Quantitative Exercise - build a PD model

Nathaniel Ahy

May 2020

Approach

1. Pycharm with scikit and pandas packages were used
2. The methods attempted were:
   - Logistic Regression
   - Linear Discriminant Analysis (LDA)
   - Quadratic Discriminant Analysis (QDA)
   - k-Nearest Neighbors
   - Random Forest (Classification and Regression Trees)
   - Gradient Boosting
   - Adaboost

Data preprocessing 1

1. All features missing more than 50% of their values were removed
2. Since there were significantly more defaults than non-defaults we tried three approaches:

   - Carry on as usual, which lead to the most accurate models predicting 'no default' every time. This is a suitable description of the results with this method it will not be discussed any further

   - Use all the remaining default cases and take equally many samples from the remaining non-default cases

   - Balance the data for models that require it, i.e. increasing the importance of classifying the underrepresented default scenarios

Data preprocessing 2

1. Normalize all features x_j such that $x_j^{new} = \frac{x_j - x_{min}}{x_{max} - x_{min}}$
2. Remove all features with correlation greater than 0.8 with any other features
3. Use principal component analysis to keep the 10 most significant features

Logistic Regression, LDA, and QDA: 50/50

1. We will refer to the scenario where we sampled equally many non-default cases as default cases as the "50/50" scenario and the aforementioned balancing scheme as the "weighted scenario"
2. In the 50/50 scenario the Logistic Regression, LDA, and QDA models achieved a 10-fold mean-test error of 0.1867, 0.1817, and 0.2427 respectively
3. The table below displays the corresponding rates of true positives (TP) [accurately classified defaults], true negatives (TN), false positives (FP), and false negatives (FP).

	Logistic	QDA	LDA
TP	0.4066	0.3529	0.4348
TN	0.4040	0.4040	0.3836
FP	0.0895	0.0895	0.1100
FN	0.0997	0.1534	0.0716

Logistic Regression, LDA, and QDA: Weighted

1. In the weighted scenario the Logistic Regression, LDA, and QDA models achieved a 10-fold mean-test error of 0.2316, 0.0291, 0.0655 respectively, where only guessing 'non-default' would entail a relative error of 0.0181
2. The table below displays the corresponding rates of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FP). It is worth noting that we could not apply weights to LDA and QDA models, however, according to a paper by Xue and Titterington [1] this doesn't lead to a significantly different result

	Logistic	QDA	LDA
TP	0.0146	0.0066	0.0022
TN	0.7538	0.9278	0.9686
FP	0.2282	0.0541	0.0133
FN	0.0035	0.0114	0.0158

k - Nearest Neighbors: 50/50 and Weighted

1. Implemented the k - Nearest neighbors algorithm for $k \in \{1, 2, ..., 50\}$
2. In this case we settled for a non-weighted method; a shortcoming of this study
3. The errors for the best k for the 50/50 and non-weighted setting were 0.1841025641025641 2 (for $k = 40$) and 0.01807 (flagging all cases as non-defaults) respectively

	50/50	Default
TP	0.4373	0
TN	0.3785	0.9819
FP	0.1151	0
FN	0.0691	0.0181

Boosting: 50/50 and Weighted

1. Adaboost was significantly outperformed by Gradient boosting and will therefore not be considered
2. Gradient boosting was tried with Maximum depth=i, Minimum sample leaves=j, $(i, j) \in \{1, 2, 3, 4, 5\}$. The smallest errors in the 50/50 and weighted setting were 0.1894 with Maximum Depth=2, Minimum Sample Leaves=4 and 0.01825 with Maximum depth=1, Minimum sample leaves=4.
3. The Gradient boosting method combats class imbalances by increasing weights on mislabeled datasets. Although, this didn't seem to be particularly successful in this setting

	50/50	Weighted
TP	0.4399	0.0001
TN	0.3708	0.9815
FP	0.1228	0.0003
FN	0.0665	0.0181

Random Forest: 50/50 and Weighted

1. The smallest errors for the Random Forest algorithm in the Weighted and 50/50 setting were: 0.01834 (for minimum trees=6) and 0.1867 (for minimum trees=5)
2. The confusion matrix was:

	50/50	Weighted
TP	0.4322	0.0002
TN	0.3811	0.9815
FP	0.1125	0.0005
FN	0.074	0.0179

Conclusion

1. In this presentation 7 methods for classifying the loans were used
2. At best they classified equally well as an all-zero model when introduced to this unbalanced dataset
3. However, as it is more important to correctly classify loans that will default an important metric here is recall: TP/(TP+FN)
4. The model with the highest recall was boosting in the 50/50 setting, although one would have to sacrifice a lot of accuracy to achieve this level of recall

Shortcomings

1. Although attempts with Univariate selection, recursive feature elimination, and principal component analysis we made too much emphasis was put on model selection as opposed to feature selection. This is most likely the reason why the models considered were outperformed by an all-zero model when using full data
2. This also lead to a more limited subset of the data to be used
3. The only data that remained was where none of the features had contemporary NaN-values, leaving us with less than half our dataset upon removal. This means it might have been better to use, for example, conditional models or find some other way to handle NaN-values instead of avoiding them
4. It may have been better to weigh the dataset instead of relying on sklearn's built in functions as our models in the weighted setting showed poor recall.

Bibliography

1 J.-H. Xue and D. M. Titterington, "Do unbalanced data have a negative effect on lda?" ScienceDirect, vol. 64, pp. 1558–1571, 2008

Appendix 1:k-NN figures

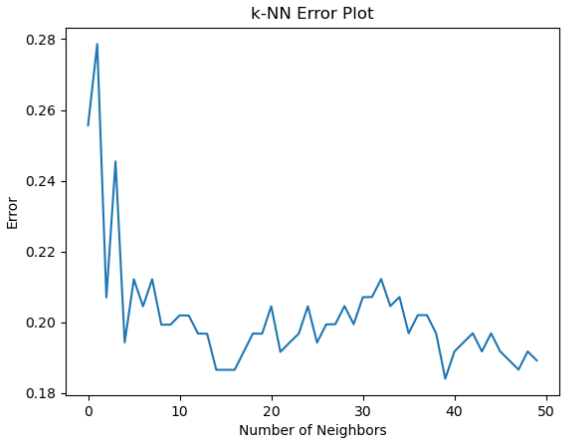
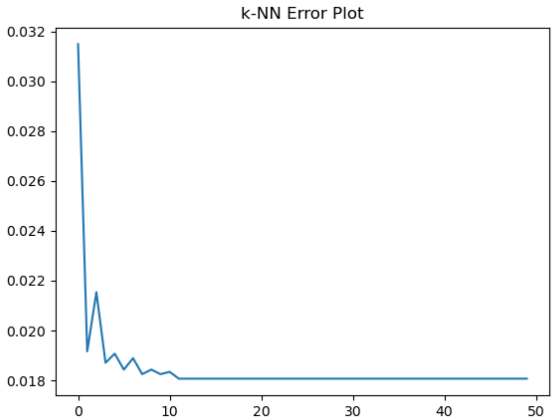


Figure: k-NN 50/50 for $k \in \{1, 2, ..., 50\}$



Appendix 2:Random Forest figures

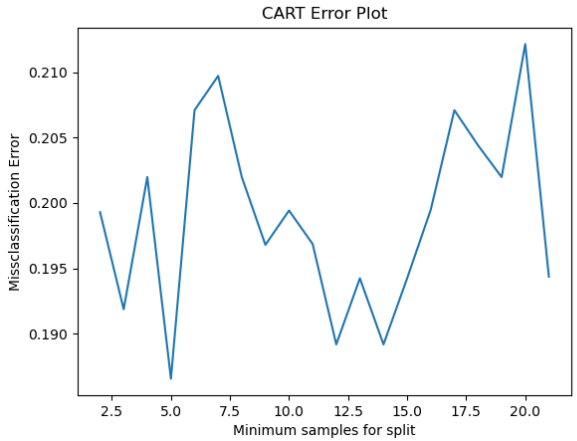


Figure: 50/50 Random Forest Classifier where the minimum number of splits, $s \in \{1, 2, ..., 50\}$