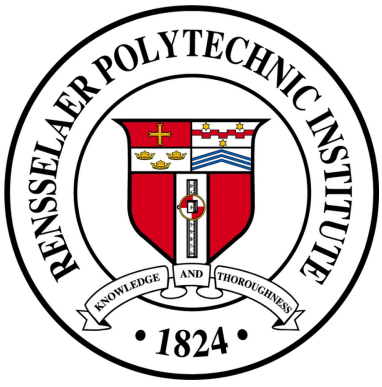
Comparative Analysis of Credit Approval Forecasting Models

Flexible and Inflexible Models

Max Troeger

ECON 4280/6280 - Econometric Methods for Big Data
Dr. Huaming Peng



Department of Economics
Rensselaer Polytechnic Institute
Troy, NY, USA
11 Dec. 2024

Contents

1 Introduction and Literature Review

Financial institutions primarily function as issuers of debt and managers of risk. Without debt, we would not be able to issue capital to projects that are not immediately viable, nor would we be able to produce infrastructure or other public goods. The issuance and management of debt are thus a necessity, and this presents financial institutions with a need to assess credit risk, the accuracy of which — according to Jayanthi (2018) — "is of uttermost importance for lending organizations"[6] who scrutinize credit applicants' demographic and socio-economic profile to minimize losses. According to Boddala and Nandipati[3], lending organizations consider a number of factors when assessing credit risk, including, "credit ratings, historical data, and predefined rules." Given the monetary relevance of forecasting accuracy, lenders employ automated parametric and non-parametric models including, "[logistic] regression, KNN, Decision trees, [and] Random forest..." to make their assessments. Building on the forecasting work of Fu and Liu[4] — including linear models, support vector machines, and tree models — we forecast credit card approvals using a variety of commonly used regression techniques and calculate each model's testing accuracy following Hyndman's[5] comparison of predicted classification and testing data. We then contrast these testing accuracies following Jayanthi's methodology[6]. For greater granularity, we also calculate type-I (false positivity rate) and type-II (false negativity rate) errors. We anticipate that type-I errors are more costly to financial industries than type-II errors because they entail extending credit to applicants with a high likelihood of default. Type-II errors, on the other hand, indicate applicants with low credit risk that were incorrectly denied. Although any error is unfavorable, type-II rejections incur no direct cost to creditors. Since, "modest improvements in scoring accuracy may result in significant savings for financial institutions"[6] we select the model with the greatest testing accuracy and the lowest type-I error.[1]

2 Data

2.1 Description

We obtain the Credit Approval Dataset originally prepared by Chiharu Sano[8] in 1992 from the University of California, Irvine's Machine Learning Repository. The dataset contains 690 observations from individuals in Japan who applied for credit and tracks 16 demographic and socioeconomic features. The class attribute, credit approved, we take as our dependent variable in our forecasting models. It contains the outcome of the credit application, noting '+' if the application was approved or '-' if it was rejected. The other 15 variables contain information like the gender, age, and employment status of the applicant among other demographic

[1] The relevant code is available at the author's website: `https://www.maxbtroeger.com/project/econ-6280.tar.gz`

features.

2.2 Processing

The dataset encodes missing values with a '?' and, for the purpose of omitting them, we encode them as 'NA'. We can confidently omit these observations because they accont only for 37 (5%) of our 690 observations. Before we continue, we assign interpretable labels to variables with obscure titles (e.g., turning variables like `A1` into `Gender`). Subsequently, we transform `Age` and `ZipCode` from strings into numeric values for analysis. We then convert the dataset's binary variables into numeric form; rendering, for example, the 'a' and 'b' of `Gender` as '0' and '1', respectively. We do the same for `PriorDefault`, `Employed`, `DriversLicense`, and `Approved`.

To perform partial least squares (PLS) and principal component regressions (PCR) later in our analysis, we standardize our data. We perform this transformation for `Age`, `Debt`, `YearsEmployed`, `Credit Score`, and `Income`. To expose more of the dataset's features to our analysis, we decompose multipart variables into binary variables. We split `ZipCode` into `ZipCode1`, `ZipCode2`, and `ZipCode3` to reduce the 'neighborhood effect' of demographic features. We also break `Citizen` into the `Citizen_g` and `Citizen_p` dummies; and we produce multi-class dummies for `MaritalStatus`, `BankCustomer`, `EducationLevel`, and `Ethnicity`.

In order to perform rigorous forecasting, we divide the dataset into training and test data. This permits us to fit our model according to the training data and assess its performance on the remaining, out-of-sample testing data. We use a measure of accuracy on the test data as our performance metric. In this paper we apportion 60% of our dataset for training and the remaining 40% for testing. Employing this divide ensures we do not overfit our models.

2.3 Summary

The features of the credit approval dataset are summarized in Table 1.

Variable	μ	σ	**Min**	**Max**
Gender	0.311	0.463	0	1
Age	0	1	-1.499	3.822
Debt	0	1	-0.961	4.609
YearsEmployed	0	1	-0.665	7.788
PriorDefault	0.534	0.499	0	1
Employed	0.439	0.497	0	1
CreditScore	0	1	-0.504	12.981
DriversLicense	0.462	0.499	0	1
Income	0	1	-0.193	18.842
Approved	0.453	0.498	0	1
ZipCode1	0.251	0.434	0	1
ZipCode2	0.296	0.457	0	1
ZipCode3	0.207	0.405	0	1
Citizen_g	0.916	0.278	0	1
Citizen_p	0.003	0.055	0	1
MaritalStatus_u	0.764	0.425	0	1
MaritalStatus_y	0.233	0.423	0	1
MaritalStatus_l	0.003	0.055	0	1
BankCustomer_g	0.764	0.425	0	1
BankCustomer_p	0.233	0.423	0	1
EducationLevel_c	0.204	0.403	0	1
EducationLevel_d	0.040	0.196	0	1
EducationLevel_cc	0.061	0.240	0	1
EducationLevel_i	0.084	0.278	0	1
EducationLevel_j	0.015	0.123	0	1
EducationLevel_k	0.074	0.261	0	1
EducationLevel_m	0.058	0.234	0	1
EducationLevel_r	0.005	0.068	0	1
EducationLevel_q	0.115	0.319	0	1
EducationLevel_w	0.096	0.295	0	1
EducationLevel_x	0.055	0.228	0	1
EducationLevel_e	0.037	0.188	0	1
EducationLevel_aa	0.080	0.271	0	1
Ethnicity_v	0.583	0.493	0	1
Ethnicity_h	0.210	0.407	0	1
Ethnicity_bb	0.081	0.273	0	1
Ethnicity_j	0.012	0.110	0	1
Ethnicity_n	0.006	0.078	0	1
Ethnicity_z	0.012	0.110	0	1
Ethnicity_dd	0.009	0.095	0	1
Ethnicity_ff	0.083	0.276	0	1

Table 1: Credit Approval Dataset Summary

3 Results

3.1 Linear Probability (Backward Stepwise Selection)

We begin with a simple linear probability model because inflexible models offer easy interpretability. We first run ordinary least squares (OLS) on an unrestricted model and then employ backward stepwise variable selection to remove ancillary features so as to augment testing accuracy, giving the following formula:

$$\begin{aligned}\widehat{P(Approved)} = 0.984 &+ 0.033 \cdot Age + 0.581 \cdot PriorDefault + 0.119 \cdot Employed + 0.037 \cdot CreditScore \\ &- 0.057 \cdot DriversLicense + 0.129 \cdot Income + 0.064 \cdot ZipCode1 - 2.312 \cdot Citizen\_p \\ &- 1.058 \cdot MaritalStatus\_u - 1.097 \cdot MaritalStatus\_y + 0.080 \cdot EducationLevel\_c \\ &+ 0.166 \cdot EducationLevel\_d + 0.166 \cdot EducationLevel\_d + 0.154 \cdot EducationLevel\_cc \\ &+ 0.112 \cdot EducationLevel\_i + 0.186 \cdot EducationLevel\_j + 0.145 \cdot EducationLevel\_q \\ &+ 0.157 \cdot EducationLevel\_w + 0.216 \cdot EducationLevel\_x + 0.209 \cdot EducationLevel\_e \\ &+ 0.084 \cdot Ethnicity\_v + 0.113 \cdot Ethnicity\_h + 0.391 \cdot Ethnicity\_n\end{aligned}$$

where $\bar{R}^2 = 0.61$ and $F = 28.92$. To calculate the testing accuracy of the linear probability model, we create the following confusion matrix:

		Test Approved		
		0	**1**	**total**
Predicted Approved	**0**	127	10	TN
	1	18	107	TP
	total	PN	PP	

Which renders a testing accuracy of

$$100\% \cdot \frac{107 + 127}{262} \approx 89.31\%$$

If we were to follow the same procedure, but instead use forward stepwise selection, we would find a lower testing accuracy rate of approximately 88.55%. For the remainder of this paper, we report results from

backward stepwise selection.

From our confusion matrix we also identify that we have type-I and type-II errors of

$$\text{Type-I} = 100\% \cdot \frac{18}{125} = 14.4\% > \text{Type-II} = 100\% \cdot \frac{10}{137} = 7.3\%,$$

respectively. That the type-I error is larger than the type-II error is a weaknesses of the linear probability model.

3.2 Logistic Probability (Backward Stepwise Selection)

Logistic regression improves on the linear probability model by restricting the values the dependent variable can take; moreover, logistic regression does not permit probabilities outside the range $P \in [0, 1]$. As in the linear probability model, we run backward stepwise regression on an unrestricted formula including all features. In so doing, we arrive at a testing accuracy of 88.93%, a slight decrease from the linear probability model, but importantly our type-I error has decreased to 11.4%. Nevertheless, our type-II error has increased to 10.8%, but the gap between the two errors is closer than in the linear model. The favorable trade off between lowered type-I error and lowered testing accuracy is a strength of the logistic probability model over its linear counterpart.

3.3 Linear and Quadratic Discriminant Analysis (LDA & QDA)

When classes in the data are well separated, as we would expect for a credit approval system, the logistic regression model breaks down. We favor linear discriminant analysis (LDA) under this assumption[6]. Fitting an unrestricted model with LDA gives a testing accuracy of 88.5%, a type-I error of 15.2%, and a type-II error of 8%. Although our performance is roughly equivalent to the logistic regression model, the increase in type-I error is undesirable.

Quadratic discriminant analysis (QDA) improves on LDA by disposing of the assumption of shared parameter variance $\hat{\sigma}^2$, allowing it to vary by index: $\hat{\sigma}_k^2$. This improvement, however, precipitously reduces testing accuracy to 74.4%. Interestingly, the type-I error given by QDA is one of the lowest observed in any of our forecasts, sitting at 6.9%. That said, QDA has the highest type-II error observed: 30.9%. The LDA and QDA forecasts for this dataset do not demonstrate a sufficient improvement over the linear and logistic probability models.

3.4 Neural Network

According to Adya and Collopy[1], effectively implemented neural networks, "show potential for forecasting and prediction." Validating a neural network, "should be based on ex ante (out-of-sample) performance" which is exactly how we are evaluating our models: by splitting our training and testing data.

For an underlying formula, we input the formula generated by backward stepwise selection on the linear probability model. In R's `neuralnet` function, provided by the package of the same name, we specify a threshold of 0.5 (where the default is 0.01) which forces the model to stop if the partial derivative error does not change by at least 0.5. This permits a faster calculation and reduces the potential for overfitting.

We arrive at the following model:

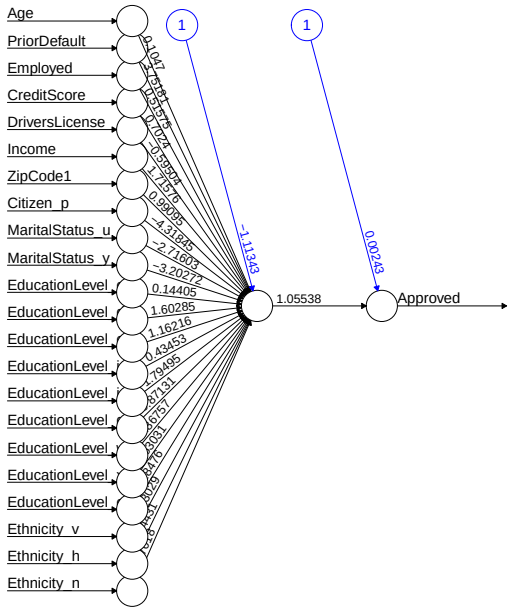


Figure 1: A Visualization of Our Credit Approval Neural Network Forecast

Which gives 90.46% testing accuracy! Our type-I and II errors are comparable to our other forecasting models at 10.3% and 8.9%, respectively. That type-I error is greater than type-II is alarming, but it is lower than the logistic probability model and certainly lower than the linear probability model. Neural networks, however, suffer from a difficulty of interpretation — aptly being called *blackbox* models — and so the increase in testing accuracy is traded for interpretability.

3.5 Polynomial Regression

As an extension of the linear probability model, we augment its unrestricted form by adding each variable's squared term and again run backward stepwise selection. Despite the added flexibility afforded by the

quadratic terms, we observe a decrease in overall testing accuracy: 87.8%. Type-I error jumps to 17.6% and type-II error sinks to 6.9%.

3.6 Spline Regression

Given the limitations of the linear probability model and the relatively poor performance of the polynomial regression, Perperoglou et al. recommend using a spline where, "instead of fitting a global polynomial, partitioning the range... into smaller intervals, utilizing an arbitrary number and position of points... called *knots*." This generates a piecewise continuous function of polynomials that can, in principle, better fit the given data than a global polynomial.[2]

By trial and error we select a model of the form

$$PriorDefault + Income + YearsEmployed + Employed + CreditScore$$

with knots at 4, 7, and 11. Notwithstanding the benefits discussed above, applying a spline on the discovered formula generates a worse performing forceast than the global polynomial, with testing accuracy falling to 82.8%. Compared to the polynomial forecast, the type-I error is slightly lower at 13.3% and the decrease in testing accuracy aggregates to type-II error, which increases to 19.5%.

3.7 Bagging and Boosting

Bagging, or Bootstrap Aggregating, and Boosting are homogeneous weak learners' models that take multiple decision trees and reduce their overall error. This improvement notwithstanding, bagging and boosting perform relatively poorly, with respective testing accuracies of 86.6% and 85.1%; type-I errors of 15.3% and 10.2%; and type-II errors of 11.8% and 17.7%.

3.8 Random Forest

Random forests, an extension of the bagging method, are weighted collections of randomly generated decision trees used for classification. This allows a better forecast because, at the aggregate level, it reduces the individual bias and overfitting of using a single decision tree.[7] We specify an unrestricted model as before and allow the algorithm to produce the best fit, giving us the following random forest:

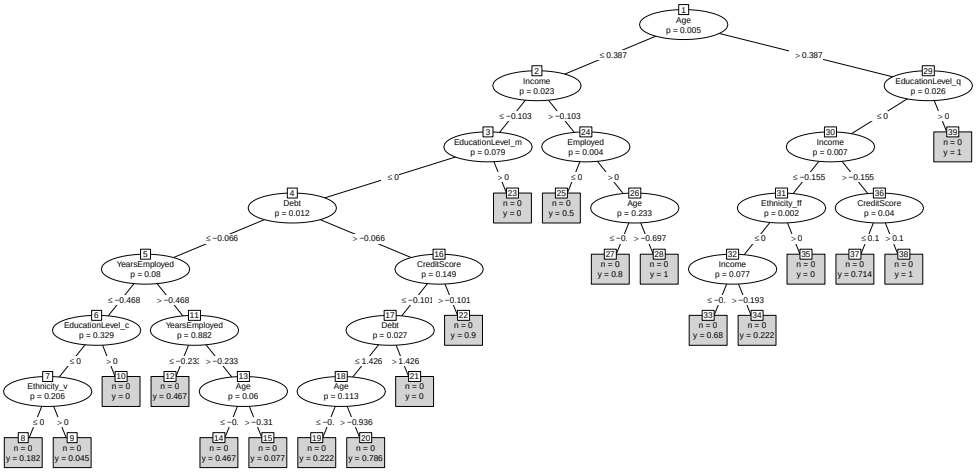


Figure 2: A Visualization of Our Random Forest Forecast

This forecasting model performs better than the spline and about the same as the polynomial forecast, with a testing accuracy of 87.8%. The random forest forecast, however, has a type-I error of 13.0% and type-II error of 11.6%. Because we care more about type-I error, this model performs with greater accuracy ($13.0\% < 17.6\%$) than the polynomial forecast.

3.9 Support Vector Machine (SVM)

SVM works by identifying the non-linear $M > p$ dimensional support-vector classifier that divides our two classes: `Approved=1` and `Approved=0`. In principle, when classes are separable SVM gives greater forecasting accuracy than a logistic regression. If not, the two will deliver similar results. We find that SVM has a testing accuracy of 88.2%, a type-I error of 14.8%, and a type-II error of 9.3%; in short, SVM performs with less overall accuracy than a logistic regression for our credit approval dataset.

3.10 Principal Component Regression (PCR)

A PCR works by separating a dataset into linear combinations of its predictors, replacing many correlated variables with a smaller set of components that capture their joint variation. This requires we standardize our dataset as discussed previously. As with the spline regression, by trial and error we select a model of the form

$$PriorDefault + Income + YearsEmployed + Employed + CreditScore$$

which also permits performance comparison between PCR and the other forecasts. PCR delivers a testing accuracy of 87.8%, a type-I error of 18.0%, and a type-II error of 6.2%; a performance comparable to the

polynomial model.

3.11 Partial Least Squares (PLS)

Although PCR aids computation by identifying linear combinations of dataset features, it does so in a manner that does not guarantee that the best linear combinations to fit the data are the best for forecasting accuracy. To improve forecasting accuracy, PLS reduces dimensionality in a manner that checks if the linear combinations are related to the response variable: it weights linear combinations according to their correlation with the response variable. These weights permit us to specify an unrestricted model, unlike in PCR, which delivers a testing accuracy of 88.9%, a type-I error of 13.3%, and a type-II error of 9.2%. Although this is a clear improvement over PCR, it is slightly less accurate than the linear probability model we started our discussion with. Nevertheless, PLS presents a slightly lower type-I error than OLS.

3.12 $\mathbf{K^{th}}$-Nearest Neighbor (KNN)

KNN aims to find the dividing line between two classes; and the smaller K is, the more flexible the model becomes, asymptotically approaching the Bayesian classifier. By convention, K is selected in the range of 5–10. We specify an unrestricted model to capture as much variance as possible and set a distance of $K = 5$ by trial and error. In so doing we arrive at a testing accuracy of 93.1%, a type-I error of 3.7%, and a type-II error of 9.0%.

3.13 Ridge Regression

Whereas a linear regression finds the parameters $\beta_0, \beta_1, \cdots, \beta_p$ such that RSS is minimized, *ridge regression* finds the set of β^R that minimize

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

where the tuning parameter $\lambda \geq 0$ controls the effect of the shrinkage penalty term. We use backward stepwise selection to drop unimportant features from our regression and use this smaller set for our ridge regression analysis. We employ cross validation to determine the optimal $\lambda = 0.03$, which delivers a testing accuracy of 89.3%, a type-I error of 14.4%, and a type-II error of 7.3%. This is *identical* to our linear probability model calculated above.

3.14 Lasso Regression

As an improvement to ridge regression, lasso drops features instead of including all those given to it. So, we feed an unrestricted model to the algorithm. Lasso determines this sparser subset by finding only the β_λ^L that minimize

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

and we again employ cross validation to select the optimal $\lambda = 0.04$, which delivers a testing accuracy of 87.8%, a type-I error of 18.1%, and a type-II error of 6.2%. Thus, we find better model performance by manually removing features in a ridge regression, than by algorithmically selecting them through lasso regression.

4 Conclusion

Table 2 summarizes the results of our forecast comparisons. Entries are first sorted in descending order by forecasting accuracy and, if the accuracy matches, in ascending order by type-I error.

Model	**Testing Accuracy**	**Type-I Error**	**Type-II Error**
KNN	93.1%	3.7%	9.0%
Neural Network	90.5%	10.3%	8.9%
Linear Probability	89.3%	14.4%	7.3%
Ridge	89.3%	14.4%	7.3%
Logistic Probability	88.9%	11.4%	10.8%
PLS	88.9%	13.3%	9.2%
LDA	88.5%	15.2%	8.0%
SVM	88.2%	14.8%	9.3%
Random Forest	87.8%	13.0%	11.6%
Polynomial	87.8%	17.6%	6.9%
PCR	87.8%	18.0%	6.2%
Lasso	87.8%	18.1%	6.2%
Bagging	86.6%	15.3%	11.8%
Boosting	85.1%	10.2%	17.7%
Spline	82.8%	13.3%	19.5%
QDA	74.4%	6.9%	30.9%

Table 2: Tabulated Testing Accuracies and Types I and II Error

Employing different forecasting techniques allows for a comprehensive capturing of the variance of provided data. Looking beyond testing accuracy and considering type-I and type-II errors allows creditors to discern the best overall model that accounts for non-zero misclassification costs. K^{th}-nearest neighbor ($K = 5$) gave the best testing accuracy and the lowest type-I error observed, making it the best model presented for forecasting credit card approvals. Neural networks, although difficult to interpret, show promise in forecasting. Nevertheless, neural networks provide only a marginal improvement in testing accuracy over linear probability models at the expense of computation time. KNN gives a higher accuracy than both and is more efficient than the presented neural network.

References

[1] Monica Adya and Fred Collopy. "How effective are neural networks at forecasting and prediction? A review and evaluation". In: *Journal of Forecasting* 17.5-6 (1998), pp. 481–495. DOI: https://doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6<481::AID-FOR709>3.0.CO;2-Q.

[2] A. Perperoglou et al. "A review of spline function procedures in R". In: *BMC Medical Research Methodology* 19.46 (2019).

[3] L. V. Boddala and V. S. Nandipati. "Credit Card Approval Prediction: A comparative analysis between Logistic Regression, KNN, Decision Trees, Random Forest, XGBoost". Bachelor's Thesis. Blekinge Institute of Technology, 2024.

[4] Z. Fu and Z. Liu. *Classifier Comparisons On Credit Approval Prediction*. URL: https://cs229.stanford.edu/proj2014/Zhoutong%20Fu,%20Zhedi%20Liu,%20Classifier%20Comparisons%20On%20Credit%20Approval%20Prediction.pdf.

[5] R. Hyndman. "Measuring forecast accuracy". In: *Business forecasting: Practical problems and solutions* (2014), pp. 177–183.

[6] M. D. Jayanthi. "Credit Approval Data Analysis Using Classification and Regression Models". In: *International Journal of Research and Analytical Reviews* 5.3 (2018).

[7] Karina Kervin. *Using random forest to predict credit defaults using Python*. URL: https://developer.ibm.com/tutorials/awb-random-forest-predict-credit-defaults/.

[8] C. Sano. *Credit Approval*. 1992. URL: https://archive.ics.uci.edu/dataset/27/credit+approval.