DSE 511: Final Project Report

Baltimore Ravens

Team members:

Su-Ann Chong, Matthew Horan, Regis Nisengwe, Prachi Patel

Introduction

In this assignment, our group was asked to provide a useful analysis of a given database of bank data provided by a Czech bank which included account information, loan information, and transaction history. The group chose to specifically focus on the status of the loan (paid off/not paid off, current or in debt) and the amount of the loan to determine an algorithm for the risk and amount approved for future loans. We used machine learning classification techniques in order to determine the risk of the loan and regression analysis in order to determine the amount expected from the loan. Although no classifier will be perfect due to the inherent randomness of people, this can provide future loan officers some insights when deciding to approve or disapprove a loan application along with some idea of whether the amount is normal.

Description of Dataset

The original dataset can be obtained from PKDD'99 Discovery Challenge Financial Dataset.
The dataset contains 8 tables with the following descriptions:

Tables	**Descriptions**
Account	each record describes static characteristics of an account
Client	each record describes characteristics of a client
Disposition	each record relates together a client with an account
Permanent order	each record describes characteristics of a payment order
Transaction	each record describes one transaction on an account
Loan	each record describes a loan granted for a given account
Credit card	each record describes a credit card issued to an account
Demographic data	each record describes demographic characteristics of a district

Data Preprocessing and Feature Selection

The initial dataset requires some data cleaning and processing. First, because the dataset is written in Czech, some of the terminologies are translated into English to allow easier understanding of our dataset. Some of the columns are disregarded because of the lack of relevance to our objectives. For example, the bank and account of the recipient of a permanent order. There are several columns that have missing values. For the ones with missing categorical data, the missing values are classified as "Unknown". For the ones with missing numerical data, the missing values are replaced by the mean value depending on the

distribution of the data. For example, there are missing values in the “unemployment rate ‘95” and “no. of committed crimes ‘95” columns. The missing values are replaced by the mean of the distribution after eliminating outliers in the data. The distribution of the two columns with missing values is shown in Figure 1.

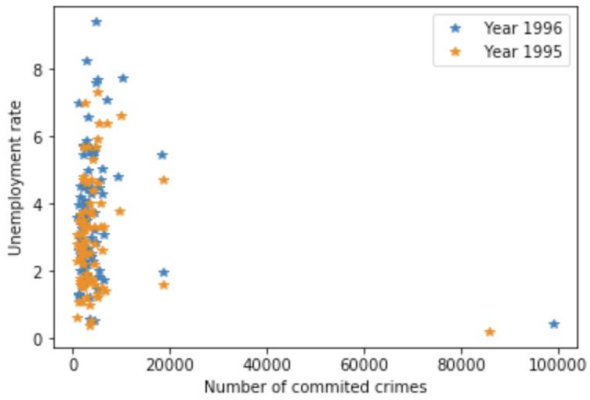

Figure 1: Missing values in the unemployment rate and number of committed crimes in 1995 is replaced by the mean of the distribution after removing any outliers.

Once the dataset is clean, we proceeded to merge all 8 tables into 1 table that is easily accessible and used for applying machine learning models. Our goal throughout this process is to be able to describe loan accounts by the overall amount, whether they are active, and whether they are current (i.e. if they are good or bad loans). Therefore, we sorted each individual loan given the rest of the data in the database. In many cases, such as adding the account and district information, a simple join allows the tables to be combined. However there are several decisions that need to be made in order to combine these tables.

For accounts with multiple users (which in all cases are no more than two), we simply created a new variable that includes the number of users on an account. There is no way of telling which user took out the loan as it is created by account. We use a similar technique to use the number of cards per account (which in some cases is zero).

Transactions and orders were where the most decisions needed to be made in combining the dataset as there were several transactions per account and usually multiple orders. In this case, we split transactions into withdrawals and credits, and summed the number of each and the total amount of each to use in our final algorithms. We used the entire order dataframe. This caused a loss of information of the amount of large/small transactions or the reason for many of these, but did provide data in a format that worked well to use with several other variables in the loan dataset that could be generalized. A series of joins allowed us to combine all eight tables into one simple dataframe that can be loaded as a .csv (saved in the repository), or recreated if given a similar dataset with the code given in our general package. Finally, we applied dummy encoding (UC regents, 2020) in order to define categorical variables.

Here is a heatmap to show the correlation between the columns of the single combined dataframe:

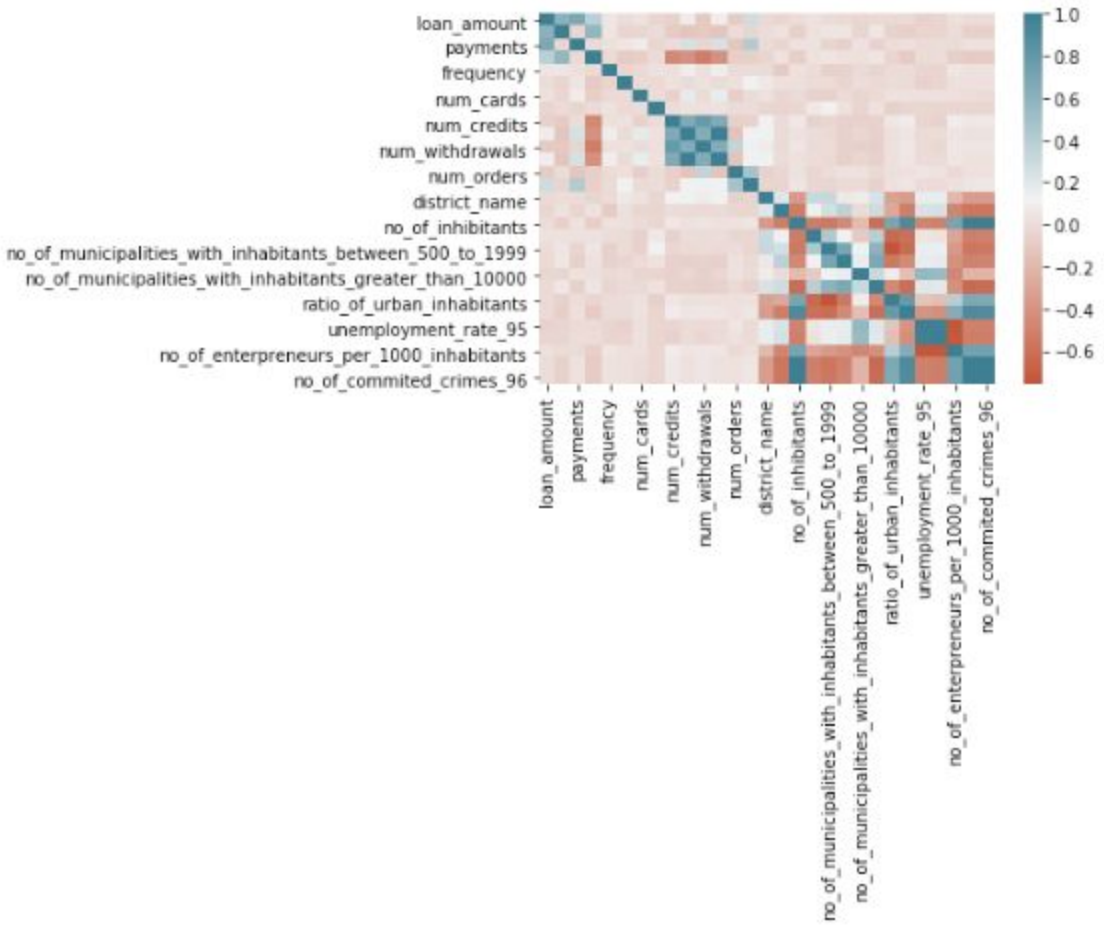

Figure 2: The correlation matrix of the single combined dataframe

From Figure 2, we observed that there are multiple features that are very strongly correlated to one another. Hence, we applied dimensionality reduction technique, specifically the principal component analysis (PCA), to extract useful uncorrelated features. Too many features often make a predictive modeling task more challenging to model because of the curse of dimensionality.

We had a total 33 features from the single combined dataframe, so it is important to do more detailed analysis to see which features affect the most. The main goal of PCA is to extract the important features using variance from the data and to express these features as a set of summary indices called principal components. The visualization of the PCA variance with the number of principal components is shown in Figure 3 in order to get a basic idea about how much information a particular feature is holding so that we can use those features as best components.

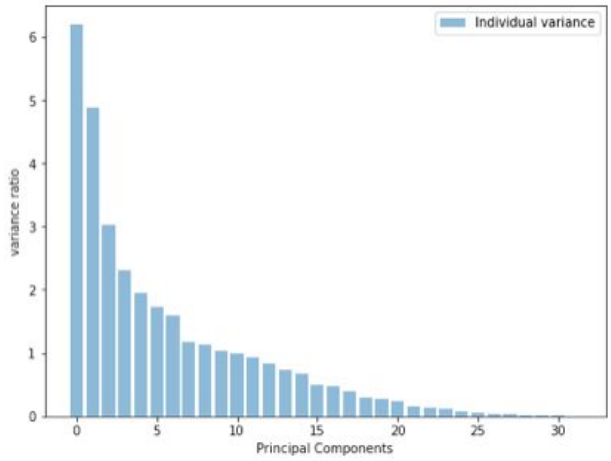

Figure 3: Distribution of the PCA variance as a function of principal components.

According to this variance chart it is clear that after 22 components we are losing more information and the first few components carry important information. After getting the PCA variance we have trained random forest using different sets of components and analyzed the result. It has seen that results vary with different components and choosing right no of components gives the best results. Below table shows the random forest classifier results with different principle components. The whole analysis can be found in the DM_reduction_data.ipynb file on github.

Random Forest Classifier	
No. of Principle Components	**Accuracy**
22	78 %
15	79 %
10	78 %
5	69 %
1	69 %

Loan-Associated Risk Using Classification Models

For classification, our final goal was to determine whether loans fall into Status ‘A’, ‘B’, ‘C’, or ‘D’. ‘A’ and ‘B’ loans both are completed, with A loans having no problem and B loans defaulting on a portion. C and D loans are both still active with C loans in good standing and D loans being overdue or having problems of some sort. The intention for anyone using these algorithms is to minimize the number of ‘B’ or ‘D’ loans that are given in the future. Initially, we

split our dataset in at 75/25 training and validation set using the train_test_split code in sklearn. For classifications, we applied Decision Tree, Support Vector Machine, Random Forest and K-Nearest Neighbors machine learning models to the dataset. All of the machine learning models, scoring metric, splitting of training and testing set, cross-validation and hyperparameter tuning are imported from sklearn module.

Decision Tree (DT) model uses a decision tree as a predictive model to predict the value of a target value based on several input variables. Figure 4 shows a 2-layer graph using the DT

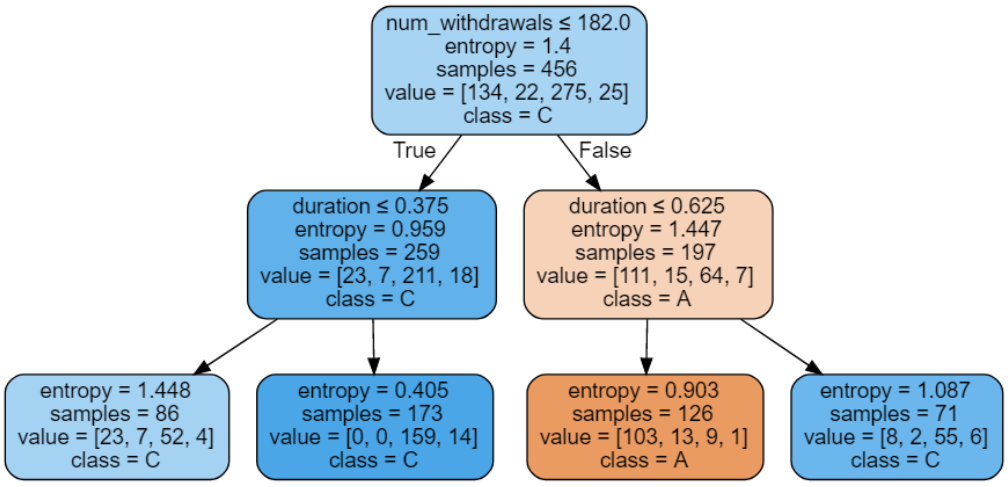

Figure 4: Decision tree provided by idealized GridSearchCV

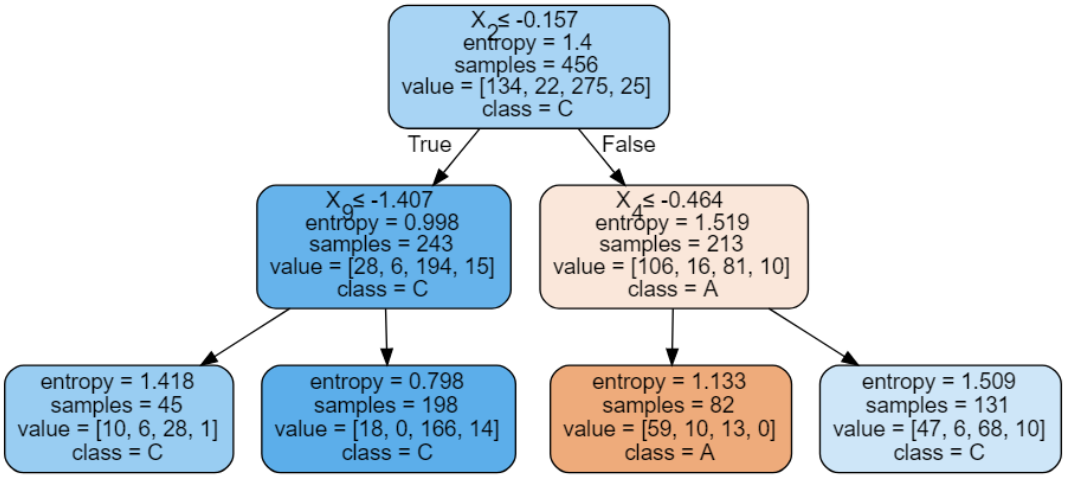

Figure 5: Decision tree provided by idealized GridSearchCV after PCA

model after finding the optimal hyperparameter using GridSearchCV (Scikit Learn, 2020). The hyperparameters chosen for the DT model are

{'criterion': 'entropy', 'max_depth': 2, 'min_samples_leaf': 1}

Support vector machine (SVM) models construct a hyperplane or set of hyperplanes in a high-dimensional data. The SVM scheme finds an optimal hyperplane that has the best separation (largest distance to the nearest datapoint) because the larger the margin, the lower the generalization error of the classifier. The hyperparameter used are:

{'C': 0.5, 'degree': 2, 'kernel': 'sigmoid', 'probability': True}

Random forest (RF) model is basically a collection of multiple decision trees. Each decision tree makes a prediction based on input variables and RF chooses the target value based on majority voting among the decision trees. The hyperparameter used are:

{‘max_depth’ : 2, ‘random_state’: 0}

K-Nearest Neighbors (k-NN) model is a non-parametric method to predict a target variable based on the majority voting within a neighborhood with k-nearest data points. The parameter found using the GridSearchCV method is {'n_neighbors': 73}. However, using the elbow method, a separate k can be approximated.

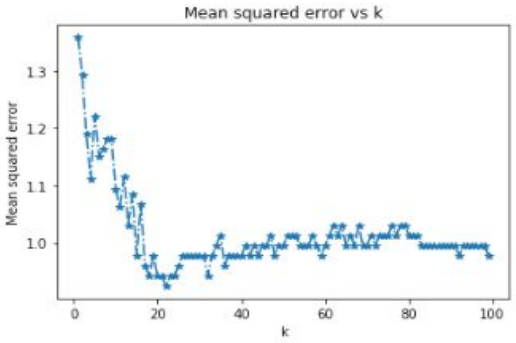

Figure 5: Using the elbow method, the optimal k is about 20.

Models	Accuracy
Decision Tree	64.2%
Support Vector Machine	73.0%
Random Forest	76.6%
K-Nearest Neighbor	70.53 %

Table 1: Accuracy based on different machine learning algorithm for loan status classification

Finally, we performed a 5-fold cross validation on the Decision tree classifier, SVM and kNN as shown in Figure 6 and 7. The decision tree always provided value over straight guessing, but was sometimes very low. SVM and kNN are much more consistent and generally better performing than decision trees.

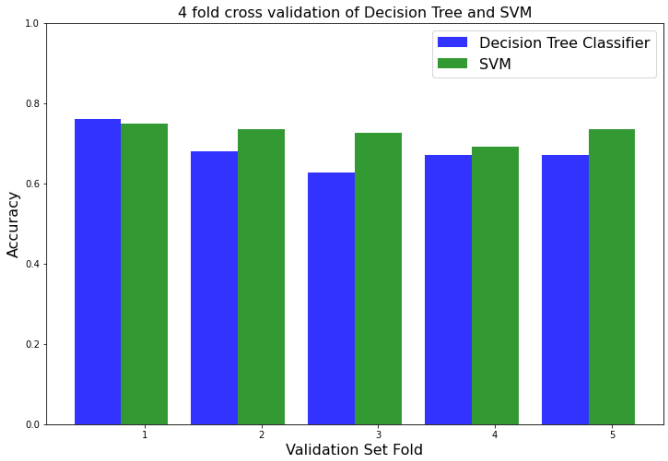

Figure 6: Accuracy of 5-fold cross validation on idealized SVM and Decision Tree

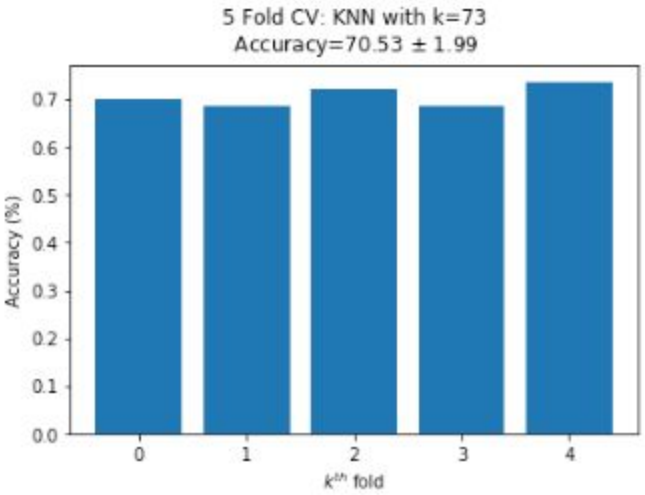

Figure 7: Accuracy of 5-fold cross validation on kNN

Loan Amount Prediction Using Regression Models

In addition to machine learning classification techniques to determine the risk of the loan, our group used regression analysis to determine the amount expected from the loan. Regression is a method used to assess the relationship between a response (dependent) variable Y and a covariate X. The covariate is also known as a predictor variable or a feature. This helps us use predictor variables to predict the response variable (Wasserman, 2019).

In our analysis, predictor (independent) variables of the regression were: average salary, number of withdrawals, duration, payments, and status. Since “status” was a categorical and nominal variable, we transformed it into a categorical but numerical variable to allow for analysis. The dependent variable of the regression was the loan amount.

Since the dependent variable is continuous, the appropriate regression analysis is the linear regression analysis. The use of many predictor variables results in a multiple linear regression analysis (Sauter, 2002).

After defining the variables, we define the model using LinearRegression() method in ScikitLearn. When this was complete, we proceeded with a dataset split in test and training sets with 80/20 split using train_test_split() method. This was followed by model fitting using the regressor.fit() method. This gave an intercept of about -0.186 and the regressor.coef gave the following coefficients:

	Coefficient
average_salary	0.014323
num_withdrawals	0.000042
duration	0.338726
payments	0.586865
status_	0.009957

The next step was to make predictions of the loan amount based on all predictor variables. This was achieved using the regressor.predict() method. A comparison of the actual values with the predicted values gave the following results:

Actual	**Predicted**
0.134248	0.135450
0.249099	0.245730
0.672778	0.566094
0.483716	0.468802
0.029660	0.028854

Using the Seaborn library (sns.regplot()), we can visualize this by plotting the actual against predicted values, shown in Figure 8.

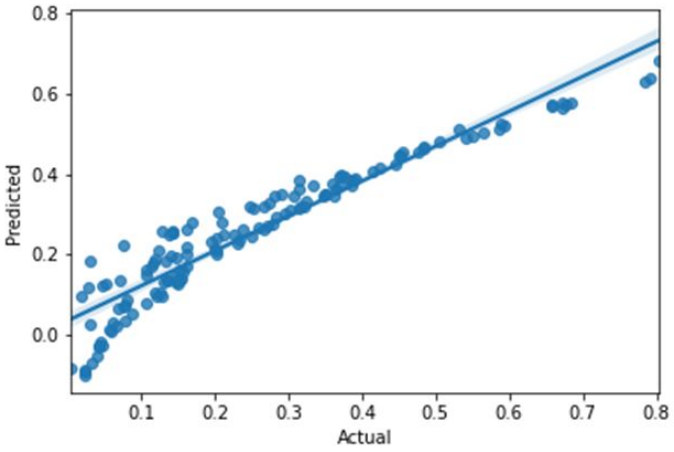

Figure 8: Linear regression on predicted values vs actual values.

A calculation of various metrics of the predictive model gave the following values:

- Mean Absolute Error: 0.0448614437320704
- Mean Squared Error: 0.003702192214587743
- Root Mean Squared Error: 0.06084564252752816

Discussion

As our classification and regression errors are searching for different things, they are by no means going to have similar accuracy rates. We can see from the previous work on regression that the RMSE is fairly good at 0.06, however most of this accuracy is contained in the central region (loans of 0.3 to 0.7 of the arbitrary values assigned). The regression model tends to overpredict the larger loans while there is quite a bit of spread in the smaller loans. This is likely due to the human unwillingness to take out too much of a loan, and potentially may be solved by completing a classification algorithm prior to doing regression analysis, though there is always the chance for overfitting with too complex of a model.

Classification on the other hand again shows a strong bias towards the majority. Simply picking items that are favorable (i.e. loans in good standing) and sorting by time will generally get pretty good results if attempting to classify these loans. However, we can note that with idealized models we can get some value out of Support Vector Machines and Decision trees and may be able to identify some cases that really are simply bad loans. Accuracy rates (even including those that are misclassified not as good or bad loans but as complete or incomplete) are around 70% where a random guess can assume at best 59% (blindly picking C, though truly random will be 25% by nature). Risk will not be eliminated, but there can be some risky loans that are simply not accepted due to the risk factors given.

Conclusion

Overall, quite a bit can be gleaned from the dataset already. We chose to simply focus on loan data in order to determine which loans are high risk as well as the amount to expect from each loan. The basic learning algorithms gave us regression results that were very close to the actual mean results. It was quite successful in predicting the mean loan and the values around the mean loan with very little error. However, it had an issue with overpredicting the true value of large loans and predicted far too many to be small loans (with little in the way of over or underpredicting). It is possible that a linear model may not be complex enough to truly find this value and that polynomial regression may be required (though again runs the risk of overfitting the data). Categorical data on the other hand excessively looks to find the favorable loans that are good buys and in all cases seems to underestimate the number of bad loans. Given the data we are given, this may be expected simply due to the number of bad loans in the dataset (very few), though in the rare case one of these models does predict a loan to default it is likely not worth taking the risk. Further work may be done in this respect to analyze the specific reasons for transactions, the maximum transaction and the mean transaction and how it impacts loan risk and amount. We simply looked at the amount of transactions and orders and the total amount.

References

1. Scikit Learn (2020), “GridSearchCV”, accessed 5 Decemember 2020 at https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

2. UC Regents (2020) “FAQ: WHAT IS DUMMY CODING?”, accessed 5 December 2020 at https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faqwhat-is-dummy-coding/#:~:text=Dummy%20coding%20provides%20one%20way%20of%20using%20categorical,all%20of%20the%20necessary%20information%20on%20group%20membership.

3. https://stackabuse.com/implementing-pca-in-python-with-scikit-learn/

4. Wasserman, 2019 “All of Statistics” Springs, access 8 December 2020 at https://www.datasciencecentral.com/profiles/blogs/free-book-on-statistics-larry-wasserman-s-all-of-statistics