Machine Learning Engineer Nanodegree

Udacity Capston Project: Home Credit

Author: Omar Villa

Date: November 2nd 2018

Contents

Definition

Overview

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Problem Statement

The Home Credit Default Risk competition is a supervised classification machine learning task. The objective is to use historical financial and socioeconomic data to predict whether or not an applicant or client will be able to repay the loan. This is a standard supervised classification task:

• Supervised: The labels are included in the training data and the goal is to train a model to learn and predict the labels from the features

• Classification: The label is a binary variable, 0 (will repay loan on time) and 1 (will have difficulty repaying the loan)

The final application is expected to predict when or who are the best candidates for credit loan.

Metrics

Our evaluation metrics will be based in a ROC Curve because is what the competition specification requests but for learning purpose we will also include a Confusion Metrix to compare results.

[1] Understanding AUC - ROC Curve

The target metric will be "TARGET" under the application_train/test datasets, results are based on loan payed or not payed

Analysis

Data Exploration

application_{train|test}.csv

This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET). Static data for all applications. One row represents one loan in our data sample.

bureau.csv

All client's previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample). For every loan in our sample, there are as many rows as number of credits the client had in Credit Bureau before the application date.

bureau_balance.csv

Monthly balances of previous credits in Credit Bureau. This table has one row for each month of history of every previous credit reported to Credit Bureau – i.e the table has (#loans in sample *# of relative previous credits* # of months where we have some history observable for the previous credits) rows.

POS_CASH_balance.csv

Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample *# of relative previous credits* # of months in which we have some history observable for the previous credits) rows.

credit_card_balance.csv

Monthly balance snapshots of previous credit cards that the applicant has with Home Credit. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample *# of relative previous credit cards* # of months where we have some history observable for the previous credit card) rows.

previous_application.csv

All previous applications for Home Credit loans of clients who have loans in our sample. There is one row for each previous application related to loans in our data sample.

installments_payments.csv

Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample. There is a) one row for every payment that was made plus b) one row each for missed payment. One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit credit related to loans in our sample.

HomeCredit_columns_description.csv

This file contains descriptions for the columns in the various data files.

Data table statistics

For this project we will use the main table's application_train.csv and application_test.csv as well the bureau.csv table. The application_train table contains 307,511 rows x 122 features each row represents an individual or loan applicant and the features is the data captured by Home Credit regarding the borrower, this table contains the TARGET feature which is the target to predict which is a binary feature where 1 means the borrower made a least one late payment and target value 0 indicates borrowed has paid on time every month.

Of the 122 features we can say that DAYS_BIRTH and DAYS_EMPLOYED have the highest correlation vs the TARGET feature, DAYS_EMPLOYED feature required some feature engineering because it had negative values and counting days employed cannot really say much about the borrower employment situation which is why we created a new feature which only indicates of the borrower haves a job or not.

Bureau table contains summary information of applicant's loans from another lenders. Each row represents a loan and by instance each applicant can have one or more loans. In order to link this table with application_train we will need to create a feature that connects both tables, more of this under the Feature Engineering section.

Empty Values or NaN entries

We found around 20 features with more than 60% of missing data or NaN values, because of this we check if the features had good enough data to keep them or if we was force to drop them, fortunately the data was usuable and strategies as Impute work to repopulate the data.

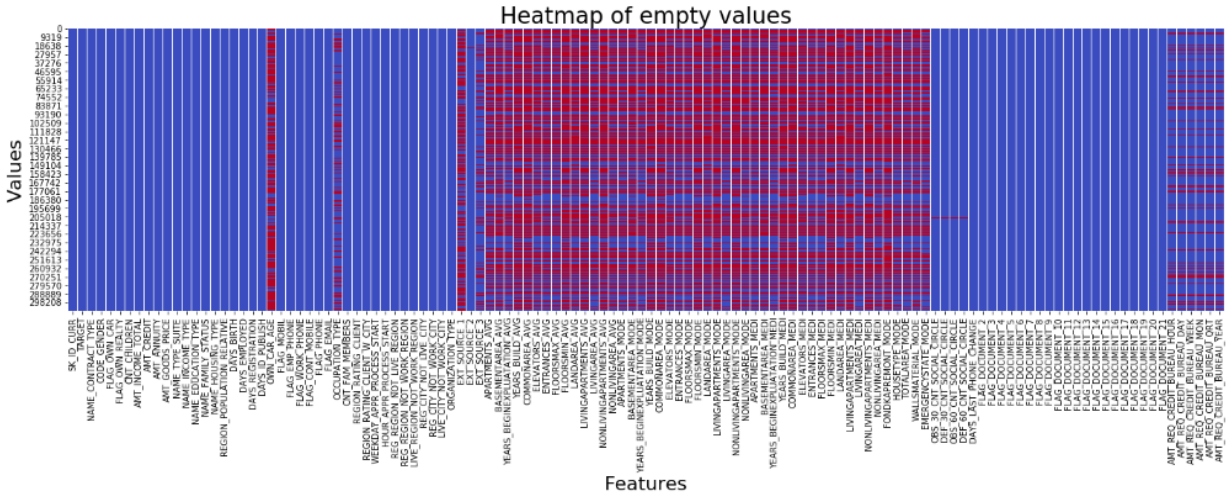

Fig 1. Heatmap shows empty values in red and populated values in blue.

Grouping Features

In order to better split the data and fill the NaN's or execute any Feature Engineering strategy we notice there was different features groups:

1. Categorical Features, divided in binary and Multi-categorical.
2. Numerical Features, divided in features with floating values and whole values at the same time each one of this two had binary entries most of these fall under the Numerical Features with whole values and the rest had several different values up to the thousands.

Once data was categorized it was easier to Engineer other features and move the new feature under another category as the CNT_CHILDREN feature who had several different values, we converted this to another binary category named HAS_CHILDREN indicating 0 for no-children and 1 for children.

This logic was applied to the bureau table but no new features where created.

Anomalies

There was five numerical features with a good level of anomalies: DAYS_BIRTH, DAYS_EMPLOYED, DAYS_REGISTRATION, DAIS_ID_PUBLISH and DAYS_LAST_PHONE_CHANGE which we only end up "fixing" DAYS_BIRTH, DAYS_EMPLOYED and DAYS_REGISTRATION and left the other two alone, critical sample of this is the creation of HAS_JOB feature from DAYS_EMPLOYED

Second anomaly was found with the CNT_FAM_MEMBERS, the size of this feature was represented as a numerical value and we decided to engineer a binary categorical feature named HAS_CHILDREN.

Feature Engineering

Of all the tables outside of application_train/test we found that bereau.csv contains a summary of information of applicant loans from other lenders, being this more useful for our target we have the hypothesis that engineering a feature out of this table that could relate with application_train table was best thing to do to better evaluate the applicants not only based on the data or loans with Home Credit but with other lenders.

The engineered featured was based on CREDIT_DAY_OVERDUE and indicates if a borrower has overdue loans with other lenders, we named this new feature HAS_CREDIT_BUREAU_LOANS_OVERDUE is of the binary type indicating of the applicant have fail to pay any of its loan with the other creditors.

During the exploratory visualization we found an anomaly with the DAYS_EMPLOYED feature, some of the numbers where way to far from the rest of the data but because all what we care for this analysis is if the borrower has a job or not is that we converted this numeric feature in to a binary feature where if the applicant had more than 0 days employed was enough to know if applicant will have the income to pay the loan from this we created the HAS_JOB feature and added to our dataset.

Exploratory Visualization

Distribution of target feature

I plotted the TARGET feature distribution and we found that we have a very small number of applicants who have paid in time their loans, pretty much a 10% of the total pays their loans on time.

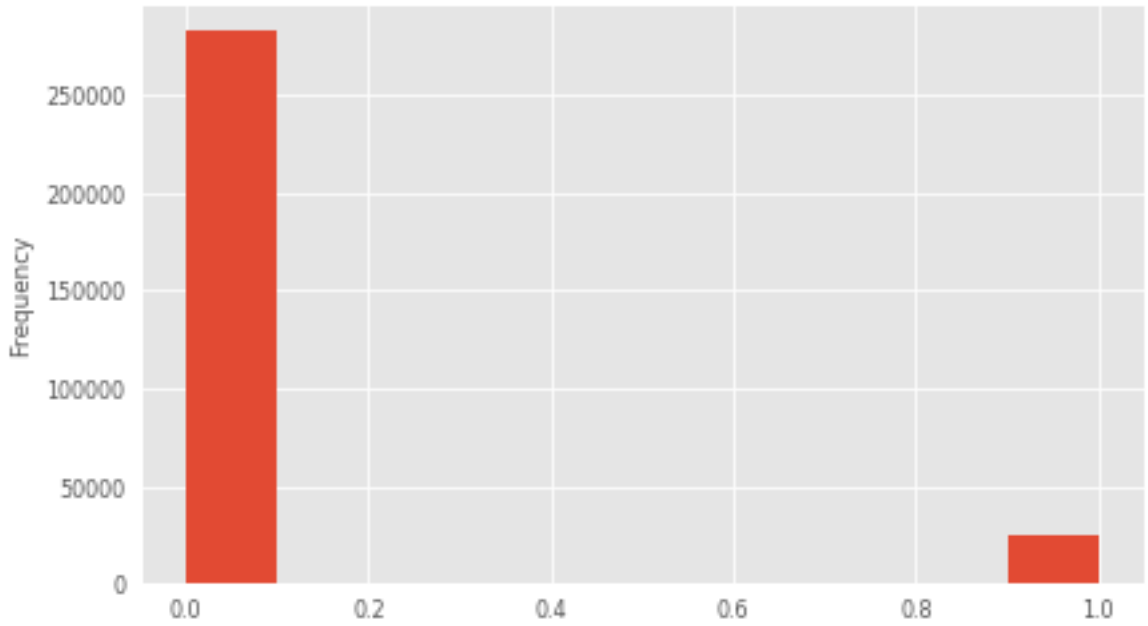

Fig 2. TARGET feature indicating 0.0 for applicants with at least one failed payment and 1 for those who haven't fail a single payment.

Based on above data we wanted to know if the missing data showed in figure 1 had any impact over our target feature, and we found that around 2000 of the total of the applicants had 60% of their data missing, but being the total number of applicants more than 300,000 we consider this number small and is why we decided to keep all the features and later impute the missing values with the mean of each feature.

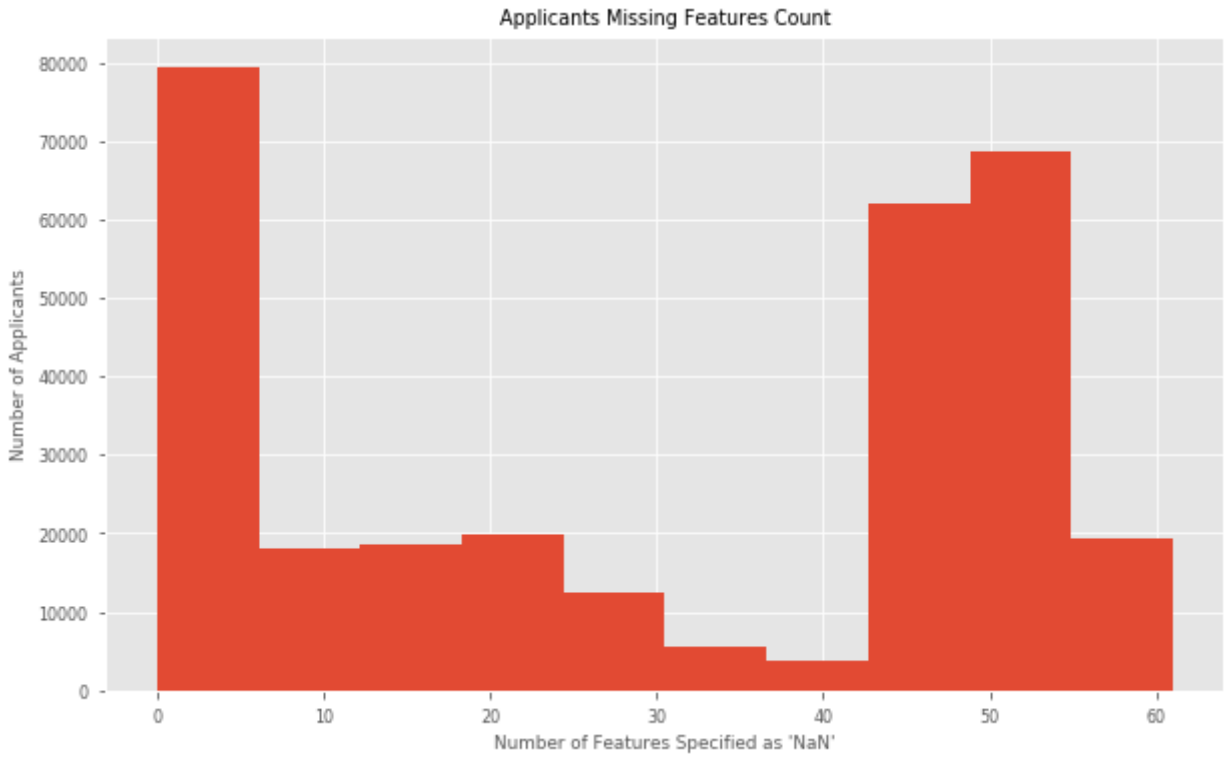

Fig 3. Number of feature with NaN values vs the Number of applicants.

We also found some key features that showed anomalies on their charts, some had negative values as DAYS_BIRTH we could only assume that Home Credit system was counting the number of days that person was alive since date of birth.

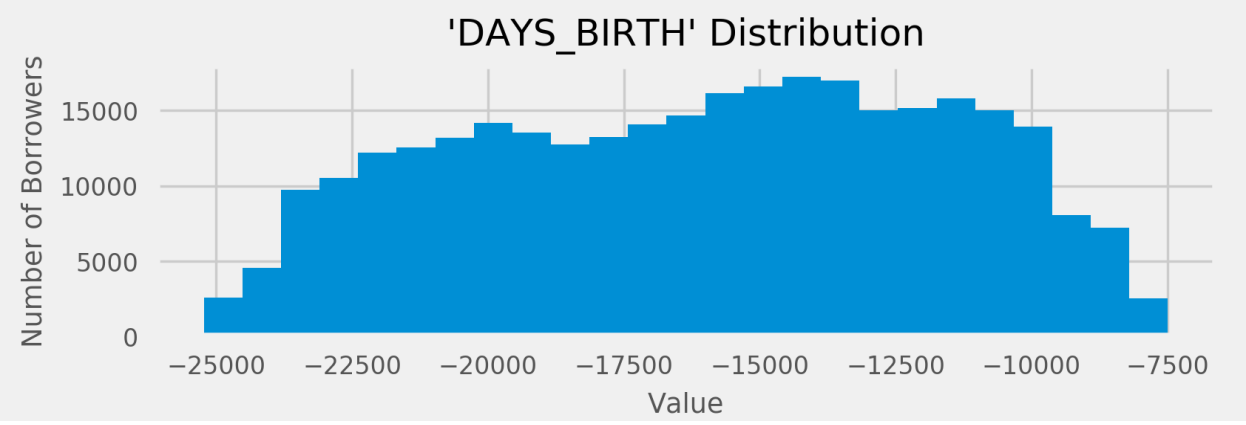

Fig 4. DAYS_BIRTH distribution in negative values anomaly.

Other important feature that we found with anomalies was DAYS_EMPLOYED, here we found some values way too far from the rest of the values this help us to make to create the HAS_JOB feature mentioned in the feature engineering section.

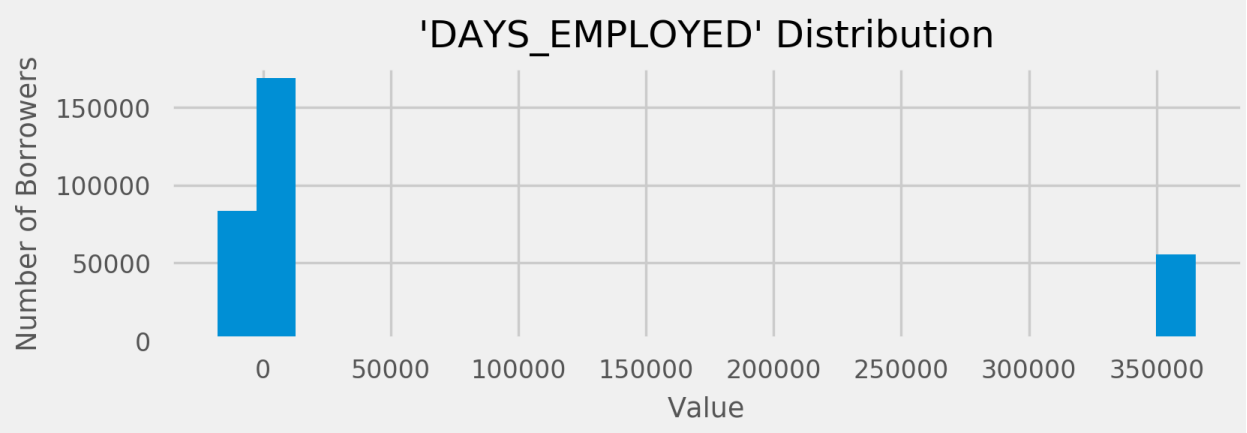

Fig 5. DAYS_EMPLOYED distribution anomaly.

Algorithms and Techniques

The classifier is a Known Neighbors, which is based on comparing neighbor data around a select point then algorithm will evaluate and count this neighbors and determine if the point in evaluation belongs to one group of data or not, the algorithm is a bit slow but work well for analysis and is not part of the regression family which is our benchmark and I think that using another family of algorithms can give us a good view of which model works better, the processing steps

Benchmark

As this is a Kaggle competition a benchmark model would be SK_ID_CURR in the test set, who must predict a probability for the TARGET variable. The submission will be evaluated on area under the ROC curve between the predicted probability and the observed target.

We will use Logistic Regression and KNN algorithms as our benchmark model because of their simplicity to create a baseline score

Our goal is not to win the competition but to present a solid model with a scientific method that can be use in other datasets so in the end we will learn the machine learning methodology being this the main objective of this project.

My Solutions will be ranked by the area under the curve (ROC) between the predicted probabilities and the observed targets indicating if the borrower will be able to pay its loan.

I will use the Logistic Regression classifier to analyze the dataset and use the ROC as the benchmark but will also use the Logistic Regression default scaler to see if there is any major difference when we use a different scaler. The training can take a bit longer than our second choice KNeighbors but because the target is a binary feature we think Logistic Regression will give us better results and with the help of the PCA dimensionality reduction performance can be drastically reduced.

Methodology

Data Preprocessing

1. We split the application_train table by 80/20% based on the cross-validation technique, even when we was provided with the application_test table, is a best practice to use the cross-validation as a pre-step to actual testing and potentially confirm that what have cross-validated it near to the actual test.
2. Fixed anomalies, as we already mentioned in our previous section and in some cases we created new features to squash this anomalies this specifically points to DAYS_EMPLOYED and CREDIT_DAY_OVERDUE.

3. Other Anomalies that required attention where the features with negative values who were out of proportion specifically DAYS_REGISTRATION and DAYS_LAST_PHONE_CHANGE for this features we log-normalized the values to align them with the rest of the features.
4. Missing data imputation, as we shown in figure 1 we found that around 20 features had more than 60% of NaN values and we decided to keep them because they didn't represented much of problem for the number of applicants who failed to pay their loans because only 2000 of the more than 300,000 applicants who fail on their payments had more than 60% of their , what we did with each one of this features was to calculate the mean of each feature and fill the empty or NaN values with the resulting mean.
5. Regarding the categorical features we One-hot encoded all the non-binary features, this was necessary because for machine learning to work we can only present numbers to be analyzed and categorical features will not be computable explaining why it was necessary to use this technique that causes and explosion of features which it can be a big setback but helps with the processing.
6. For the One-hot encoded features who were binary, it was necessary to replace the NaN values with a zero, this was necessary because NaN is not part of a binary value as 0 and 1 so we imputed all this values to 0 using a replacement technique.
7. As final step of the preprocessing we scaled all numeric values in a range of 0.0 to 1.0.

Implementation

1. Logistic Regression classifier. I felt that starting with this classifier was the best option because our target feature is a binary feature and logistic regression is king on this matter and one of the main principles of machine learning is to start with the most simple algorithm before we get in to more complex classifiers, our initial training give us a prediction score of 74%, this using the default values of scikit learn Logistic Regression classifier.
2. GridSearchCV. Using a grid we was able to increase our 74% score to 91% something to mention is that using our ROC AUC Scorer function we scored only 50% accuracy and the resulting hyperparameters of the GridSearch where totally different or opposite while we use the ROC AUC Scorer function we got a C=1000 and penalty=l2 and with the default scorer best score hyperparameters where C=0.001 and penalty=l1
3. KNeighbor classifier. This classifier help us to compare with the benchmark created by the Logistic Regression classifier

Refinement

To refine or analysis we used the Principal Component Analysis to reduce the number of features or dimensionality of the dataset. PCA will help us to find the properties that show as much variations across the dataset fitting the PCA algorithm to the numerical features using 17 components reaching a 90% of variance, we think with this can reduce the characteristics of our problem and increase the performance of the Logistic Regression and KNeighbors algorithm without impacting the results we obtained before the refinement.

Results

Model Evaluation and Validation

In the begging I couldn't tell which algorithm to use or which one will work better this is why I randomly picked the Logistic Regression and KNeighbors models, first one based on our target feature which is a binary feature.

During the development of the model we set a validation and testing parameters using cross-validation to split our data and GridSearchCV to find the best or most convenient hyper-parameters for either the Logistic Regression or Known Neighbors models. The best hyper-parameters for our model found through a Grid Search CV selection for the Logistic Regression classifier was C=0.001 and Penalty=l1 when we was using the ROC AUC Scaler but when we use the default Logistic Regression scaler we also notice a significant improvement on the accuracy of the model

The selection of PCA as our refinement process was based on the number of dimensions or data had, with more than 200 features in our dataset it was almost obvious that a reduction was necessary not only to improve performance but to see if the solution could be achieve with less variables.

Furthermore, after the final hyper-parameter tuning and training the model we found a 91.92% score when we use the default scaler for our Machine Learning algorithms, we think this number must be in the space of over-fitted model because using the ROC AUC scaler we got a 50.62% accuracy for our first runs on the training data and after we run KNeighbors against the PCA data ROC AUC only improve a 3% up to 53.62% but the Logistic Regression give us a 75% score it went up from 20% from 50.62%, telling us that our initial approach was correct regarding the Logistic Regression method but when we applied the GridSearch best score values the results improved by a 5% before PCA values application.

Justification

Using an Intel i5-86000K CPU @ 3.60GHz with 6 cores and a total of 8GB of RAM our testing took only a couple of minutes to finish with the Logistic Regression algorithm vs KNeighbors that took a couple of hours to run each time, our results were also way better with this method.

With this results we can tell Home Credit that we have a 75% accuracy model on selecting applicants that will pay their loan which it can give more confidence on their selection process.

Conclusion

Free-Form Visualization

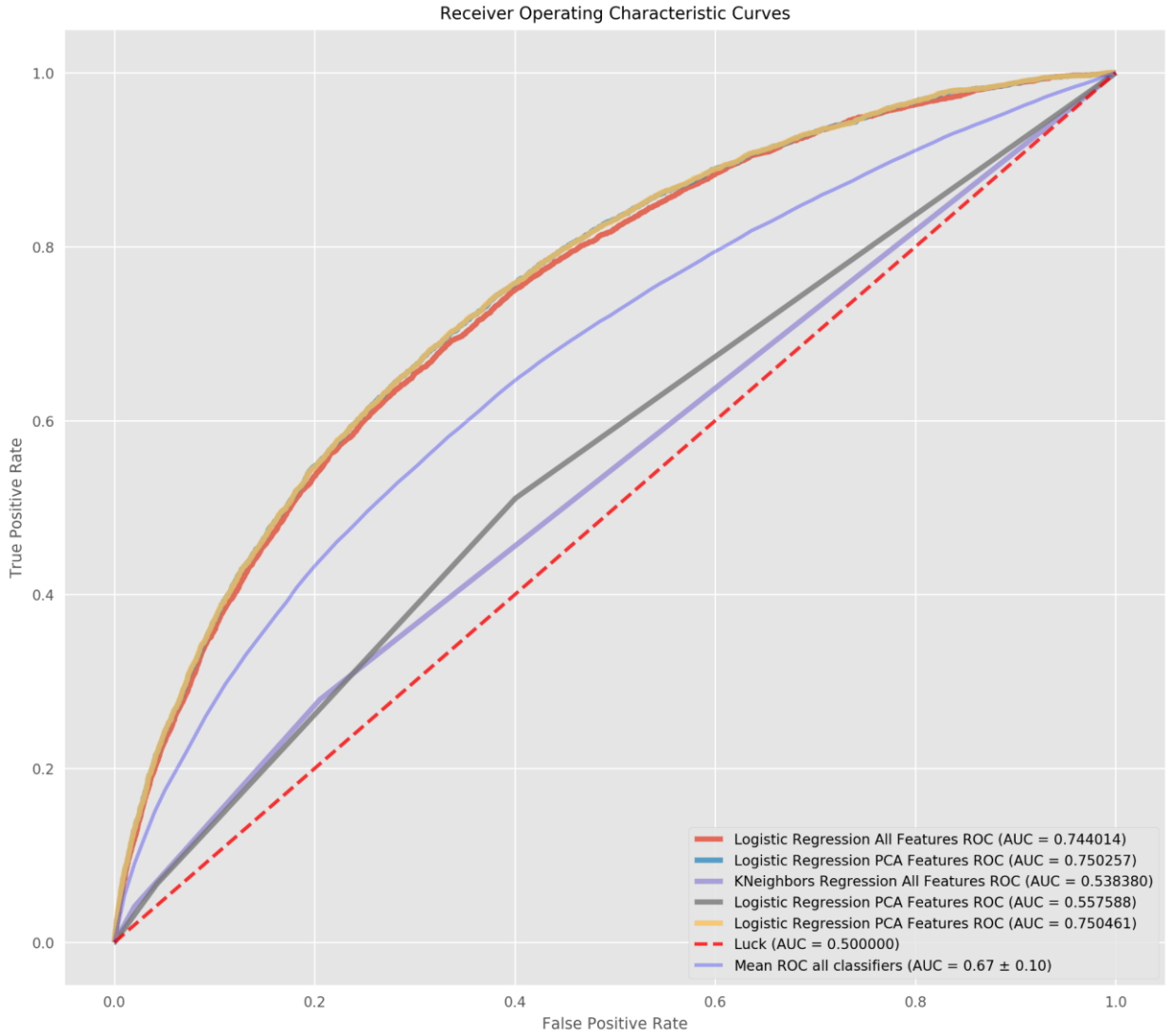

Fig 6. ROC Curves of each classifier and features set.

Reflection

My reflection of all this project is that is impossible at my level to do all this on my own, starting with the processing of digesting the data and features, then how it can be break apart and be able to give it consistency what I mean is to keep the number of features cohesive across the data sets specially between Train and Test.

The second part of the challenge was to understand when it was the right time to apply certain concepts, like when is better do Impute the data or when is better log transform a feature and which features to transform.

Once I thought I had the data ready for analysis or put it through a Machine Learning algorithm I learned that there was a lot of things that I needed done before this and I scratched my project 3 times because of this, finally I found a good template or blueprint that helped me with this phase of the project I learned that splitting the data by numeric and non-numeric and identify each one of this types as binary and multivariate is probably the best approach to take, but also can lead to overwork or unnecessary steps as creating features that we don't need or that won't represent any change to the output.

With all this completed again I learned the hard way that going back to our code and rerun pieces from other sections like running the code in the preprocessing and then use the output of this variables in the refinement section it produces a lot of errors and is why I copied the code to this section of the analysis so I ensure that the variables and lists I was generating where all part of the preprocessing and completely fresh for the analysis and it prove that once I ran the all notebook it didn't generated any errors on a single execution start to end.

Personally my biggest frustration was the need to google a lot of things, I'm the type of engineer who likes to know things from the top of its head but this is a so challenging subject that I found that almost impossible, at the same time googling or researching topics or looking for pieces of code that might help me it created a mess because I didn't had a blueprint on how to approach this type of problem, definitely this will change in the future and I learned that I should be ashamed of consulting and using others peoples code or blueprint to get to my goal while I understand what I'm doing, the concepts and the outcome of this code.

An last is the physical limitation that a computer or PC brings to this subject some of my executions took 2 or 3 hours to only realized that I made a mistake or submitted the wrong list to a function or things like this, there was a lot of down time because I only had 8GB of ram in my PC, definitely I'm upgrading and now I understand the power of the cloud when it comes to analyze bigger datasets.

Improvement

The improvement here was far from competitive from kaggle kernels but I was able improve my own initial evaluation, I was able to use the tools I consider where the right ones for this type of project and showed that there was certain level of improvement during the training and testing of the data, what surprised me the most was my intuition to use the Logistic Regression model and how well this worked, I'm sure there are other algorithms out there that I can add and test now that I have the preprocessing ready to be executed, but I wanted to stick with the initial proposal of only use two models.

Finally thanks to the GridSearch technique I was able to tell that I was using the best possible parameters for my classifiers, even if where not the best classifiers I can tell for sure I was using the best parameters for the Logistic Regression and KNeighbors algorithms, this part was the one that took the longest because doing a GridSearch is not a compute cheap process and it took a total of 10-15 hours spread in different runs and after several mistakes, I think next step is to learn to use the cloud resources in a better way and avoid this long waiting and waste of times.

References

Brownlee, J. (n.d.). *how-to-one-hot-encode-sequence-data-in-python*. Retrieved from machinelearningmastery.com: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/

Brownlee, J. (n.d.). *https://machinelearningmastery.com*. Retrieved from how-to-transform-data-to-fit-the-normal-distributio: https://machinelearningmastery.com/how-to-transform-data-to-fit-the-normal-distribution/

Brownleed, J. (n.d.). *feature-selection-machine-learning-python*. Retrieved from machinelearningmastery.com: https://machinelearningmastery.com/feature-selection-machine-learning-python/

Brownleed, J. (n.d.). *machinelearningmastery.com*. Retrieved from handle-missing-data-python: https://machinelearningmastery.com/handle-missing-data-python/

Credit, H. (2018, September). *home credit default risk*. Retrieved from https://www.kaggle.com/c/home-credit-default-risk: https://www.kaggle.com/c/home-credit-default-risk

Dellinger, J. (n.d.). *github.com*. Retrieved from Capstone Project Report - Home Credit: https://github.com/jamesdellinger/machine_learning_nanodegree_capstone_project/blob/master/report.pdf

Lee, T. (n.d.). *a-home-for-pandas-and-sklearn-beginner-how-tos*. Retrieved from kaggle.com: https://www.kaggle.com/timolee/a-home-for-pandas-and-sklearn-beginner-how-tos

Moffitt, C. (n.d.). *pandas_transform*. Retrieved from pbpython.com: http://pbpython.com/pandas_transform.html

Narkhede, S. (n.d.). *understanding-auc-roc-curve*. Retrieved from towardsdatascience.com: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Scikit-Learn. (2017). *machine_learning_map*. Retrieved from http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

VanderPlas, J. (n.d.). *PythonDataScienceHandbook/05.04-feature-engineering*. Retrieved from github.com: https://jakevdp.github.io/PythonDataScienceHandbook/05.04-feature-engineering.html

Weed, J. (n.d.). *github*. Retrieved from Capstone Project - ML Engineer Nanodegree: https://github.com/udacity/machine-learning/blob/master/projects/capstone/report-example-1.pdf