Tree methods

Denis Tsvetkov and Lukas Ostrovskis

January 2023

1 A single decision tree model

5 feature model

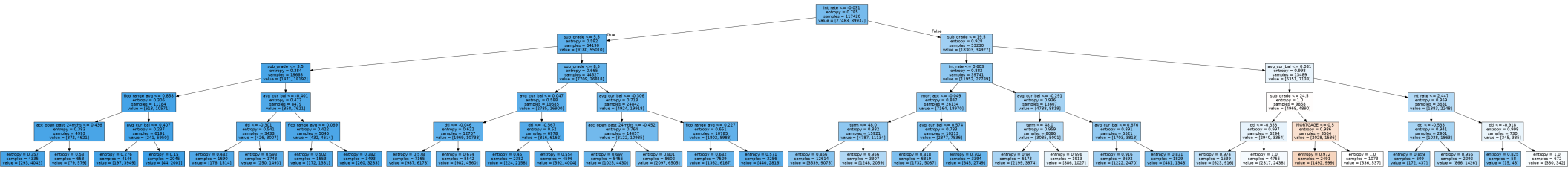
We trained the decision tree model with the 5 most important features identified by the regression model with the elastic net regularization as their importance looks the most plausible based on both model performance and feature definitions. The features are: *sub_grade*, *int_rate*, *WY*, *pub_rec_bankruptcies*, *mo_sin_old_il_acct*.

The feature that the model chose for splitting at the root node was *int_rate*. Below is a table that shows the information gain that could be obtained by each feature at the root node.

Feature	Threshold value	Expected entropy	Information gain
sub_grade	11.5	0.74483	0.04018
int_rate	-0.031	0.74456	0.04045
WY	N.A.	0.78499	0.0000173
pub_rec_bankruptcies	0.838	0.78428	0.00074
mo_sin_old_il_acct	-0.879	0.78439	0.000627

Table 1: Information gain from features to determine the root node

The plot of the final decision tree is provided on the next page. It is important to note that allowing the model to reach a depth of 5 results in a pretty big tree model, so it might be necessary to zoom in to inspect the individual nodes.


int_rate <= -0.031
entropy = 0.785
samples = 117420
value = [27483, 89937]
True
False
sub_grade <= 3.5
entropy = 0.752
samples = 64190
value = [9180, 55010]
sub_grade <= 19.5
entropy = 0.928
samples = 53230
value = [18303, 34927]
sub_grade <= 3.5
entropy = 0.354
samples = 19663
value = [1431, 18232]
sub_grade <= 8.5
entropy = 0.855
samples = 44527
value = [7749, 36818]
int_rate <= 0.803
entropy = 0.892
samples = 39741
value = [13952, 27789]
avg_cur_bal <= 0.081
entropy = 0.998
samples = 13489
value = [4351, 7138]
fico_range_avg <= 0.858
entropy = 0.306
samples = 11184
value = [615, 10571]
avg_cur_bal <= -0.001
entropy = 0.473
samples = 8479
value = [816, 7631]
avg_cur_bal <= 0.047
entropy = 0.598
samples = 15635
value = [2785, 12850]
avg_cur_bal <= -0.306
entropy = 0.718
samples = 24842
value = [4926, 19918]
mort_acc <= -0.049
entropy = 0.847
samples = 26134
value = [7354, 18970]
avg_cur_bal <= -0.291
entropy = 0.936
samples = 13607
value = [4786, 8819]
sub_grade <= 24.5
entropy = 1.0
samples = 9858
value = [4968, 4890]
NORTEAST <= 0.5
entropy = 0.966
samples = 3564
value = [2028, 1536]
int_rate <= 2.447
entropy = 0.959
samples = 3655
value = [1393, 2248]
acc_open_past_24mths <= 0.436
entropy = 0.998
samples = 4993
value = [572, 4621]
avg_cur_bal <= 0.497
entropy = 0.237
samples = 6191
value = [243, 5950]
dti <= 0.348
entropy = 0.541
samples = 3413
value = [426, 3007]
fico_range_avg <= 0.069
entropy = 0.422
samples = 5066
value = [632, 4614]
dti <= -0.048
entropy = 0.622
samples = 12707
value = [1969, 10738]
dti <= 0.587
entropy = 0.52
samples = 6978
value = [816, 6162]
acc_open_past_24mths <= -0.452
entropy = 0.704
samples = 14057
value = [3122, 10935]
fico_range_avg <= 0.227
entropy = 0.651
samples = 10785
value = [1802, 8983]
term <= 48.0
entropy = 0.382
samples = 18914
value = [1839, 9075]
avg_cur_bal <= 0.574
entropy = 0.763
samples = 10213
value = [2377, 7836]
term <= 48.0
entropy = 0.959
samples = 9166
value = [3085, 5001]
avg_cur_bal <= 0.676
entropy = 0.891
samples = 5521
value = [1703, 3818]
dti <= 0.193
entropy = 0.997
samples = 6294
value = [2940, 3354]
NORTEAST <= 0.5
entropy = 0.966
samples = 3564
value = [2028, 1536]
dti <= 0.533
entropy = 0.941
samples = 2901
value = [1038, 1863]
dti <= 0.018
entropy = 0.999
samples = 732
value = [345, 385]
entropy = 0.217
samples = 4335
value = [293, 4042]
entropy = 0.53
samples = 658
value = [79, 579]
entropy = 0.278
samples = 4146
value = [197, 3949]
entropy = 0.13
samples = 2045
value = [46, 2001]
entropy = 0.662
samples = 1690
value = [176, 1514]
entropy = 0.593
samples = 1723
value = [250, 1493]
entropy = 0.502
samples = 1553
value = [172, 1381]
entropy = 0.382
samples = 3499
value = [260, 3239]
entropy = 0.719
samples = 7165
value = [987, 6178]
entropy = 0.674
samples = 5542
value = [982, 4560]
entropy = 0.48
samples = 2292
value = [324, 2158]
entropy = 0.554
samples = 4686
value = [492, 4004]
entropy = 0.697
samples = 5455
value = [1025, 4430]
entropy = 0.801
samples = 8602
value = [2097, 6505]
entropy = 0.682
samples = 7529
value = [1362, 6167]
entropy = 0.571
samples = 3256
value = [440, 2816]
entropy = 0.856
samples = 12614
value = [3039, 9575]
entropy = 0.956
samples = 3307
value = [1248, 2059]
entropy = 0.918
samples = 6519
value = [1732, 4787]
entropy = 0.702
samples = 3694
value = [645, 2749]
entropy = 0.94
samples = 6173
value = [2199, 3974]
entropy = 0.996
samples = 1913
value = [886, 1027]
entropy = 0.916
samples = 3692
value = [1222, 2470]
entropy = 0.831
samples = 1829
value = [481, 1348]
entropy = 0.976
samples = 1539
value = [623, 916]
entropy = 1.0
samples = 4755
value = [2317, 2438]
entropy = 0.972
samples = 2491
value = [1492, 999]
entropy = 1.0
samples = 1073
value = [536, 537]
entropy = 0.859
samples = 609
value = [172, 437]
entropy = 0.916
samples = 2292
value = [866, 1426]
entropy = 0.825
samples = 56
value = [15, 41]
entropy = 1.0
samples = 676
value = [330, 342]

Furthermore, we decided to test 5 different prediction thresholds on our tree classifier - 0.5, 0.6, 0.7, 0.8, 0.9. The table with the confusion matrices for each of the thresholds can be found below.

0.5		0.6		0.7		0.8		0.9	
1.15	22.29	5.17	18.27	9.39	14.06	17.88	5.57	22.49	0.96
1.02	75.53	6.15	70.4	13.58	62.97	38.41	38.14	63.69	12.86

Table 2: Confusion matrices for the decision tree model with different prediction thresholds

This table is quite interesting to analyze in the context of our scenario.

On the one hand, it might be tempting to conclude that the model with a threshold of 0.5 is the best since it results in the highest accuracy (76.68%). However, it is important to acknowledge that it also produces a significant amount of false positives (predicting no default but the loan actually defaulting). This can be explained by the imbalance in training data: over 70% of the training samples are loans that do not default, hence the model is biased to guess that a loan is usually good.

On the other hand, we argue that it is obvious that a high percentage of false positives is expensive for loan providers but we can also support this claim by referring to Hull [1], who states at the end of Section 3.11 that revenue can be maximized by picking a threshold such that $1000TP - 4000FP$ is maximized. We tried to maximize the formula proposed by Hull by trying each of the thresholds and the resulting values are -13630, -2680, 6730, 15860 and 9020 respectively. Therefore, the performance metric proposed by Hull seems to indicate that it is most optimal to pick a threshold of 0.8 for the model.

While this looks counter-intuitive, as the accuracy is only 56%, it results in significantly more false negatives which cost less than false positives and thus the resulting model would most likely maximize the revenue for a loan provider but result in a stricter process to give out a loan.

Consequently, we decided to pick the 0.8 threshold. Table with the confusion matrix, true positive rate, and precision can be found below.

Confusion matrix		TPR	Precision
17.88	5.57	0.4982	0.8725
38.41	38.14		

Table 3: Confusion matrix, TPR, and precision for threshold = 0.8

To gain even more insight into the performance of the model we also generated the ROC curve for the model. The ROC curve plot is shown below.

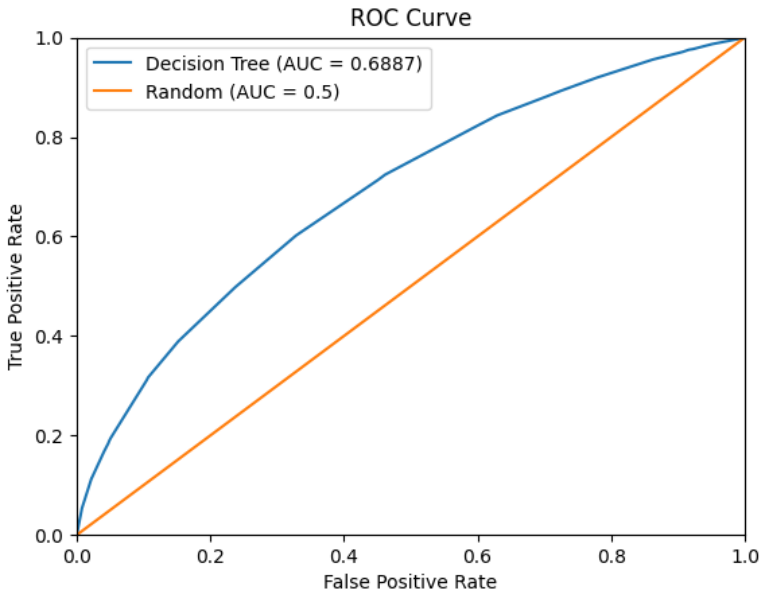


Figure 1: ROC for the Decision Tree (5 features)

It is visible that the model obtains an AOC of 0.6887.

10 feature model

To see if the number of features was an important factor in the analysis we redid it using 10 features. For this, we decided that it might be worthwhile to train a Random Forest model and look at the variable importance diagram that it generates. The features there are obviously effective for generating decision trees and might give us further improvement. According to the diagram, the 10 most important features are *int_rate*, *sub_grade*, *term*, *fico_range_avg*, *dti*, *acc_open_past_24mths*, *mort_acc*, *avg_cur_bal*, *Not Verified*, *MORTGAGE*. We now tried using these 10 features to train a decision tree model.

We again tried to maximize the formula proposed by Hull by trying each of the aforementioned thresholds and the resulting values are -13630, -2600, 8360, 16980, and 9380 respectively. We can again deduce that 0.8 is the optimal threshold. Table with the confusion matrix, true positive rate, and the precision for the optimal threshold can be found below.

Confusion matrix		TPR	Precision
16.76	6.68	0.5708	0.8673
32.85	43.7		

Table 4: Confusion matrix, TPR and precision for threshold = 0.8

While the optimal threshold does not change and remains 0.8, it is clear from the values that the new model is better at maximizing the formula and must therefore be better than the 5-feature decision tree. We can also support this claim by looking at the AUC of the model, which is 0.6957 and is greater than the 0.6887 obtained by the 5-feature model. The ROC curve plot is shown below.

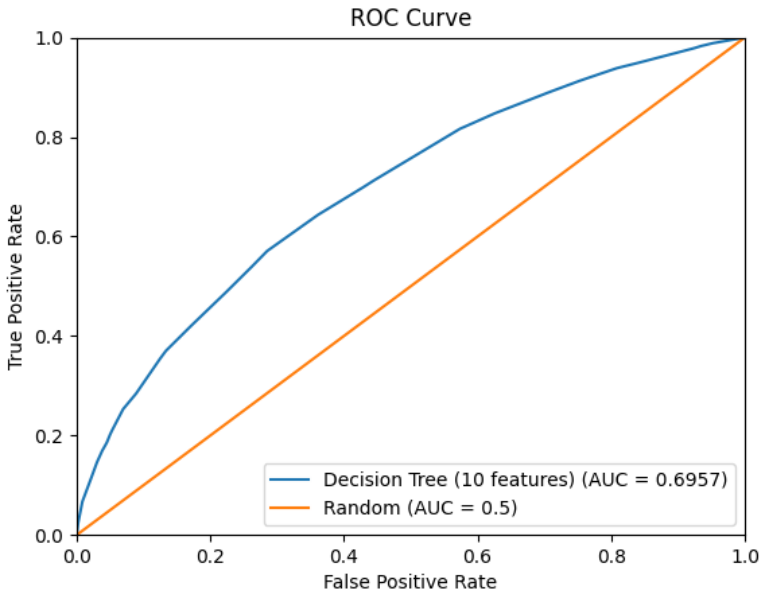


Figure 2: ROC for the Decision Tree (10 features)

Consequently, we conclude that the 10-feature decision tree model is the better one. It has a larger AUC and achieves a better performance as measured by the performance formula we're using.

2 Tree ensemble methods

To create a *RandomForestClassifier*, we looked at the documentation of the random forest classifier class[1] and decided to explicitly provide these parameters:

- criterion (the measure for splitting; we'll use "entropy" as we did for decision trees)
- max_depth (maximum depth of the tree; important for the bias-variance tradeoff)
- oob_score (whether to use out-of-bag samples; necessary for calculating the OOB error)
- random_state (controls the randomness; necessary for consistent results)
- max_features (number of features to consider for a split; we picked 2 values as defined in the assignment)
- n_estimators (number of trees in the forest; 100 by default, worthwhile to try more)
- warm_start (reuse the solution of the previous call to reuse and add more estimators; useful to see how OOB error changes based on the number of estimators)

First of all, we ran a grid search[2] for the *max_depth*, *max_features*, and *n_estimators* parameters to get an initial idea of good values for the hyperparameters. The search resulted in the following values:

- max_depth = 20

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
[2] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

- max_features = "sqrt"
- n_estimators = 500

Based on these findings, we decided to try the following parameter combinations for growing forests:

- max_depth = 10 and max_features = "sqrt"
- max_depth = 20 and max_features = "sqrt"
- max_depth = 10 and max_features = "log2"
- max_depth = 20 and max_features = "log2"

Furthermore, to investigate the effect of the bootstrap sample size we chose an interval of [100, 500] for the sample sizes with a step of 100. In order to evaluate the performance of the models, we created a graph utilizing OOB estimates in relation to the sample size. The resulting plot is provided below.

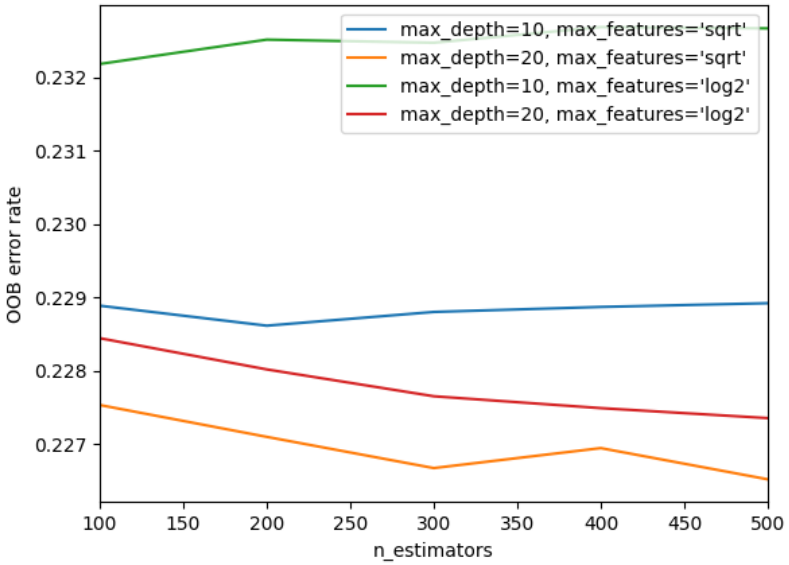


Figure 3: Performance of Random Forest models based on their OOB error estimates

Firstly, it is immediately noticeable that the models with a maximum depth of 10 are inferior to those with a depth of 20. This could be explained by the large amount of data and features we are working with: lower tree depth can result in slight underfitting and is not enough to capture the essence of all the information.

Secondly, we can see that the optimal bootstrap sample size for trees of depth 10 is around 200. On the other hand, for trees of depth 20, it keeps decreasing for the entire interval and most likely converges at an optimal value that is larger than 500.

As a result, the best Random Forest classifier seems to be the one with a maximum depth of 20, maximum features equal to the square root of all features, and a bootstrap sample size equal to 500.

Moreover, to evaluate the OOB error estimation, we compared the OOB error of the best model to the error estimate of our test set. Evaluating the model on the test set results in an error of 0.228, which is only marginally higher than the observed OOB error of 0.2267. We can thus conclude that the OOB error estimation was sufficiently accurate to be a reasonable metric in choosing the best model and did not result in us overfitting the dataset.

Lastly, after finding the best model, we looked at which variables had the largest influence on the model. The importance of a feature is calculated as the mean criterion decrease obtained by that feature. We used a convenient attribute of the *RandomForestClassifier*, *feature_importances_*[3] which does exactly that. The variable importance diagram with the 10 most important variables can be found below.

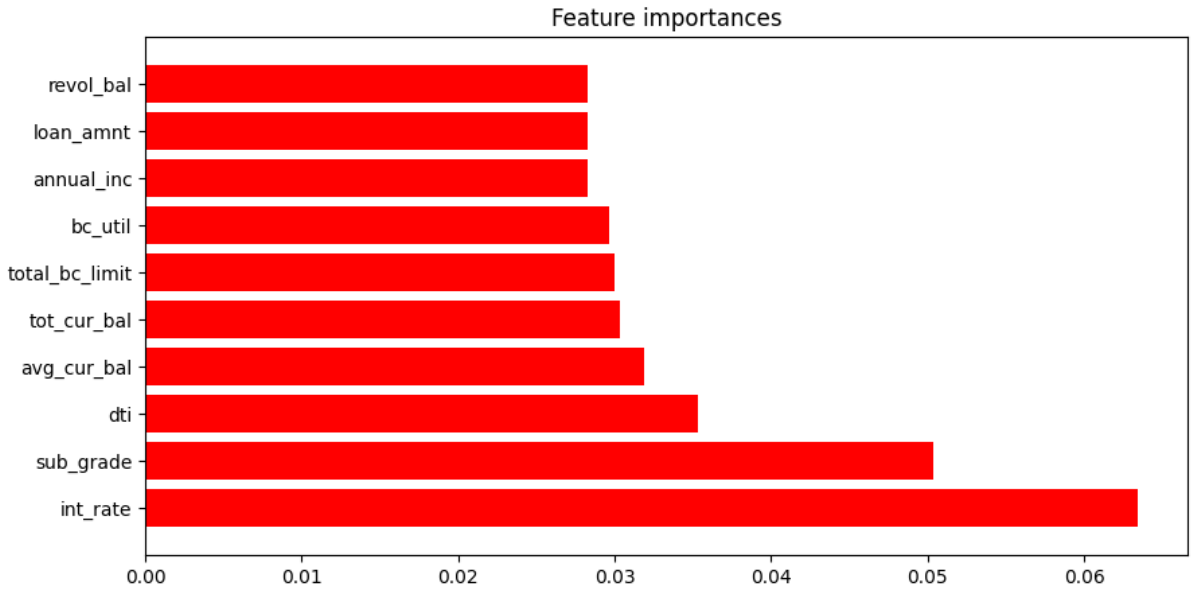


Figure 4: Variable importance diagram

As we can see, *int_rate* and *sub_grade* are the most important variables according to the model. This confirms that these are crucial features in determining the outcome of a loan. Furthermore, since these variables seem the most influential on the Random Forest model, we thought that it can be a good idea to try using them on a single Decision Tree model (1) as well, and as we already discussed, it did result in better performance.

3 Best final tree model

After comparing single Decision Tree and Random Forest models, we came to the conclusion that Random Forest models achieve better performance. For the Random Forest model, we again focused on maximizing the formula proposed by Hull [1] by trying each of the thresholds we used for single Decision Tree models and the resulting values are -10710, 460, 12720, 19220, 10020 respectively. We can again deduce that 0.8 is the optimal threshold but the obtained values are significantly larger

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.feature_importances_`

than the one found by a single Decision Tree using any of the thresholds. Table with the confusion matrix, true positive rate, and the precision for the optimal threshold can be found below.

Confusion matrix		TPR	Precision
18.23	5.22	0.5238	0.8849
36.45	40.1		

Table 5: Confusion matrix, TPR and precision for a Random Forest model with threshold = 0.8

Moreover, the AUC of the model is 0.7184, which is greater than the 0.6957 obtained by the 10-feature single Decision Tree model. The ROC curve plot is shown below.

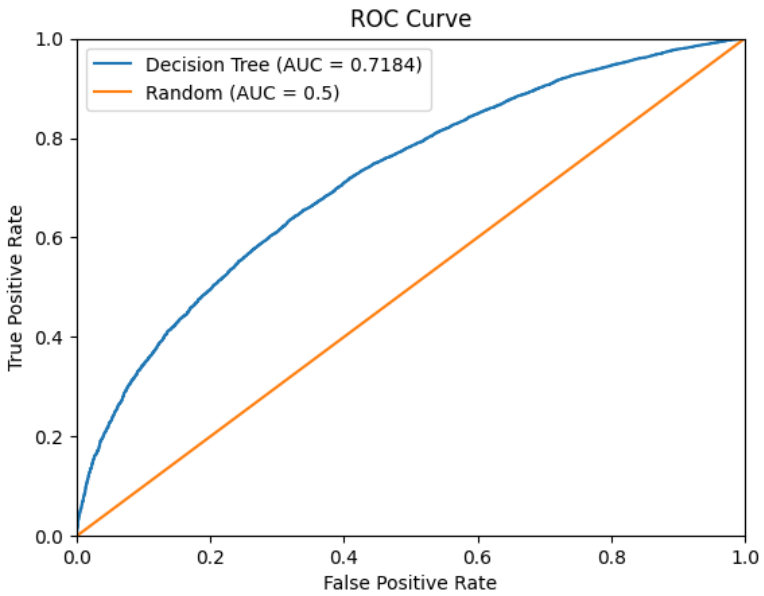


Figure 5: ROC for the Random Forest

References

[1] Hull, J. C. (2019). Machine Learning in Business: An Introduction to the World of Data Science.