Coby Davis, ML Final Project, Dec 20 2019

1 Introduction

The purpose of this report is to document and give analysis on the task of identifying credit default probabilities for a large dataset of individuals and their behaviours.

2 Data Dictionary

The way I created features for this project first requires me to perform many aggregations on each of the sub files because there is not one dataset with an individual identifier row; the data is much more segmented. The general strategy I used was inspired by most of the top 10 Kaggle users in their approach: Look at each subfile, perform feature engineering on important datum, aggregate certain features to which there is more than 1 ID associated to, left join these new features to the overall dataframe, and then repeat for the remaining subfiles. For categorical columns, I used one hot encoding as this was a popular option among Kagglers and is a requirement for XGBoost (it needs numerical variables). For some of these subfiles I did not know what would be the appropriate business sense varible to create. But my thinking was that if I take enough of the "standard" feature transformations on the usual features, that XGBoost will have an easier time learning the non-linear relationships between the variables because I have already hade different versions to be learned upon. In addition, I chose to keep empty values as they were and not try to impute because boosted trees can handle these values as a valid input.

For some reason, there were many date columns that had the value 365.243, to which I replaced with Nan. I am assuming this was an error.

To perform one hot encoding on the categorical columns, I define a function called one_hot_encoder whose input is a dataframe. This function identifies the N (N=0 is possible) columns that are filled with non-numeric values and then creates N columns corresponding to each of these categories and then for each item in the list fills in a 1 or 0 depending on which category this entry happens to be in.

2.1 Application Train Test

For this dataset, I used all the features in addition to
CREDIT_TO_ANNUITY_RATIO= ratio of credit to annuity
CREDIT_TO_GOODS_RATIO=ratio of credit to goods
INC_PER_CHLD= the normalized amount of income in your family
INC_BY_ORG=income by the categorical 'organization' in the dataset
EMPLOY_TO_BIRTH_RATIO=ratio of days employed to days since birth

ANNUITY_TO_INCOME_RATIO= ratio of annuity to total income
External sources seem to provide credit scores. These will be very useful later because they turn out to be very important features.
SOURCES_PROD=product of other credit scores. Higher is better.
EXT_SOURCES_MEAN= mean of external credit scores
SCORES_STD=standard deviation of credit scores
CAR_TO_BIRTH_RATIO= ratio of days since car purchased to age
CAR_TO_EMPLOY_RATIO=ratio of days since car purchased to days since employment
PHONE_TO_BIRTH_RATIO= ratio of last phone change to age
PHONE_TO_EMPLOY_RATIO= ratio of last phone change to days since employment
CREDIT_TO_INCOME_RATIO=ratio of credit to income

2.2 Bureau And Balance

After aggregating upon 'SK_ID_BUREAU', I used a combination of max,min,mean, and sum on the variables. Specifically:
'DAYS_CREDIT': ['min','max','mean','var'],
'DAYS_CREDIT_ENDDATE': ['min','max','mean'],
'DAYS_CREDIT_UPDATE': ['mean'],
'CREDIT_DAY_OVERDUE': ['max','mean'],
'AMT_CREDIT_MAX_OVERDUE': ['mean'],
'AMT_CREDIT_SUM': ['max','mean','sum'],
'AMT_CREDIT_SUM_DEBT': ['max','mean','sum'],
'AMT_CREDIT_SUM_OVERDUE': ['mean'],
'AMT_CREDIT_SUM_LIMIT': ['mean','sum'],
'AMT_ANNUITY': ['max','mean'],
'CNT_CREDIT_PROLONG': ['sum'],
'MONTHS_BALANCE_MIN': ['min'],
'MONTHS_BALANCE_MAX': ['max'],
'MONTHS_BALANCE_SIZE': ['mean','sum']
I also used the mean for the one-hot categorical features.

2.3 Previous Applications

Similarly, I used aggregations on the numeric features:
prev['APP_CREDIT_PERC'] = ratio of application on previous to previous credit 'AMT_ANNUITY': ['min','max','mean'],
'AMT_APPLICATION': ['min','max','mean'],
'AMT_CREDIT': ['min','max','mean'],
'APP_CREDIT_PERC': ['min','max','mean','var'],
'AMT_DOWN_PAYMENT': ['min','max','mean'],
'AMT_GOODS_PRICE': ['min','max','mean'],
'HOUR_APPR_PROCESS_START': ['min','max','mean'],

'RATE_DOWN_PAYMENT': ['min','max','mean'],
'DAYS_DECISION': ['min','max','mean'],
'CNT_PAYMENT': ['mean','sum'],

2.4 Pos Cash

The aggregations I used are:
'MONTHS_BALANCE': ['max','mean','size'],
'SK_DPD': ['max','mean'],
'SK_DPD_DEF': ['max','mean']
I also use'POS_COUNT'to count the available pos cash accounts

2.5 Installments Payments

The following two formulae capture the percentage and difference paid in each installment (amount paid and installment value)
'PAYMENT_PERC'='AMT_PAYMENT'/'AMT_INSTALMENT'
'PAYMENT_DIFF'='AMT_INSTALMENT'-'AMT_PAYMENT'
The following two formulae capture the days past due and days before due
'DPD'='DAYS_ENTRY_PAYMENT'-'DAYS_INSTALMENT'
'DBD'='DAYS_INSTALMENT'-'DAYS_ENTRY_PAYMENT'
'DPD'='DPD'.apply(lambda x: x if x $>$ 0 else 0)
'DBD'='DBD'.apply(lambda x: x if x $>$ 0 else 0)
Again I preform aggregations on the numeric features. 'NUM_INSTALMENT_VERSION': ['nunique'],
'DPD': ['max','mean','sum'],
'DBD': ['max','mean','sum'],
'PAYMENT_PERC': ['max','mean','sum','var'],
'PAYMENT_DIFF': ['max','mean','sum','var'],
'AMT_INSTALMENT': ['max','mean','sum'],
'AMT_PAYMENT': ['min','max','mean','sum'],
'DAYS_ENTRY_PAYMENT': ['max','mean','sum']

2.6 Credit Card Balance

In this category, I apply all the aggregations to all the variables: ie. 'min', 'max', 'mean', 'sum', 'var']

3 Modelling

3.1 Performance Measures

The only measure I choose to use for this project is the ROC-AUC measure. Mainly because this is what the Kaggle competition basing success upon and

there is a great deal of notebooks and commentary online that deals with this specific metric. The ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. The higher the AUC, better the model is at predicting defaults as defaults and not-defaults as not-defaults.

3.2 Algorithm

Based on my research, the three state of the art ML algorithms are currently LightGBM or XGBoost or CatBoost. I settled on XGBoost because I noticed most of the top Kagglers used LightGBM (it is much faster I beleive in this context), and I wanted to see if I could get a similar result with a slightly different implementation to benefit my own learning. In addition, gradient boosting methods are known to be better performing than generic gradient boosting methods for a vareity of reasons. The get_fscore() function in XGBoost allows me to evaluate the importance of each of the features I am using. This will be important later because my initial feature set is so big (800), that I will need to run XGBoost on a smaller subset of the data (10,000 samples) , then isolate the important features (take the top 50), and then rerun XGBoost on this subsampled group of features. In addition, I use cross validation in the form of the k-folds parameter in XGBoost. It works by splitting the dataset into k-parts. Each split of the data is called a fold. The algorithm is trained on k-1 folds with one held back and tested on the held back fold. This is repeated so that each fold of the dataset is given a chance to be the held back test set. After running cross validation I end up with k different performance scores that I can summarize using a mean and a standard deviation. The result is a more reliable estimate of the performance of the algorithm on new data given your test data. It is more accurate because the algorithm is trained and evaluated multiple times on different data. I use k=3.

3.3 Parameter Tuning

The code I used to run XGBoost can be found in the appendix with the rest of the code. I did very extensive tuning for this project, most likely more than what was necessary. The main function I used to look for parameters was GridSearchCV. I use XGBClassifier whichis an sklearn wrapper for XGBoost in order to use GridSearchCV. This allows us to use sklearn's Grid Search with parallel processing in the same way we did for GBM However, given the huge number of features I had to start with, I ran XGBoost with my initial parameters on a small subset of the dataset to find which features were most important, and then chose the top 50 features, to be used for prediction. The steps I will be following are:

1. Choose a relatively high learning rate. A learning rate of 0.1 works in this case.

2. Determine the optimum number of trees for this learning rate. XGBoost has a function called as "cv" which performs cross-validation at each boosting iteration and thus returns the optimum number of trees required.

3. Tune tree-specific parameters (max_depth, min_child_weight, gamma, subsample, colsample_bytree) for decided learning rate and number of trees.

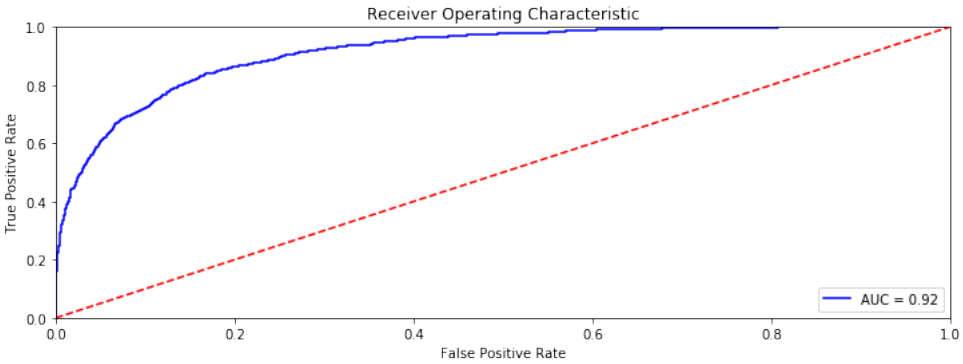
4. Tune regularization parameters (lambda, alpha) for XGBboost which can help reduce model complexity and enhance performance.

5. Lower the learning rate and decide the optimal parameters .

As my initial conditions I used:
max_depth = 5 : This should be between 3-10. I've started with 5 but this does not seem to make a difference in this example
min_child_weight = 1 : A smaller value is chosen because it is a highly imbalanced class problem and leaf nodes can have smaller size groups.
gamma = 0 : A smaller value like 0.1-0.2 can also be chosen for starting. This will anyways be tuned later.
subsample, colsample_bytree = 0.8 : This is a commonly used used start value.
scale_pos_weight = 1: Because of high class imbalance.

After running this example on the smaller dataset, I choose my 50 best features, and I also can see that I got 140 as the optimal estimators for 0.1 learning rate. This value was too high for the speed of my computer so I lowered it to 100. The results for the smaller dataset and its 50 most important features are below:

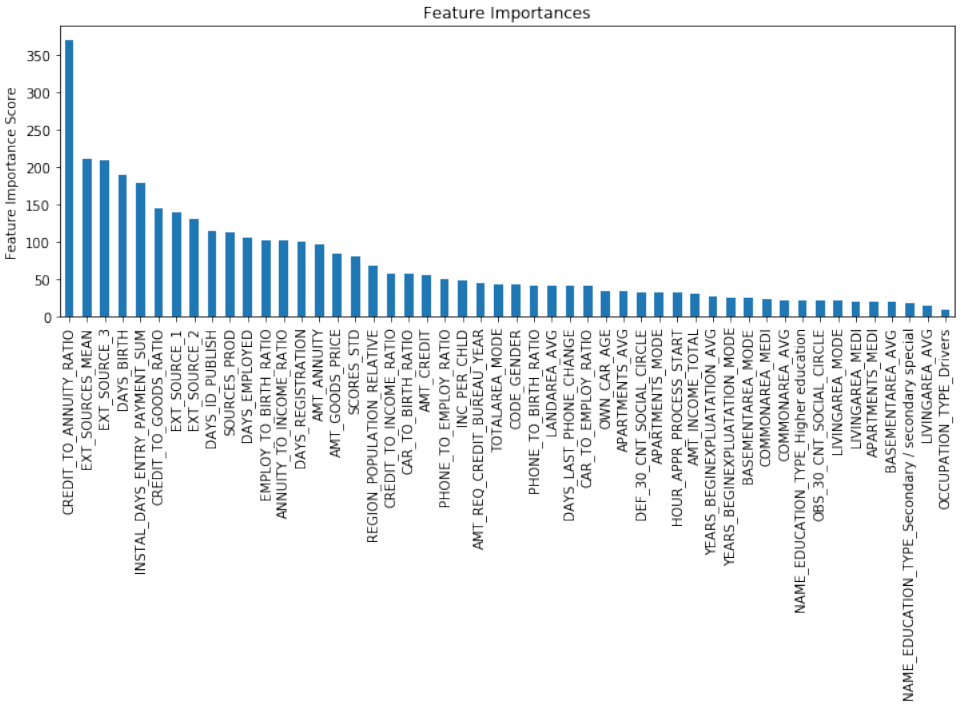
Accuracy :	0.943
AUC Score:	0.915171
Best Iteration:	44

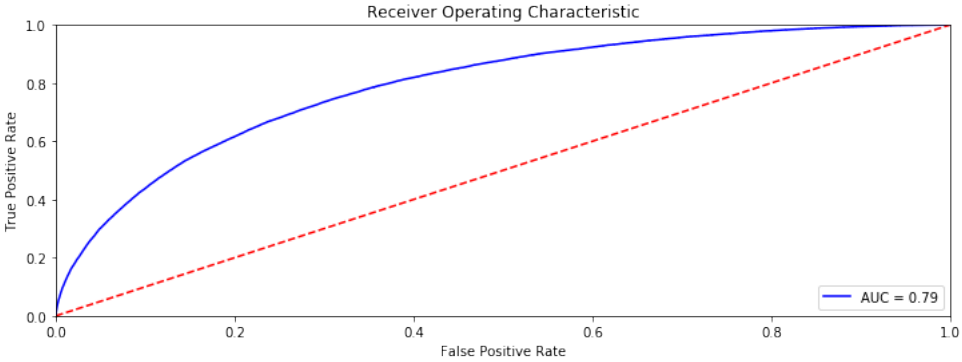


Note that the AUC is extremely high. This is probably due to the fact that the model is heavily overfitting on this smaller dataset, but all we need here

Accuracy :	0.91
AUC Score:	0.7945
Best Iteration:	212

is a rough idea of the features that led to this impressive perfomance. After selecting the top 50 features to be my predictors, I redo the model using the entire dataset:





The top features are:

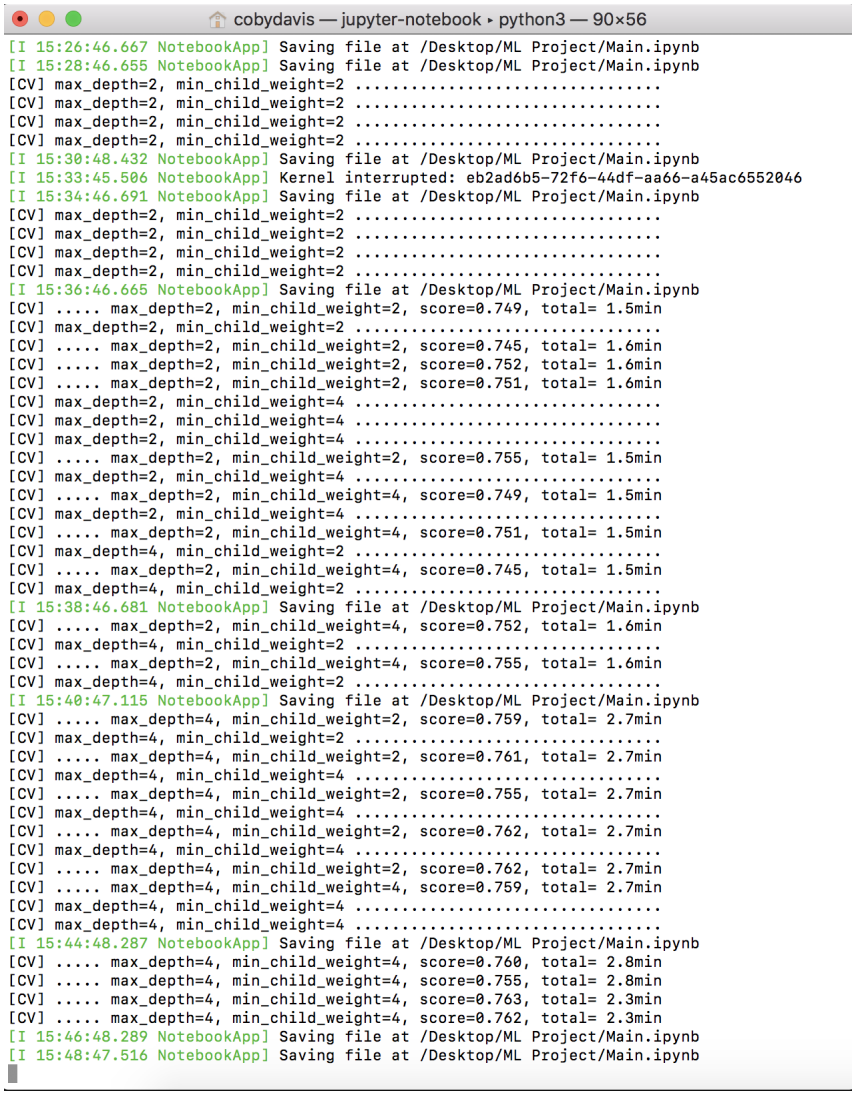
CREDIT_TO_ANNUITY_RATIO
EXT_SOURCES_MEAN
DAYS_BIRTH
EXT_SOURCE_3
INSTAL_DAYS_ENTRY_PAYMENT_SUM
EXT_SOURCE_1
CREDIT_TO_GOODS_RATIO
DAYS_ID_PUBLISH
EXT_SOURCE_2
DAYS_EMPLOYED
DAYS_REGISTRATION
EMPLOY_TO_BIRTH_RATIO
AMT_ANNUITY
SOURCES_PROD
ANNUITY_TO_INCOME_RATIO
SCORES_STD
CREDIT_TO_INCOME_RATIO
AMT_CREDIT
PHONE_TO_EMPLOY_RATIO
AMT_GOODS_PRICE
CAR_TO_BIRTH_RATIO
REGION_POPULATION_RELATIVE

We will analyze the importance of these features in the business write up. Note that for each run of the hyperparameter tuning section, the recalibration will cause the feature importance's to change, however the changes are very small and the overall order may adjust slightly however, the core features remain roughly stable under the parameter tuning.
Next I tune max_depth and min_child_weight by searching in a 2 to 6 by 2 to 6 grid skipping even values. After determining the optimal values within this search range, I run this again searching one above and one below the optimal value to see if there is a better choice that is an even number.
In this case 'max_depth'=4, and 'min]_child_weight'=2 are the optimal parameters. GridSearchCV returns a score for each combination and the score in this case was 0.759796. We use this as the metric to determine optimal parameters going forward. Here is the output while running these operations:

```
Starting Gridsearch
Fitting Parameter Test 4
Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   5 tasks      | elapsed:  3.1min
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:  4.7min
[Parallel(n_jobs=4)]: Done  16 out of  20 | elapsed:  8.5min remaining:  2.1min
[Parallel(n_jobs=4)]: Done  20 out of  20 | elapsed: 10.8min finished

{'max_depth': 4, 'min_child_weight': 2}
```



After running (for another 15 minutes), the optimal parameters were deter-

Accuracy :	0.935
AUC Score:	0.8124
Best Iteration:	228

mined to be: 'max_depth'=5, and 'min]_child_weight'=3

Next I search for a gamma in between 0 and 0.5 with 5 steps. Gamma being 0.1 turns out to be optimal for this case.

Before proceeding, I re-calibrate the number of boosting rounds for the updated parameters.

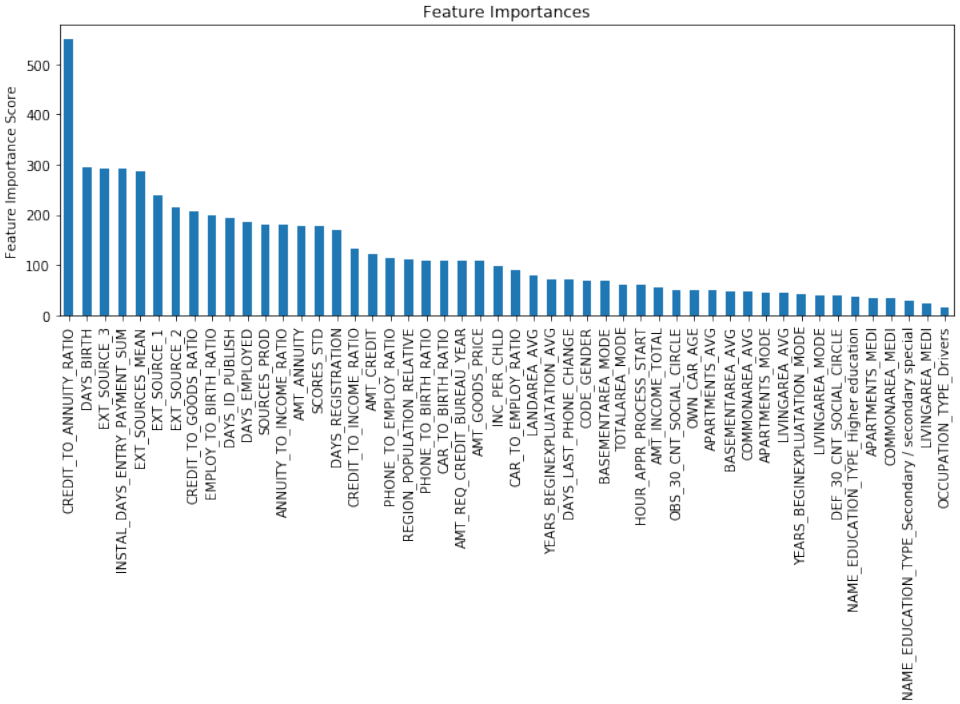
Next I tune subsample and colsample_bytree by looking in a 0.6 to 0.9 by 0.6 to 0.9 grid with 5 steps in between.

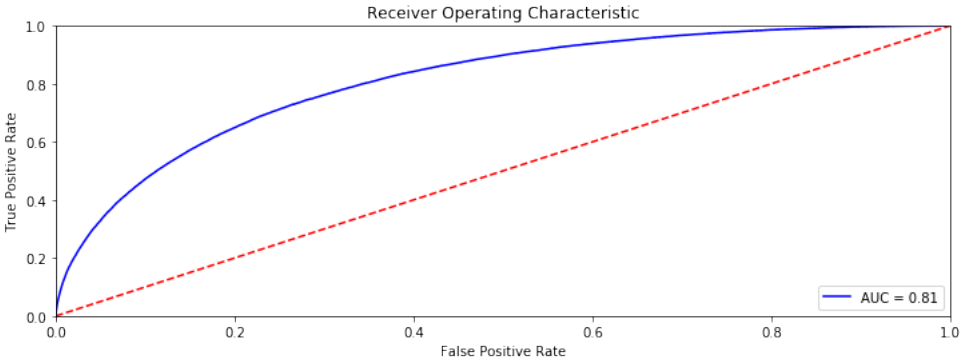
I then do another seach in the +-0.05 range of the optimal parameters found in the previous step.

I now tune the regularization parameters by gridsearching for an alpha in between 0.01 0.1 1 10 100. After finding this parameter, I look in another smaller grid around it.

I then apply the new optimal parameters to the model and recalibrate.

Lastly, I lower the learning rate and add more trees (n_estimators=5000 now). The final values for AUC are:





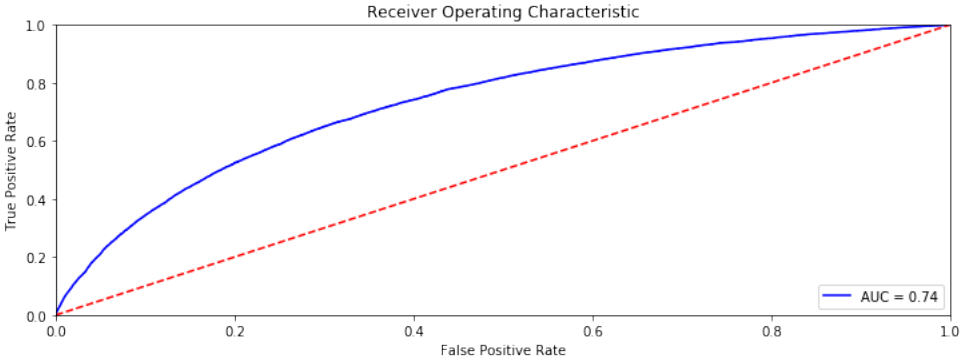
4 Logistic Regression

For this section, I use the exact same variables as I did in XGBoost however, I employ the technique of calculating WOE and IV for each of the features as I have done in a previous MMF project. The idea behind this specific type of feature transformation is to identify the features that most importantly contribute to the default probability and allows to cut down the number of variables to a much more reasonable number, in this case 20 chosen features based on IV rank.

In order to calculate the WOEs the algorithm requires that each feature has to be divided into bins and each ID was placed in one of the bins for each feature. This selection process is based on the quantiles of the distribution for that feature. As a recap WOE describes the relationship between a predictive variable and a binary target variable. IV measures the strength of that relationship. After ranking the features, the top features were:

VAR_NAME	IV
EXT_SOURCES_MEAN	6.218968e-01
EXT_SOURCE_2	3.156339e-01
EXT_SOURCE_3	2.436972e-01
SOURCES_PROD	1.341009e-01
EXT_SOURCE_1	8.645165e-02
DAYS_BIRTH	8.492034e-02
DAYS_EMPLOYED	6.952718e-02
AMT_GOODS_PRICE	6.076463e-02
CREDIT_TO_GOODS_RATIO	5.613675e-02
EMPLOY_TO_BIRTH_RATIO	5.588825e-02
NAME_EDUCATION_TYPE_Higher education	4.921436e-02

The AUC was 0.74 for the logistic model, even after the variable transformations using WOE, which was to be expected. One additional thing I used in this method was from a previous project (I know that at least one other group will have similar code and method because we worked together on the previous project) is SMOTE (Synthetic Minority Over-sampling Technique in order to predict more defaulted customers. SMOTE synthetically introduces new flagged individuals that do not actually lie within the dataset by using features that flagged individuals have in common from the features table. This allows the model to select the choice of the ratio of flagged to non-flagged customers. The threshold of 0.3 was chosen.



As a reminder WOE=$ln(\frac{Distribution of Non Defaults}{Distribution of Non Defaults})$

5 Business Component

In this section I will discuss the importance of credit models, and how the findings from the previous analysis relate to the business side of the equation.
The business value of the findings are twofold. One, the power to produce a model that can predict the default probability of a population and can be reused and modified in the future. Second, there is immense value in looking at which features that the XGBoost model deems as predictive. Knowing these features can help us build more powerful datasets in the future and drive the direction that we move in the data collection and storage space. If we are to trend in the right direction on looking for important variables, this is far more important than any technically complex model one can come up with. In this particular case, there are some strikingly predictive features that have well founded business rationalizations that I will explain. To recap, the XGBoost model was run on 50 of the 800 (a lot of them were one hot encoded categoricals and arbitrary functions like max and mean), features. And some of the ones listed below will be discussed.

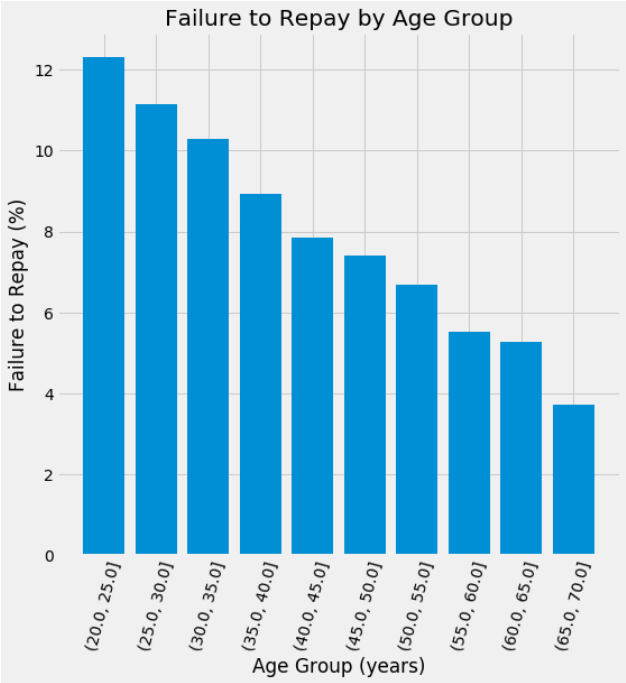
CREDIT_TO_ANNUITY_RATIO
EXT_SOURCES_MEAN
DAYS_BIRTH
EXT_SOURCE_3
INSTAL_DAYS_ENTRY_PAYMENT_SUM
EXT_SOURCE_1
CREDIT_TO_GOODS_RATIO
DAYS_ID_PUBLISH
EXT_SOURCE_2
DAYS_EMPLOYED
DAYS_REGISTRATION
EMPLOY_TO_BIRTH_RATIO
AMT_ANNUITY
SOURCES_PROD
ANNUITY_TO_INCOME_RATIO
SCORES_STD
CREDIT_TO_INCOME_RATIO
AMT_CREDIT
PHONE_TO_EMPLOY_RATIO
AMT_GOODS_PRICE
CAR_TO_BIRTH_RATIO
REGION_POPULATION_RELATIVE

Firstly, we see that there is a repetitive presence of the variables derived from EXT_SOURCE. It is my belief that this a credit score that was determined by an outside entity, for example it could be a FICO score. We see that all 3 scores

in addition to their means and product and standard deviations all make it into the top features. This tells us that instead of looking for heaps of data about an individuals lifestyle and previous banking actions, we should look to see if there were any other credit evaluations performed on the individual recently. A business objective would be to invest in looking for ways to find these other credit scores or to invest resources in determining other credit-score-like variables in these peoples lives that may act in a similar way to credit scores. If one can use the work and resources that another has already put into this individuals classsification, it is certainly worth the effort.

In addition there is a theme of ratios of combinations of loan credit, loan annuity, income, etc. This is a perfect example of a predictive feature having a very transparent business interpretation. These variables tell us that if the loan is worth a large amount, and you have a low income, or a high loan annuity, but little loan credit, then you are more likely not to be able to repay the loan. This set of features come predominantly from the initial application test file. This again tells us that loads of the extra data provided by the other files are not necessarily important, and thus if we know what to look for in the future, we can focus our efforts on isolating this data and thus have more time for analysis and building more robust and concise models that have less moving parts.

One more important theme that arises in the top few features is the presence of a time variable. For example DAYS_EMPLOYED and DAYS_BIRTH and DAYS_REGISTRATION all depend on the length of time the customer is alive in one way or another. This has the implication of age correlating to the default rate. As proof, below is a histogram of age group vs. default. This could have the entrepretation of the older you are, or the longer you have had an income, or the longer you have had a bank account (all somewhat equivalent), the more likely you are not only to be financially stable, but also to have the means and ability to repay loans either through savings built up through the years or through knowledge and experience gained throughout ones lifetime.



6 Technical Component

The model I chose to use, XGBoost has many desirable features for a ML model. XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we are fored to run grid-search and only a limited values can be tested. Also XGBoost has a very useful function called as "cv" which performs cross-validation at each boosting iteration and thus returns the optimum number of trees required. With all of these extra optimizations I might not otherwise get, I am confident that the model performs close to the upper limit of performance for this dataset. I used gridsearch in a very systematic way to squeeze the most performance out of the model increasing the AUC by 2%. The tuning of the hyperparameters strongly impacts the performance of the model, and if I had more computational power, I could have most definitely done a better job isolating the learning rate and number of estimator parameters. In terms of the metric used, AUC of the ROC, a slight backdraw is that this method does not tell you exactly where to assign the threshold. Would you rather have high precision, accuracy, f1, and so on? It depends entirely on the business case. The way I would implement this in the real world would be to discuss the business objectives with the client and determine the importance of false negatives, true negative, false positives, and true positives. Depending on his or her answers, this would help me assign the

threshold because at the moment I do not know how devastating each case of a false positive actually is. It could be harmless or it could be destructive. This is a gap in the method. Another gap is that I could not run the algorithm on all 800 features and all the data at once. So I had to run XGBoost on a smaller subset, then choose the most important features, and then use these features as a baseline going forward. So in theory, there are some features that have predictivity that I simply missed out on due to the fact that the sample was relatively small compared to the entire dataset. For the logistic regression, the WOE and IV method can be traced back to the 1950's and their validity is extremely well founded and well known especially within the credit risk industry. Another gap in logistic regression is to assume in the first place that the relationship between the features and the probability of default could be modelled in a linear way. There is most likely a very non-linear relationship that relates the two, but the version of "linearization" of the problem allows logistic regression to be employed. However, this is not too large a problem because linear regression is known to be a very good model for these sorts of problems and the generalized linear model is commonplace in ML.

7 Next Steps Component

As VP of the credit risk division, I would focus our efforts on determining the validity of the model that was just developed and I would also focus on data collection for the future and more aggresively pursuing the data that is more likely to be predictive instead of the data types that have virtually no impact, like obscure account nuances that appear in many of the datasets. To test the models, I would define an observation period and a collection period, where the observation period follows the collection period by 3 years. In this time, we would monitor if the loans defaulted and under what conditions. If these strongly correlate with our predictions we will know the validity of the model. I would also look into producing different models for each unique situation. Individuals, small businesses, loans for education, equipment etc. All of these categories have real world business impacts that cannot be thought of if they are all aggregated into one data set. For example, on student loans the most likely predictor is the degree/future occupation of the applicant. However, if this is considered to be a general loan, and the only info we have available is previous banking info, then we will not have access to this critical information. Similarly, if someone wanted to get a loan to launch a paper company, this would be a very likely business to default, but if there is no segmentation among loans types, we would not see this information. In addition, I would invest in a way to synthetically replicate the features we found to be of importance in this study. For example, it is possible that there are other tests (not necessarily credit based) that we can find on each user that is very similar in nature, but not directly related to their banking. This could possibly be the interest rates they are paying on their mortgage which is most likely derived from some other credit score, or possibly the number of available credit they have with another bank

(they cannot exceed a certain threshold without a set credit limit). After the model goes into production I would need to ensure that the model is working correctly and is taking into account real world economic factor such as interest rates, the stock market, and other world events that could affect the statistics of a population paying back their loans.