value of a potential credit customer is a credit score, which is typically a single numeric value between 300 and 850 from which a categorical variable consisting of levels like “bad”, “fair”, “good”, and “excellent” can be derived by binning the credit score into ranges. This metric is often based on four main factors: (1) demographic information about the applicant, such as where they live, how old they are, their marital status, their employment status, etc.; (2) the applicant’s existing debt, payments, and interest rates, including the purpose of said debt; (3) the applicant’s credit history, particularly its duration and the timeliness of their payments; and (4) the applicant's overall financial situation, including current and historical balances in checking or savings accounts (Fenjiro, 2018). Unfortunately, we do not have access to all this data about the potential credit card holders in our dataset, and thus cannot produce a credit score as descried above.

Data Sources

The dataset which we have been provided—posted to Kaggle by Xiao Song—consists of two separate tables stored in the flat files *application_record.csv* and *credit_record.csv* (Song, 2020). The first table has demographic information from credit card applications, hereafter referred to as the *application table,* with each row representing an application and keyed by individual ID (Song, 2020). The second table has payment status data keyed by individual ID and month (Song, 2020), hereafter referred to as the *payment history table*. It is not the case that each ID is represented in both tables. In fact, there are three distinct groups of individual IDs: (1) 9,528 IDs that appear only in the second table, i.e., for which we have no application but for which we do have payment history data; (2) 36,457 IDs that appear in both tables, i.e., those for which we have both application and payment history data; and (3) 402,053 IDs that appear only

in the first table, i.e., those for which we have an application but for which we have no payment history data.

The Application Table: application_record.csv

The unit of observation in the application table is a single application, with a total of 438,557 applications represented. For each application, the table has 18 variables: (1) a unique identifier for each applicant; (2) whether the applicant is male or female; (3) whether the applicant owns a car; (4) whether the applicant owns any real estate; (5) how many children the applicant has; (6) the applicant's total annual income; (7) the applicant's income type, categorized as *commercial associate*, *pensioner*, *state servant*, *student*, or *working*; (8) the applicant's highest achieved level of education, categorized as *academic degree*, *higher education*, *incomplete higher*, *lower secondary*, or *secondary / secondary special*; (9) the applicant's marital status, categorized as *civil marriage*, *married*, *separated*, *single / not married*, or *widow*; (10) the applicant's housing type, categorized as *co-op apartment*, *house / apartment*, *municipal apartment*, *office apartment*, *rented apartment*, or *with parents*; (11) the applicant's age in days; (12) the number of days the applicant has been either employed or unemployed; (13) whether the applicant provided a mobile phone number; (14) whether the applicant provided a work phone number; (15) whether the applicant provided a phone number; (16) whether the applicant provided an e-mail address; (17) the applicant's occupation type, categorized as *accountants*, *cleaning staff*, *cooking staff*, *core staff*, *drivers*, *HR staff*, *high skill tech staff*, *IT staff*, *laborers*, *low-skill laborers*, *managers*, *medicine staff*, *private security staff*, *realty agents*, *sales staff*, *secretaries*, *security staff*, or *waiters/barmen staff*; and (18) the number of people in the applicant's family. There are 47 IDs that each have two applications associated with them, which are in fact different applications submitted at different times. None of these duplicated

IDs appear in the payment history table.

The Payment History Table: credit_record.csv

The unit of observation for the payment history table is a month of payment status history associated with an individual. There are multiple rows for individuals who have more than one month of payment history. For each row, there are three attributes: (1) a personal identifier for the person, which matches with the identifier in the applications record table when a person is represented in both tables; (2) how many months ago the row's data is from; and (3) the payment status for the given person in that month, categorized as *1-29 days past due, 30-59 days past due, 60-89 days overdue, 90-119 days overdue, 120-149 days overdue, overdue or bad debts* / *write-offs for more than 150 days, paid off that month,* or *no loan for the month*.

Methodology

The methodology of this project can be summarized in three steps. First, we generated a *reliability measure* for each ID directly from its payment status history when possible. This created the *reliability measure* for the IDs for which we have payment history only and for IDs for which we have payment history and an application. Second, we built predictive models using IDs for which we have both an application and a payment status history that estimates the *reliability measure* as its target variable based on the application fields. Third, we can use the best model to estimate the *reliability measure* from the application fields for those IDs for which we have an application but no payment history. These three steps will ensure that we have a consistent measure of payment reliability as long as we have at least a credit card application or a payment status history.

Generate a Reliability Measure from Payment Status History

Two different approaches were taken to generating a *reliability measure* for each ID.

The first was to create a single number that represented reliability, as a credit score does. The second was to use the distribution of payment statuses itself for modeling, allowing for the flexibility to interpret the meaning of the predicted distribution afterwards.

Numeric Reliability Scores: Reliability as a Single Number

To generate a single number that represents an ID's payment reliability, each payment status in the payment history table was encoded as an integer based on how "bad" that score as perceived to be, with "bad" payment scores encoded as negative numbers, "good" statuses as positive numbers, and "neutral" payment statuses as zero according, as shown in the table on the right. The more negative the encoding, the more that status is considered to reflect poor reliability. With each payment status now encoded as an integer that indicates how good or bad it is, we generated a score for each ID based on all the months' payment statuses associated with that ID in the payment history table. We used weighted sums and weighted means with different weighting functions so that payment statuses further in the past affected the score less. We also tried applying min-max regularization to the scores before and after the sum or mean was calculated. The result of all of this was ten different versions of a reliability score as represented by a single number, which we called NORM_SUM, LOG_SUM, NORM_MEAN, LOG_MEAN, MIN_MAX_LOG_SUM, MIN_MAX_LOG_MEAN, MIN_MAX_NORM_SUM, MIN_MAX_NORM_MEAN, MIN_MAX_SUM, and MIN_MAX_MEAN. By weighting each payment status score such that more recent payment statuses are worth more than older payment statuses, we allow for the idea

STATUS	ENCODING
Paid on time	1
No loan for the month	0
1-29 days past due	-1
30-59 days past due	-2
60-89 days overdue	-3
90-119 days overdue	-4
120-149 days overdue	-5
overdue or bad debts write-offs for more than 150 days	-6

that someone can show improvement with regard to their reliability in making payments. The weighted scores calculated using sums allow for those with more payment status history to achieve either more negative and, more positive scores than those with less payment status history, with the idea that our more extensive knowledge about them allows us to make a more certain judgement about their reliability.

Reliability as a Distribution of Payment Statuses

The second technique for generating a reliability score is very straight forward. We used the probability mass function (PMF) of the payment status variable for an ID, which is simply a mapping of each payment status to the percentage of months that the person has that payment status. The idea is that if we know what percentage of months a person pays on time, does not use their credit card, is one month behind on payments, is two months behind on payments, and so forth, then we would have a good indication of how reliable they are in making their payments.

Predictive Models

We trained three different kinds of models, all of which used application fields for their predictor variables and either the numeric scores or payment status PMF as their target variable. In what follows, we will briefly provide three things for each of the models trained: (1) a description of the datasets on which it was trained and tested; (2) a description of the model itself, including what Python library was used to create it; and (3) the results of fitting the model to the dataset.

Linear Regression

Dataset. The dataset used for the linear regression consisted of 36,457 rows from the application table, the IDs of which also exist in the payment history table. Numeric reliability

scores for these IDs were generated directly from the payment statuses associated with these IDs in the payment history table, as described above.

Model Description. Using the formula API of the statsmodels Python library, we fit ordinary least squares linear regression models (*Statsmodels.formula.api.ols—Statsmodels*, n.d.) as descriptive models rather than as predictive models. All application fields were used one at a time as predictors, with the exception of ID. For the target variables, the numeric reliability scores assigned to each ID from the payment history table were used. We thus fit a linear model for each possible predictor variable that targeted each of the ten possible numeric reliability scores. The goal was to find if there was any particular version of the numeric reliability score that was more predictable than the others, and to find which predictor variables explained the most variance of each of the scores.

Results. Of all the application fields used as predictor variables, *occupation type* explained the most variance in all of the versions of the reliability scores. Unfortunately, the r-squared values of the models show that it never explains even one half of one percent of the variance in a reliability score. This is shown in the table on the right. Ordinarily least squares linear regression is not a useful technique for this situation.

Numeric Reliability Score	Percentage of Variance Accounted for by the *occupation type* application field
NORM_SUM	0.163%
LOG_SUM	0.189%
NORM_MEAN	0.298%
LOG_MEAN	0.288%
MIN_MAX_LOG_SUM	0.401%
MIN_MAX_LOG_MEAN	0.304%
MIN_MAX_NORM_SUM	0.433%
MIN_MAX_NORM_MEAN	0.217%
MIN_MAX_SUM	0.146%
MIN_MAX_MEAN	0.246%

Decision Tree Regression

The r-squared values of the descriptive linear regression models showed that no one application field could explain a significant amount the variance in the numeric scores. We

trained a decision tree regressor as a predictive model to see if any combinations of application fields could be used to accurately predict any of the ten versions of our numeric reliability scores.

Dataset. We started with the same dataset as the linear regression models, consisting of 36,457 rows with each row representing an application. Numeric reliability scores for these IDs were generated directly from the payment statuses associated with these IDs in the payment history table, as described above. It was split into a training set of 27,342 rows and a test set of 9,115 rows so that we could fit and evaluate the predictive models.

Description. Using the DecisionTreeRegressor class from the tree's module of the sklearn Python library (*Sklearn.tree. DecisionTreeRegressor—Scikit-learn 0. 23. 2 documentation*, n.d.), we built a separate decision tree regressor model for each version of the numeric score using all the application fields except ID as predictors. Each of them was fit to the training set and then asked to predict the value of the numeric scores in the test set. We then compared the root mean squared error of the model's prediction to the root mean squared error of following the null hypothesis by guessing the mean for that numeric reliability score.

Results. Using the predictions of each of the ten-decision tree regressors resulted in greater error or roughly the same error compare to guessing the mean of its target variable, as shown in the table to the right.

Numeric Reliability Score	Model RMSE	Null Hypothesis RMSE
NORM_SUM	12.02	12.71
LOG_SUM	6.37	6.88
NORM_MEAN	0.53	0.57
LOG_MEAN	0.35	0.38
MIN_MAX_LOG_SUM	4.08	4.06
MIN_MAX_LOG_MEAN	.017	0.19
MIN_MAX_NORM_SUM	8.51	8.50
MIN_MAX_NORM_MEAN	0.17	0.18
MIN_MAX_SUM	5.62	6.01
MIN_MAX_MEAN	5.62	6.00

This indicates that even interactions between application fields are not useful for predicting the numeric reliability scores as they have been calculated. Therefore, we

concluded that we should not use any version of our numeric reliability scores as the target variable for our predictive models.

A Custom Predictive Clustering Tree

Predictive clustering trees are a generalization of decision trees (Stepišnik and Kocev, 2020) that we can use to predict the PMF of payment statuses for a given application, i.e., their payment reliability represented as a distribution. We did not find a Python library that could be used to easily accomplish exactly what we wanted, so we created our own DecisionTree class for this purpose. To predict the PMF of payment statuses associated with a given application, our DecisionTree class used a different arrangement of the data than that used for the linear and decision tree regression models descried above. In what follows, we provide a description of its dataset as well as descriptions of each of the three phases of our predictive clustering tree model: the clustering phase, the prediction phase, and the analysis phase.

Dataset. We merged the application and payment history tables on ID, keeping one row for every month of payment history for IDs that are in both tables. This results in multiple rows per ID if that ID has multiple months of payment history, with each row having all the application fields and the payment status for that month and ID. The resulting dataset has 777,715 rows of application data and monthly payment statuses representing up to 60 months of payment history for 36,457 unique IDs. This was split into a training set of 583,286 rows and a test set of 194,429 rows. We also trained a separate DecisionTree on the records from this dataset that did not have a payment status of zero. The justification for doing so is that it is possible to argue that not using a card at all does not necessarily indicate responsible behavior of the credit card holder because there are many different reasons that someone might not use their credit card.

The Clustering Phase. The goal of this phase of the model is to group records from the training set into clusters such that the application fields of records in a cluster have similar values (analogous to minimizing the intra-cluster distance) whereas the cluster-wise PMFs of the payment status variables vary significantly (analogous to maximizing the inter-cluster distance). In order to accomplish this, we built our DecisionTree class according to Algorithm 19.1 in Zaki & Meira's *Data Mining and Machine Learning* (Zaki & Meira, 2020, p.488), using application fields as the predictor variables. This ensures that records in a given leaf node are similar according to application field values. To evaluate which application field value made for the best split at each decision node, the DecisionTree class seeks to maximize the CART metric with regard to the payment status variable, which "prefers a split point that maximizes the difference between the class probability mass function for the two partitions" (Zaki & Meira, 2020, p.490). When each leaf node is created, the records in that leaf node are put into a pandas DataFrame and appended to a list of leaf nodes, with the prediction for that leaf node being its index in this list (analogous to a cluster label).

There is a balance to be struck in finding enough clusters to capture as much of the continuous nature of the probabilities in the PMF as possible without overfitting the model to training data. In order to prevent the tree from completely overfitting the training data, we set it to create a leaf node whenever the number of records in a node was less than or equal to the number of payment statuses equal to -6 found in the training set. This was admittedly an arbitrary value to choose, but it works as a proof of concept for this model. We did not investigate the tuning of such hyperparameters to find their optimal values.

The Prediction Phase. When the DecisionTree is asked to make its prediction for a given application, it uses the application field values to determine which leaf node cluster that

application belongs in. It then returns the index of its leaf_nodes_ list in which the records for that cluster from the training set are stored as a pandas DataFrame. If provided a sequence of application records, it will perform this process for each in sequence and return a sequence of index values indicating the leaf node cluster in which the corresponding applications are predicted to belong.

The Analysis Phase. The index value returned by the DecisionTree in the prediction phase can then be used to access the records from the training set from that cluster. From here it is simple to compute the PMF of the payment status variable for records in the leaf node cluster and use that as the prediction for the new application that is predicted to belong in that cluster. In this way, the model serves as something similar to a multi-dimensional regressor that estimates the PMF of payment statuses when given the application fields as inputs. By binning the predicted PMF at this point, it is also possible to use the DecisionTree as a classifier.

Results with Zeros. First, we used the DecisionTree as a regressor in the three-phase manner described above. It grouped the training records into 839 leaf node clusters with maximally different PMFs for the payment status variable. It was then asked to predict the PMFs for the records in the testing set. When evaluating the results of the model, we compared both the element-wise mean absolute error (MAE) and the element-wise root mean square error (RMSE) of the model as compared to the null hypothesis of guessing the overall PMF of the training set. As can be seen in the table below, using the model results in an overall reduction of error as compared to the null hypothesis.

Payment Status	-6	-5	-4	-3	-2	-1	0	1
Model MAE	0.003201	0.000530	0.000692	0.001946	0.022934	0.318632	0.240531	0.321158
Null Hypothesis MAE	0.003345	0.000532	0.000691	0.001950	0.023193	0.346195	0.271116	0.366605
Model RMSE	0.027589	0.007930	0.009535	0.016963	0.075694	0.378512	0.328630	0.366272
Null Hypothesis RMSE	0.027637	0.007918	0.009531	0.016984	0.076450	0.399247	0.349916	0.391888

To show how this can also be used as a classifier rather than a regressor, we then binned the predicted PMFs to predict whether someone would have a majority of non-negative payment statuses. We similarly binned the actual PMFs in the test set. When thus used as a binary classifier on the test set records, the model improved upon the null hypothesis of guessing the majority case from the training set every time. The null hypothesis results in an overall accuracy rate of 52.58%, with true positive and false positive rates of 100% and true negative and false negative rates of 0%. The model achieves an overall accuracy rate of 59.27%, with a true positive rate of 83.83%, a true negative rate of 32.05%, a false positive rate of 67.95%, and a false negative rate of 16.17%.

Results without Zeros. When the zeros are dropped from the dataset, all of the other payment statuses grow to a higher percentage in the overall PMF of the payment status variable. Differences in the PMF of the leaf node clusters are thus going to have different proportions of records with negative and non-negative payment statuses. This was not the case when the zeros were included in the dataset because the PMF could be different by a proportional difference of 0s and 1s. The DecisionTree found 703 leaf node clusters in the training set that had no records with payment statuses of zero. It was then used to make predictions on a test set that had no payment statuses of zero. When used as a regressor to predict the PMF of the status variable, this tree also showed an overall reduction of elementwise MAE and elementwise RMSE, as shown in the table below.

Payment Status	-6	-5	-4	-3	-2	-1	0	1
Model MAE	0.004000	0.000702	0.000903	0.002449	0.028201	0.033401	NA	0.338755
Null Hypothesis MAE	0.004271	0.000715	0.000914	0.002463	0.028438	0.379760	NA	0.382791
Model RMSE	0.033842	0.011561	0.012922	0.021262	0.086809	0.391223	NA	0.389607
Null Hypothesis RMSE	0.034170	0.011550	0.012921	0.021275	0.087691	0.419353	NA	0.419362

When used as a binary classifier to predict whether an application would have a majority

of non-negative payment statuses in the same manner as the previous tree, this tree performed much better than the null hypothesis. The null hypothesis resulted in an overall accuracy rate of 38.99%, with true positive and false positive rates of 100% and true negative and false negative rates of 0%. The model achieves an overall accuracy rate of 61.56%, with a true positive rate of 64.94%, a true negative rate of 59.40%, a false positive rate of 40.60%, and a false negative rate of 35.06%.

Discussion and Conclusions

This project serves as a proof of concept for using the probability mass function of the payment status variable as reliability score, and using our custom DecisionTree predictive clustering tree class as the predictive model to apply to IDs that have applications but no payment status history. It is a flexible framework that can easily serve as both a regressor and a classifier. Interestingly, because the model stores the records for each of its leaf nodes, the leaf node cluster label can be added as a feature to the long-term database. This would allow for the predictions of the tree to be updated over time as people continue to either make or not make their payments on time each month. Those new payment statuses could be taken into account when computing a leaf node cluster's PMF when making future predictions.

Acknowledgements

We would like to thank Bellevue University and Dr. Brett Werner for providing the resources and course material to help us complete our project and fulfill the requirements towards completing our master's program. We would like to thank our families for supporting us while we spent time preparing and completing this project to fulfill the academic requirements of this class.

References

Fenjiro, Y. (2018, September 7). *Machine learning for Banking: Loan approval use case.* Medium. https://medium.com/@fenjiro/data-mining-for-banking-loan-approval-use-case-e7c2bc3ece3

Sklearn.tree.Decisiontreeregressor—Scikit-learn 0. 23. 2 documentation. (n.d.). Retrieved November 21, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

Song, X. (2020, March 24). *Credit Card Approval Prediction*. Kaggle. https://www.kaggle.com/rikdifos/credit-card-approval-prediction?select=credit_record.csv

Statsmodels.formula.api.ols—Statsmodels. (n.d.). Retrieved November 21, 2020, from https://www.statsmodels.org/stable/generated/statsmodels.formula.api.ols.html

Stepišnik, T., & Kocev, D. (2020). Oblique predictive clustering trees. *ArXiv:2007.13617 [Cs, Stat]*. http://arxiv.org/abs/2007.13617

Zaki, M.J., & Meira, Jr., W. (2020). *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (Second Edition). Cambridge University Press.