MDB ANALYTICAL MODELING

Problem B: German Credit Data

Andrea Sabia

Student ID: e197047

Table of Contents

Introduce the dataset. 2

Describe the chosen models; and explain your rationality of these choices. 3

Technical execution of model that you have selected. 3

ROC analysis. 5

Interpretation of your results and compare the analytical performance of at least 2 models. 6

Results from Logistic Regression: 6

Results from Random Forest: 6

Results from KNN: 7

Results from K-Means: 7

Results from Association Rules: 8

Comparison and interpretation: 9

My own insights in this problem based on what I have done. 10

Introduce the dataset.

This dataset is a commonly used dataset in Machine Learning practice. It contains outcomes from 900 loan applications. Given 20 independent variables, the problem at hand asks us to predict whether the applicant will default or not on the loan. The dataset is made of the following variables:

```
RangeIndex: 900 entries, 0 to 899
Data columns (total 21 columns):
Default             900 non-null int64
checkingstatus1     900 non-null object
duration            900 non-null int64
history             900 non-null object
purpose             900 non-null object
amount              900 non-null int64
savings             900 non-null object
employ              900 non-null object
installment         900 non-null int64
status              900 non-null object
others              900 non-null object
residence           900 non-null int64
property            900 non-null object
age                 900 non-null int64
otherplans          900 non-null object
housing             900 non-null object
cards               900 non-null int64
job                 900 non-null object
liable              900 non-null int64
tele                900 non-null object
foreign             900 non-null object
dtypes: int64(8), object(13)
```

Figure 1

The variables with type “object” are encoded and categorical. The variables with type “int64” are numerical and continuous.

We can describe the continuous variables as follows:

	Default	duration	amount	installment	residence	age	cards	liable
count	900.000000	900.000000	900.000000	900.000000	900.000000	900.000000	900.000000	900.000000
mean	0.300000	20.732222	3268.436667	2.965556	2.866667	35.622222	1.412222	1.155556
std	0.458512	11.790178	2839.812099	1.116133	1.102127	11.429309	0.575806	0.362635
min	0.000000	4.000000	250.000000	1.000000	1.000000	19.000000	1.000000	1.000000
25%	0.000000	12.000000	1359.500000	2.000000	2.000000	27.000000	1.000000	1.000000
50%	0.000000	18.000000	2308.500000	3.000000	3.000000	33.000000	1.000000	1.000000
75%	1.000000	24.000000	3965.250000	4.000000	4.000000	42.000000	2.000000	1.000000
max	1.000000	60.000000	18424.000000	4.000000	4.000000	75.000000	4.000000	2.000000

Figure 2

Figure 2 gives the most common statistics on the variables at hand, including our target variable default. In particular, we can see that the variables duration, amount and age are rather sparse, whereas the other variables have a range of 1-4 or 1-2. In the data preparation, this will be taken into consideration, by considering these variables as categorical.

By analyzing the data, it is possible to see there are no missing values. Hence, no need for any imputation.

After the preprocessing, we enter the modeling stage with 55 variables (due to the "dummification" of categorical variables)

Describe the chosen models; and explain your rationality of these choices.

My hypothesis is that the best model for this approach are Logistic Regression and KNN. First and foremost, we have a target variable, meaning this problem is a supervised machine learning problem. Secondly, the target variable is categorical, and so are most of the independent variables, which means that the 2 above algorithms should perform rather well.

We are not treating a multiclass problem, but a binary problem, for which Logistic Regression is highly suited.

KNN is arguably the most "simple" out of the supervised machine learning techniques, as the outcome of a new record is based on majority voting of the closest neighbors in the training set. I can hence argue that, unless very specific situation arises, such as a perfect balance between the 2 possible outcomes, this algorithm will perform worse than Logistic Regression. This is because if one of the 2 classes has more records, it will systematically win the majority voting on which this algorithm is based.

I will also test a random forest, an algorithm which performs really well on most types of data. Random forest is an algorithm based on the decision trees. It is a method that falls in the category of ensamble learning, as it pools from the knowledge of multiple iterations, and in this case, it pools from N iterations of decision trees. This method is so powerful since it pools from N uncorrelated decision trees, leading to the greatest amount of learning that can be done. However, as with decision trees, we need to pay attention to the potential overfitting of this model.

To prove that unsupervised machine learning techniques don't work as well as supervised techniques, I will run a K-Means analysis, and also use association rules on the dataset.

Technical execution of model that you have selected.

For Logistic Regression, KNN and Random Forest, after having explored the dataset, I dummified the categorical variables, and normalized the continuous ones (with a min-max scaling algorithm). After doing so, we had 55 variables in the model. Subsequently, I split the dataset in training and test set. As the preprocessing is complete, we can now start with the executions of the models.

The first model I ran was a logistic regression, as I hypothesized it was going to be the best model. I first ran it with all the variables possible, and then used the output from the first model to select only the significant variables (p-value<0.05, which is the selected level of

significance). When doing so, you arrive to figure 3, with only 13 variables instead of 55. You actually lose in terms of performance (AUC is 0.78 vs. 0.79), but with a lot less information, you almost manage to get a similar result. The remaining of the analysis will be carried out on the full dataset however, as with so few records, the number of variables does not impact performance of the machine.

```
Optimization terminated successfully.
         Current function value: 0.493204
         Iterations 7
                         Results: Logit
================================================================
Model:                 Logit            Pseudo R-squared: 0.207
Dependent Variable:    Default          AIC:              620.8043
Date:                  2020-02-27 10:34 BIC:              678.0292
No. Observations:      603              Log-Likelihood:   -297.40
Df Model:              12               LL-Null:          -374.96
Df Residuals:          590              LLR p-value:      5.1953e-27
Converged:             1.0000           Scale:            1.0000
No. Iterations:        7.0000
----------------------------------------------------------------
                     Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
----------------------------------------------------------------
amount                3.5270   0.7142  4.9387 0.0000  2.1273  4.9268
checkingstatus1_A13 -1.6441   0.5162 -3.1850 0.0014 -2.6558 -0.6324
checkingstatus1_A14 -1.5172   0.2306 -6.5780 0.0000 -1.9692 -1.0651
history_A33         -1.0291   0.4375 -2.3525 0.0187 -1.8865 -0.1717
history_A34         -0.9806   0.2438 -4.0217 0.0001 -1.4585 -0.5027
purpose_A41         -1.2194   0.3983 -3.0614 0.0022 -2.0002 -0.4387
savings_A65         -0.7289   0.2990 -2.4382 0.0148 -1.3148 -0.1430
installment_4        0.4249   0.1852  2.2948 0.0217  0.0620  0.7878
status_A93          -0.6265   0.2074 -3.0202 0.0025 -1.0331 -0.2199
residence_2          0.3114   0.2206  1.4114 0.1581 -0.1210  0.7437
property_A124        0.7383   0.3011  2.4519 0.0142  0.1481  1.3284
tele_A192           -0.7016   0.2202 -3.1863 0.0014 -1.1332 -0.2700
foreign_A202        -1.8837   0.7801 -2.4146 0.0158 -3.4128 -0.3547
================================================================
```

Figure 3

For Random Forest, I followed roughly the same procedure as per the Logistic regression. However, we do not have variables significance in this particular model, hence I had to settle for variable importance. As with logistic regression, the AUC is rather similar (lower by 3 percentage points) when you decrease the number of variables, hence I decided to keep all the variables in the model.

I implemented a KNN model with a search parameter, looking for the optimal N neighbours. The model searched through K=1 to K=10, and arrived to the decision the optimal number is 6. Indeed, the AUC for this specific model is significantly higher (10 percentage point) than the average of the others.

K-means on the other hand requires only continuous data, otherwise the model does not converge well. Indeed, in this case, I only used 3 continuous variable for K-means, hence explaining the poor performance.

For association rules, I had to bin the continuous variables instead of normalizing them as I did in the previous models. I used quintiles for this binning, and created 4 classes. We now have 67 features. I set the min support to 0.05, and min confidence at 0.6. This generates slightly less than 50000 rules.

ROC analysis.

As it is possible to see from figure 4, Logistic Regression and Random Forest are very similar in performance, followed by KNN. The AUC of K-Means is completely on the wrong side. Even by switching the automatic classification, the AUC would be in the area of 0.60, which is still a very poor AUC.

The ROC curve compares the true positive rate against the False positive rate at all possible cut-off points. It essentially shows the trade of between specificity and sensitivity. We can use this graph to select the cut-off point which leads to the best possible performance for our specific case. Indeed, we should wish to minimize the False negative cases, as not predicting if someone will default is highly costly. This graph helps us optimize this choice.

For example, if we look at the figure 4, for a false positive rate of 0.2, we have a true positive rate of roughly 0.6 for both logistic regression and random forest, whereas it is only of 0.4 for the KNN classifier.

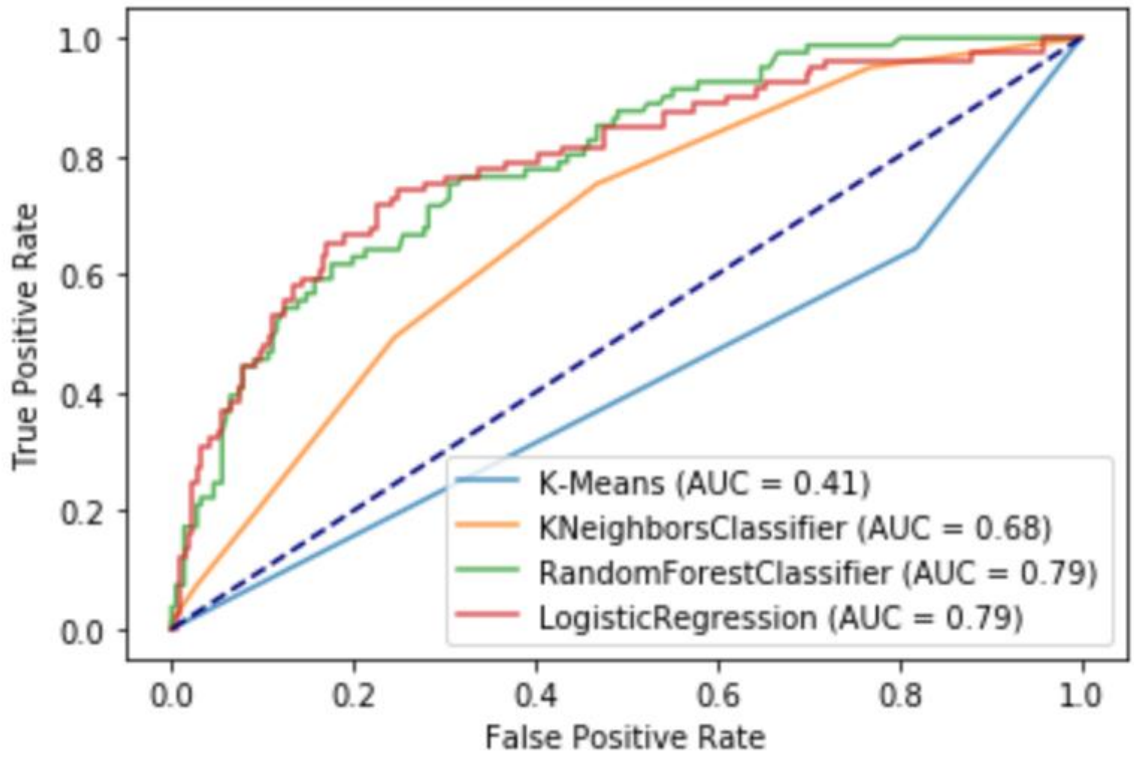

Figure 4 ROC Curves for 4 models

Interpretation of your results and compare the analytical performance of at least 2 models.

Results from Logistic Regression:

```
In [97]: print(classification_report(y_test,predictions))
                  precision    recall  f1-score   support

               0       0.83      0.88      0.85       216
               1       0.61      0.53      0.57        81

        accuracy                           0.78       297
       macro avg       0.72      0.70      0.71       297
    weighted avg       0.77      0.78      0.78       297


In [98]: confusion_matrix(predictions,y_test)
Out[98]: array([[189,  38],
                [ 27,  43]])
```

Figure 5

Results from Random Forest:

```
In [101]: print(classification_report(y_test,randforest))
                  precision    recall  f1-score   support

               0       0.82      0.91      0.86       216
               1       0.65      0.46      0.54        81

        accuracy                           0.78       297
       macro avg       0.73      0.68      0.70       297
    weighted avg       0.77      0.78      0.77       297


In [102]: confusion_matrix(randforest,y_test)
Out[102]: array([[196,  44],
                 [ 20,  37]])
```

Figure 6

Results from KNN:

```
In [32]: print(classification_report(y_test,knn))
```

```
              precision    recall  f1-score   support

           0       0.75      0.91      0.82       216
           1       0.44      0.20      0.27        81

    accuracy                           0.71       297
   macro avg       0.60      0.55      0.55       297
weighted avg       0.67      0.71      0.67       297
```

```
In [33]: confusion_matrix(knn,y_test)
```

```
Out[33]: array([[196,  65],
                [ 20,  16]])
```

Figure 7

Results from K-Means:

As this method has essentially been discarded by AUC in terms of performance, I will not focus on it too much. However, we should note how the performance shows how inadequate the unsupervised techniques are when we are looking at a supervised problem. Across all metrics, this model performs poorly (precision, recall, accuracy, AUC, and F1-Score).

```
In [41]: print(classification_report(y,km))
```

```
              precision    recall  f1-score   support

           0       0.54      0.18      0.27       630
           1       0.25      0.64      0.36       270

    accuracy                           0.32       900
   macro avg       0.40      0.41      0.32       900
weighted avg       0.46      0.32      0.30       900
```

```
In [40]: confusion_matrix(km,y)
```

```
Out[40]: array([[114,  96],
                [516, 174]])
```

Figure 8

Figure 9 is a visualization of the algorithm on the axis of age and duration, 2 of the most important variables in the model. This does not tell us much, but we can see how the algorithm has split the data. The red dots are the centroids.

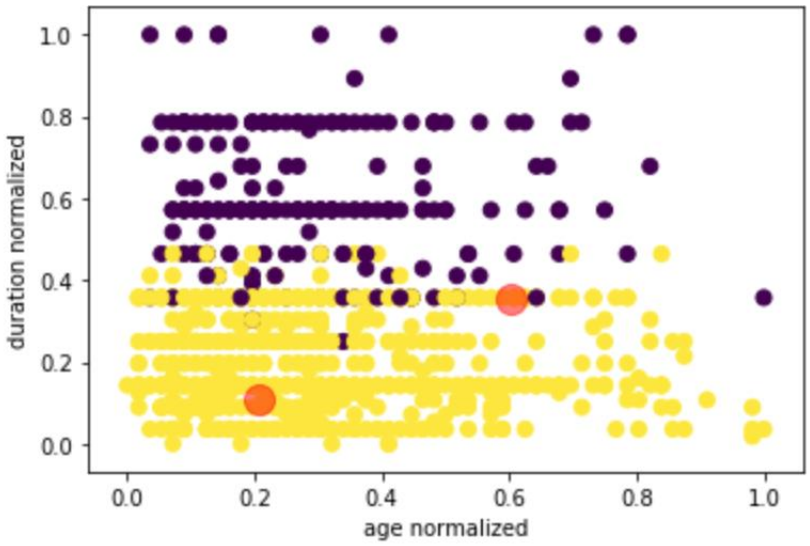

Figure 9

Results from Association Rules:

First and foremost, this technique cannot be used to plot an ROC curve. However, we can interpret the output below to get some insights and compare its performance to that of the other models.

If we take the first combination of antecedents and consequents, we can see that the antecedents have a support of 7.5%, meaning the happen together in the dataset 7.5%. Out of those cases, they lead to default 66.6% of the time (as per the "Confidence" metric). From this rule onwards, we have less and less support and confidence. This means that this technique can be used for predicting default. However, the accuracy with which it does so is very low. Hence we can discard this model from the analysis here on out.

```
In [103]: #Top rules which lead to default
          parameter1 = rules[(rules['S']== True) & (rules['lift']>0)]
          parameter1.sort_values('confidence', ascending=False).head(5)
```

Out[103]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
33318	(others_A101, savings_A61, job_A173, foreign_A...	[Default]	0.113333	0.3	0.075556	0.666667	2.222222	0.041556	2.100000
15298	(savings_A61, job_A173, checkingstatus1_A11, o...	[Default]	0.115556	0.3	0.075556	0.653846	2.179487	0.040889	2.022222
45958	(otherplans_A143, others_A101, savings_A61, jo...	[Default]	0.088889	0.3	0.057778	0.650000	2.166667	0.031111	2.000000
33270	(otherplans_A143, others_A101, savings_A61, jo...	[Default]	0.091111	0.3	0.057778	0.634146	2.113821	0.030444	1.913333
33231	(others_A101, history_A32, job_A173, foreign_A...	[Default]	0.084444	0.3	0.053333	0.631579	2.105263	0.028000	1.900000

Figure 10

Comparison and interpretation:

There are various ways we can compare the analytical performance of the above models. Arguably the most effective one is ROC analysis, as presented in the previous section. Another powerful method for comparing performance is the confusion matrix, along with all of it metrics, such as sensitivity, specificity or F1-Score.

All of the confusion matrices for the techniques can be found above. There are very few significant differences between Random Forest and Logistic Regression. Accuracy is essentially the same (0.78), and the difference in performance is just due to the misclassification type (Logistic Regression misclassified more true positives, whereas Random Forest misclassified more true negatives). The real difference can be found when comparing KNN and Logistic Regression. Indeed, the accuracy of KNN is 6 percentage points lower than the one from Logistic Regression. However the difference can be best seen in terms of F1 score, which is the harmonic average of precision and recall, which are 2 very important metrics in classification. Indeed, the F1 score for a default is 0.27 for KNN, against a 0.57 of Logistic Regression. As we are more interested in better predicting the applicants who will default, using KNN is not the best approach.

Lastly, we can visualize a Precision vs. Recall curve, which shows the tradeoff between being precise and sensitive. As previously mentioned, the difference between Random Forest and Logistic Regression is minimal in terms of AUC, Accuracy, and F1 score. This graph also shows us that the precision and recall are similar at similar cut-off points for the two algorithms. We can also definitively discard KNN (GridsearchCV in the figure 11), whose performance is consistently worse than that of the other 2 algorithms.

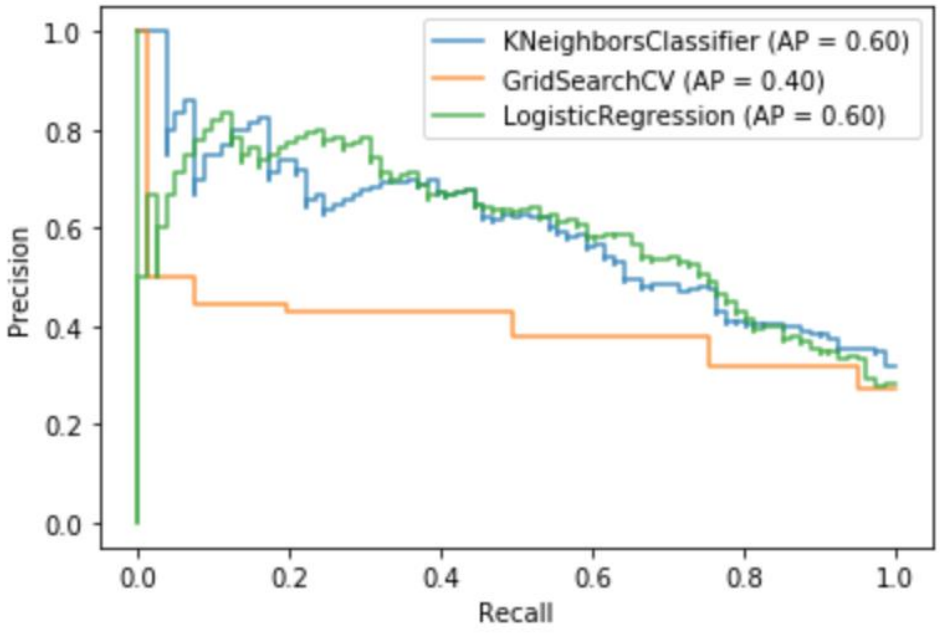

Figure 11 Precision Recall Curve

Now that we have compared the results, we can interpret what these models mean.

From the output of the logistic regression, we can see the coefficients of the independent variables, alongside their significance. We can hence see which variables have a greater effect

on predicting default. We also have this opportunity with the Random Forest. Please find below the top 10 most important variables according to the Random Forest.

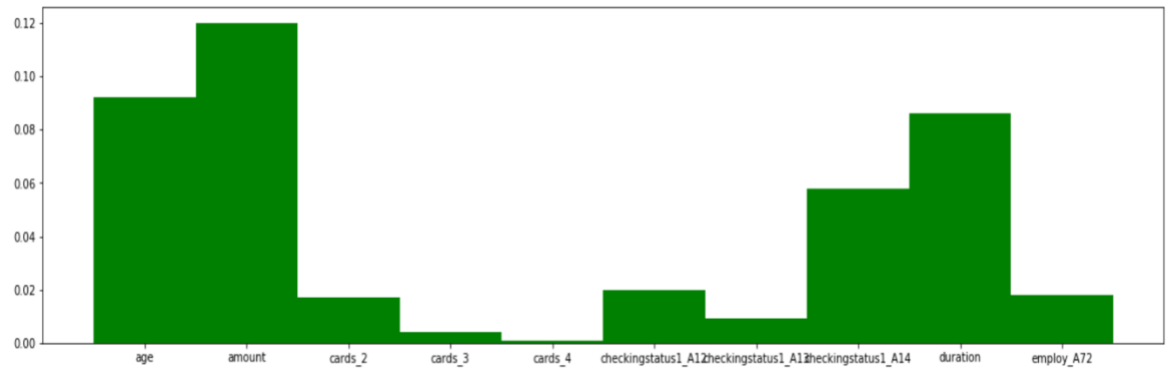

Figure 12

We can see that certain variables are considered important and significant by both models. Indeed, amount is the most important variables in both models. The output from the logistic regression tells us that, all else equal, an increase in 1 unit of the amount, leads to an increase of the probability of default of 3.5. Essentially, the more money you borrow, the more likely you are to default. Another variable which is shared by the models is checkingaccountstatus_14. The coefficient of this variable in the logistic regression is negative, which means that if you do not have a checking account, you are less likely to default.
Other important variables according to the random forest are age, which can be understandable as the older you are, the higher your income, the more chances of paying back the loan. The duration of the loan seems also very important according to the output. The variable cards also seems important (albeit much less). Obviously your employment status is relevant. In this case, the model believes it can use the fact that you have been employed for less than 1 year to understand better if you will default or not on your loan.

My own insights in this problem based on what I have done.

The most important insight is that the best models are the supervised ones. If you have a target variable, it will be easier for the machine to understand what leads to that specific output. I can conclude that using either Logistic Regression or Random Forest is acceptable, as they both perform rather well on this data. Obviously, with more datapoints their accuracy would increase.

The confusion matrices are all showcasing the scenario for which the cutoff is 0.5. In this case we are saying that it is as important do detect who will default as it is to detect who will not default. As we know, there is higher cost in not detecting who will default, as a loss will be made by the bank, rather than wrongly accusing someone who will not default, of defaulting, as the loss here is simply a loss in terms of opportunity cost, which is definitely lower. By using the command Predict_Proba in ScikitLearn, we can adjust this threshold. However, as we do not have information on the exact “cost” of making one mistake or the other, I have decided

to leave the threshold as it is. Indeed, in this case, we should compare the models using AUC and Precision Recall Curves, as they tell us the magnitude of various parameters across the various different cut-off points.

In the section above, I have detected which ones are the important variables in the models. We can use those ones (or all of them, as I did), to create the models. In case we have more records, which is an ideal situation, we would only use the important, significant variables, as they would not be computationally heavy, and lead to similar results.

Another important metric that I have not covered as there was no need for it, is the overfitting. In our models there is no overfitting, but in case we had a lot more variables, it would have been useful to draw a comparison between the performance of the model on the training set and on the testing set, to check for overfitting of the training set.