Gideon Blinick

Capstone 1: Final Report

Problem Statement

Goal

The goal of this project is to effectively predict credit card payment default using demographic factors, credit data, repayment statuses, bill statements, history of payments, and a variety of data mining and machine learning techniques.

To do this, we will be using the "Default of Credit Card Clients Dataset" available on Kaggle and the UCI repository. This dataset was the subject of a 2009 paper in the journal *Expert Systems with Applications* called "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients". In that paper, the authors use 6 data mining techniques to predict credit card default. Their use of data mining is motivated by the desire to answer 2 questions:

1) Is there are difference of classification accuracy among the six data mining techniques?
2) Could the estimated probability of default produced from data mining methods represent the real probability of default?

This project will focus on answering the first of those questions, with 2 subtle adjustments. First, while the authors of the paper used Artificial Neural Networks and Discriminant Analysis as 2 of their techniques, we will use Support Vector Machines and Random Forests; and second, while the authors used classification *accuracy* as an important metric, we will instead use classification *recall*. This is motivated by the problem context, as will be explained.

Client Interest

The client for this project is, naturally, credit card companies. Such companies have a significant interest in predicting which customers will default on their payments because such defaults cost them money, and thus, they would rather not extend money to individuals with a high probability of default. A good prediction model will enable them to lend to good customers.

A good prediction model will also enable credit card companies to make early and effective interventions with existing customers who are likely to default. Such interventions may take the form of debt and financial counseling, enabling the customer to maintain good credit and enabling the company to receive payment. In worse circumstances, an intervention may allow credit card companies to be paid ahead of other creditors.

In summary, credit card companies are the primary stakeholder in this project, the decision being improved is the acceptance and rejection of credit applications as well as the decision of who to extend interventions to, and improvement of the decision results in greater profit for the company.

It is worth mentioning that this problem is not too much different from other problems such as bankruptcy prediction, and that a solution to one problem might generalize easily to the other. Hence, other stakeholders to this problem could include banks and other large creditors.

Dataset Description

Overview

From Kaggle: “This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.”

There are 25 variables:

ID: ID of each client

LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit

SEX: Gender (1=male, 2=female)

EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)

MARRIAGE: Marital status (1=married, 2=single, 3=others)

AGE: Age in years

PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)

PAY_2: Repayment status in August, 2005 (scale same as above)

PAY_3: Repayment status in July, 2005 (scale same as above)

PAY_4: Repayment status in June, 2005 (scale same as above)

PAY_5: Repayment status in May, 2005 (scale same as above)

PAY_6: Repayment status in April, 2005 (scale same as above)

BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)

BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)

BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)

BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)

BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)

BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)

PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)

PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)

PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)

PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)

PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)

PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)

default.payment.next.month: Default payment (1=yes, 0=no)

How the Data Was Obtained

This dataset was obtained from Kaggle (https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset/home) .

It was sourced from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients) and the paper referenced earlier can be found here: https://bradzzz.gitbooks.io/ga-dsi-seattle/content/dsi/dsi_05_classification_databases/2.1-lesson/assets/datasets/DefaultCreditCardClients_yeh_2009.pdf.

Data Cleaning and Wrangling

Since the dataset was loaded from Kaggle, it was fairly clean. Nevertheless, inconsistencies were found in the data that needed to be corrected.

First, the data was examined for missing or Null values using the Pandas .info() method. No columns had Null values, and values of 0 were part of the domain in the columns in which they were found, so nothing needed to be done.

Next, outliers in the columns were examined. It was determined that no values should be excluded from the dataset because although there were plenty of values that met the statistical definition of being an outlier (greater than the 3rd quartile value + 1.5 times the interquartile range or less than the 1st quartile value minus 1.5 times the interquartile range) these values all made sense in context and there was no evidence that values had been entered incorrectly, or that the data was corrupt in any way.

And yet, some values did need to be changed because they violated the description of the dataset. There were three instances where this occurred.

In the first instance, it was discovered that the PAY_X columns (the columns determining how many months a customer was behind on credit) contained values of -2 and 0 when they should only have contained values of -1 (representing no months behind on repayment) and positive integers (representing the number of months behind payment for whom this applied). Given the definition of the column, it doesn't make sense to have negative values (because you cannot be ahead of bills, only behind or on time). Therefore, the decision as made to replace all negative values in these columns with 0.

In the second instance, it was discovered that the Marriage column contained values of 0, when it should have only contained values of 1 for "Married", 2 for "Single", or 3 for "Other". The logical decision was made to recode these values as 3, since the values of 0 and 3 effectively were 2 values for "Other".

Finally, in the third instance, there were values of 0 for Education which the dataset description did not account for. Furthermore, there was a column for "Other" and 2 values for "Unknown". Since all of these values represent essentially the same thing, it was decided to group all of these values under one

coding. Therefore, values of 0 (unaccounted for in the dataset description), and 5 and 6 (both values for “unknown”) were recoded to values of 4 representing “Other”.

Additionally, it was also the case the PAY_0 column was awkwardly named, since the other PAY_X columns had values in range(2, 6) inclusive. Therefore, the column was renamed to PAY_1, which also made it conform to the convention used to name the BILL_AMTX and PAY_AMTX columns.

Exploratory Data Analysis

There were 2 main components to the Exploratory Data Analysis stage of this project. First, correlations between our variable of interest or dependent variable, 'default' were examined. And second, correlations and relationships between various independent variables were looked into.

For the calculation of the correlation between default and other variables, a simple Pearson correlation was calculated for each of the variables and default, and their results were ordered from highest to lowest. It was discovered that none of the variables had significant or even moderate correlation (defined as having an absolute correlation values greater than 0.5), but that the LIMIT_BAL variable (measuring the credit extended to a customer) and the PAY_X variables (representing repayment status over 6 months) were most correlated with default. In the case of LIMIT_BAL, it was more negatively correlated with default than any other variable, having a correlation value of about -0.15, and in the case of the PAY_X variables it was found that they had correlations ranging from about 0.24 for PAY_6 all the way to about 0.4 for PAY_1 with the strength of the correlation increasing with later time (that is, lower X in PAY_X).

In the case of the PAY_X variables, it made sense that they would be more strongly correlated with default than any other variable or variable group, since they measure repayment status which can intuitively be understood as being very related to default. Furthermore it made sense that the correlation would increase with time since being behind on repayments in September should be more correlated with defaulting in October than being behind on repayments in April.

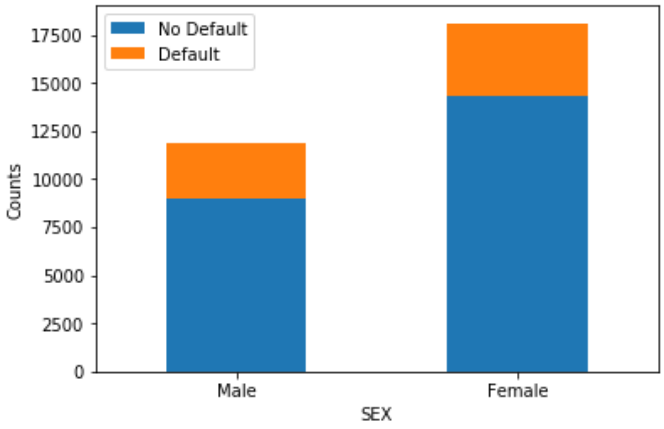
In the case of the LIMIT_BAL variable and the reason for the negative correlation with default, a working hypothesis is that the credit agency only extended more credit to individuals who they were more confident could pay back. Therefore, individuals with more credit should have had greater repayment abilities, and thus lower rates of default.

As for the second element of Exploratory Data Analysis, that of finding relationships among variables, a number of relationships were discovered, but only the top 4 will be mentioned for brevity.

They are:

1) Men default at a higher rate than women

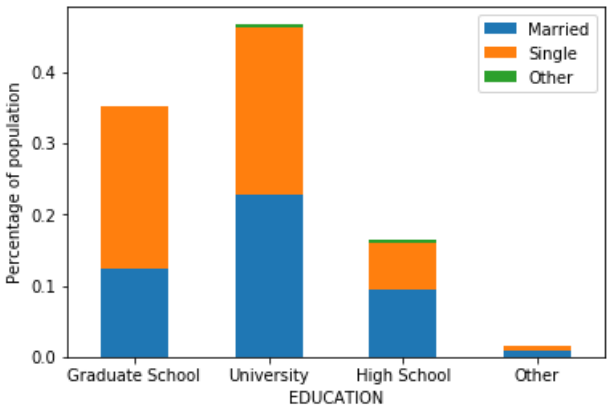
This can be seen in the following chart:



And table:

default	No	Yes	Percentage Default
SEX			
Male	9015	2873	24.2%
Female	14349	3763	20.8%

2) People with only a high school education have higher rates of marriage than people with graduate school education.

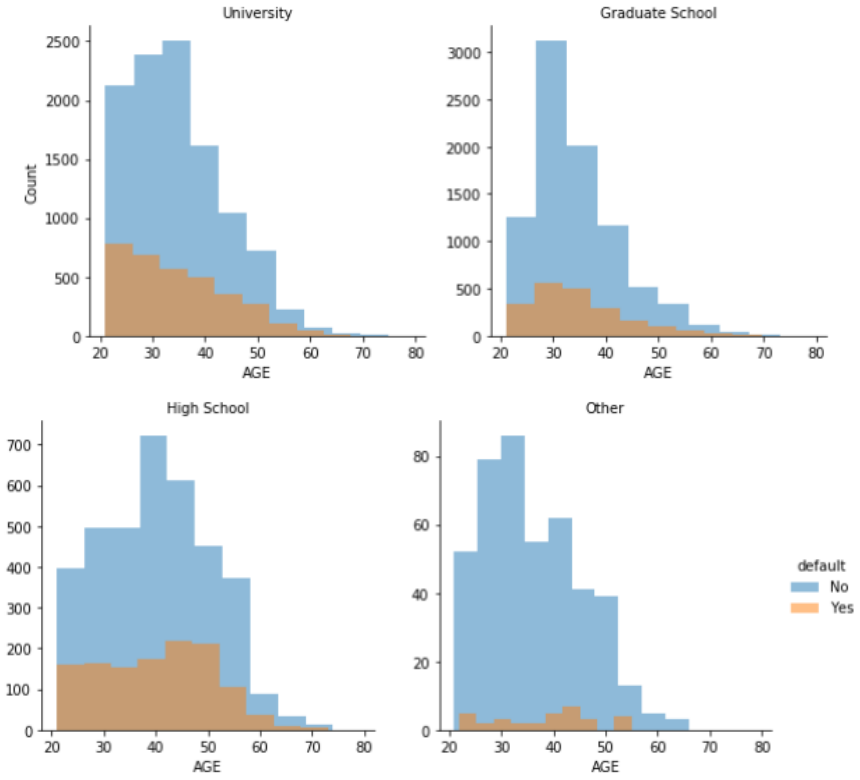


MARRIAGE	Married	Single	Other	Total
EDUCATION				
Graduate School	35.2%	64.3%	0.5%	100.0%
University	48.8%	50.0%	1.2%	100.0%
High School	58.2%	38.8%	3.0%	100.0%
Other	50.0%	48.3%	1.7%	100.0%

3) The mean age of people with only a high school education (about 40.3) is considerably higher than the mean age of people with a graduate school education (about 34.2).

default	No	Yes
EDUCATION		
Graduate School	34.1	34.6
High School	40.3	40.2
Other	36.0	38.2
University	34.7	34.7

The histograms below also show that more education is associated with younger age. Additionally, from both the chart and the below histograms, we see that the ages of defaulters and non-defaulters is not significantly different.



4) Married individuals are more likely to default than singles (23.5% vs. 20.9%)

default	No	Yes	Percentage Default
MARRIAGE			
Married	10453	3206	23.5%
Single	12623	3341	20.9%
Other	288	89	23.6%

After obtaining these insights, inferential Statistics were not applied to the dataset to determine if they were statistically significant, because the dataset constitutes a population of credit card holders and not a sample.

These finding are interesting to us, and we may speculate about why they are true, but we must remember that they are by-products of our process to achieve our goal, which is predicting default using 6 different machine learning techniques and determining if there is a difference among them. So we will move on to that task now.

In-depth Analysis (Machine Learning)

We want to predict credit card default using the 23 features we have available to us containing information about customer demographics and customer financial information. We also want to see if there is a difference in prediction ability between different machine learning models.

To do this, we will test 6 different models of classification to see which produces the best results: Logistic Regression, K Nearest Neighbors (KNN) , Support Vector Machine (SVM), Decision Tree, Random Forest, and Naïve Bayes

Before fitting each model to the data we must do two things:

1) Preprocess the data;
2) Split the data into training and test sets

These two steps are the same for every model, and therefore we need only do them once. After completing them we must:

3) Instantiate the model or estimator (this can be combined with the first step into a scikit-learn Pipeline() object);
4) Specify the Hyperparameter space over which we are optimizing our hyperparameters;
5) Create a GridSearchCV() or RandomizedSearchCV() object from the estimator and hyperparameter space;
6) Fit the CV object to the training set (this takes the most time of any of the steps);
7) Create a list of predictions by using the .predict() method on the estimator with the test set;
8) Score the model

We begin by preprocessing the data. We make good use of scikit-learn's Pipeline() object and preprocessing features here. We divide our features into 2 lists, one containing categorical features and the other containing numeric/continuous features. We then apply appropriate transformations: to the continuous features, we apply the MinMaxScaler() transformation so that all values are rescaled to be between 0 and 1; to the categorical features we apply the OneHotEncoder() transformation. These transformations are important because many classifiers like KNN use distance metrics to make classifications. We wrap each of these Pipeline transformations in a ColumnTransformer() object so that they can be applied to their appropriate features.

We split the data using scikit-learn's train_test_split() function. We are now ready to fit our models to the data.

Before doing that, it is important to clarify how we will score each model. The most natural and intuitive metric is accuracy. However, there are 2 problems with using accuracy alone:

1) Our dataset features an imbalance of about 78% to 22% where 78% of the records are labelled as 'no default' and 22% were labeled as 'default'. In classification problems, the greater the imbalance in the dataset, the less informative accuracy is as a metric, since high accuracies can be achieved by predicting the more common label.
2) In the context of our problem, not all mistakes made by a model are equal. There are 2 kinds of errors we can make:
    a. False positives: predicting a person will default when he/she won't

b. False negatives: predicting a person won't default when he/she will

In our problem, our key stakeholder is any credit-extending institution. For such institutions, the cost of a false negative is far higher than the cost of a false positive. Therefore, in our scoring system, we should assign greater value to models that produce fewer false negatives at the cost of more false positives. This trade-off is well-known in classification problems as precision-recall trade-off. In our case we want to maximize recall (the ratio of correctly predicted defaulters to the number of actual defaulters) at the cost of precision (the ratio of correctly predicted defaulters to the total number of predicted defaulters).

So we now know that recall should be our main scoring metric. In addition to accuracy, precision, and recall, we will also keep track of the AUC score for each model, which tracks the trade-off between true-positive rate (higher is better) and false-positive-rate (lower is better)

Finally, before running our models, we create a dummy model that will predict 'no default' for every record in the testing set so we have a natural baseline with which to compare the accuracy, precision, recall, and AUC of our models.

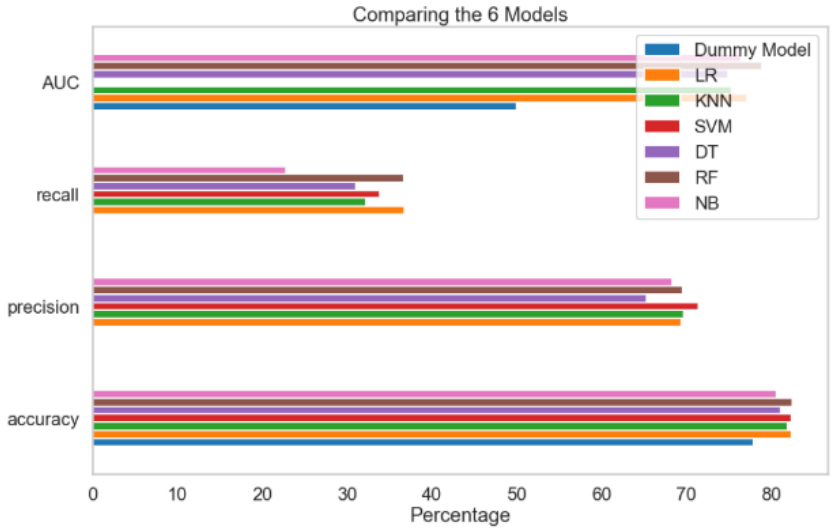
We fit the models to the training data, following the steps above, predict on the test set, and score each of the models, all the while appending each model score to a table tracking each model and its scores.

After fitting the models and scoring them, we obtain a table of our models and their scores:

	Dummy Model	LR	KNN	SVM	DT	RF	NB
accuracy	0.779	0.824	0.819	0.824	0.811	0.825	0.806
precision	0.000	0.694	0.697	0.714	0.653	0.696	0.683
recall	0.000	0.368	0.322	0.339	0.311	0.367	0.228
AUC	0.500	0.771	0.753	NaN	0.749	0.789	0.765
Time to Train	0.045	77.851	278.351	971.990	9.830	125.211	0.062

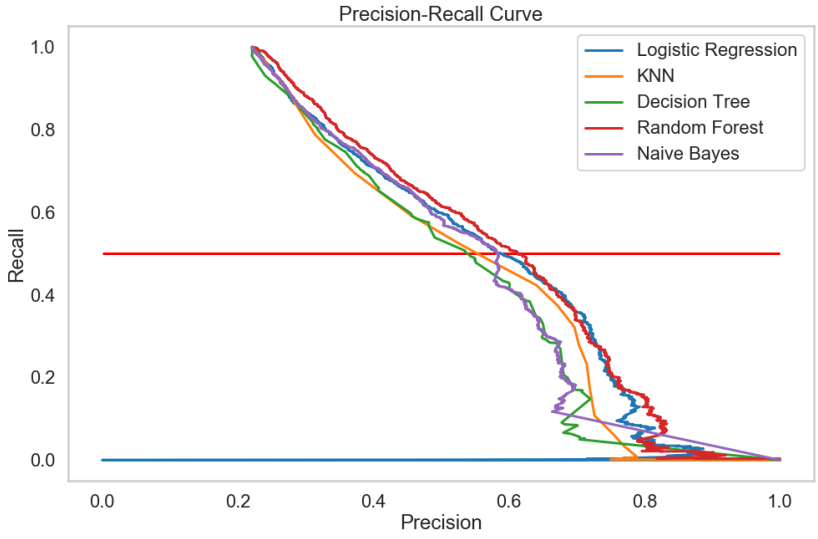
Looking at the results, we can make two key points. First, that all of our models performed better than the dummy model. The dummy model had precision and recall scores of 0 since it never predicted default and had an accuracy of about 78% since that was the percentage of non-default records in our dataset. In contrast, our models achieved accuracies ranging from 80.6% for Naïve Bayes to 82.5% for Random Forest. This means our models are working since they are finding some means by which to distinguish defaulters from non-defaulters. The second point is that by recall alone, Logistic Regression performed the best (36.8%) by a hair over Random Forest (36.7%).

The following chart also provides a helpful way of looking at the performance of the models:



From the table and chart alone, it appears that either Logistic Regression or Random Forest is the best classifier. It is hard to give one the edge since they are so close in all metrics. However, the table and chart don't tell the whole story. What we also need to do is look at a precision-recall graph which shows how recall changes with precision. Different threshold values for the probability at which we classify a customer as defaulting or not will result in different points on the curve. A precision-recall curve gives us a way of standardizing precision so that we can see how each classifier performs in terms of recall at that level of precision.

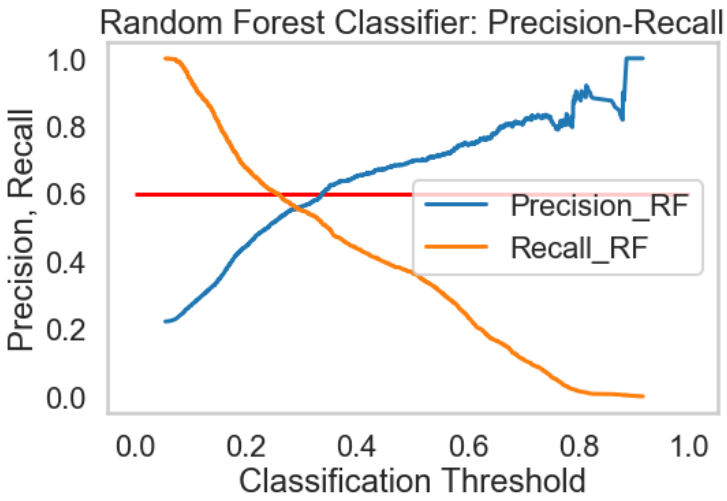
When we plot precision-recall curves for each of our models (except for SVM – this cannot be done since SVC() in scikit-learn has no predict_proba() method), we obtain the following figure:



We see that the Random Forest model produces the highest recall for a given precision over all the models except for values of precision between about 0.63 and 0.78. In that range, Logistic Regression actually produces higher values of recall. Because the values for precision within our table fell within that range, Logistic Regression had a higher recall then Random Forest in our table. This can be changed

by altering the threshold probability value for prediction so that we end up at different points on our precision-recall curves. This will allow us to attain higher values of recall, which we desire.

The figure below shows us how precision and recall change as a function of threshold for the Random Forest Classifier:



We see that at classification values below about 0.3, recall is greater than precision. Therefore, we should change the probability threshold used by our classifier from the default level of 0.5 to achieve greater recall. When we do this and use a level of 0.25, we achieve a recall of 60.6%, precision of 51.5%, and accuracy of 78.6%. Clearly, then, our accuracy (and precision) have suffered as a result of using a new, lower value for threshold, which makes sense. On the other hand, our recall is way up, which is what we really care about. We can also output the confusion matrix that accompanies using a threshold value of 0.25. We notice that it contains more false positives than false negatives, as it should:

PREDICTION TRUE	pay	default	Total
pay	5872	1137	7009
default	785	1206	1991
Total	6657	2343	9000

We conclude that Random Forest performed best. By altering the probability threshold for classification, we are able to achieve higher rates of recall than we are able to with all other classifiers. Since this is our key scoring metric, this makes Random Forest our best model for the particular challenge of classifying customers as defaulting or not.

Approaches Taken That Did Not Produce Better Results

The above description of the in-depth analysis performed describes the sequence of steps that were taken to achieve the best results. A number of different ways of processing features were attempted, and their effects on accuracy were observed. They were:

1) Not scaling the PAY_X (repayment status) columns: This resulted in worse accuracy, recall, precision, and AUC.
2) Binarizing the PAY_X columns. The idea here was to treat every customer as either being behind on payments or on time without regard to how far behind they were on payments. As might be expected if we choose to lose information, this did not lead to better performance.
3) Getting rid of all PAY_X, BILL_AMTX (bill amounts), and PAY_AMTX (repayment amounts) columns except for the ones latest in time (September). Surprisingly, this had only a very small effect on accuracy and recall! A natural reason for this is that if we are determining default in October, we really only need to look at the customer's state of affairs in September. Ultimately, it was decided to keep the other columns because they did add a very small amount of performance, but the implication from this approach is that having more data about a customer's current financial information tells us a lot about their default probability.

Future Work

Looking back upon our results, there are 5 things that can be done in future work in order to obtain better performance – that is higher recall and/or accuracy, precision, and AUC.

1) The first is feature engineering. There wasn't any feature engineering done in this project, and feature engineering can often deliver real value. One natural idea that could be implemented is to consider interaction terms among the features. This could be implemented with something like sklearn.preprocessing.PolynomialFeatures
2) The second is more comprehensive EDA. This suggestion is actually upstream of feature engineering, since better EDA allows the data scientist to have a better grasp on the data, and hence what features should be involved in feature engineering. The techniques used here would be more scatter plots, correlation matrices, and visualization.
3) The third is sampling techniques to handle the class imbalance in our data: upsampling, downsampling, and SMOTE are all techniques designed to deal with imbalances and using them in our project might have improved performance.
4) The fourth is using more advanced models. Using Neural Networks and Discriminant Analysis as the initial paper did makes a lot of sense. Using models that have been developed since the dataset came out, like Gradient Boosting and Ada-Boost, would also probably deliver value.
5) A fifth improvement could be to use unsupervised learning as part of the process. The idea here would be to use dimensionality reduction to see how many features can be disposed of without sacrificing model performance that much, which would make for a much better, leaner model.

Conclusion and Recommendations

In this project, we predicted credit card default with a variety of models and found that there was a difference among the models, with Random Forest producing the best results for our purposes.

Based on this result and our observation that performance decreased only slightly when using just the last month's data, we can make 2 recommendations:

1) The client should use a Random Forest classifier, if they are to use only one model.
2) The client should give much greater importance to data that is recent, and should seek to gather more data about the client prior to the point in time at which they may default, as this will likely improve their predictions about default.