Mira Kasari, mkasari
Sairathan Rajuladevi, srajulad
Kelly McManus, kellymcm
Cole Thomas, nhthomas

April 29, 2022

Case Study - Phase 3
95-828: Machine Learning for Problem Solving

Throughout this phase, we employed predictive analytics to predict how likely a loan is to default and employed predictive models to develop various investment strategies.

Part 1: Predictive Models of Default

Question 1

Carefully explain:

(i) How did you set up your model training and evaluation?
(ii) Which model hyper-parameters did you tune (for each model)?
(iii) Which performance measure(s) did you use? Report your evaluation results.

We imported all necessary packages and loaded the data from our pickle file from Phase 2. We employed feature engineering on continuous features to ensure all columns are of the same type. Our outcome column is based on 'loan_status', where the outcome is 'True' if loan_status is 'Charged Off' or 'Default', and 'False' otherwise.

We fit various classification models such as Naive Bayes, l1 regularized logistic regression, l2 regularized logistic regression, decision tree, random forest classifier, and a multi-layer perceptron. The features we tuned for each model are as follows:

- **Dummy Classifier (Baseline):** We used a Dummy Classifier as our baseline model. Whichever model we select, we want to ensure that it performs better than a random classifier. We used a "uniform" strategy for the baseline classifier to generate predictions. With this strategy, the model predicts all samples to be in the negative class and assigns an outcome value of False. There were no hyperparameters to tune for the baseline model.

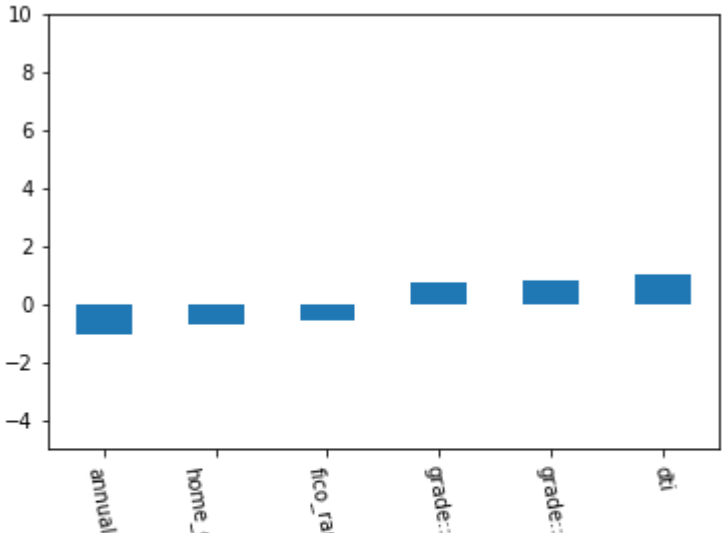
- **Naive Bayes:** We trained a Gaussian Naive Bayes model to predict if a loan defaulted. We tuned one hyperparameter for this model, "var_smoothing". The "var_smoothing" parameter adds a constant to the variances to smooth the model predictions. We tested via cross-validation all values in np.logspace(0,-9, num=100). The best performing

var_smoothing = 0.657933224657568.

- **L1 Regularized Logistic Regression:** We trained two logistic regression models. First, the L1 logistic regression, or Lasso Regression. We set the model parameters of penalty = 'l1' and solver = 'liblinear' to specify the L1 logistic regression. Using cross-validation, we tuned the hyperparameter C across the values [0.01, 0.1, 1, 10, 100, 1000]. C is a hyperparameter that has a negative impact on model regularization. A smaller value of C indicates stronger regularization. The best performing value was C = 0.1. This is a smaller value in our cross-validation parameter list and indicates that the best performing model has strong regularization.

- **L2 Regularized Logistic Regression:** We also trained a L2 logistic regression model, or Ridge Regression. The model parameters were set to penalty = 'l2' and solver = 'liblinear'. Similarly, we tuned the hyperparameter C for the L2 regression model with cross-validation across the values [0.01, 0.1, 1, 10, 100, 1000]. The best hyperparameter value was also C = 0.1 indicating a model with strong regularization.

  For the L2 model, we plot the top 3 features with the most positive and negative weights. This is seen in the graph below. The features with the top positive weights are dti, grade::E, grade::G. Having positive weights, these features indicate that a sample is more likely to be classified into the positive class, the loan was defaulted. This makes sense as loans with lower grades are riskier and a high debt-to-income ratio indicates that an individual has previous financial instability. On the other hand, the features with the top negative weights are fico_range_low, home_ownership::MORGAGE, and annual_inc. Large values for all three of these features indicating that a sample is most likely to be classified into the negative class. This aligns with the fact that having a high income, having a mortgage and having a high fico score indicate financial stability.



- **Decision Tree Classifier:** We fit a decision tree tree to provide an intuitive option for loan classification. Using cross-validation, we tuned four hyperparameters:
    - *Max_depth:* This hyperparameter specifies the maximum depth of the decision tree. We test values the values [3, 5, 7, 9] for the parameter. The value of this parameter in the best performing model evaluation is max_depth = 3. This will produce a generalized decision tree model.
    - *Criterion*: The criterion specifies the method of measuring Gain in the decision tree splits. We cross-validated the values ['gini', 'entropy']. Entropy was selected as the best measurement for split Gain for this hyperparameter.
    - *Min_sample_split*: This hyperparameter indicates the minimum number of samples in a node needed to split the node further into a deeper tree. We tested values in the range of 0-10 and 2 was identified as the best value for min_smaple_split. This indicates that a node must have a minimum of 2 nodes to be split. Combined with max_depth, max_depth is being used as the main pruning technique for our decision tree.
    - *Min_sample_leaf:* This hyperparameter indicates the minimum number of nodes required to be at a leaf node. We tested values between 1-5 and min_samples_leaf = 1 performed best. This indicates the tree would be fully grown, however because there is a max_depth = 3, the tree would not get to this point.

- **Random Forest Classifier:** We also tuned a Random Forest Classifier, which is an ensemble model of decision trees. We tuned the same four hyperparameters as the decision tree.
    - *Max_depth:* We conducted cross-validation across the values [3, 5, 7] and resulted with the best performing value of max_depth = 7.
    - *Criterion*: We test the same techniques for split as decision tree and criterion = ‘gini’ performed better on the random forest classifier.
    - *Min_sample_split*: This hyperparameter was tested with values in the range of 0-10. A value of min_sample_split = 9 was selected as best indicating a node requires 9 samples to be split.
    - *Min_sample_leaf:* Similarly, we tested values in the range of 1-5 for min_sample_leaf hyperparameter and a value of 2 was selected.

  The Random Forest Classifier developed into a more complex tree compared to the pruned decision tree.

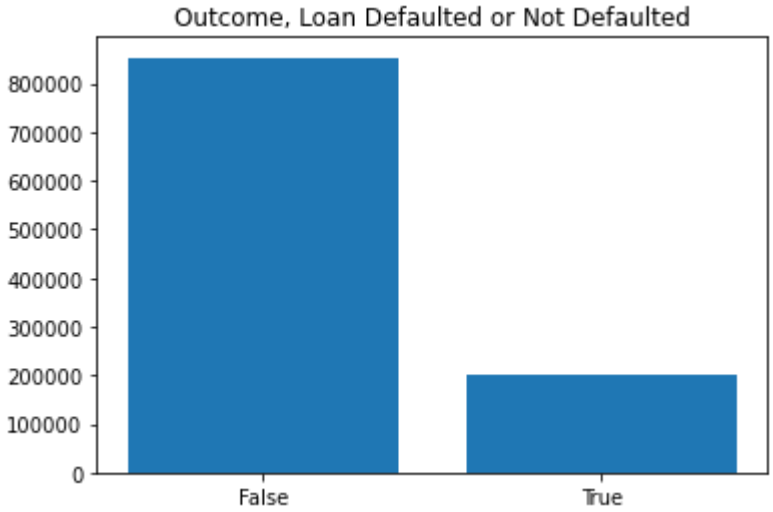
- **Multi-Layer Perceptron:** Lastly, we tuned a Multi-Layer Perception Classifier with five hyperparameters. Neural networks are very sensitive to tuning, therefore it was necessary to conduct cross-validation to ensure good performance. We tuned the following hyperparameters:

- Activation: This parameter is the activation function that is used at each layer in the neural network. We tested ['tanh', 'relu', 'logistic'] and ‘logistic’ was selected as the best activation function.
- Alpha: The alpha value is a penalty parameter. We tested [0.0001, 0.05] and 0.0001 was selected as best performing indicating that a small penalty is applied in the network for regularization.
- Hidden_layer_sizes: This hyperparameter specifies the number of neurons in a hidden layer. We tested [(10,30,10),(20,)] and a value of (20, 0) performed best.
- Learning_rate: Lastly, we tuned the learning_rate, a parameter that indicates the rate at which weights are updated in the neural network. We tested values ['constant','adaptive'] and ‘constant’ learning rate performed best.

All best performing model hyperparameters can be found in the table below.

Model	Best Performing Hyperparameters
Dummy Classifier	{}
Naive Bayes	{'var_smoothing': 0.657933224657568}
L1 Logistic Regression	{'C': 0.1}
L2 Logistic Regression	{'C': 0.1}
Decision Tree	{'criterion': 'entropy' , 'max_depth': 3 , 'min_samples_leaf': 1 , 'min_samples_split': 2}
Random Forest Classifier	{'criterion': 'gini' , 'max_depth': 7 , 'min_samples_leaf': 2 , 'min_samples_split': 9}
Multi-Layer Perception	{'activation': 'logistic' , 'alpha': 0.0001 , 'hidden_layer_sizes': (20,) , 'learning_rate': 'constant' , 'solver': 'adam'}

We noticed that the dataset is imbalanced. There are four times the number of False outcome samples, than True.



Because of this imbalance, we decided to use the recall score as our performance evaluation metric. Recall is calculated by True Positives / (True Positives + False Negatives). Contrary to accuracy, recall is a nice metric to show correct classification into the positive class. Accuracy score can be misleading as it will misrepresent the number of correctly classified samples due to the models preference to classify the negative class with imbalance data. The ideal value for recall score is 1, therefore we want our model's recall to be as close to 1 as possible.

A secondary metric we used for evaluation was the Area Under the ROC Curve (AUC). An AUC of 1 indicates that all samples are perfectly classified. Therefore, we also want our model to have an AUC as close to 1 as possible.

Model	**Recall Score**	**Area Under Curve (AUC)**
Dummy Classifier (Baseline)	0.0	0.5
Naive Bayes	0.02475	0.68
L1 Regularized Logistic Regression	0.04667	0.7
L2 Regularized Logistic Regression	0.03558	0.7
Decision Tree	0.0	0.67
Random Forest Classifier	0.12171	0.7
Multi-Layer Perceptron	0.03713	0.7

Analyzing recall score and AUC across the tuned models, the Random Forest Classifier proves to perform the best amongst our models in our hypothesis space at predicting whether a loan

defaulted. We defined the Random Forest Classifier as YOUR MODEL for the remainder of our analysis.

Question 2

What are some advantages and disadvantages of using these data splitting procedures?

We split our data **randomly**, with 70% going into the training set and 30% going into the testing set. We set our default_seed to 1.

When running our models, it is desirable for our training dataset and testing dataset to cover the same dataset. Splitting the data randomly allows each observation has an equal chance of being selected and bias in each split is removed. No pre-analysis is required of the dataset to randomly split.

However, if the data does not follow a uniform distribution and is more complex, randomly splitting the data can cause an imbalance in the resulting datasets. This can lead to inaccurate model metrics, and the estimate of the model error can have a high variance.

In the Lending Club dataset, all loans have dates of when they were issued. We can use these dates to set a cut-off date that creates a training dataset that contains all loans issued before this date, and a testing dataset that contains all loans issued after this date. During cross-validation, we use the sliding window technique within the training set to create folds for each k-fold. Setting an appropriate cut-off date allows the model to train and cross-validate on a wide variety of fluctuations in the economy during different time periods in the dataset.

However, this cut-off date is very important, and extensive knowledge of the dataset and the U.S. economy is required to set an appropriate date that splits the data into time periods that are most representative of the sample. This can be very difficult, and our testing set may still vary greatly from the training set which can lower our model metrics significantly. For example, this dataset includes the year 2016 to 2019. At the end of 2019, the start of the Covid-19 pandemic began, shocking many factors of the economy, including skyrocketing unemployment rates and plummeting small business revenue. Although we don't see these astounding numbers until 2020 hits, there are still significant changes in the 2019 Q4 dataset. Setting a cut-off date in 2018 or even in the beginning of 2019 can cause the testing set to include numbers that are unfamiliar to the training set and can lead to lower metric scores.

Finally, there is the element of human error. We would need to account that these dates may not have the same guidelines to when a loan is considered issued, which can cause irregularities in the dataset.

Question 3

Carry out this investigation:

(i) *Provide a list of aforementioned features that are derived by LendingClub and any other features that correlate/reflect those.*

The features that are derived from LendingClub are as follows: grade, dti, verification_status, revol_util, loan_status (FICO range high/low derived from FICO, not LendingClub).

- *grade:* This is the assigned grade by LendingClub that signifies the loan's risk of default.
- *dti:* This is the ratio calculated using the borrower's total monthly debt payments on the total debt obligations, divided by the borrower's self-reported monthly income.
- *verification_status*: This value indicates if and how the income was verified: by LendingClub, if the source was verified, or not verified at all.
- *revol_util*: This value is the revolving line utilization rate, which is calculated by finding the amount of credit the borrower is using relative to all available revolving credit.
- *loan_status*: This value is the current status of the loan

(ii) *Fit a (L1 or L2 regularized) Logistic Regression model using only one of the features you identified in (i). What is the predictive power as compared to that for the models you trained in part 1.?*

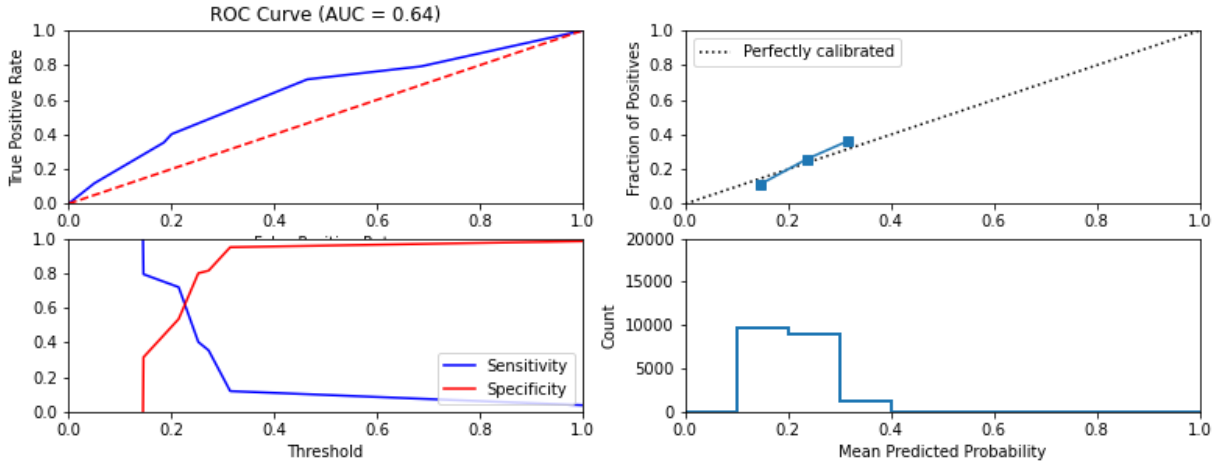
When using only the feature 'grade' to fit a L1 and L2 logistic regression with hyperparameter tuning via cross validation , the predictive power of the model, namely AUC, decreases and out resulting models both have a recall of 0% and an AUC of 0.64, compared to our best model which has a recall of 12.17 and AUC of 0.7. The AUC being higher than 0.5, or random, tells our team that a considerable portion of the predictive power of the model is related to this feature. We tested a few other features from LendingClub and saw a similar pattern. As an investor, this tells us that if we relied on this feature, it would mean that we have an upstream model, the model that LendingClub uses to derive these features, influencing our model. Our returns and default predictions could be impacted if there was an error or trading/economic regime change that the upstream model failed to capture.

```
============================================================
  Model: Only Grade - l1 Regularized Logistic Regression
============================================================
Fit time: 0.68 seconds
Optimal parameters:
{'C': 0.01}

Accuracy-maximizing threshold was: 1
Recall:  0.0
              precision    recall  f1-score   support

  No default     0.8061    1.0000    0.8926     16122
     Default     0.0000    0.0000    0.0000      3878

    accuracy                         0.8061     20000
   macro avg     0.4031    0.5000    0.4463     20000
weighted avg     0.6498    0.8061    0.7196     20000
```



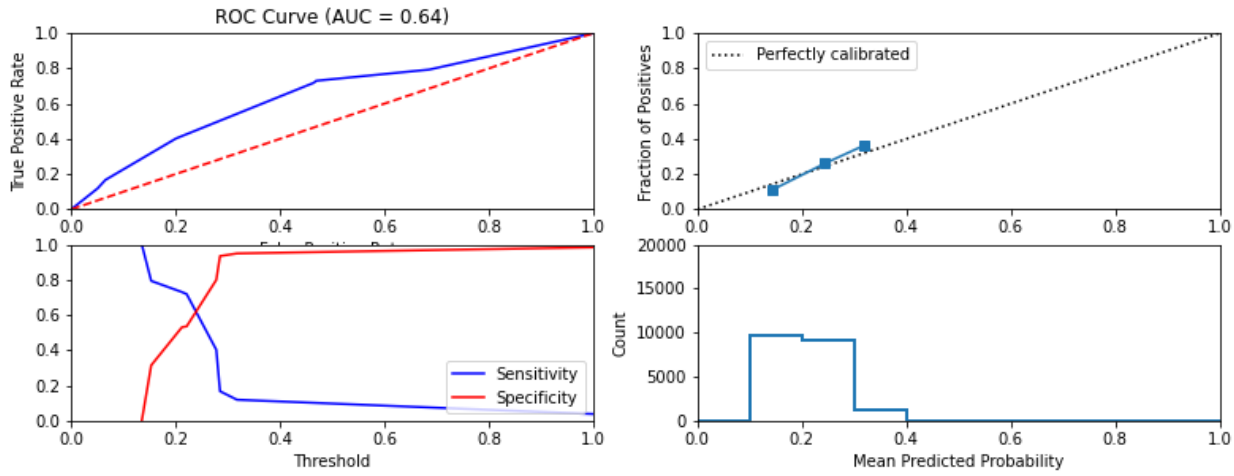

```
Similarity to LC grade ranking:  0.6833463888312478
Brier score: 0.1497074263440674
Were parameters on edge? : True
Score variations around CV search grid : 0.0
[0.80546667 0.80546667 0.80546667 0.80546667 0.80546667 0.80546667]
```

```
=========================================================
  Model: Only Grade - l2 Regularized Logistic Regression
=========================================================
Fit time: 0.48 seconds
Optimal parameters:
{'C': 0.01}

Accuracy-maximizing threshold was: 1
Recall:  0.0
              precision    recall  f1-score   support

  No default     0.8061    1.0000    0.8926     16122
     Default     0.0000    0.0000    0.0000      3878

    accuracy                         0.8061     20000
   macro avg     0.4031    0.5000    0.4463     20000
weighted avg     0.6498    0.8061    0.7196     20000
```




```
Similarity to LC grade ranking:  0.7023150890457355
Brier score: 0.1494565112766282
Were parameters on edge? : True
Score variations around CV search grid : 0.0
[0.80546667 0.80546667 0.80546667 0.80546667 0.80546667 0.80546667]
```

(iii) *Remove the features you identified in (i), refit your models onto remaining features and report new performance measure(s). What are your conclusions?*

Removing the features that were derived from LendingClub and rerunning the model, we see that random forest classifier still has the highest recall score of 0.04000, and Naive Bayes still has the lowest recall score of 0.00675, apart from the dummy classifier and random forest classifier. Each new score is approximately a third of the original scores because we averaged over 100 random splits versus only one split in the original models and removed the LendingClub derived model. Seeing as these scores and model ranks were maintained into the new models, this shows that LendingClub's statistical features had a slight impact on the scoring outcome but did not

impact the ranking of our model performance.

```
# Remove LendingClub derived features
removed_features = ['grade', 'dti', 'verification_status', 'dti', 'revol_util', 'loan_status']
your_features = set(your_features) - set(removed_features)

# Prepare the train, test data for training models with new features
data_dict = prepare_data(feature_subset = your_features)

avg_metrics_for_hundred_splits_classification(dc, data_dict, hundred_train_test_splits, 'Dummy Classifier')
avg_metrics_for_hundred_splits_classification(gnb, data_dict, hundred_train_test_splits, 'Naive Bayes Classifier')
avg_metrics_for_hundred_splits_classification(l1_logistic, data_dict, hundred_train_test_splits, 'L1-Logistic Classifier')
avg_metrics_for_hundred_splits_classification(l2_logistic, data_dict, hundred_train_test_splits, 'L2-Logistic Classifier')
avg_metrics_for_hundred_splits_classification(decision_tree, data_dict, hundred_train_test_splits, 'Decision Tree Classifier')
avg_metrics_for_hundred_splits_classification(random_forest, data_dict, hundred_train_test_splits, 'Random Forest Classifier')
avg_metrics_for_hundred_splits_classification(mlp, data_dict, hundred_train_test_splits, 'Multi-layer Perceptron Classifier')

# Prints average recall +/- standard deviation of 100 random fits
```

```
Dummy Classifier: 0.0 +/- 0.0
Naive Bayes Classifier: 0.0067534811758638465 +/- 0.005543278661864567
L1-Logistic Classifier: 0.015407426508509544 +/- 0.00805491420273299
L2-Logistic Classifier: 0.01594636410520887 +/- 0.007461855740788212
Decision Tree Classifier: 0.0 +/- 0.0
Random Forest Classifier: 0.04000773594636411 +/- 0.007651713931465534
Multi-layer Perceptron Classifier: 0.015843218153687465 +/- 0.007427611516278324
```

Question 4

Moving forward, pick the best-performing model in part 3. We will refer to it as Your Model. After modifying YourModel to ensure you did not include any features calculated by LendingClub, you want to assess the extent to which YourModel's scores agree with the grades assigned by LendingClub. How can you go about doing that? What is your observation?

grade	Empiracle Pct Default	Predicted Pct Default
A	6.586345	0.000000
B	13.652633	0.000000
C	22.370100	1.257976
D	29.518267	7.371484
E	36.078431	16.392157
F	43.691589	23.364486
G	37.606838	30.769231

To see if our model's score agrees with the grades from LendingClub, we first needed to obtain the empirical default rate by grade in the dataset. We see that as grade increases (moves towards Z), the percentage of default increases accordingly. We see one exception to this rule where F is greater than G, however, this could be due to our sampling approach and size. Next, we needed to obtain our predicted percentage of default to compare against the empirical baseline. For this, we used our models classification output of default vs. not-defaulted. After this step, we merged these two data frames together to compare the output.

Our model, while having high accuracy, suffered from poor recall scores. Because we did not use SMOTE, oversampling, or undersampling, the model could obtain a high accuracy by simply predicting that the loans did not default as evidenced by our lower than empirical predicted default rates. For classes A-B, as an investor, our team felt reasonably confident in these scores. For grades C-F, our model did not perform well. We grossly underestimated the predicted default rates which could result in serious underperformance in a real-world scenario. One notable exception to this rule is grade G where we predicted ~7% below the empirical default rate. This presents a unique opportunity to our strategy.

We know that grade G yields the highest of any investment available from LendingClub and our original investment guidelines recommended that we invest in assets outside of traditional stocks and bonds. Since we can reasonably predict defaults in this asset class, we could tune our strategy to favor a mix of safer investments (A-B) with high yield assets, namely G.

If time allowed, one additional item we could have considered was creating our own scoring system or internal risk calculation, similar to what is done at firms like Moody's, S&P, and Fitch by the borrower credit data embedded in the original dataset (ex. FICO score, revolver balance, revolver utilization, number of open credit lines, credit utilization just to name a few). For example, if FICO was lower than 700, revolver utilization was 70% and number of credit lines was greater than 8, we could engineer a binary feature flag for high risk individuals.

Question 5

Next you will assess the stability of YourModel over time. To this end, analyze whether YourModel trained (using the Random data splitting procedure in part 2. for cross validation) in 2010 performs worse in 2017 than YourModel trained on more recent data 2016. What conclusion can you draw? Is your model stable?

Our model is not stable over time. In 2010 our recall score was 0.13 while in 2016 our model had a recall score of 0.20. The conclusion we can draw from this is that more recent training data performs better than aged data. If our model enters a production environment, we would need to be conscious of this and continually monitor and update our model. Model stability over time is crucial for a wide variety of applications, especially if there are financial impacts. Notably, our model with 2016 data outperformed our model fitted with random samples of data from earlier in this case study. Since recall is paramount in classification problems, the conclusion our team could draw is that our team should strongly consider changing the input data to match current borrower conditions.

A next step our team should take before deploying to a production environment is backtesting on different years. While we did randomly sample data in the first part of the case, would our 2016 data perform better in 2011 than a model trained with 2010 data? There are seemingly endless

ways to analyze a model's performance, however, we can say for certain that training on more recent data resulted in better recall performance.

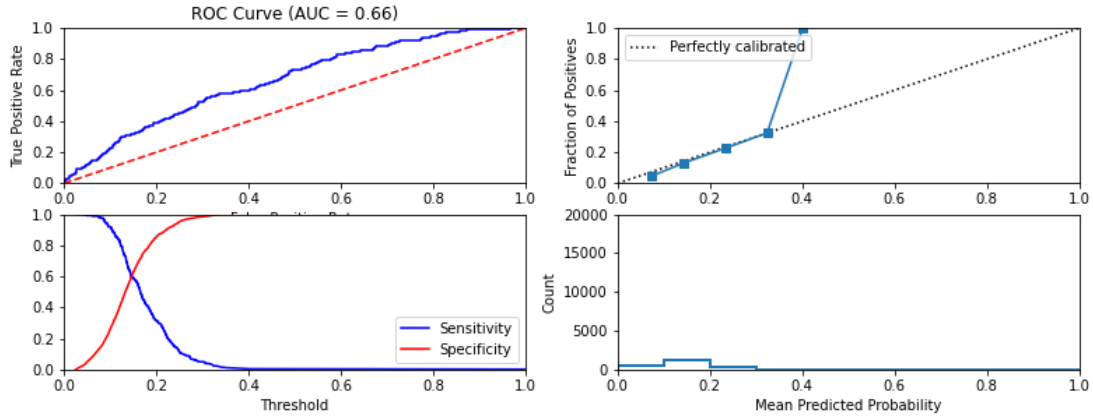
It's commonplace to perform this type of benchmarking in industry and corporate model risk teams request this type of analysis before certifying a model. Changes in prevailing interest rates, economic conditions, and wage growth are a few common metrics that could result in regime change and would require further model tuning and evaluation.

```
============================================================
  Model: Random Forest (2010)
============================================================
Fit time: 2.37 seconds
Optimal parameters:
{}

Accuracy-maximizing threshold was: 0.2469472876711781
Recall:  0.1274131274131274
              precision    recall  f1-score   support

  No default     0.8791    0.9437    0.9102      1741
     Default     0.2519    0.1274    0.1692       259

    accuracy                         0.8380      2000
   macro avg     0.5655    0.5356    0.5397      2000
weighted avg     0.7979    0.8380    0.8143      2000
```

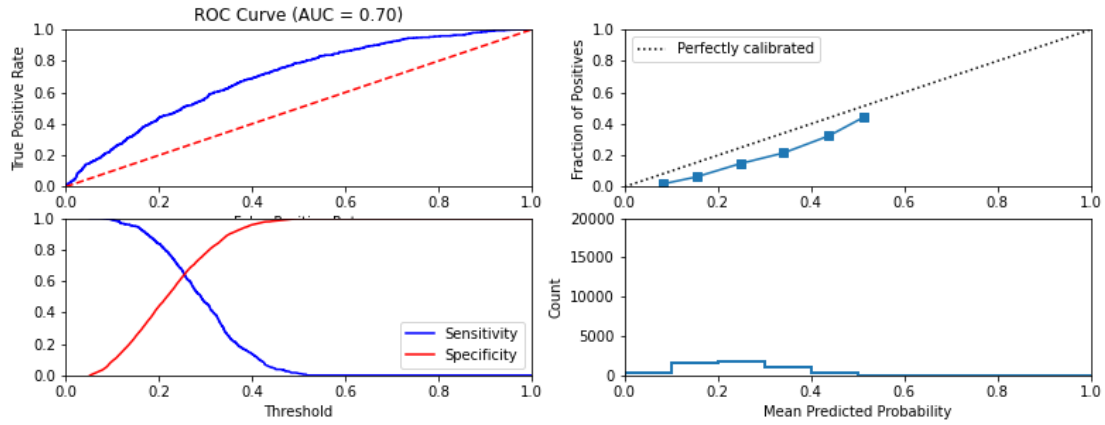


```
============================================================
  Model: Random Forest (2016)
============================================================
Fit time: 1.89 seconds
Optimal parameters:
{}

Accuracy-maximizing threshold was: 0.3684070747240593
Recall:  0.19790104947526238
              precision    recall  f1-score   support

  No default     0.8818    0.9208    0.9009      4333
     Default     0.2779    0.1979    0.2312       667

    accuracy                         0.8244      5000
   macro avg     0.5798    0.5594    0.5660      5000
weighted avg     0.8012    0.8244    0.8115      5000
```



Question 6

Now go back to the original data (before cleaning and feature selection) and fit YourModel to predict the Default likelihood using all of the features. Does anything surprise you about the performance of this model (averaged on out-of-sample test datasets) compared with the other models you have fit earlier?

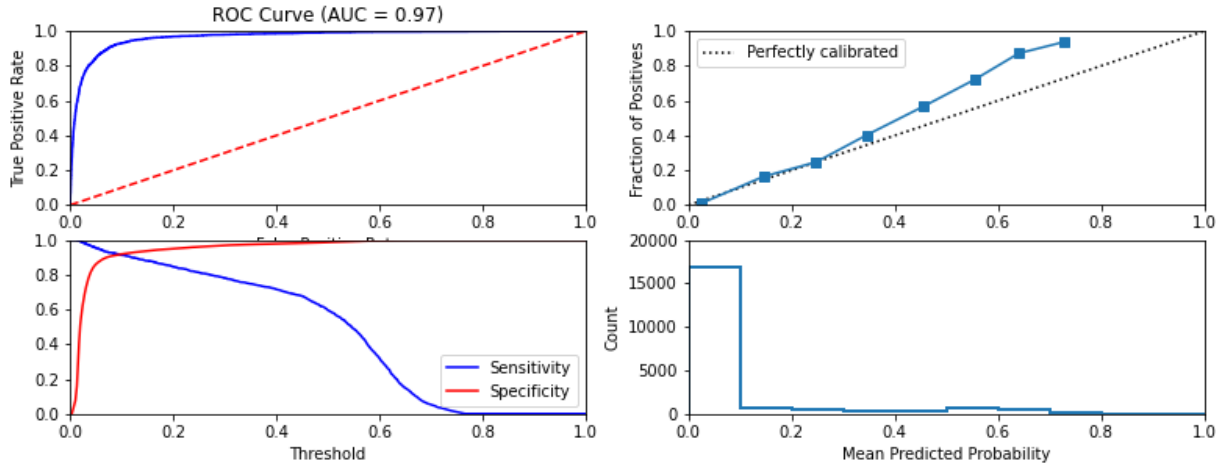
What is surprising to us is that the performance is so high. Well really, it isn't that much of a surprise because of data leakage, however, a recall of 0.60 is outstanding for our use case compared against our baseline of 0.12. One reason for this performance is that we cut out about one million records that were still current loans through our cleaning and outlier detection process from Phase 2. The additional records and additional features may be misleading our model. They do not provide value to the core goal of the model and so in training, provide excess noise that may mislead the model or are indicative of the loan's future default state. Overall, leaving the data in the original state will mislead us to believe that we have fit a good model. After considering the role of each feature and data values and conducting feature engineering, we developed a model that focuses more on the goal of predicting defaulted loans even though it has a lower recall score.

```
============================================================
  Model: Random Forest (Original Data)
============================================================
Fit time: 5.33 seconds
Optimal parameters:
{}

Accuracy-maximizing threshold was: 0.4837927710450953
Recall:  0.6244158878504673
              precision    recall  f1-score   support

  No default     0.9655    0.9845    0.9749     18288
     Default     0.7901    0.6244    0.6976      1712

    accuracy                         0.9536     20000
   macro avg     0.8778    0.8044    0.8362     20000
weighted avg     0.9505    0.9536    0.9512     20000
```




```
Similarity to LC grade ranking:  0.3180748874544905
Brier score: 0.03650054846948019
Were parameters on edge? : False
Score variations around CV search grid : 0.0
[0.92176667]
{'model': RandomForestClassifier(max_depth=7, min_samples_leaf=2, min_samples_split=9,
```

Part 2: Investment Strategies

Question 7 - Sai

First, build the three regression models described above: (1) regressing against all returns, (2) regressing against returns for defaulted loans, and (3) regressing against returns for nondefaulted loans. In each case, use each one of the four return variables you calculated in Phase II as your target variable (recall M1, M2, M3(2.3%), and M3(4%)) and try (L1 and L2 regularized) linear regression, random forest regression, and multi-layer NN regression. Report the performance results in corresponding entries in Table 3.1. Do they perform well? Can you tell?

We fit various regression models such as l1 regularized linear regression, l2 regularized linear

regression, a multi-layer perceptron regressor, and a random forest regressor. The performance measure we used was accuracy.

- **L1 regularized linear regression model**: We trained a linear regression model with Lasso regularization. We tuned one hyperparameter using cross-validation.
    - *alpha*: This parameter multiplies the L1 term, and the value 0 is not recommended. Using cross-validation, we tested the possible values [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100].

- **L2 regularized linear regression model:** We trained a linear regression model with Ridge regularization. We tuned one hyperparameter using cross-validation. So that none of our parameters were on the edge, we continuously updated the upper bound in order to find the optimal value.
    - *alpha*: This parameter indicates the regularization strength and must be a positive value. Using cross-validation, we tested the possible values with the possible values [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100, 200, 300, 400, 500, 600, 1000, 1100, 1200, 1300, 1400, 1500].

- **Multi-layer perceptron regressor:** We trained a multi-layer perceptron regressor which was tuned with two hyperparameters using cross-validation.
    - *activation*: This parameter indicates the activation function to be used for the hidden layer. Using cross-validation for all hyperparameters, the possible 'activation' values we tested are 'identity', 'logistic', 'tanh', and 'relu'. The identity function is the no-op activation function. The logistic function is the logistic sigmoid function. The tanh function is the hyperbolic tan function. The relu function is the rectified linear unit function.
    - *alpha:* This parameter indicates the L2 regularization penalty. The possible 'alpha' values we tested are 0.0001, 0.001, 0.01, 0.1, and 1.
    - *hidden_layer_sizes*: This value is a tuple where the ith element corresponds to the number of neurons in the ith hidden layer. The possible value are (10, 30, 10) and (20,).
    - *learning_rate:* This value indicates the learning rate schedule for weight updates. The possible values are 'constant' and 'adaptive'. 'Constant' is a learning rate that remains constant throughout the model. 'Adaptive' keeps the learning rate constant, but only as long as the training loss keeps decreasing. If it decreases, the learning rate is divided by 5.
    - *solver:* This value indicates the solver that will be used for weight optimization. The possible values are 'sgd' and 'adam.' 'SGD' is the stochastic gradient descent-based solver. 'Adam' is also a stochastic gradient descent-based solver,

however this works well on large datasets versus ‘lbfgs’ which a quasi-Newton method optimizer.

- **Random forest regressor:** We trained a random forest regressor and tuned three hyperparameters using cross-validation. Although unsuccessful, we continuously updated the bounds so as to keep the parameters away from the edge as much as possible.
    - *n_estimators:* This parameter indicates the number of trees in the forest.
    - *max_depth*: This parameter indicates the maximum depth of the tree. We tested the possible ‘n_estimators’ values of [10, 100]. The possible ‘max_depth’ values we tested were [None, 3, 5].
    - *min_impurity_decrease:* This parameter indicates the threshold for when a node will be split, if the split leads to a decrease of impurity less than this value. The possible ‘min_impurity_decrease’ values we tested were [0.0, 1.0, 3.0].

All of our best hyperparameters can be found in the table below:

	ret_PESS	ret_OPT	ret_INTa	ret_INTb
L1 Linear Reg.	{‘a’: 0.001}	{‘a’: 1e-8}	{‘a’: 0.001}	{‘a’: 0.001}
L2 Linear Reg.	{‘a’: 300}	{‘a’: 5}	{‘a’: 1000}	{‘a’: 200}
Multi-Layer	{‘activation’: ‘tanh’, ‘a’: 0.001, ‘hidden_layer_sizes’: (10, 30, 10), ‘learning_rate’: ‘constant’, ‘solver’: ‘adam’}	{‘activation’: ‘logistic’, ‘a’: 0.0001, ‘hidden_layer_sizes’: (20,), ‘learning_rate’: ‘constant’, ‘solver’: ‘adam’}	{‘activation’: ‘tanh’, ‘a’: 0.01, ‘hidden_layer_sizes’: (10, 30, 10), ‘learning_rate’: ‘constant’, ‘solver’: ‘adam’}	{‘activation’: ‘relu’, ‘a’: 0.1, ‘hidden_layer_sizes’: (20,), ‘learning_rate’: ‘adaptive’, ‘solver’: ‘adam’}
Random Forest	{‘n_estimators’: 150, ‘max_depth’: 7, ‘min_impurity_decrease’: 0.0}	{‘n_estimators’: 200, ‘max_depth’: 7, ‘min_impurity_decrease’: 0.0}	{‘n_estimators’: 150, ‘max_depth’: 5, ‘min_impurity_decrease’: 0.0}	{‘n_estimators’: 150, ‘max_depth’: 7, ‘min_impurity_decrease’: 0.0}

Performance metrics for our models can be seen in the table below. The performance metric that was used was the R^2 score which reports the amount of variation from the independent variable that explains the dependent variable. Keep in mind that these metrics are averaged over a hundred splits of data.

	M1	M2	M3 (2.3%)	M3 (4%)
L1 Linear Reg	0.01225	0.02983	0.02359	0.03049
L2 Linear Reg	0.02572	0.02975	0.02475	0.03208
Multi-Layer	0.01161	0.02307	0.02617	0.03701
Random Forest	0.03189	0.0364	0.03309	0.04228

For the next step, we computed the R^2 values for the separated default and non-default loans. This was done through the 'separate' argument in the fit_regression function.

Shown below are the R^2 values predicted for the models trained on **defaulted loans**.

	M1	M2	M3 (2.3%)	M3 (4%)
L1 Linear Reg	0.28564	0.71379	0.33050	0.18470
L2 Linear Reg	0.28162	0.71017	0.36942	0.19283
Multi-Layer	0.23855	0.69310	0.33302	0.21799
Random Forest	0.36959	0.71375	0.45543	0.29448

Shown below are the R^2 values predicted for models trained on **non-defaulted loans**.

	M1	M2	M3 (2.3%)	M3 (4%)
L1 Linear Reg	0.13412	0.13292	0.05432	0.06937
L2 Linear Reg	0.13839	0.13735	0.05830	0.07328
Multi-Layer	0.10220	0.10213	0.05643	0.07074
Random Forest	0.13517	0.13282	0.06385	0.07880

R^2 scores of 0.7 and above generally show a good correlation between the two variables. For the models run on the unseparated data, all of our R^2 scores for each column for each model are extremely low- below 0.05. This means that less than 5% of each dependent variable predicts our outcome variable and thus our models are not necessarily performing well from a statistical point of view. On the separate defaulted loans, we can see that all the R^2 values are much higher, with the M2 column having around an average value of 0.7. This indicates that our model is

much better at making return predictions on loans that have defaulted, which makes sense as the returns for these loans may be subject to lesser variation in the explanatory variables, such as with lower than average returns.

From our analysis, the Random Forest Regressor is our best model among all four of these models.

Question 8

Next, implement each of the investment strategies described above using the best performing regressor from part 7.

(i) Suppose you were to invest in 1000 loans using each of the four strategies, what would your returns be? Average your results over 100 independent train/test splits.

The following are the resulting return rates for each strategy averaged over 100 independent train/test splits for each strategy and return method.

```
strategy: Random
ret_PESS: 0.004481863525567513
ret_OPT: 0.05675962800977735
ret_INTa: 0.43798568971741547
ret_INTb: 1.298731371235703

strategy: Default-based
ret_PESS: 0.002962581117136989
ret_OPT: 0.05692440731943666
ret_INTa: 0.4364921645154196
ret_INTb: 1.3048948535896918

strategy: Return-based
ret_PESS: 0.018773859764228358
ret_OPT: 0.056912631578948034
ret_INTa: 0.4377107940070313
ret_INTb: 1.2925903478397163

strategy: Default-return-based
ret_PESS: 0.029130493945367677
ret_OPT: 0.05592992946779278
ret_INTa: 0.4387829196720559
ret_INTb: 1.2932841270566036
```

(ii) Include the best possible solution (denoted Best) that corresponds to the top 1000 performing loans in hindsight, that is, the best 1000 loans you could have Picked. Fill in the corresponding entries in the table below.

Strategy	M1	M2	M3 (2.3%)	M3 (4%)
Random	0.00448	0.05676	0.43799	1.29873
Default	0.00296	0.05692	0.43649	1.30489
Return	0.01877	0.05691	0.43771	1.29259
Default-Return	0.02913	0.05593	0.43878	1.29328

Best	0.09924	0.23048	0.61313	1.85667

```
strategy: Best
ret_PESS: 0.09924090075017031
ret_OPT: 0.23047947276628133
ret_INTa: 0.6131312179013871
ret_INTb: 1.8566677801674383
```

(iii) Based on the above table, which data-driven investment strategy performs best? What can you tell about using the Random strategy? Does it cause you any loss? Why do you think that is the case? How do the data-driven strategies compare to Random as well as BEST?

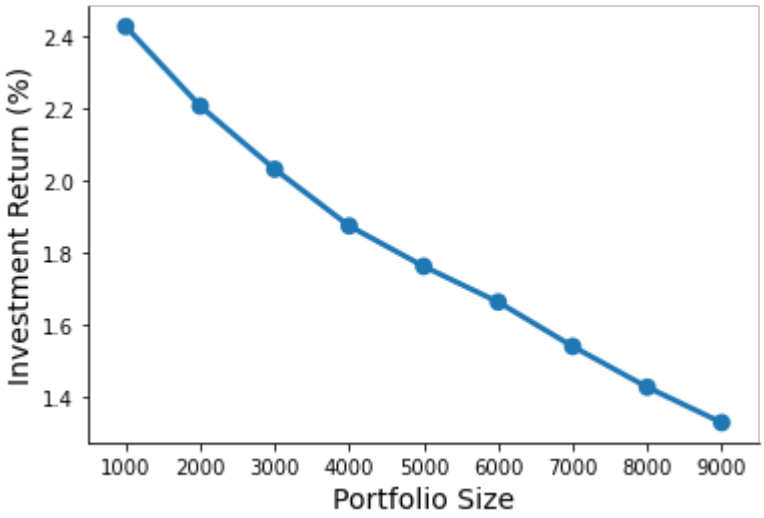
Based on the table above, the default-return based strategy and default strategy perform the best. The default-return strategy performs the best on the M1 and M3 (2.3%) methods, with the return strategy coming in as second best for M1 and the random strategy coming in second best for M3 (2.3%). The default strategy performs the best on the M2 and M3 (4%) methods, with the return strategy coming in second best for M2 and the random strategy coming second best for M3 (4%).

The random strategy did surprisingly much better than we expected. Although it was not the best strategy for any method, it sometimes outperformed our data-driven models. It outperformed the default strategy for M1, the default-return strategy for M2, the return strategy and default strategy for M3 (2.3%), and the default-return strategy and return strategy for M3 (4%). The 'best' strategy is the best by a significant amount for each method.

The random strategy returns are all positive and do not cause us any loss. In the random strategy, each loan has an equal chance of being selected, regardless of its expected or predicted return, which allows us to include 'bad' loans that we would otherwise not consider. According to Investopedia (2020), a higher level of default risk can lead to a higher return, and leads us to setting higher interest rates in an attempt to offset the default risk. Seeing how this strategy is actually comparable to the other strategies, the positive returns show us that LendingClub has set higher interest rates to balance the default risk.

Question 9

The strategies above were devised by investing in top 1000 loans. You are worried, however, that if you wanted to increase the number of loans you wished to invest in, you would eventually "run out" of good loans to invest in. Test this hypothesis using the best-performing data-driven strategy from part 8. Specifically, plot the return (using the M1 return calculation, averaged over 100 runs) versus your portfolio size (i.e., number of loans invested in). What trend do you observe? Why do you think that is the case?



We see that as we choose the best loans and subsequently add more bonds to our portfolio, our total return decreases. From the data we know that as we add additional loans into our portfolio that are not as investment worthy, they have a higher likelihood of default. This will eat away at our total return as we are saddled with unreturned principal and interest. From the chart we know that the portfolio with 1,000 loans had the highest return and then our total return fell gradually. Perhaps when this strategy is implemented, starting from a portfolio size of 100 loans would be a better strategy.

Outside of simply return based calculations on historical data, we can broaden our horizons and look at issues relating to the market itself. We identified the following pain points as an investor that compounds our portfolio size problem mentioned above: 1. single issuer and market maker - supply shortage and 2. limited product offerings.

1. Single issuer and market maker - supply problem

Since we are only dealing with one firm and one market maker, we are at the mercy of their offerings. In this LendingClub case, our return woes could be compounded by lack of available next best high quality investments. For instance, LendingClub is competing in a saturated market with well known firms such as AmericanExpress, Capital One, Kiva, SoFi, and Prosper just to name a few. Would a person seeking a loan start with LendingClub? Maybe they will, maybe they won't, however, this certainly factors into the supply side of the market. If high quality borrowers would choose LendingClub as their first stop, perhaps the slope of our return wouldn't be so steep.

2. Limited product offerings

In traditional capital markets, there is no shortage of debt instruments. Wall Street firms even engineer their own asset backed structured products to satisfy the needs of market participants. In this case, LendingClub offers only a few categories of product. If they were to finance other

kinds of debt instruments the universe of available products and investment opportunities would increase. S&P's Market Intelligence product does a great job at ranking all various kinds of debt instruments and LendingClub could take inspiration from their exotic product offerings.

References

Kagan, Julia. (2020). *Default Risk*. Investopedia. https://www.investopedia.com/terms/d/defaultrisk.asp