Credit Card Approval Prediction Model

Petya Buchkova, Bogdan Iulian, Lei Xian, Tiberiu Mitran

May 19, 2019

Abstract. This paper describes a data analysis of a credit card approval dataset, together with a prediction model 1 using Random Forest and Logistic Regression. The features of the dataset are described, analyzed, and their predictive strength is evaluated.

1 Introduction

In this paper, we attempt to develop a model to predict the approval of credit card applications and examine the importance/correlation of given features on the application outcome. The Credit Card Approval dataset[4], used for analysis, is from the UCI Machine Learning Repository and it contains anonymized data of customers of a Japanese bank . In order to create a reliable model we conducted an analysis of the aforementioned dataset using well established techniques along with a comparison of different machine learning algorithms

2 Research Question

Based on the UCI credit card approval dataset, what model has the highest predictive power to forecast credit card applications approval? Which attribute(s) have the highest influence?

3 Method

In the following section, we describe the methods used to help us answer the research question

3.1 Data Preprocessing[6]

For achieving better results from the machine learning algorithms, we used this technique to clean the dataset from inconsistent and incomplete values thus creating robust data. Subsequent steps such as **data cleaning** and **sampling** were followed.

3.2 Exploratory Data Analysis[11]

Using this strategy we made use of various visualization techniques, to analyze the data in order to discover what the data can reveal about the research topic and gain insights on how the data is constructed and spread across the dataset.

3.3 Model Evaluation & Selection[12]

Using different machine learning techniques like grid search[5] and cross-validation [13] to select and fine tune the model.

4 Dataset

The Credit Approval Dataset is a collection of credit card applications and approval decisions. Taken from UCI's Machine learning repository[4], it is interesting due to the good mix of attributes. Moreover, it offers the challenge of working with anonymized data. To make it easier to work with the dataset and inspired by previous work[8] with it, we gave the variables working names based on the type of data. The dataset has 15 attributes, most of them (95%) holding 690 observations. As the topic of this paper is to predict whether a credit card application would be approved, we take approval status as the dependant variable.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

Table 1: Credit Card Approval Dataset

5 Analysis

5.1 Dependant variable

The dependant variable is approval status, represented by the non-numerical values "+" for approved and "-" for not approved. The dataset contains more instances that correspond to "Denied" status than instances corresponding to "Approved" status. Specifically, out of 690 instances, there are 383 (55.5%) applications that got denied and 307 (44.5%) applications that got approved.

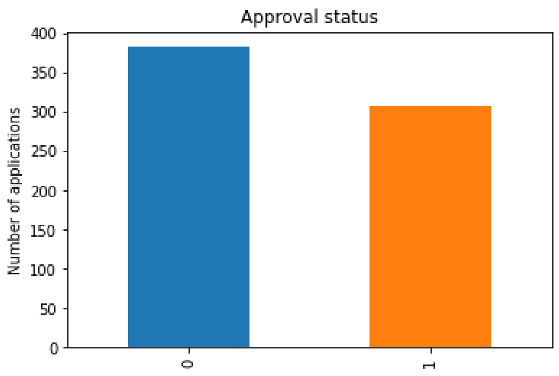

Figure 1: Approval Status

5.2 Correlation

Looking at the Pearson's correlation[2] among the prediction vectors and the target vector, we can see from Table 2 what independent variables have strong correlation with the dependant variable.

The attributes that have the highest correlation to the target vector is *PriorDefault.Employed*, *CreditScore* and *YearsEmployed* also have high correlations with the target vector. We also can see, *Male*, *Ethnicity*, *DriversLicense* and *ZipCode* don't have much influence, which make sense.

	ApprovalStatus
Married	-0.191431
BankCustomer	-0.187520
Citizen	-0.100867
ZipCode	-0.094851
Gender	-0.028934
Ethnicity	0.000877
DriversLicense	0.031625
EducationLevel	0.130026
Age	0.133304
Income	0.175657
Debt	0.206294
YearsEmployed	0.322475
CreditScore	0.406410
Employed	0.458301
PriorDefault	0.720407
ApprovalStatus	1.000000

Table 2: Correlation with the dependent variable

Figure 2 is a heatmap that demonstrates the correlation amongst all attributes and the dependant variable, which give a further understanding of the correlations of all attributes. For example, we can see that *Employed* has a relevant high relation with *EducationLevel.* The understanding of the correlation helps us with deciding what attributes should we use to building a predictive model.

6 Credit Card Approval Prediction

6.1 Data preprocessing

6.1.1 Missing values

As previously mentioned, 5% of our data consists of missing values. Ignoring them can affect

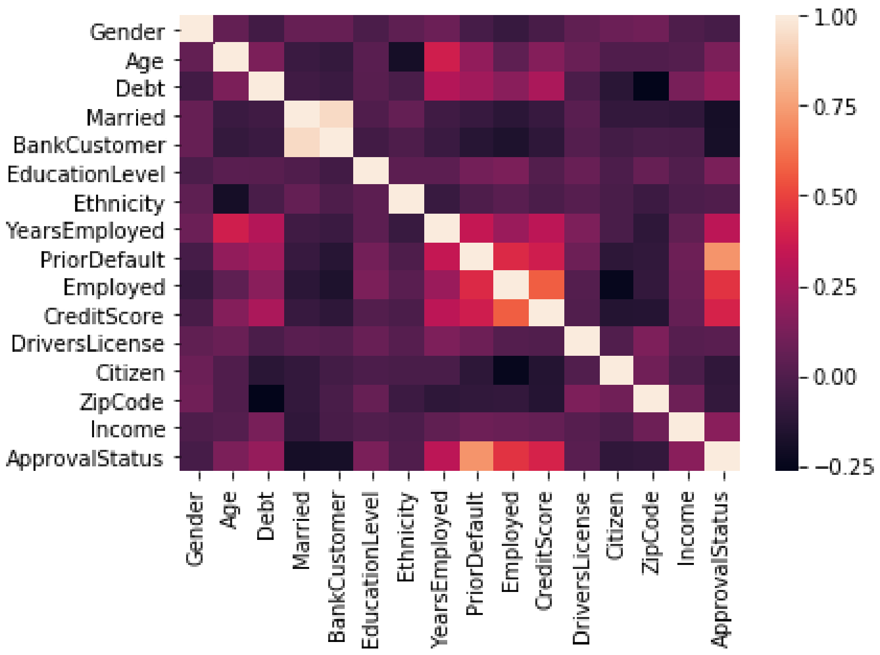

Figure 2: Correlation Heatmap

the performance of our machine learning model heavily. To tackle this problem, we used **mean imputation**[7] to take care of the missing values in the numeric columns. However, we need another strategy for the categorical data and we used the **most common feature value**[7].

6.1.2 Data transformation

Some machine learning models perform better or require data in strictly numeric format. Therefore, to solve this issues and achieve faster computation, we converted all categorical values into numeric ones using label encoding[14].

6.1.3 Data Splitting

The training set is used to build up our model which sees and learns from this data. Validation allows us to fine tune the model parameters on the unknown instances and one of the most useful outcomes of it is the ability to check for overfitting. Adding a validation step gives us more chances to improve the performance. After the model is completely trained, we use a test dataset to provide an unbiased evaluation of a final model fit on the training dataset.

6.1.4 Feature Scaling

By plotting histograms on the non-nuisance features we observed that the data does not have a normal distribution. Also, some of the attributes have data that varies in a big range, to handle these outliers , we use MinMaxScaler to rescale our data so they are ranging from 0 to 1, this should give better performance to our model.

6.2 Model selection

For this project we looked into different classifiers, more precisely Naïve Bayes[10], Random Forest[9] and Logistic Regression[1]. We found that Naïve Bayes, used mostly for probability such as spam filters for example, is performing worse than the other two. Logistic Regression was one of the models selected due to its performance with binary classification. Moreover, the features of the datasets are not conditionally independent, so Naive Bayes would have a higher bias but lower variance compared to logistic regression[3]. After an initial evaluation of the most common performance metrics(Table 3), we decided to go along with Logistic Regression and Random Forest and take them one step further by applying parameter tuning on both of them.

	Logistic Regression	Random Forest	Naive Bayes
AUC	0.918	0.910	0.876
F1	0.873	0.860	0.827
Precission	0.910	0.884	0.752
Recall	0.845	0.841	0.920
Accuracy	0.780	0.846	0.780

Table 3: Initial models results on cross validation

Using GridSearch we found the best parameter combination and we used them on the validation set in order to see the metrics. After running the tuned models on the test, the results were slightly better in favor of Logistic Regression, as shown in Table 4.

Based on Sokolova et al. research on classification algorithms[reference to Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation], we know we cannot base our decision solely on the commonly accepted metrics such as Accuracy, F-score and ROC, therefore we chose to compute the Youden's index in order to find the optimal threshold.

	Logistic Regression	Random Forest
AUC	0.943	0.901
F1	0.903	0.902
Precission	0.934	0.942
Recall	0.879	0.866
Accuracy	0.630	0.898

Table 4: Tuned models results

7 Findings

From on our research on UCI's Credit Card Approval dataset, we were able to find the most correlated values that influence approval - *PriorDefault*, *Employed*, *CreditScore* and *YearsEmployed*.We tested and tuned Logistic Regression and Random Forest, and achieved 63% and 89% accuracy. We can therefore conclude that the model with the highest predictive power to forecast credit card applications approval is Random Forest.

References

[1] George Ebow Bonney. "Logistic regression for dependent binary observations". In: *Biometrics* (1987), pp. 951–973.

[2] UWE Bristol. "Pearson's Correlation Coefficient". In: *URL: http://learntech.uwe.ac.uk/da/Default.aspx?pageid=1442* ().

[3] Sanghamitra Deb. "Naive Bayes vs Logistic Regression". In: *URL:* $https://medium.com/@sangha_deb/naive-bayes-vs-logistic-regression-a319b07a5d4c$ ().

[4] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository.* 2017. URL: `http://archive.ics.uci.edu/ml`.

[5] Rohan Joseph. "Grid Search for model tuning". In: *URL: https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e?fbclid=IwAR3ay2XYANtZWSKhlSK-UqBs1AObP5CKf-glDFFwMqWNU7XDmRulVeP6IrQ* (2018).

[6] SB Kotsiantis, Dimitris Kanellopoulos, and PE Pintelas. "Data preprocessing for supervised leaning". In: *International Journal of Computer Science* 1.2 (2006), pp. 111–117.

[7] SB Kotsiantis, Dimitris Kanellopoulos, and PE Pintelas. "Data preprocessing for supervised leaning". In: *International Journal of Computer Science* 1.2 (2006), p. 3.

[8] Ryan Kuhn. "Analysis of Credit Approval Data". In: *URL: http://rstudio-pubs-static.s3.amazonaws.com/73039.html* ().

[9] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[10] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. "Introduction to information retrieval". In: *Natural Language Engineering* (2008).

[11] Wendy L Martinez, Angel R Martinez, and Jeffrey Solka. *Exploratory data analysis with MATLAB.* Chapman and Hall/CRC, 2017.

[12] Sebastian Raschka. “Model evaluation, model selection, and algorithm selection in machine learning 2018”. In: *URL: https://sebastianraschka.com/pdf/manuscripts/model-eval.pdf* (2019).

[13] Sunil Ray. “Improve Your Model Performance using Cross Validation (in Python and R)”. In: *URL: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/* (2018).

[14] “sklearn.preprocessing.LabelEncoder”. In: *URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html* ().