A Prototype of Decision Support System for Credit Risk Performance

Yunjia Ma
yma54@simon.rochester.edu

Yinan Wang
ywang401@simon.rochester.edu

Hang Zou
hzou8@simon.rochester.edu

Jingyi Wu
jwu121@simon.rochester.edu

April 28, 2022

The HELOC dataset and more information about it, including instructions to download, can be found here.
Our interactive interface can be found here.

1 Introduction

Society increasingly relies on credit to make purchases and other financial decisions right now. Credit scores are a financial tool that determines more than just the loans you can get and your pay interest rates. Landlords use them to decide who can rent their apartments. Insurance companies use credit scores to set premiums for car and home insurance. Bank and credit card companies use credit scores to determine who can apply for and repay credit cards. People need good credit to live well and comfortably.

In this project, we utilized the dataset of Home Equity Line of Credit(HELOC) applications to predict whether they will pay back their HELOC account. Our vision is to develop a predictive model to access credit risk and combine this with an interactive interface that the bank/credit card company's sales representatives can use to decide whether to accept or reject an application. To achieve this end, we applied eight predictive models in machine learning to help us in model selection. They are Logistic Regression, Linear SVM, Naive Bayes, LDA, Decision Tree, Random Forest, ADABoost, KNN, Polynomial Kernel SVM, and Radial Basis Function (RBF) Kernel SVM. For our interface, we created a simple platform that allows sales representatives to investigate our previous model and adjust different predictors to easily understand applicants' credit risk.

2 Evaluate Predictive Models

- **Prepare the Data for ML Algorithms**

The dataset has 9871 rows and 24 columns. The first 23 columns (Features) contain applications' credit histories which the company knew before the approval, and the last column (RiskPerformace) holds the outcome of the loan disbursement. This step aims to handle missing values and transform the data matrix to smooth model training and evaluation steps.

The dataset was separated randomly into a train set and a test set with a test size of 0.2, representing 20% of the dataset in the test split. We also used a cross-validation approach and further split the train set for model selection with a test size of 0.25. Thus, the proportion of train to validation to test is 0.6:0.2:0.2, which means we would use 60% of the data to train models, 20% of the data for model selection, and the remaining for final evaluation. Since the value in the last column is labeled as "Bad" and "Good", we converted them to 0 and 1 for easy computation. Three missing values are introduced in the data dictionary, encoded by -7, -8, and -9 in the dataset. Combining with the feature Explanation, we plotted the default risk as a function of the ExternalRiskEstimate and the corresponding counts. The missing value -9 means "no bureau record or no investigation." Therefore, all values -9 were removed where the feature (ExternalRiskEstimate) was missing because it would cause bias in prediction. The other missing values, -7 and -8, seem to break the trend of risk, which behave differently from the rest. Therefore, a pipeline was assigned to extend the train set and test set with binary variables that indicated these two values and replaced these two values with the average in each column. After all, the transformed dataset was ready for further modeling selection. Based on the preprocessed train set, the correlation matrix is created to show correlation coefficients among all features in preparation for later feature selection, as shown below in Figure 1.

- **Model Training and Evaluation/Selection**

The linear models we applied are Logistic Regression, Linear SVM, Bernoulli Naive Bayes, and LDA. The classification models we used are Decision Tree, Random Forest, ADABoost, KNN, Polynomial Kernel SVM, and Radial Basis Function (RBF) Kernel SVM. After performing hyper-parameter tuning to improve model performance, we generated accuracy for all train, validation, and test sets and then chose the best models for further feature engineering.

First, we used the train set to train four linear models and compare them. It is because linear models often work well, and we can easily explain predictions from coefficients. For logistic regression, we set the parameter max_iter to 10000 to ensure algorithm coverage, so the estimation process would reach a stable parameter value in iteration. We used the default settings for the other three models to see performances. To store coefficients and intercepts corresponding to models, we generated a data frame `linear_coefficients` and a new data frame `linear_coefficients_scaled`, which holds the linear coefficients scaled by intercept values to help us understand the differences between models.The following Table 1 includes the accuracy of four models.

Model Name	Train Acc	Vali Acc	Test Acc
Logistic Regression	0.740	0.738	0.738

Linear SVC	0.518	0.528	0.518
Bernoulli Naive Bayes	0.673	0.665	0.672
LDA	0.740	0.737	0.738

Table 1: Accuracy of Linear Models

Next, we switch to working with Tree-Based Models: Decision Tree, Random Forest and ADABoost. Decision trees provide an effective approach to decision-making because they can list issues clearly so that all choices can be challenged, which helps usefully analyze the possible consequences of a decision. We visualized the tree and found the accuracy is close to other models we trained before. Therefore, we applied grid search and generated random values for the hyper-parameters to see any improvement. As a result, it appears that we don't seem to have significantly improved performance, and we may have simply chosen the decision stump. We created an ensemble of trees Random Forest to see further performance rather than a single increasingly complex tree. By tuning the hyper-parameters with the best param we gathered in a grid search, our final performance has significantly increased compared to the decision stump. The other ensemble of trees we used is ADABoosting. We didn't do a grid search at this time because the accuracy has already outperformed the previous models by using the default hyperparameters. The following Table 2 includes the accuracy of these three models.

Model Name	Train Acc	Vali Acc	Test Acc
Decision Tree	0.732	0.697	0.697
Random Forest	0.734	0.736	0.733
ADABoost	0.746	0.740	0.736

Table 2: Accuracy of Tree-Based Models

The other classification models we used in the prediction are KNN, Polynomial Kernel SVM, and RBF Kernel SVM. For KNN model, as one of the most used classification models, its n_neighbor ranging from 1 to 50 are tested for later selection for the optimal. As the plot shown below (Figure 2), the testing accuracy of KNN model starts to converge to around 0.70 since n_neighbor equals 15. Therefore the optimal value for n_neighbor is 15. Polynomial Kernal SVM is a kernel function commonly used in support vector machines (SVMs) and other kernelized models. For Polynomial Kernal SVM, the degree is tested from range 1 to 10, and the result of test accuracy peaks at a degree equal to 5, shown in Figure 3; therefore, 5 is chosen as the optimal. RBF Kernel is similar to KNN. It has the advantages of KNN and also overcomes the space complexity problem, so we simply tried it with the default setting.

Model Name	Train Acc	Vali Acc	Test Acc

KNN	0.726	0.697	0.708
Polynomial Kernel SVM	0.737	0.725	0.729
RBF Kernel SVM	0.735	0.717	0.729

Table 3: Accuracy of Other Classification Models

- **Feature Selection**

Feature selection is used to reduce irrelevant, redundant features or noise. As the correlation graph (Figure 1) shows, we see some features are highly correlated with other features. The feature selection makes a tradeoff between the accuracy and number of features to keep the sparsity principle, lower computational cost, and better model interpretability.After we chose those three models, we used a threshold of 0.2 to filter the average contribution of each feature to the linear function. The feature selection is based on the combination of average contribution and correlation. Moreover, Instead of using training, validation, and test sets, it applied cross-validation to validate the accuracy and checked the overfitting. As the accuracy shows below, about ten features can perform similarly (among 0.01 difference) as the original feature sets.

For Logistic Regression model, 11 features are preserved `ExternalRiskEstimate, AverageMInFile, NumSatisfactoryTrades, PercentTradesNeverDelq, MSinceMostRecentDelq, MaxDelq2PublicRecLast12M, PercentInstallTrades, NumInqLast6M, NetFractionRevolvingBurden, NetFractionInstallBurden, NumRevolvingTradesWBalance`, with the final cross-validated accuracy of 0.727, which is very close to the test accuracy before feature engineering, and that means the abandoned features have little impact to the model.

For LDA model, 11 features are preserved and they are `ExternalRiskEstimate, PercentTradesNeverDelq, AverageMInFile, NumSatisfactoryTrades, NumInqLast6M, NetFractionRevolvingBurden, PercentInstallTrades, NumRevolvingTradesWBalance,MSinceMostRecentDelq, NetFractionInstallBurden, MSinceMostRecentDelq=-7`. The final cross-validated accuracy is 0.726.

For Random Forest model, 10 features are selected: `ExternalRiskEstimate, NetFractionRevolvingBurden, AverageMInFile, NumBank2NatlTradesWHighUtilization, PercentTradesNeverDelq, PercentTradesWBalance, MSinceMostRecentInqexcl7days, MSinceMostRecentDelq, MaxDelq2PublicRecLast12M, MSinceOldestTradeOpen`. The final cross-validated accuracy is 0.72.

- **Best Model Out**

Overall, the model with the best performance is Logistic Regression, with a cross-validated accuracy of 0.727. This trained model is saved for further prediction generation in the interface site. In the Interface, the predicted accuracy is similar to 0.727 as the cross-validated accuracy.

4 Interface Design

We used Streamlit to build an interactive interface that sales representatives in a bank or credit card company can use to decide on accepting or rejecting applications. The users can get the predictive credit risk by simply entering the value of variables through sidebars.

When building the interface, we first create the title of the interface. Then we build the control panel for 11 features. The sidebar is easy for customers who don't have any technical proficiency in adjusting these parameters. After that, we applied the adjusted logistic regression model to the interface.

The interface displays the input value, variables' coefficient, and coefficient*value of all variables, which indicates how those variables affect the risk assessment. Besides, the interface gives the prediction value, model accuracy, and the final assessment below the data frame. Our users can explain the assessment by comparing multiplication values and determining the most influential variable.

5 Summary

Combining all the knowledge from class, the project develops a predictive model and builds a decision support system. The whole project includes preprocessing data for machine learning, training models by applying learning algorithms and validating models to select the best one, selecting features to improve model performance, and applying the adjusted model to build an interactive decision support system. The predictive model that we developed and applied to the interface is a logistic regression model with an accuracy of 0.727. The interactive DSS interface is designed in minimalism, as it can be understood and operated by everyone.

APPENDIX

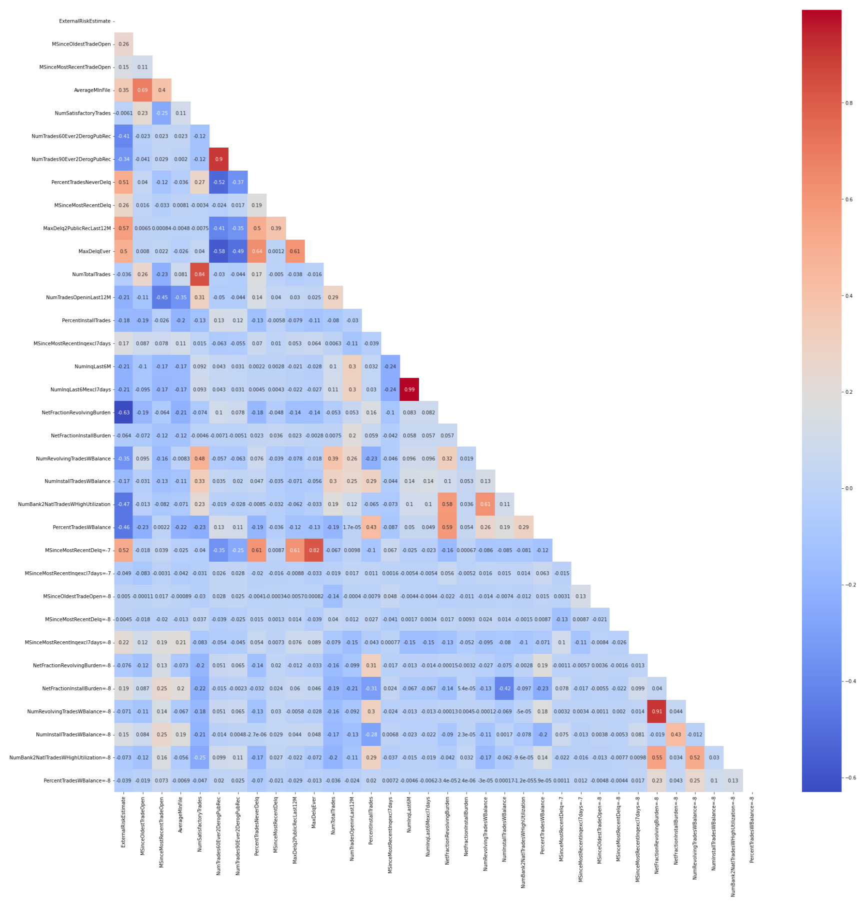


Figure 1: Correlation Matrix

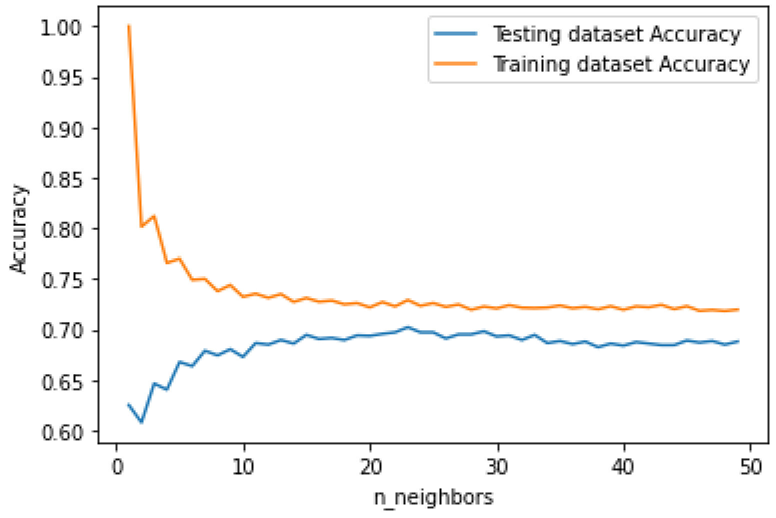


Figure 2: KNN Model

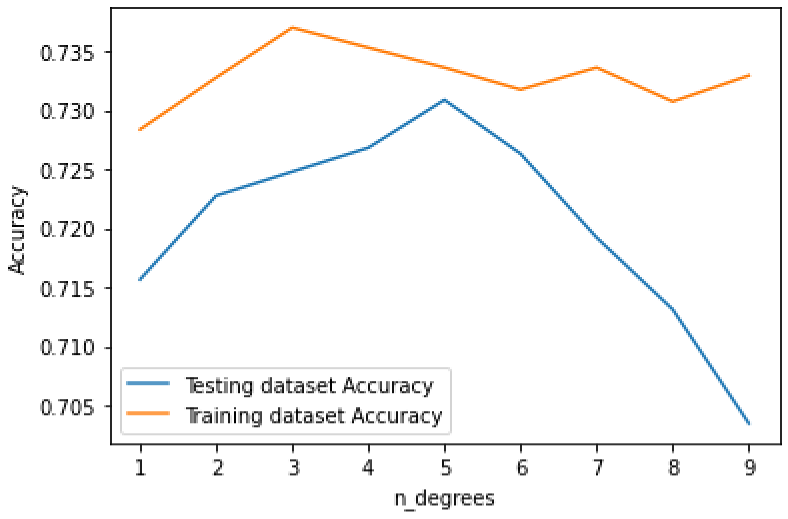
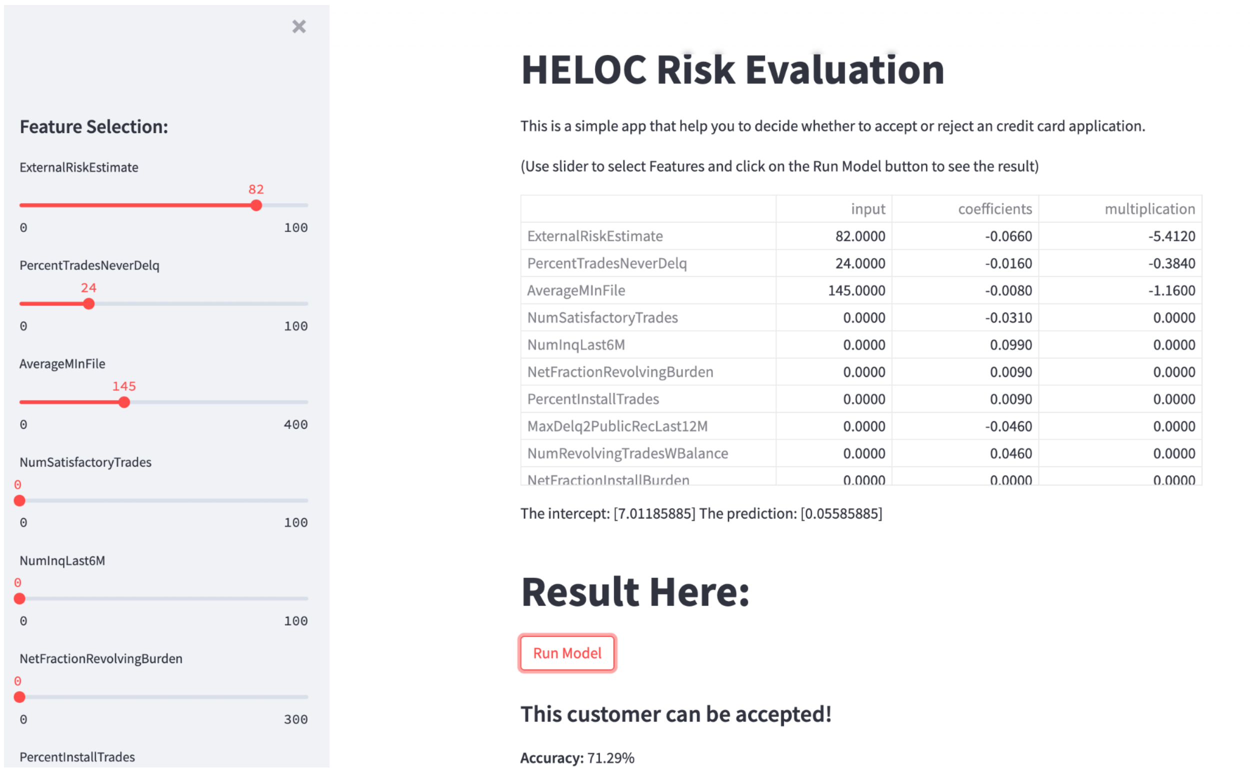


Figure 3: Polynomial Kernal SVM Model



HELOC Risk Evaluation

This is a simple app that help you to decide whether to accept or reject an credit card application.

(Use slider to select Features and click on the Run Model button to see the result)

	input	coefficients	multiplication
ExternalRiskEstimate	82.0000	-0.0660	-5.4120
PercentTradesNeverDelq	24.0000	-0.0160	-0.3840
AverageMInFile	145.0000	-0.0080	-1.1600
NumSatisfactoryTrades	0.0000	-0.0310	0.0000
NumInqLast6M	0.0000	0.0990	0.0000
NetFractionRevolvingBurden	0.0000	0.0090	0.0000
PercentInstallTrades	0.0000	0.0090	0.0000
MaxDelq2PublicRecLast12M	0.0000	-0.0460	0.0000
NumRevolvingTradesWBalance	0.0000	0.0460	0.0000
NetFractionInstallBurden	0.0000	0.0000	0.0000

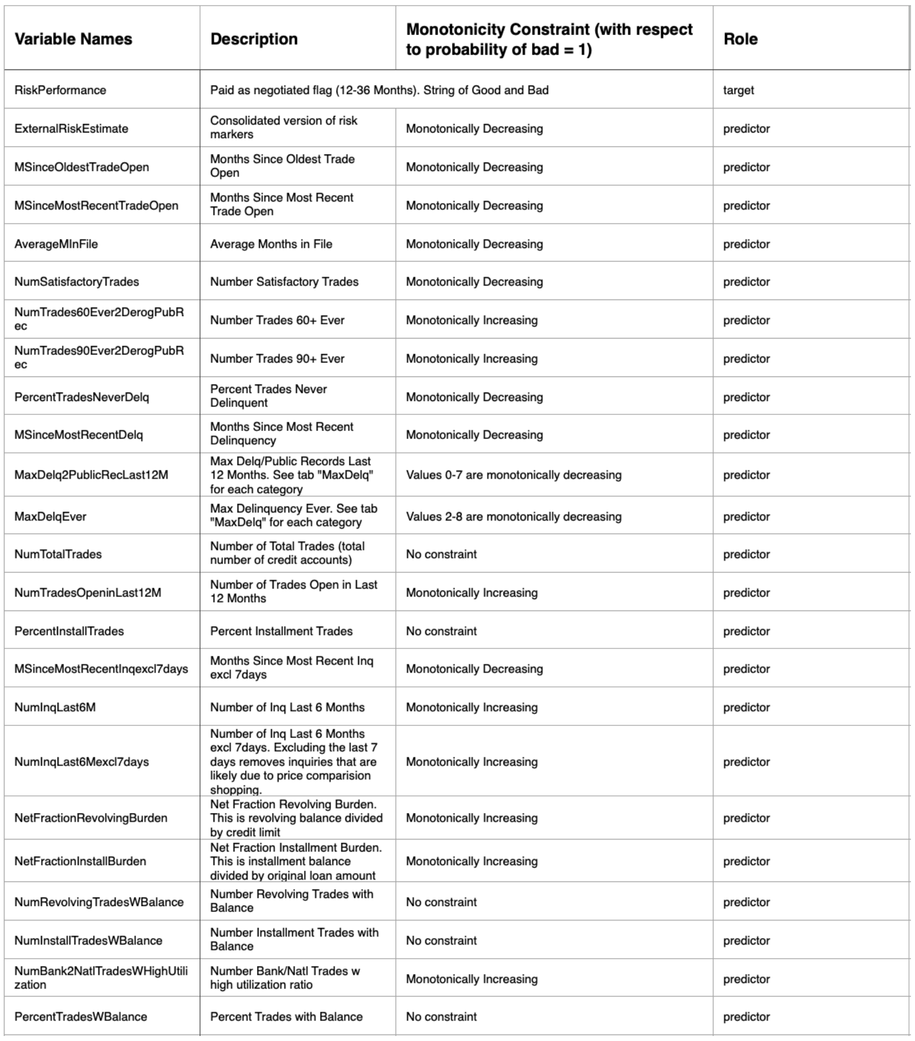
The intercept: [7.01185885] The prediction: [0.05585885]

Result Here:

This customer can be accepted!

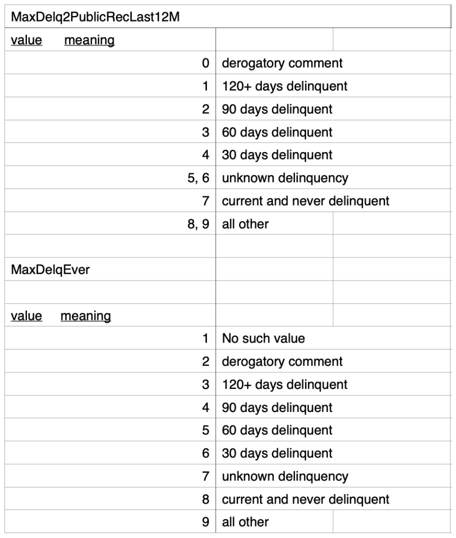
Accuracy: 71.29%

Figure 4: Interface Design



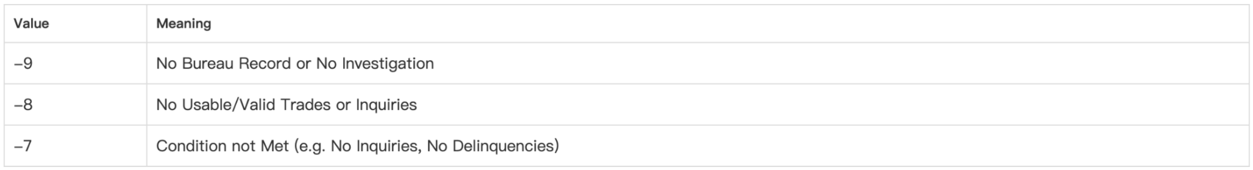
Variable Names	Description	Monotonicity Constraint (with respect to probability of bad = 1)	Role
RiskPerformance	Paid as negotiated flag (12-36 Months). String of Good and Bad		target
ExternalRiskEstimate	Consolidated version of risk markers	Monotonically Decreasing	predictor
MSinceOldestTradeOpen	Months Since Oldest Trade Open	Monotonically Decreasing	predictor
MSinceMostRecentTradeOpen	Months Since Most Recent Trade Open	Monotonically Decreasing	predictor
AverageMInFile	Average Months in File	Monotonically Decreasing	predictor
NumSatisfactoryTrades	Number Satisfactory Trades	Monotonically Decreasing	predictor
NumTrades60Ever2DerogPubRec	Number Trades 60+ Ever	Monotonically Increasing	predictor
NumTrades90Ever2DerogPubRec	Number Trades 90+ Ever	Monotonically Increasing	predictor
PercentTradesNeverDelq	Percent Trades Never Delinquent	Monotonically Decreasing	predictor
MSinceMostRecentDelq	Months Since Most Recent Delinquency	Monotonically Decreasing	predictor
MaxDelq2PublicRecLast12M	Max Delq/Public Records Last 12 Months. See tab "MaxDelq" for each category	Values 0-7 are monotonically decreasing	predictor
MaxDelqEver	Max Delinquency Ever. See tab "MaxDelq" for each category	Values 2-8 are monotonically decreasing	predictor
NumTotalTrades	Number of Total Trades (total number of credit accounts)	No constraint	predictor
NumTradesOpeninLast12M	Number of Trades Open in Last 12 Months	Monotonically Increasing	predictor
PercentInstallTrades	Percent Installment Trades	No constraint	predictor
MSinceMostRecentInqexcl7days	Months Since Most Recent Inq excl 7days	Monotonically Decreasing	predictor
NumInqLast6M	Number of Inq Last 6 Months	Monotonically Increasing	predictor
NumInqLast6Mexcl7days	Number of Inq Last 6 Months excl 7days. Excluding the last 7 days removes inquiries that are likely due to price comparision shopping.	Monotonically Increasing	predictor
NetFractionRevolvingBurden	Net Fraction Revolving Burden. This is revolving balance divided by credit limit	Monotonically Increasing	predictor
NetFractionInstallBurden	Net Fraction Installment Burden. This is installment balance divided by original loan amount	Monotonically Increasing	predictor
NumRevolvingTradesWBalance	Number Revolving Trades with Balance	No constraint	predictor
NumInstallTradesWBalance	Number Installment Trades with Balance	No constraint	predictor
NumBank2NatlTradesWHighUtilization	Number Bank/Natl Trades w high utilization ratio	Monotonically Increasing	predictor
PercentTradesWBalance	Percent Trades with Balance	No constraint	predictor

Figure 5: Feature Explanation



MaxDelq2PublicRecLast12M	
value	meaning
0	derogatory comment
1	120+ days delinquent
2	90 days delinquent
3	60 days delinquent
4	30 days delinquent
5, 6	unknown delinquency
7	current and never delinquent
8, 9	all other
MaxDelqEver	
value	meaning
1	No such value
2	derogatory comment
3	120+ days delinquent
4	90 days delinquent
5	60 days delinquent
6	30 days delinquent
7	unknown delinquency
8	current and never delinquent
9	all other

Figure 6: MaxDelq Table



Value	Meaning
−9	No Bureau Record or No Investigation
−8	No Usable/Valid Trades or Inquiries
−7	Condition not Met (e.g. No Inquiries, No Delinquencies)

Figure 7: Missing Values