CKME 136 Data Analytics: Capstone Course
Ryerson University
Fall 2018

Final Report

Richard Harman

Default Risk Analysis in the Low-Income Segment

Contents

1. Introduction

Major financial institutions do not typically service well the needs of lower income families. Lending to low-income or first-time borrowers is challenging when there is little or no credit history. This creates a segment of ‘under-banked’ individuals struggling to access formal, safe and fairly priced credit. For established banks, this represents a profit growth opportunity, while for new entrants offering financial services, an attractive underserved market. However, to be successful in this segment, a financial institution must find non-traditional and creative ways to identify and mitigate default risk.

The work that follows explores a dataset from a lender operating in this under-banked segment. The dataset contains details of loan applicants at the time of application plus a target variable indicating whether the applicant ultimately defaulted (target = 1) or not (target = 0). We look for relationships between the characteristics of a loan applicant and their risk of default. For example, are people employed in a certain industry more likely to default, does length of employment, family status or size and location of residence impact default risk? Lastly, through the use of machine learning algorithms, we attempt to predict the likelihood of default for a new loan applicant.

Exploratory analysis uses the Python data manipulation and visualization modules numpy, pandas and matplotlib. Predictive modelling uses the Python machine learning module scikit-learn. The prediction task is a binary classification problem – likely to default: yes or no? The models investigated were Logistic Regression, Random Forest and AdaBoost.

All analysis and code can be seen in a collection of Jupyter Notebooks at:

https://github.com/netvigate888/credit_default_prediction

Please see the README file at the above repository for instructions on how best to view Jupyter Notebooks on GitHub.

2. Literature Review

A Machine Learning Approach for Predicting Bank Credit Worthiness [1]

15 models were investigated to predict customer credit card default. The models included Logistic Regression, K-Nearest Neighbor, Gaussian Naïve Bayes, Neural Networks, Support Vector Machines and multiple ensemble methods such as Random Forests, AdaBoost and Gradient Boosting. The subject dataset contained details of credit card customers from Taiwan with a binary target variable indicating whether a customer had defaulted or not. The dataset was unbalanced with 22% defaulting and 78% not defaulting. Accuracy, precision, recall, specificity and F-score were used to evaluate the models. The authors claim all models performed well, achieving accuracies from 76% to 98%, with the exception of Nearest Centroid and Gaussian Naïve Bayes. Of the 23 attributes, five with the most predictive power could be used to produce very similar results when compared to the full feature set.

A Comparison of Machine Learning Algorithms for the Prediction of Past Due Service in Commercial Credit [2]

The authors aim to predict commercial entities that will fall behind on their non-financial payment obligations (i.e. payments to suppliers, service providers, etc.). A large real-world dataset from Equifax was used containing 36 datasets each with over 11 million observations and 305 attributes. A binary target variable was created indicating whether the commercial entity was behind on payments or not. Significant data cleaning, missing value imputation and dimensionality reduction were performed to reduce the dataset to 16 independent variables. Three algorithms were tested: Logistic Regression, Decision Trees and Neural Networks. Accuracy, KS-statistic and ROC AUC were used to evaluate the models. Neural Networks produced the highest accuracy (at 93%) and Decision Trees had the highest KS-statistic (at 0.70). Both Neural Networks and Decision Trees outperformed Logistic Regression when using the ROC AUC measure. However, the actual numerical results were not shown in the paper.

Machine Learning Application in Online Lending Risk Prediction [3]

The author aims to predict defaulting loan applicants based on a combination of three datasets from an online Chinese lender, online phone records and third-party credit rating agencies. After amalgamation and removing observations with missing data, the dataset had over 8990 attributes and over 301,000 observations. A Random Forest model and an XGBoost model were trained and tested. The parameters of the Random Forest model were optimized using a precision-recall curve methodology, while the parameters of the XGBoost model were optimized using the ROC AUC measure. The prediction results of each model were compared using the KS-statistic. XGBoost outperformed Random Forest with a KS statistic of 0.72 versus 0.65.

Credit Default Mining Using Combined Machine Learning and Heuristic Approach [4]

The authors propose a two-step approach for predicting credit card default that combines machine learning with a custom rules-based algorithm (termed the “Heuristic Approach”). Machine learning is used to calculate a probability of default using static historical data, while the custom algorithm is used to calculate a probability of default based on “live” transaction data. The final probability of default is a linear combination of the two previously calculated probabilities. The dataset used appears to be the same Taiwan credit card dataset from [1]. Several algorithms were investigated for the machine learning step; the authors chose “Extremely Random Trees” (ER Trees) based on accuracy, precision, recall and F-score measures. Results of the final Heuristic Approach were compared to pure machine learning using ER Trees and the best results achieved in previous research on the Taiwan dataset. The authors claim the Heuristic Approach achieved accuracy of 93.1% while using ER Trees only had accuracy of 95.8%. Recall was 92.1% for the Heuristic Approach and 85.9% for ER Trees. The authors further claim that both approaches outperformed existing research on the Taiwan dataset for both accuracy and recall measures.

Personal Bankruptcy Prediction by Mining Credit Card Data [5]

A bankruptcy prediction system was built using credit card data from a major Canadian bank. This is the only paper reviewed that attempted to use the patterns in sequence/time-series data rather than the more common approach of simple aggregation (which can lead to loss of information). For each attribute, the sequence data was encoded to categorical or ordinal values (e.g. if a data point in the sequence represents ‘good’ behavior, it was encoded as 0, if it represents ‘bad’ behavior it was encoded as 1, if ‘worse’ behavior then 2, and so on). A K-Means algorithm was applied using a custom distance measure (based on conditional probability that a sequence belongs to a particular group) to cluster the encoded sequences. Based on the resulting clusters, *‘bankruptcy sequences’* were manually identified and then used to classify each sequence as either ‘bankrupt’ or ‘not bankrupt’. Once all the attributes for each client were classified in this way, a Support Vector Machine model was built as the final bankruptcy predictor. The results of the final predictor were compared, using ROC curves, to both an SVM model built using simple aggregation of the original data and credit bureau credit-scoring. The predictor compared well to credit bureau credit-scoring with a very similar ROC curve and materially outperformed the SVM model built on simple aggregated data.

Implications of Literature Review

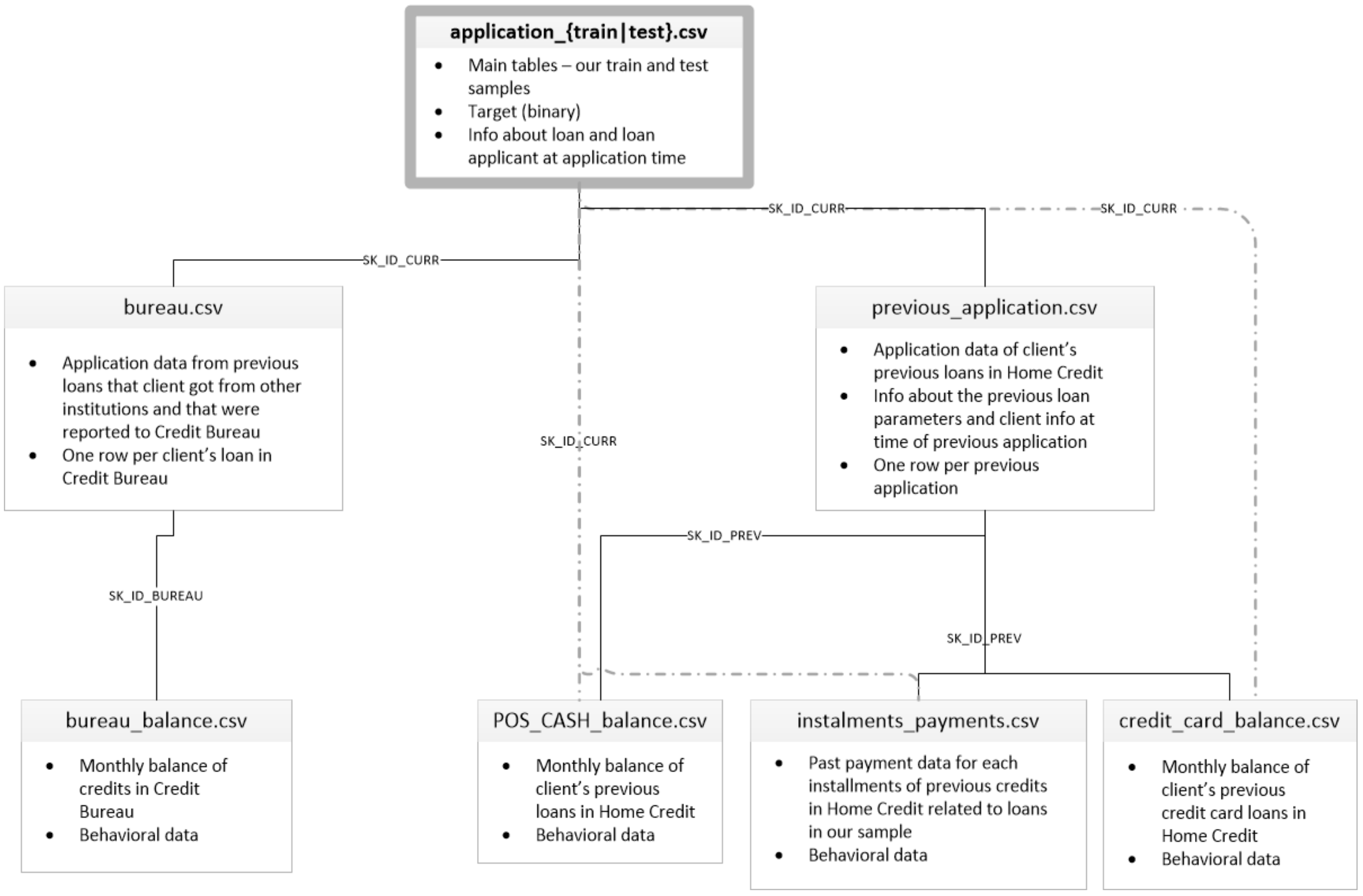
[1] showed there is a wide range of models available for default classification that are all capable of producing good results. [2] demonstrated that good prediction results can be achieved after significant dimensionality reduction of a very large dataset to a small number of attributes. While Neural Networks produced the best overall result, it was only by a small margin compared to Decisions Trees thus making Decision Trees potentially the better choice given their ease of interpretation. [3] showed the better performance of the XGBoost model versus Random Forests. [4] proposed a novel two-step approach using Extremely Random Trees that achieved good results. [5] introduced a methodology to make use of the patterns in sequential/time-series data that improved the results of a Support Vector Machine model.

Based on the review, Decision Trees and their ensemble derivatives such as Random Forests and Boosted Trees (e.g. XGBoost) appear to be the best solution for default prediction. These models show good results and are easier to interpret than Neural Networks. Lastly, the use of patterns in sequential/time-series data, as opposed to simple aggregation, can further enhance prediction results.

3. Data Description

The dataset was provided by Home Credit B.V., a lender operating in the 'under-banked' segment, for a Kaggle competition[1]. There are eight CSV files containing details of loan applicants at the time of application plus information (where available) on their historical repayment performance. Figure 1 shows a brief description of each file plus the relationships between the files.

Figure 1: Brief descriptions and relationships between files


application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_BUREAU
SK_ID_PREV
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

Source: https://www.kaggle.com/c/home-credit-default-risk/data

[1]Data Source: https://www.kaggle.com/c/home-credit-default-risk/data

Summary Details

application_train.csv (app)

This is the main data file that contains information about each applicant at the time of loan application. It contains attributes such as gender, type of loan, if the applicant owns a home, size and monthly payment of loan, type of employment, details of where the applicant lives, documentation provided and so on. This file also contains the target variable labeled as 1 = default and 0 = non-default.

application_test.csv (not used)

This is the data used to test models when submitting to the Kaggle competition. It was not used in this work because the data is unlabeled. That is, the target variable is missing.

bureau.csv (bureau)

Information about previous loans with other financial institutions for applicants in *application_train.csv* that have been reported to a credit bureau. For each applicant in *application_train.csv*, there may be zero, one or more entries in *bureau.csv* for each of their previous loans.

bureau_balance.csv (bureau_balance)

Basic historical information (where available) about the monthly balance of loans contained in *bureau.csv*. The information is a simple flag for each historical month (C = closed, X = unknown, 0 = nothing overdue, 1 = a payment is 0-30 days overdue, 2 = a payment is 31-60 days overdue, and so on).

previous_application.csv(prev_app)

Details on previous loan and credit card applications to Home Credit B.V. with attributes such as monthly payment, amount, purpose and interest rate of previous loan or credit card. Each applicant in *application_train.csv* may have zero, one or more entries in *previous_application.csv* for each of their previous loan or credit card applications to Home Credit B.V.

pos_cash_balance.csv (pos_cash_bal)

Status of the historical monthly balance of the previous loans in *previous_application.csv*. Information such as number of instalments left to pay, whether the loan is active or closed and whether there are payments past due is included.

instalments_payments.csv (install_pay)

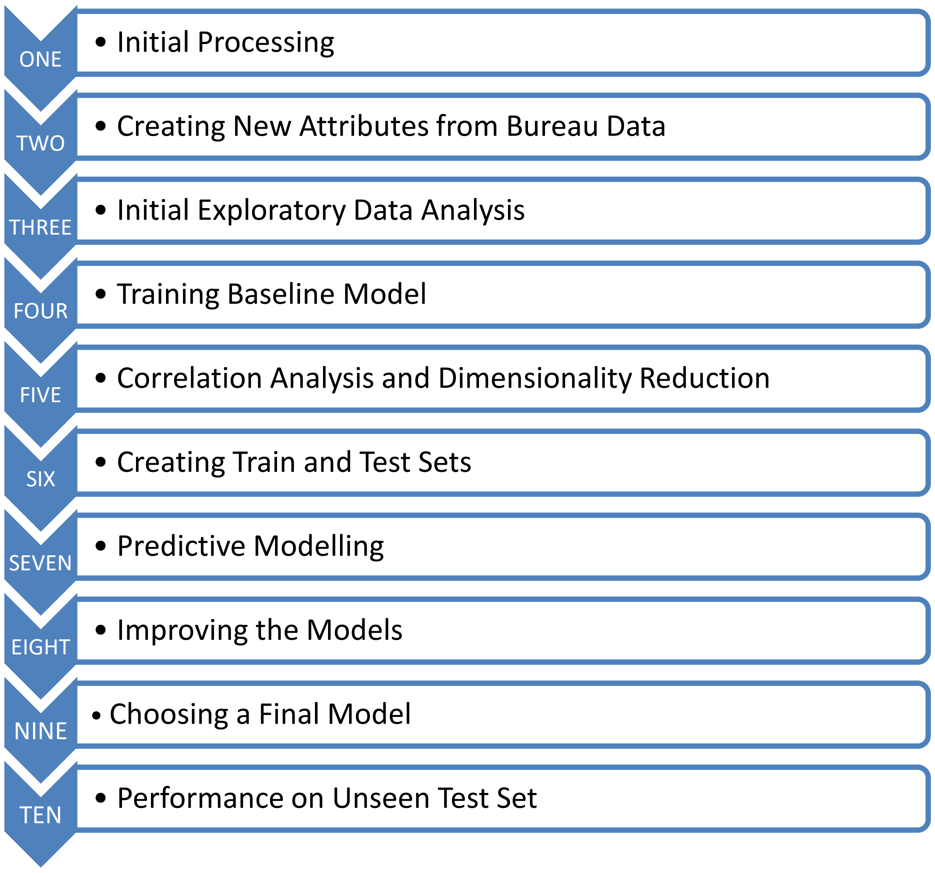
Details on the repayment history of previous loans and credit cards given by Home Credit B.V. for previous applications in *previous_application.csv*. Details such as instalment amount required and instalment amount actually paid are included.

credit_card_balance.csv (credit_card_bal)

Historical monthly information on previous credit cards given by Home Credit B.V. for previous applications in *previous_application.csv*. Contains details such as outstanding monthly balance, credit card limit, monthly payment required, monthly payment actually made and so on.

A data dictionary containing descriptions of all attributes for each file was provided by Home Credit B.V. and is available at: https://www.kaggle.com/c/home-credit-default-risk/data

4. Approach, Methodology and Results


ONE
• Initial Processing
TWO
• Creating New Attributes from Bureau Data
THREE
• Initial Exploratory Data Analysis
FOUR
• Training Baseline Model
FIVE
• Correlation Analysis and Dimensionality Reduction
SIX
• Creating Train and Test Sets
SEVEN
• Predictive Modelling
EIGHT
• Improving the Models
NINE
• Choosing a Final Model
TEN
• Performance on Unseen Test Set

Step 1: Initial Processing

During initial processing, we took a first look at the data, checked dimensions, counted missing values, removed some extreme ‘outliers’, cleaned up inconsistent values and deleted some low variance attributes.

Full details and code for initial processing can be seen at:

https://github.com/netvigate888/credit_default_prediction/tree/master/01_Initial_Processing

1) Loading the Data

The data files were loaded into Python pandas dataframes:

File Name	Dataframe Name
application_train.csv	app
bureau.csv	bureau
bureau_balance.csv	bureau_balance
previous_application.csv	prev_app
pos_cash_balance.csv	pos_cash_bal
installments_payments.csv	instal_pay
credit_card_balance.csv	credit_card_bal

2) Dimensions and Attribute Types for Each Dataframe

The dimensions and attribute types of each dataframe were checked.

Table 1: Dimensions and attribute types for each dataframe

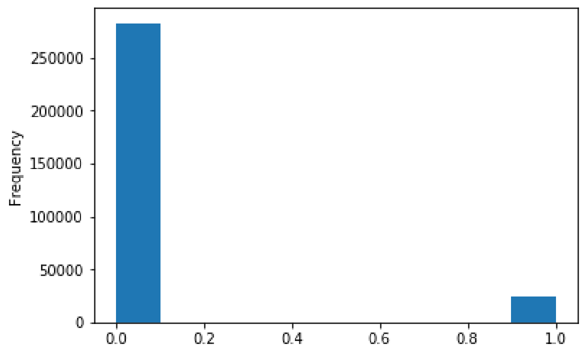
	app	bureau	bureau_balance	prev_app	pos_cash_bal	instal_pay	credit_card_bal
Attributes	122	17	3	37	8	8	23
Observations	307511	1716428	27299925	1670214	10001358	13605401	3840312
Categorical	16	3	1	16	1	0	1
Numeric	106	14	2	21	7	8	22

Given the large size of the dataset and mix of static and historical time series data, the decision was made, for the purposes of this report, to focus analysis on the first three dataframes only: app, bureau and bureau_balance.

3) Balance of Target Variable

The target variable is contained in the main dataframe *app* and is imbalanced. See Figure 2. The challenges brought by an imbalanced dataset were addressed during the design of the prediction experiments. See *Step7, Section 1) Experiment Design*.

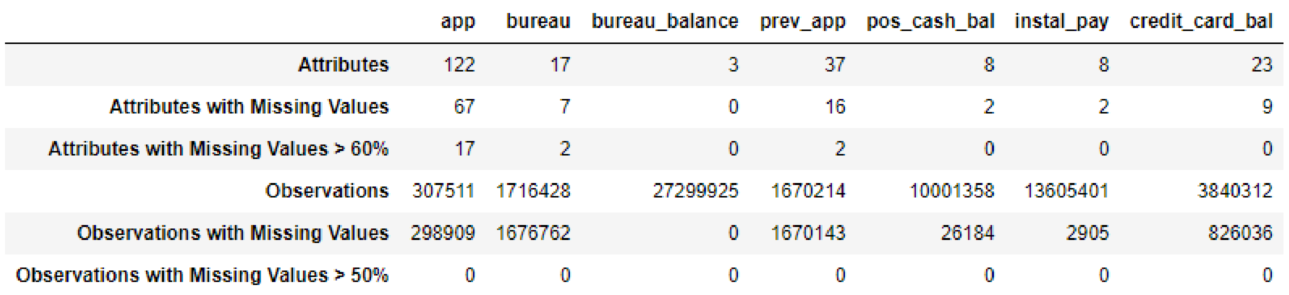
Figure 2: Imbalanced nature of target variable



	Count	Percent (%)
0	282686	91.9
1	24825	8.1

4) Missing Values

Table 2: Missing value counts for each dataframe



	app	bureau	bureau_balance	prev_app	pos_cash_bal	instal_pay	credit_card_bal
Attributes	122	17	3	37	8	8	23
Attributes with Missing Values	67	7	0	16	2	2	9
Attributes with Missing Values > 60%	17	2	0	2	0	0	0
Observations	307511	1716428	27299925	1670214	10001358	13605401	3840312
Observations with Missing Values	298909	1676762	0	1670143	26184	2905	826036
Observations with Missing Values > 50%	0	0	0	0	0	0	0

The data contains various missing values so the initial dataframes of interest (*app*, *bureau* and *bureau_balance*) were examined more closely. A rather arbitrary level of 80% was chosen as the threshold to remove attributes or observations with too many missing values. None of the three dataframes had attributes with missing values greater than 80%. The closest was the attribute *AMT_ANNUITY* within *bureau* with 71.5% missing values. As such, no attributes were removed for having too many missing values.

Similarly, there were no observations with missing values greater than 80% (in fact, there were no observations with missing values greater than 50%). Therefore, no observations were removed for having too many missing values.

5) Data Types

The data type of each attribute across the three dataframes was checked and found to be appropriate.

6) Distributions, Consistency and Variance of Attributes

a) Numerical Attributes

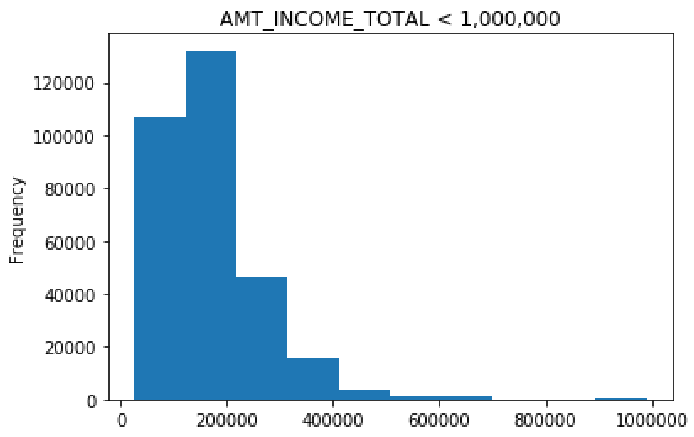
The numerical attributes of each dataframe were examined using the summary statistics: count, mean, standard deviation, minimum, maximum and 25%, 50%, 75% quantiles. Distributions, consistency and variance were further checked using histogram plots and frequency tables.

Some attributes were 'noisy' with a handful of potential 'outliers' or values inconsistent with the definition of the attribute. The volume of 'outliers' was very low, typically well below 1% of all observations, so in most cases the 'outliers' were left as-is. In cases where there were obviously extreme values, the 'outliers' were set to NaN. Similar to 'outliers', inconsistent values were very low in volume and set to NaN.

Also, a number of numerical attributes were removed at this stage for having low variance.

Figure 3: Example of potential 'outliers'

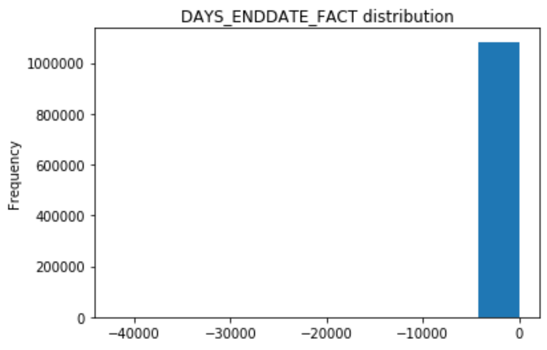
```
Number of observations above 100,000,000 is: 1
Number of observations above 1,000,000 is: 250
Percentage of observations above 1,000,000 is: 0.08%
```



The attribute above represents the total income of an applicant and comes from the main *app* dataframe with 307,511 observations. There is a small number of potential 'outliers' above 1,000,000 (which were left as-is) and one obviously extreme value above 100,000,000 (which was set to NaN).

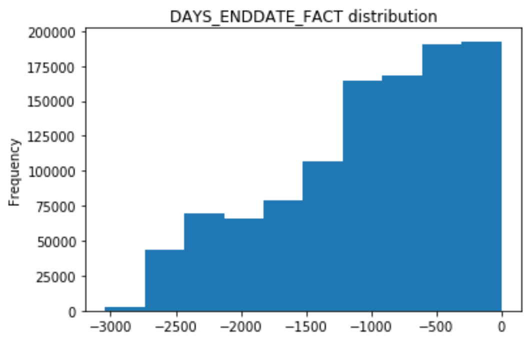
Figure 4: Example of a single extreme 'outlier'

```
Distribution with extreme value.
```



```
Frequency count of observations below -3000:

-3042.0      1
-42023.0     1
Name: DAYS_ENDDATE_FACT, dtype: int64
```

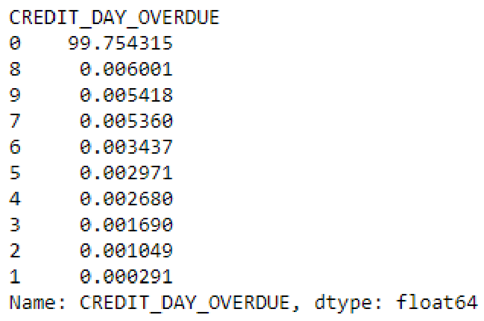
```
The distribution looks better with the extreme value removed.
```



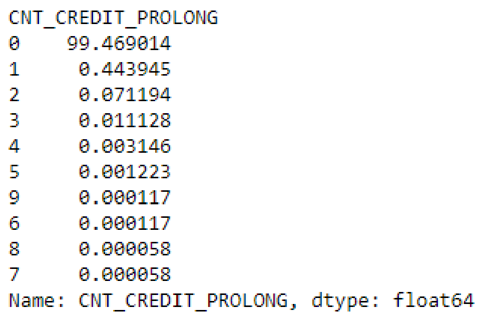
The attribute above comes from the *bureau* dataframe and represents the number of days since the credit in the bureau data was closed (only applicable for closed credits). There was an extreme 'outlier' below -42,000 (equivalent to over 115 years) which was set to NaN. As can be seen above, the distribution looks much better with the 'outlier' removed.

Table 3: Example of low variance attributes

Frequency tables for observation values expressed as percentage.



```
CREDIT_DAY_OVERDUE
0    99.754315
8     0.006001
9     0.005418
7     0.005360
6     0.003437
5     0.002971
4     0.002680
3     0.001690
2     0.001049
1     0.000291
Name: CREDIT_DAY_OVERDUE, dtype: float64
```



```
CNT_CREDIT_PROLONG
0    99.469014
1     0.443945
2     0.071194
3     0.011128
4     0.003146
5     0.001223
9     0.000117
6     0.000117
8     0.000058
7     0.000058
Name: CNT_CREDIT_PROLONG, dtype: float64
```

The two attributes above come from the *bureau* dataframe and represent the number of days a credit is overdue and the number of times a credit was extended respectively. Over 99% of the values were zero so both attributes were removed.

b) Categorical Attributes

The number of unique levels for each categorical attribute in the three dataframes was checked. Most of the attributes had a small number of levels with the exception of OCCUPATION_TYPE and ORGANIZATION_TYPE from the *app* dataframe with 18 and 58 levels respectively. Categorical attributes were one-hot encoded at a later stage.

Table 4: Dataframe dimensions following initial processing

Dataframe	Dimensions (rows x columns)
app	307,511 x 99
bureau	1,716,428 x 12
bureau_balance	27,299,925 x 3

Step 2: Creating New Attributes from Bureau Data

Recall that the *bureau* dataframe contains information about credits extended by other financial institutions to applicants in the main dataframe *app*. *bureau_balance* contains repayment histories (where available) for the credits in *bureau*.

In this step, we created new applicant level attributes from the data in *bureau* and *bureau_balance*.

Full details and code can be seen at:

https://github.com/netvigate888/credit_default_prediction/tree/master/02_Attributes_from_Bureau_data

1) A New Attribute from *bureau_balance*

For each credit in *bureau* there is (where available) a time series in *bureau_balance* containing details of the repayment history of the applicant for that credit. The information is a simple flag indicating, at that time, the status of the credit:

X = unknown
C = closed
0 = no payment overdue
1 = payment 0-30 days overdue
2 = payment 31-60 days over due
3= payment 61-90 days over due
4 = payment 91-120 days over due
5 = 120+ days overdue, sold or written off

We reduced each time series to a single flag based on a hierarchical labeling scheme starting with the worst status down to the best status:

1. First look for a '5'; if the series contains a '5' then it is labeled as '5'
2. Next look for a '4'; if the series contains a '4' then it is labeled as '4'

and so on down to 'X'

This new attribute was then joined with the *bureau* dataframe using the unique credit ID (SK_BUREAU_ID) as a key.

Not every credit in *bureau* had a repayment history in *bureau_balance*. As such, there were some NaN values in our new attribute following the join. The NaN values were replaced with 'NoHist' to show that there was no repayment history available for that credit.

2) New Attributes from *bureau*

Each applicant in the main *app* dataframe may have zero, one or many credits in *bureau*. The data in *bureau* needed to be combined at the individual applicant level.

All the numeric attributes in *bureau* were aggregated at the individual applicant level with count, maximum, minimum and mean values.

All categorical attributes in *bureau* were first one-hot encoded and then aggregated at the individual applicant level with a count and a ‘fraction’ value representing the proportion of credits with that categorical value for each applicant.

3) Joining New Attributes with *app*

The newly created attributes were joined with the main dataframe *app* using the unique applicant ID (SK_ID_CURR) as the key. The *bureau* and *bureau_balance* dataframes were no longer needed; all future worked focused on the expanded *app* dataframe.

Table 5: *app* dataframe dimensions following addition of new attributes

Dataframe	Dimensions (rows x columns)
app	307,511 x 191

Step 3: Initial Exploratory Data Analysis

The *app* dataframe was explored to identify the characteristics of defaulting loan applicants.

The full exploratory analysis and code can be seen at:

https://github.com/netvigate888/credit_default_prediction/tree/master/03_Initial_EDA

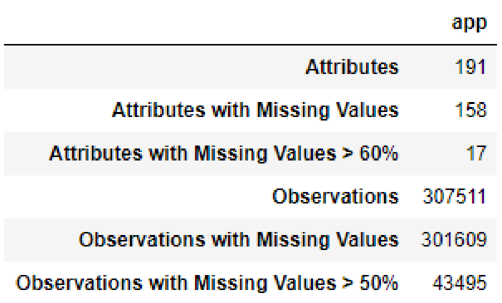
Recall that the target attribute is split 8.1% default and 91.9% non-default. See Figure 2. Therefore, the average default rate across the dataset for all applicant types is 8.1%.

Some examples of the exploratory analysis are shown below.

1) Checking Missing Values

Missing values were checked again given that many new attributes were added to the main dataframe *app*.

Table 6: Missing values in the expanded *app* dataframe



	app
Attributes	191
Attributes with Missing Values	158
Attributes with Missing Values > 60%	17
Observations	307511
Observations with Missing Values	301609
Observations with Missing Values > 50%	43495

The increase in observations with missing values above 50% (from zero in the original *app* dataframe) is due to the newly created attributes from *bureau* and *bureau_balance*. There are applicants in *app* that do not have data in *bureau* or *bureau_balance*. These are the observations that now have the increased number of missing values. There were no attributes or observations with missing values above 80%.

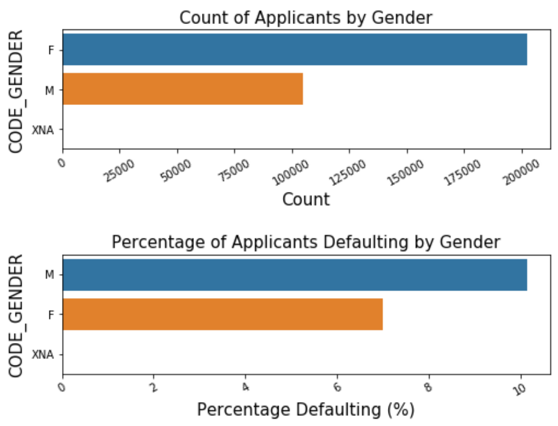
2) Exploring the Data

a) Some Categorical Attributes

i. Gender

There are significantly more woman applicants than men. Also, woman appear less likely to default than men.

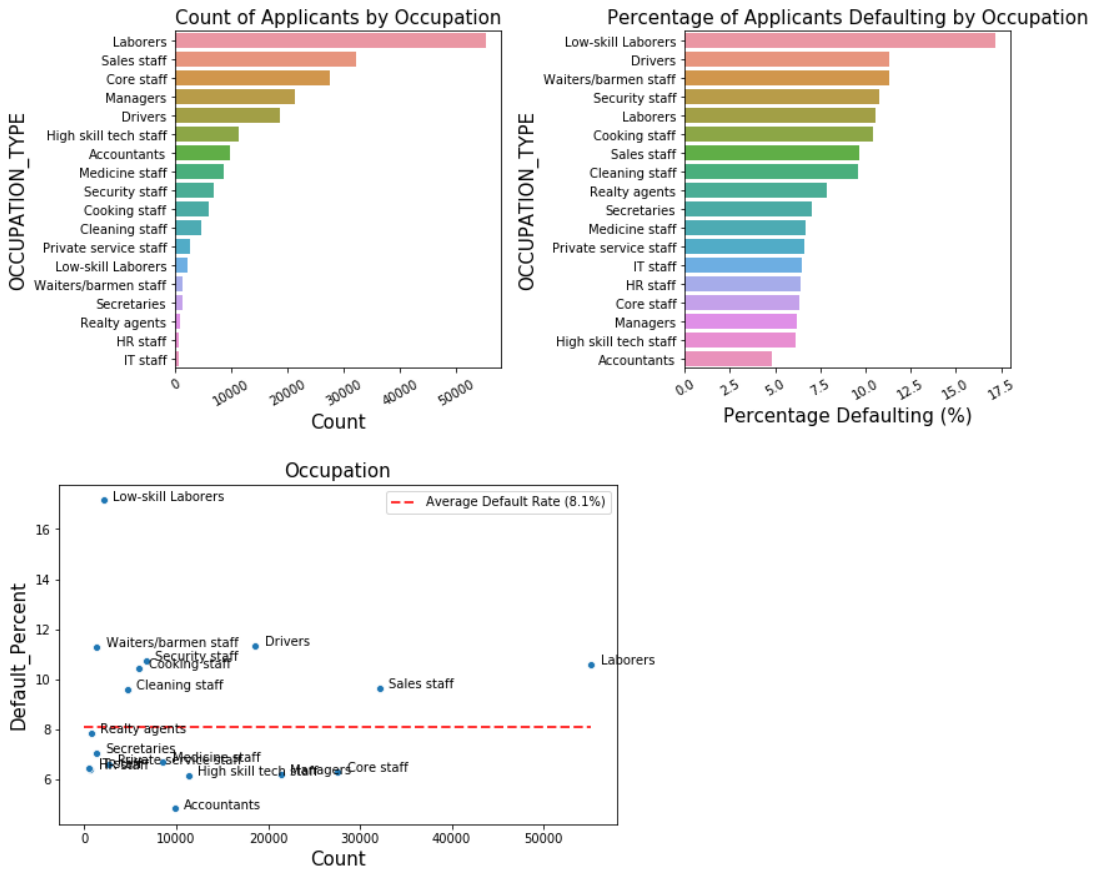
Figure 5: Counts and default percentages by gender



ii. Occupation

'Laborers', 'Sales staff' and 'Drivers' appear to be creating more risk for the business as they make up a large proportion of applicants and have a higher default rate than average around 10-11%. The business should consider changing its mix of applicants by targeting occupations currently with low volume and lower risk of default such as 'IT staff' and 'HR staff'.

Figure 6: Counts and default percentages by occupation

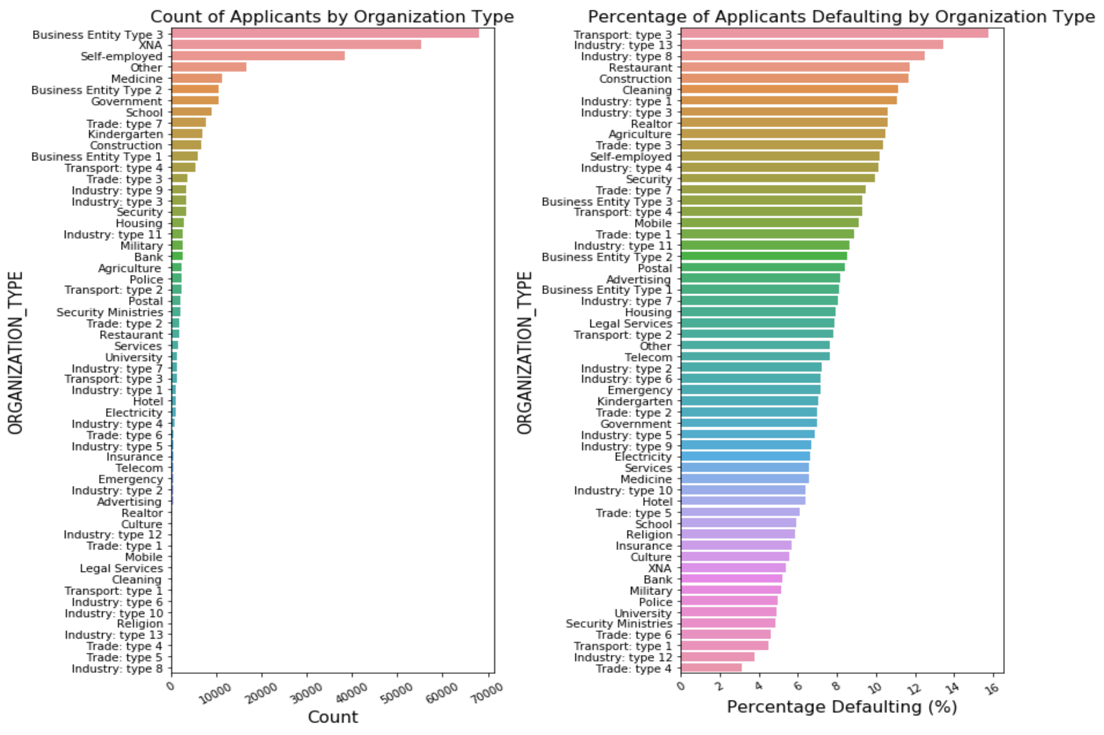


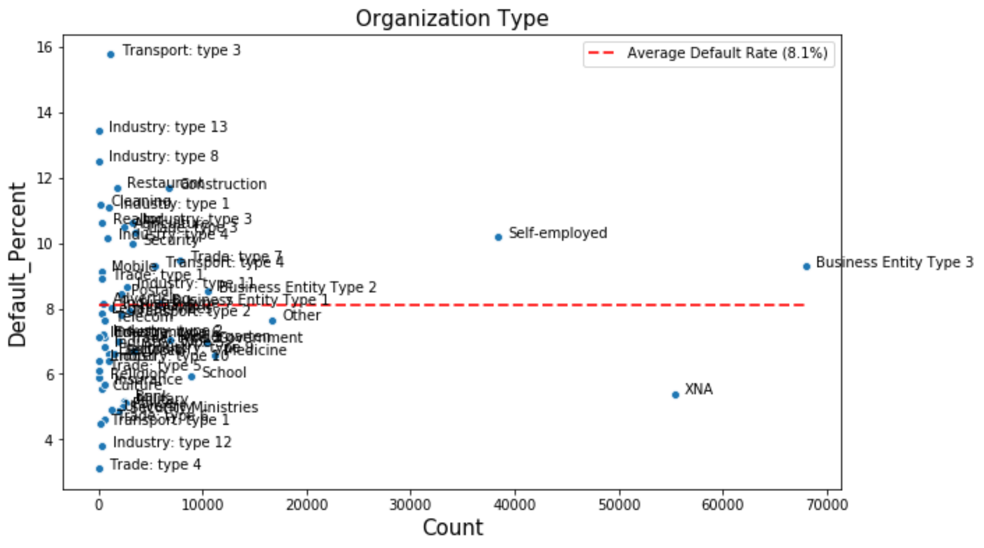
iii. Organization Type

Two of the high-volume organization types, 'Business Entity Type 3' and 'Self-employed', have higher defaults rates than average in the range of 9-10%. The business should attempt to reduce these organization types.

It was discovered during correlation analysis (see *Step 5*) that organization type 'XNA' corresponds to retired persons, who make up a large proportion of applicants and have a default rate well below the average (around 5%). Home Credit B.V. should continue to focus on such applicants.

Figure 7: Counts and default percentages by organization type



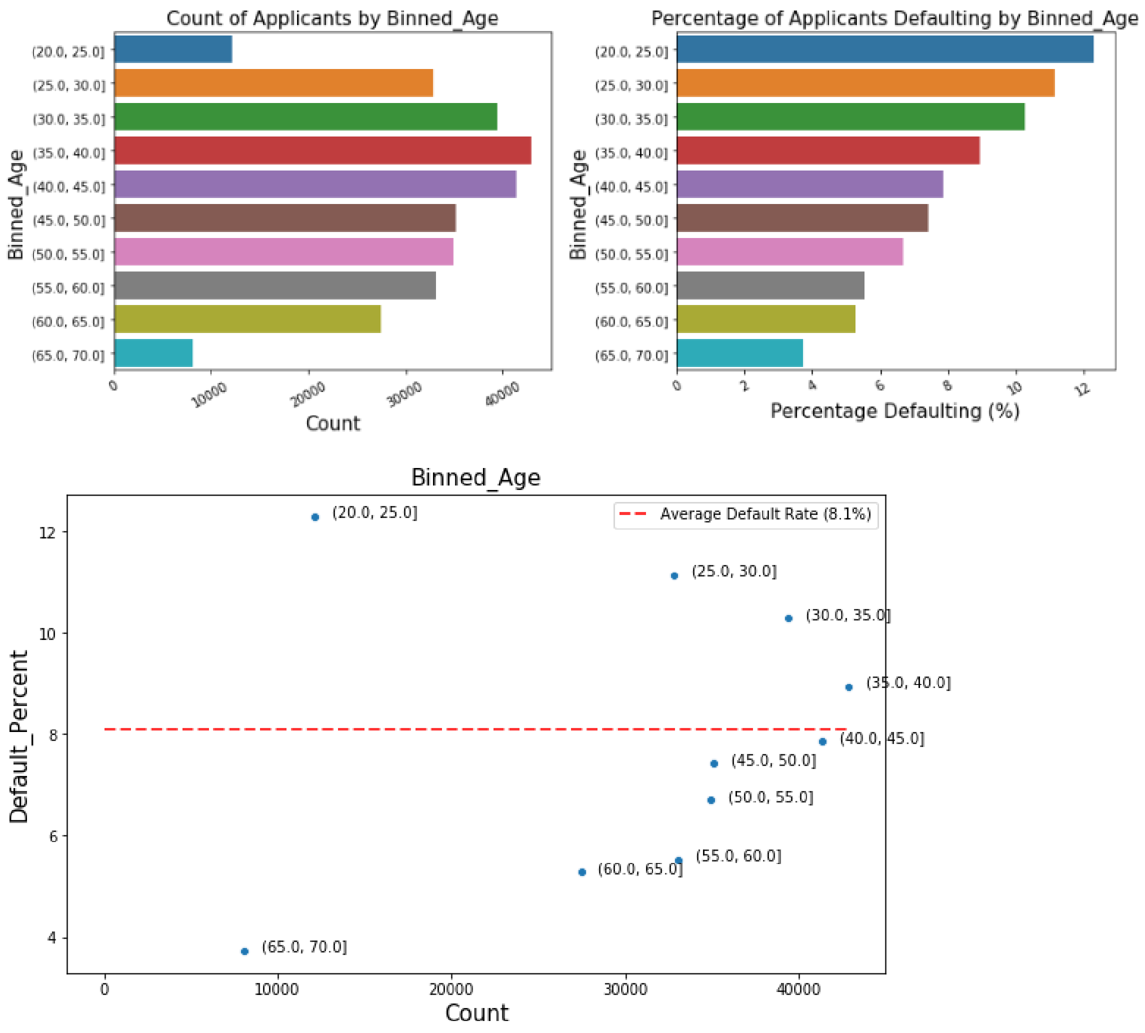


b) Some Numerical Attributes

i. Age

The attribute DAYS_BIRTH was converted to years and split into 'bins' to allow for analysis and plotting. There appears to be a correlation between age and default rate with younger applicants more likely to default. The age range 25-40 years represents the largest risk to the business with a high number of applicants and higher than average default rates. The range 20-25 years has the highest default rate but only represents a small proportion of applicants.

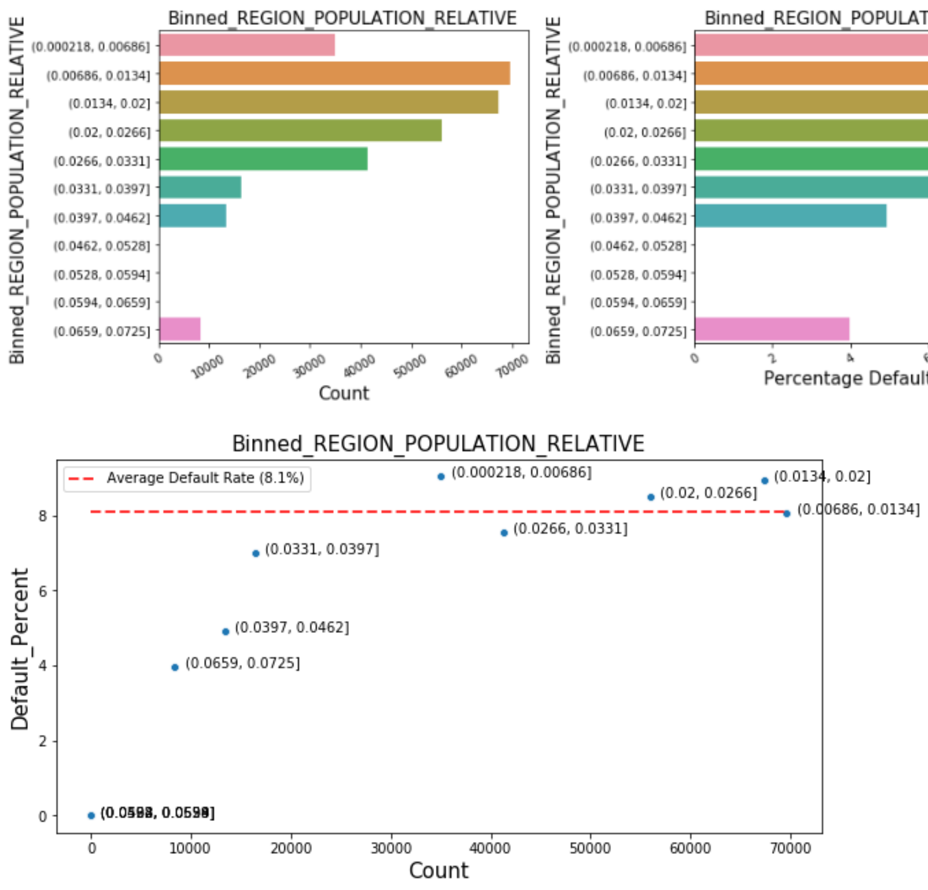
Figure 8: Counts and default percentages by binned age



ii. Regional Population

The default rate tends to drop as the relative size of population rises. The largest portion of applicants comes from the lower population areas suggesting Home Credit B.V. may be doing most of its business in small towns and rural areas.

Figure 9: Counts and default percentages by binned regional population

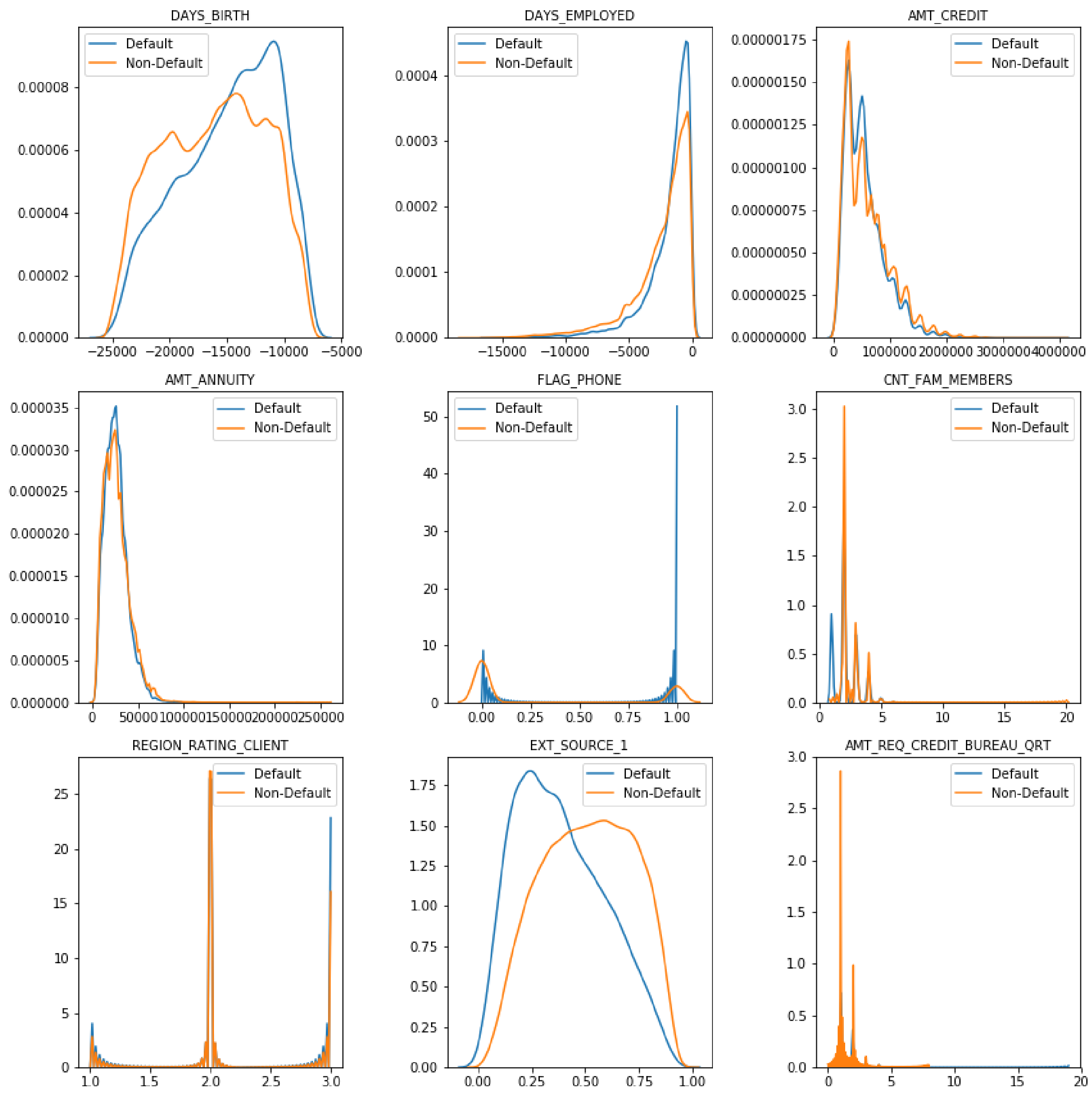


iii. Density Plots for some Original Attributes

Density plots, split between defaulting and non-defaulting applicants, for numerical attributes were examined. A difference between the distributions may indicate an attribute with predictive power.

Unfortunately, there's no obvious difference in the distributions shown below.

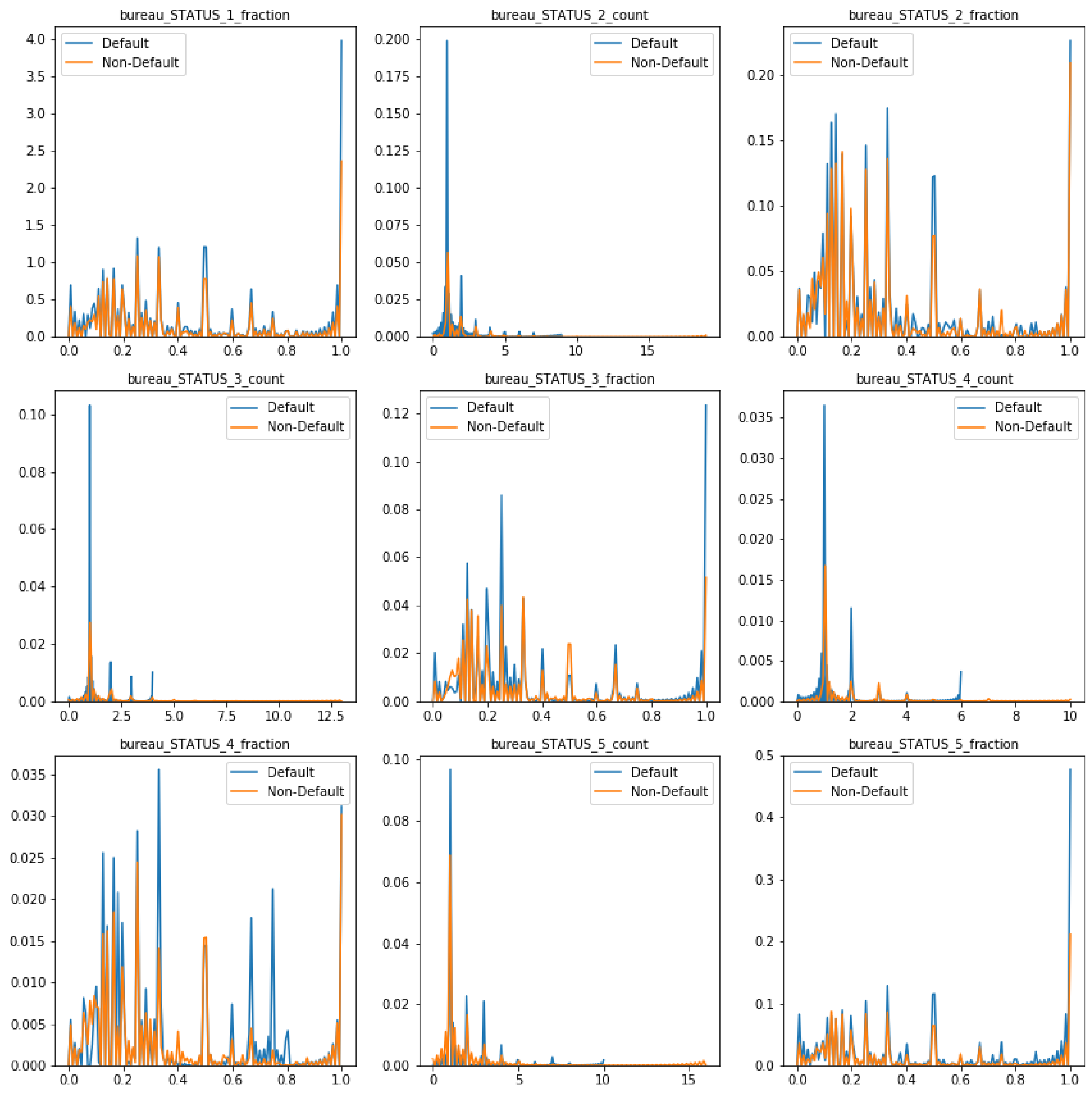
Figure 10: Density plots for some original attributes



iv. Density Plots for some New Attributes from Bureau Data

Density plots for some of the new attributes created from the bureau data are shown below. The distributions are erratic but there does appear to be some areas of difference between the default and non-default distributions. For example, within 'bureau_STATUS_2_count', the default distribution has a higher concentration of observations at the 1 and 2 counts. The non-default distribution appears to be much more skewed to the right with a long thin tail.

Figure 11: Density plots for attributes created from bureau data



Step 4: Training Baseline Model

Baseline predicative performance metrics were established with a Logistic Regression model using default hyperparameters (from Python's scikit-learn module) and the preprocessed *app* dataframe BEFORE adding the newly created bureau attributes. The aim was to train a simple model with the base data to demonstrate the improvements that could be achieved with the addition of new attributes from the bureau data, different models and hyperparameter tuning.

Experiment design and performance metrics are explained in *Step7, Section 1) Experiment Design*.

Full details and code can be seen at:

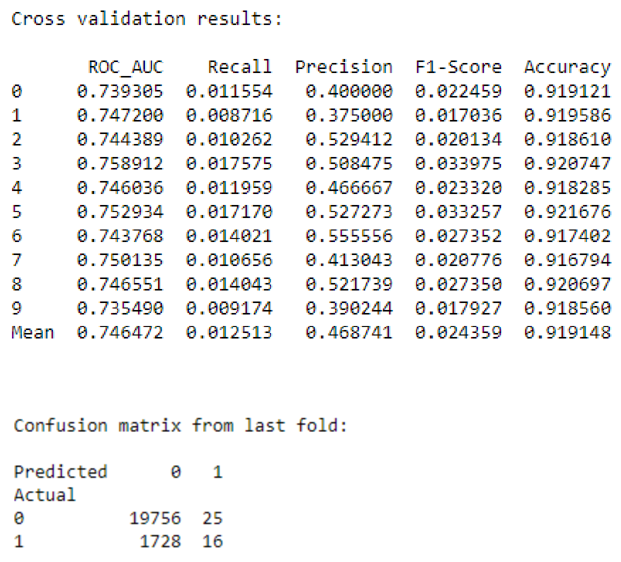
https://github.com/netvigate888/credit_default_prediction/tree/master/04_Create_Baseline_Model

First the data required some further processing to prepare for predictive modelling:

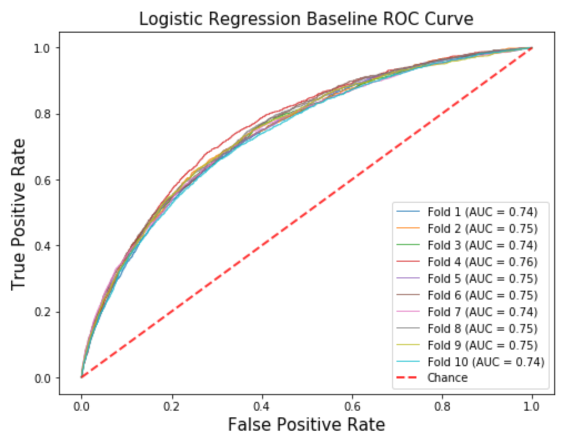
- All categorical attributes were one-hot encoded.
- The data was split into a 70% training set and 30% test set on a stratified basis around the target attribute.
- Label sets were created for the training and test sets containing the target values.
- Missing values were imputed using median values derived from the training set only. The derived missing values were then applied to the test set.
- All attributes in the training set were 'standard scaled' to have a mean of zero and standard deviation of 1. The scaling learned from the training set was then applied to the test set.

The Logistic Regression model was trained using the training set and ten-fold cross-validation.

Table 7: Baseline model results



```
Cross validation results:

       ROC_AUC    Recall   Precision  F1-Score  Accuracy
0     0.739305  0.011554   0.400000  0.022459  0.919121
1     0.747200  0.008716   0.375000  0.017036  0.919586
2     0.744389  0.010262   0.529412  0.020134  0.918610
3     0.758912  0.017575   0.508475  0.033975  0.920747
4     0.746036  0.011959   0.466667  0.023320  0.918285
5     0.752934  0.017170   0.527273  0.033257  0.921676
6     0.743768  0.014021   0.555556  0.027352  0.917402
7     0.750135  0.010656   0.413043  0.020776  0.916794
8     0.746551  0.014043   0.521739  0.027350  0.920697
9     0.735490  0.009174   0.390244  0.017927  0.918560
Mean  0.746472  0.012513   0.468741  0.024359  0.919148


Confusion matrix from last fold:

Predicted      0    1
Actual
0          19756   25
1           1728   16
```



The baseline model results are rather weak. Mean Recall is just 1% and mean Precision 47%. Accuracy looks high at 92% but this is a misleading measure given the imbalanced nature of the data; simply assigning all observations to 0 (non-default) would also give an Accuracy of 92%. The model is struggling with the imbalanced data; the vast majority of observations are being classified as 0 (non-default).

Step 5: Correlation Analysis and Dimensionality Reduction

Returning to the full data set including the attributes created from the bureau data, correlation analysis was performed to identify and remove attributes with a strong correlation to another attribute.

Full analysis and code can be seen at:

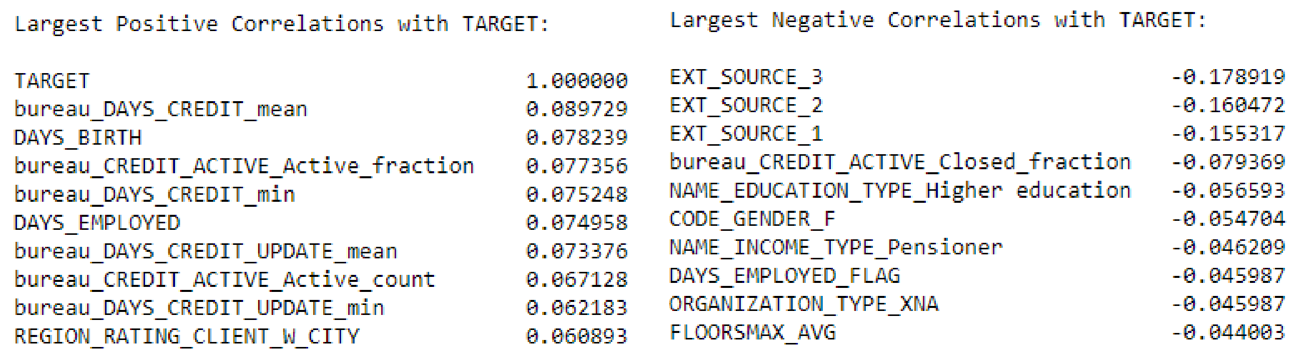
https://github.com/netvigate888/credit_default_prediction/tree/master/05_Correlation_Analysis_and_Dim_Reduction

First, all categorical attributes were one-hot encoded for correlation analysis.

1) Correlation with Target Attribute

All correlations with the target attribute were calculated.

Table 8: Attributes with strongest correlation to target



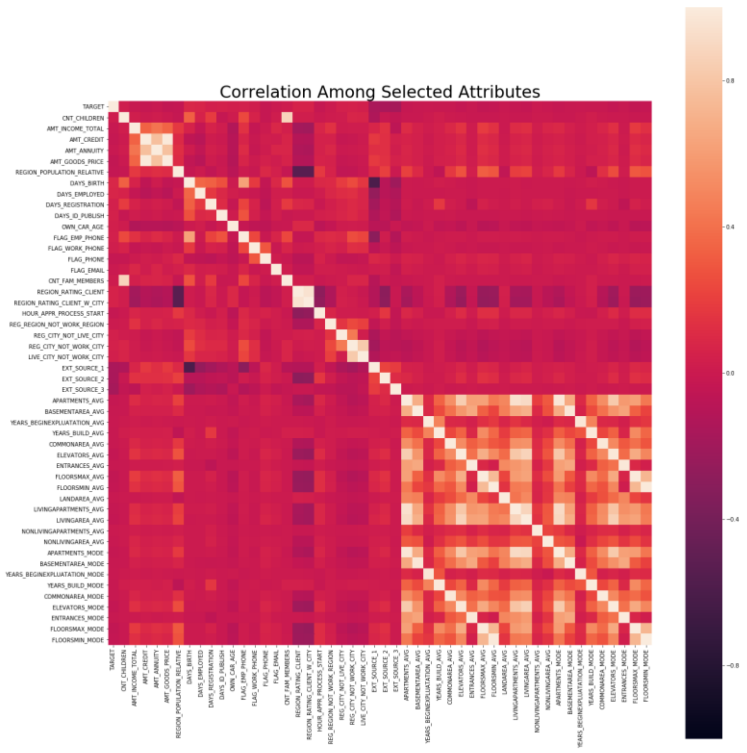
```
Largest Positive Correlations with TARGET:          Largest Negative Correlations with TARGET:

TARGET                                  1.000000    EXT_SOURCE_3                             -0.178919
bureau_DAYS_CREDIT_mean                 0.089729    EXT_SOURCE_2                             -0.160472
DAYS_BIRTH                              0.078239    EXT_SOURCE_1                             -0.155317
bureau_CREDIT_ACTIVE_Active_fraction    0.077356    bureau_CREDIT_ACTIVE_Closed_fraction     -0.079369
bureau_DAYS_CREDIT_min                  0.075248    NAME_EDUCATION_TYPE_Higher education     -0.056593
DAYS_EMPLOYED                           0.074958    CODE_GENDER_F                            -0.054704
bureau_DAYS_CREDIT_UPDATE_mean          0.073376    NAME_INCOME_TYPE_Pensioner               -0.046209
bureau_CREDIT_ACTIVE_Active_count       0.067128    DAYS_EMPLOYED_FLAG                       -0.045987
bureau_DAYS_CREDIT_UPDATE_min           0.062183    ORGANIZATION_TYPE_XNA                    -0.045987
REGION_RATING_CLIENT_W_CITY             0.060893    FLOORSMAX_AVG                            -0.044003
```

None of the above correlations show any significant strength but it's interesting to note that a few of the attributes created from the bureau data (labeled with a prefix of 'bureau') are among those most correlated to the target.

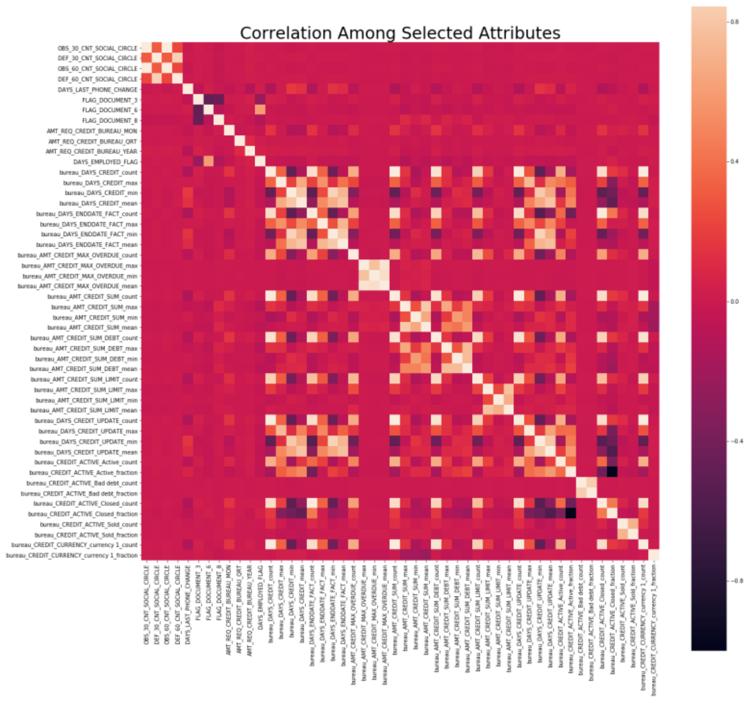
2) Correlation Among Attributes

The heatmaps below show example correlations among attributes.

Figure 12: Correlation among selected attributes



The heatmap clearly shows that strong correlations exist between attributes. For example, there's a high positive correlation between AMT_GOODS_PRICE and AMT_CREDIT. There are also numerous high correlations among the housing data attributes such as BASEMENTAREA_AVG and BASEMENTAREA_MODE. These attributes will be candidates for removal.

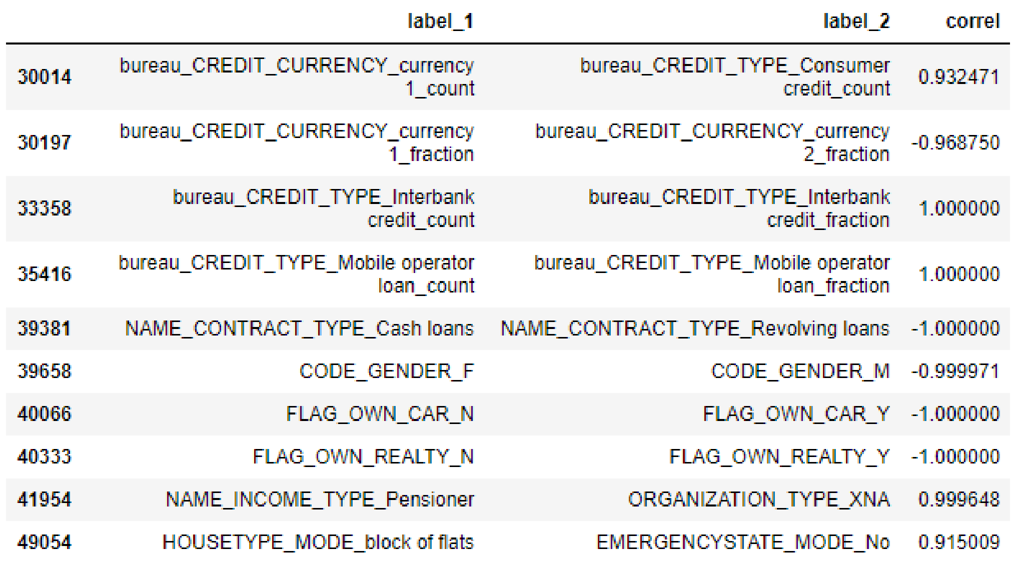


This heatmap shows many strong correlations among the attributes that were created from the bureau data. Again, these attributes will be candidates for removal.

3) Removing Attributes with Strong Correlations

All unique pairs of attributes with an absolute correlation greater than 90% were identified. A further list of one attribute from each pair was created; these attributes were removed. In total, 54 attributes were removed. The updated dataset was saved for use in next steps.

Table 9: Examples of attribute pairs with absolute correlation greater than 90%



	label_1	label_2	correl
30014	bureau_CREDIT_CURRENCY_currency 1_count	bureau_CREDIT_TYPE_Consumer credit_count	0.932471
30197	bureau_CREDIT_CURRENCY_currency 1_fraction	bureau_CREDIT_CURRENCY_currency 2_fraction	-0.968750
33358	bureau_CREDIT_TYPE_Interbank credit_count	bureau_CREDIT_TYPE_Interbank credit_fraction	1.000000
35416	bureau_CREDIT_TYPE_Mobile operator loan_count	bureau_CREDIT_TYPE_Mobile operator loan_fraction	1.000000
39381	NAME_CONTRACT_TYPE_Cash loans	NAME_CONTRACT_TYPE_Revolving loans	-1.000000
39658	CODE_GENDER_F	CODE_GENDER_M	-0.999971
40066	FLAG_OWN_CAR_N	FLAG_OWN_CAR_Y	-1.000000
40333	FLAG_OWN_REALTY_N	FLAG_OWN_REALTY_Y	-1.000000
41954	NAME_INCOME_TYPE_Pensioner	ORGANIZATION_TYPE_XNA	0.999648
49054	HOUSETYPE_MODE_block of flats	EMERGENCYSTATE_MODE_No	0.915009

Table 10: Dimensions following one-hot encoding and removal of attributes with strong correlations

Dataframe	Dimensions (rows x columns)
app	307,511 x 261

Step 6: Creating Train and Test Sets

At this stage final preparations were made for predicative modelling. Up to this point, we have examined the data, set extreme 'outliers' and inconsistent values to NaN, added new attributes created from the bureau data, one-hot encoded categorical attributes and removed attributes with a strong correlation to another attribute.

The code for this step can be seen at:

https://github.com/netvigate888/credit_default_prediction/tree/master/06_Train_Test_Set_Creation

As with the baseline model, the following further processing steps were performed:

- The data was split into a 70% training set and 30% test set on a stratified basis around the target attribute.
- Label sets were created for the training and test sets containing the target values.

- Missing values were imputed using median values derived from the training set only. The derived missing values were then applied to the test set.
- All attributes in the training set were 'standard scaled' to have a mean of zero and standard deviation of 1. The scaling learned from the training set was then applied to the test set.

The training, test and label sets were saved for use in the predictive modelling step.

Step 7: Predictive Modelling

Three models suitable for binary classification and employing different methodologies were trained and tested:

1. Logistic Regression – a statistical model
2. Random Forest – a decision tree-based ensemble method
3. AdaBoost – a boosting algorithm using decision trees

All results and code can be seen at:

https://github.com/netvigate888/credit_default_prediction/tree/master/07_Predictive_Models

1) Experiment Design

Each model was trained and tested using ten-fold cross-validation on the training set only. The test set remained unseen at all times until the final testing stage (see *Step 10*).

Sampling Methodology

In addition to training and testing with the data in its original imbalanced form, two sampling techniques were also used to create balanced training sets:

1. Over-sampling the minority class (target = 1) using Synthetic Minority Over-sampling Technique (SMOTE)
2. Randomly under-sampling the majority class (target = 0)

The over/under-sampling was executed within each fold, on only the fold training set, after the data had been split into its training and test sets for the fold. This ensured there was no 'bleeding' of the fold training data into the fold test data as a result of the sampling. Consequently (and correctly), the fold test set remained in its imbalanced form.

Performance Metrics

Given the predication task is binary classification, the following metrics were selected and calculated from the confusion matrix at each fold:

Recall (True Positive Rate):	TP / (TP+FN) The percentage of defaulting applicants correctly classified.
Precision:	TP / (TP+FP) The percentage of predicated defaulting applicants that were actually defaulting applicants.
F1-Score:	2 x Recall x Precision / (Recall + Precision) The harmonic mean of Recall and Precision. Provides a measure of the 'balance' achieved between Recall and Precision.
Accuracy:	(TP + TN) / (TP+FP+TN+FN) The percentage of applicants correctly classified. Where: TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative
ROC_AUC:	Each model is also capable of returning a prediction probability for each class of the target. For reference only, the area under the Receiver Operating Characteristic curve (ROC) was also calculated.

Which Performance Metric to Optimize?

As discussed above in *Step 4: Training Baseline Model*, Accuracy is not a good measure for an imbalanced dataset. Recall, Precision and the F1-Score provide a more detailed insight into performance. However, when optimizing prediction models, an improvement in Recall typically leads to a decrease in Precision and vice-versa. A choice is usually made, based on the context of the problem, to optimize one metric over the other, while maintaining the F1-Score as high as possible. For Home Credit B.V., this amounts to deciding which is the more expensive mistake (i) incorrectly classifying an applicant as defaulting (false positive) and turning down the application or (ii) incorrectly classifying an applicant as non-defaulting (false negative), approving the application and suffering the future default?

A simple cost function:

Loan size:	$5,000
Interest Rate:	10%
Loan term:	12 months

Cost of incorrectly classifying an applicant as defaulting:	loss of revenue = $5,000 x 10% = $500
Cost of incorrectly classifying an applicant as non-defaulting:	loss of loan principal = $5,000

The cost of incorrectly classifying an applicant as non-defaulting is materially more expensive. Therefore, it's preferable to minimize false negatives which is equivalent to maximizing Recall.

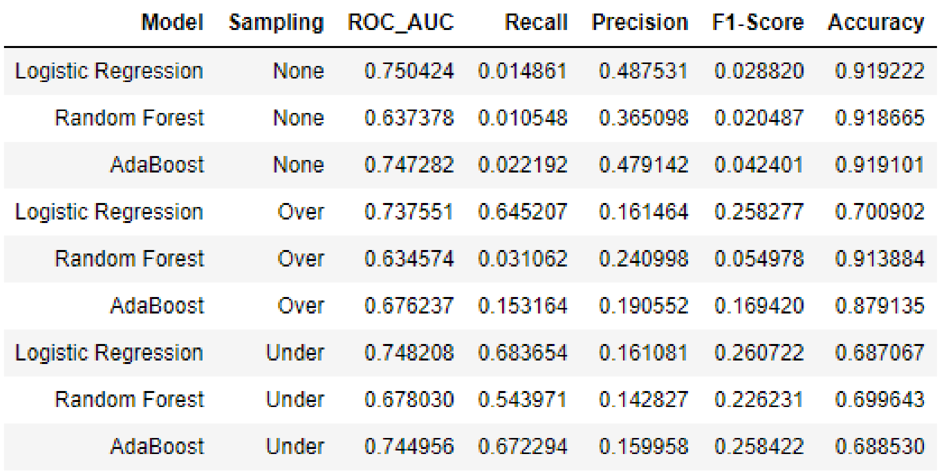
The choice was made to maximize Recall over Precision while keeping the F1-Score as high as possible.

2) Initial Training and Testing

All three models were initially trained and tested with the following conditions:

1) Ten-fold cross-validation
2) Default hyperparameters (from Python's scikit-learn module)
3) Three sampling options: no sampling, over-sampling and under-sampling

Table 11: Summary mean results from cross-validation for each model and sampling technique



Model	Sampling	ROC_AUC	Recall	Precision	F1-Score	Accuracy
Logistic Regression	None	0.750424	0.014861	0.487531	0.028820	0.919222
Random Forest	None	0.637378	0.010548	0.365098	0.020487	0.918665
AdaBoost	None	0.747282	0.022192	0.479142	0.042401	0.919101
Logistic Regression	Over	0.737551	0.645207	0.161464	0.258277	0.700902
Random Forest	Over	0.634574	0.031062	0.240998	0.054978	0.913884
AdaBoost	Over	0.676237	0.153164	0.190552	0.169420	0.879135
Logistic Regression	Under	0.748208	0.683654	0.161081	0.260722	0.687067
Random Forest	Under	0.678030	0.543971	0.142827	0.226231	0.699643
AdaBoost	Under	0.744956	0.672294	0.159958	0.258422	0.688530

While the results for Logistic Regression with no sampling are still poor, both Recall and Precision saw small improvements versus the baseline results. This suggests that the addition of the new attributes from the bureau data likely helped. With no sampling, AdaBoost had the best results (albeit still poor) and Random Forest had the worst results.

With over-sampling, Logistic Regression and AdaBoost saw real improvements with Recall and the F1-Score rising but with Precision falling. However, all three models saw their best improvements in Recall and the F1-Score with under-sampling. **Going forward, all models were assessed using under-sampling.**

3) Model Tuning

Selected hyperparameters for the three models were tuned using a 'grid search' approach to maximize Recall. A new balanced training set was created by randomly under-sampling the majority class (i.e. target = 0 non-default) to facilitate the grid search. **The balanced training set was used for hyperparameter tuning only.**

Table 12: Best hyperparameters determined by grid search

Model	**Best Hyperparameters**
Logistic Regression	'C': 0.001, 'penalty': 'l1'
Random Forest	'max_depth':10, 'max_features': 32, 'min_samples_leaf': 10, 'n_estimators': 500
AdaBoost	'max_depth':1, 'learning_rate': 1, 'n_estimators': 300

The models were then trained and tested again using the original training set, hyperparameters determined by the grid search, ten-fold cross-validation and under-sampling.

Table 13: Model results using best hyperparameters

	ROC_AUC	Recall	Precision	F1-Score	Accuracy
Logistic Regression	0.721779	0.725579	0.135938	0.228970	0.605565
Random Forest	0.744967	0.682176	0.159073	0.257980	0.683230
AdaBoost	0.750388	0.679599	0.163093	0.263047	0.692637

All three models saw an improvement in Recall. Random Forest and AdaBoost also saw an improvement in Precision (and consequently the F1-Score). **Going forward, all models were assessed using the tuned hyperparameters.**

Step 8: Improving the Models

A number of experiments were performed with the aim of improving model performance.

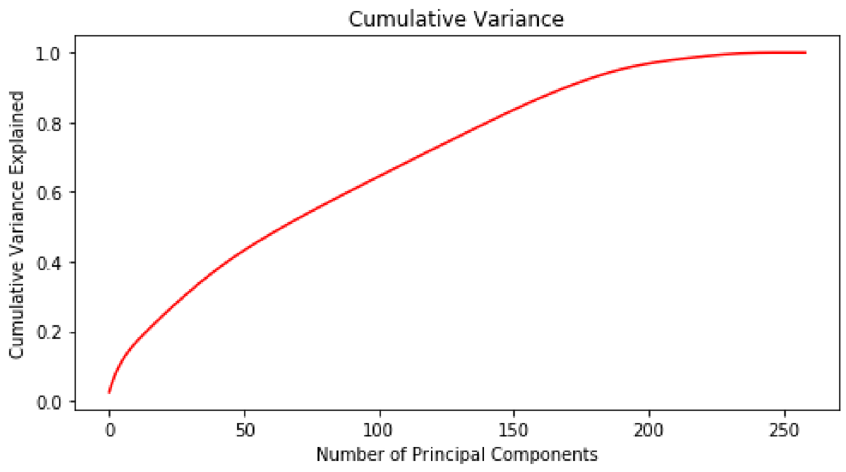
All experiment results and code can be seen at:

https://github.com/netvigate888/credit_default_prediction/tree/master/08_Improve_Models

1) Principal Component Analysis

The training set has a large number of attributes (259 excluding the target). Principal Component Analysis (PCA) was performed to see if the number of attributes could be reduced which, in turn, might lead to improved performance.

Figure 13: Cumulative variance explained by number of principal components



Based on the chart above, a large number of principal components is still required to explain the majority of variance. The Random Forest model was retrained using the first 175 principal components to test the impact on results with a reduced number of principal components.

Table 14: Random Forest results using first 175 principal components

	ROC_AUC	Recall	Precision	F1-Score	Accuracy
Mean result from 10-fold cross-validation	0.714354	0.651728	0.146399	0.239079	0.665103

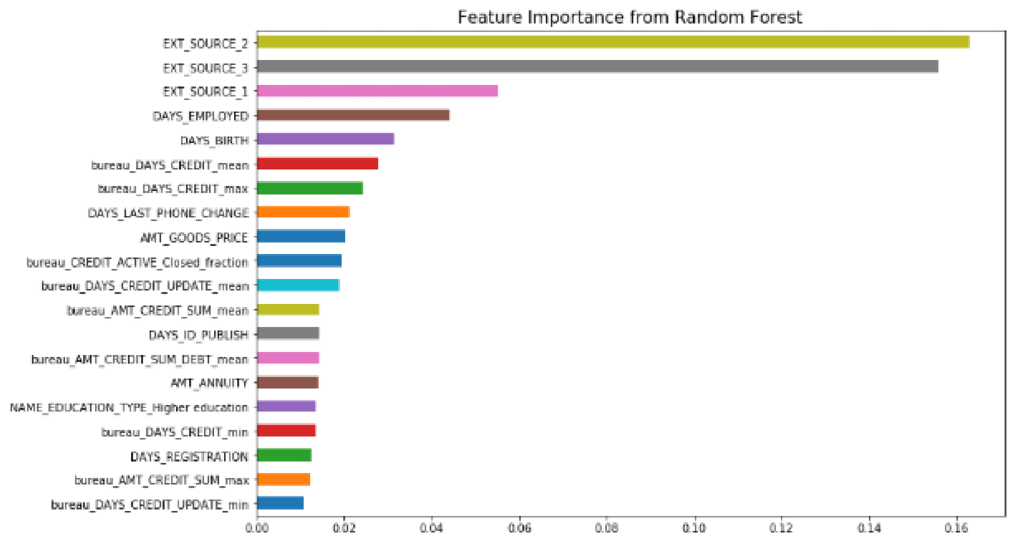
```
Confusion matrix from last fold:

Predicted      0     1
Actual
0          13040  6741
1            631  1113
```

The results were slightly worse than using the full original attribute set. In fact, any increase in the number of principal components still gave results worse than the full original attribute set. Reducing dimensions via PCA did not help performance.

2) Most Important Attributes from Random Forest

The top 20 most important attributes were extracted from the Random Forest model.

Figure 14: Most important attributes from Random Forest



The external credit agency scores appear to be the most important attributes by quite a large margin. It's interesting to note that nine of our created attributes (with prefix ‘bureau’) appear in the top 20.

The Random Forest model was retrained and tested using the top 20 attributes only to understand the impact on results.

Table 15: Random Forest results using top 20 most important attributes

	ROC_AUC	**Recall**	**Precision**	**F1-Score**	**Accuracy**
Mean result from 10-fold cross-validation	0.736140	0.670896	0.155482	0.252449	0.679267

```
Confusion matrix from last fold:

Predicted      0     1
Actual
0          13271  6510
1            576  1168
```

The results from the top 20 most important attributes were only slightly worse than using the full attribute set. This could be an important consideration when designing a model for production – managing 20 attributes would be significantly less work and much less prone to error than managing 259 with only a small drop in performance.

3) Creating New Ratio Attributes

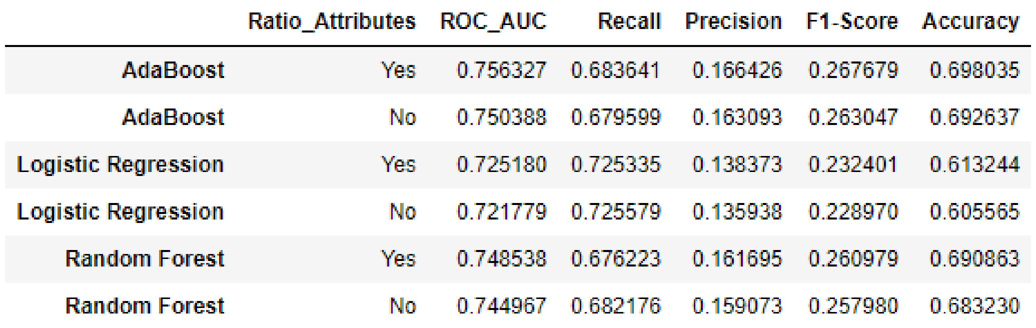
New attributes were engineered based on ratios of the original attributes. The ratios were chosen in an attempt to extract new relevant information from the data. In total, 22 new attributes were created. Some examples are shown below:

• Credit to Annuity:	Amount of Credit Borrowed / Size of Credit Payment
• Credit to Income:	Amount of Credit Borrowed / Amount of Income
• Active Credits to Age:	Number of Active Credits / Age
• Active Credit to Income:	Number of Active Credits / Amount of Income

The new attributes were added back to the original dataset before correlation analysis was performed. Correlation analysis was once again performed on the new expanded dataset, high correlation attributes removed and new training and test sets created. **Please note that the new training and test sets had the exact same observations in each set as the original training and test sets.**

All three models were trained and tested using ten-fold cross-validation on the new expanded training set. The new expanded test set remained unseen at all times.

Table 16: Summary of mean results from cross-validation with and without ratio attributes



	Ratio_Attributes	ROC_AUC	Recall	Precision	F1-Score	Accuracy
AdaBoost	Yes	0.756327	0.683641	0.166426	0.267679	0.698035
AdaBoost	No	0.750388	0.679599	0.163093	0.263047	0.692637
Logistic Regression	Yes	0.725180	0.725335	0.138373	0.232401	0.613244
Logistic Regression	No	0.721779	0.725579	0.135938	0.228970	0.605565
Random Forest	Yes	0.748538	0.676223	0.161695	0.260979	0.690863
Random Forest	No	0.744967	0.682176	0.159073	0.257980	0.683230

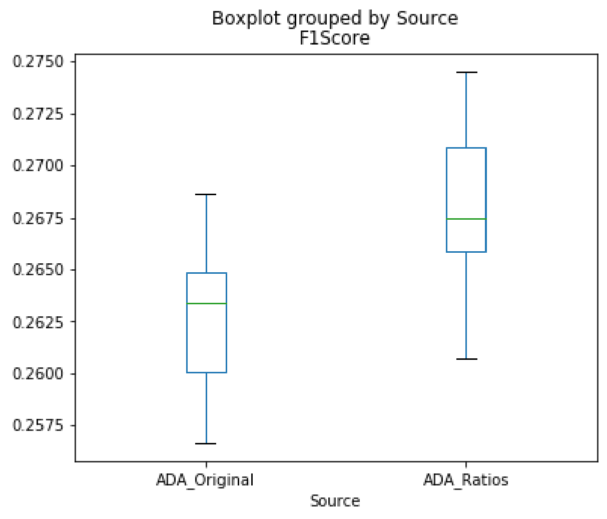
Logistic Regression and Random Forest saw a small drop in Recall but all other metrics improved including the F1-Score (despite the drop in Recall). AdaBoost saw improvements in all metrics and has the highest F1-Score among the three models.

Testing for a Significant Improvement in Performance

A one-tailed paired t-test was performed using the AdaBoost results from ten-fold cross-validation with and without the new ratio attributes to test for a statistically significant improvement in results. A 5% significance level was used.

Null Hypothesis: mean of result with ratio attributes <= mean of result without ratio attributes
Alternative Hypothesis: mean of result with ratio attributes > mean of result without ratio attributes

Figure 15: AdaBoost results from cross-validation with and without ratio attributes

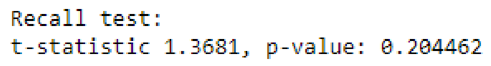


```
Recall test:
t-statistic 1.3681, p-value: 0.102231

Precision test:
t-statistic 4.2168, p-value: 0.001125

F1Score test:
t-statistic 3.7645, p-value: 0.002227

ROC_AUC test:
t-statistic 7.1739, p-value: 0.000026

Accuracy test:
t-statistic 6.5840, p-value: 0.000051
```

With the exception of Recall, all p-values were less than 5% so the null hypothesis was rejected for all metrics except Recall. This shows that all metrics (except Recall) had a statistically significant improvement in their mean values after adding the new ratio attributes.

A two-tailed paired t-test was performed, using a 5% significance level, to see if there had been any significant change in Recall.

Null Hypothesis: mean of Recall with ratio attributes = mean of Recall without ratio attributes
Alternative Hypothesis: mean of Recall with ratio attributes <> mean of Recall without ratio attributes



```
Recall test:
t-statistic 1.3681, p-value: 0.204462
```

The p-value was larger than 5% so the null hypothesis was not rejected. This indicates that there had been no significant change in the mean value of Recall after adding the ratio attributes. This is a good result given the improvements in the other metrics. The ratio attributes have helped reduce the false positives (as shown by the improved Precision) without any significant drop in Recall. That is, the new ratio attributes helped reduce the number of non-defaulting applicants incorrectly classified as defaulting.

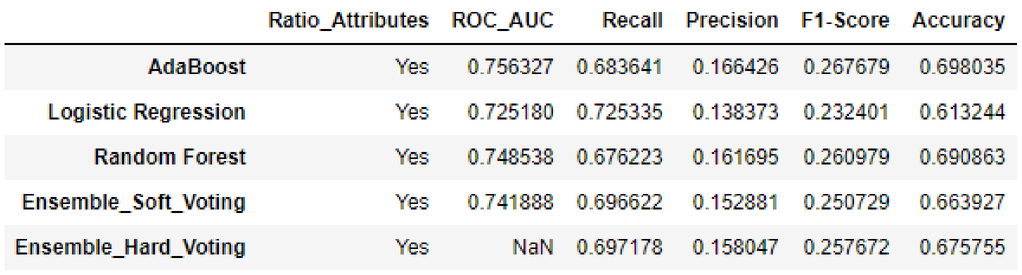
4) An Ensemble Voting Classifier Model

An ensemble voting classifier was built from the three models Logistic Regression, Random Forest and AdaBoost to see if it could outperform the individual models. Two voting schemes were tested:

1. Soft Voting: each observation was classified based on the highest probability returned from the three models
2. Hard Voting: each observation was classified based on a simple majority vote from the three models

The ensemble voting classifier was trained and tested using the two voting methods, expanded training set containing the ratio attributes and ten-fold cross-validation.

Table 17: Summary of mean results from cross-validation



	Ratio_Attributes	ROC_AUC	Recall	Precision	F1-Score	Accuracy
AdaBoost	Yes	0.756327	0.683641	0.166426	0.267679	0.698035
Logistic Regression	Yes	0.725180	0.725335	0.138373	0.232401	0.613244
Random Forest	Yes	0.748538	0.676223	0.161695	0.260979	0.690863
Ensemble_Soft_Voting	Yes	0.741888	0.696622	0.152881	0.250729	0.663927
Ensemble_Hard_Voting	Yes	NaN	0.697178	0.158047	0.257672	0.675755

The ensemble voting classifier did not outperform the existing models. It produced results somewhere between AdaBoost/Random Forest and Logistic Regression.

Step 9: Choosing a Final Model

In order to determine a best performing model, the F1-Scores from cross-validation were used to test for a significant difference between the models.

All results and code can be seen at:

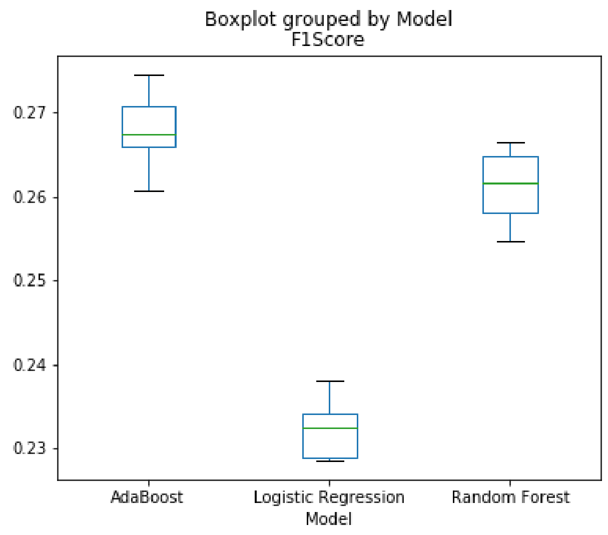
https://github.com/netvigate888/credit_default_prediction/tree/master/09_Choosing_Final_Model

1) F1-Score ANOVA Test

A One-Way Analysis of Variance (ANOVA), at a 5% significance level, was performed on the F1-Scores from the three models.

Null Hypothesis:	the mean F1-Scores for all models are equal
Alternative Hypothesis:	at least one of the mean F1-Scores is different

Figure 16: ANOVA on F1-Scores



```
ANOVA table for F1-Score:

            sum_sq    df           F        PR(>F)
Model     0.007020   2.0  195.111056  8.880799e-17
Residual  0.000486  27.0         NaN           NaN
```

The p-value from the ANOVA table is very small and well below 5%. The null hypothesis was rejected showing that at least one of the mean F1-Scores is significantly different.

2) F1-Score t-tests

A series of one-tailed paired t-tests, at a significance level of 5%, was performed to establish if a model's mean F1-Score is significantly higher than another model's mean F1-Score.

Table 18: A series of t-tests

```
Testing F1-Score: Is Random Forest significantly higher than Logistic Regression?
H_0: Random Forest <= Logistic Regression
H_a: Random Forest > Logistic Regression
t-statistic 27.0509, p-value: 3.1223898635376625e-10
```

```
Testing F1-Score: Is AdaBoost significantly higher than Logistic Regression?
H_0: AdaBoost <= Logistic Regression
H_a: AdaBoost > Logistic Regression
t-statistic 40.5877, p-value: 8.329278796488487e-12


Testing F1-Score: Is AdaBoost significantly higher than Random Forest?
H_0: AdaBoost <= Random Forest
H_a: AdaBoost > Random Forest
t-statistic 5.0979, p-value: 0.00032348277550732913
```

In all tests above, the p-value is below the 5% significance level (or below 5%/3 if a Bonferroni correction is preferred). Therefore, the null hypothesis was rejected in each test and we conclude that the mean F1-Score for AdaBoost is significantly higher than both Random Forest's and Logistic Regression's mean F1-Score. The AdaBoost model had the best performance based on F1-Scores.

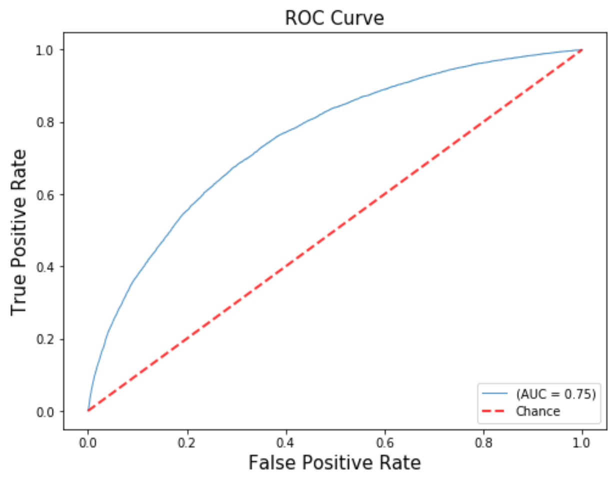
Step 10: Performance on Unseen Test Set

The AdaBoost model was trained on the full training set (including the ratio attributes) using under-sampling and then tested on the unseen test set.

All results and code can be seen here:

https://github.com/netvigate888/credit_default_prediction/tree/master/10_Performance_on_Test_Set

Figure 17: AdaBoost results from test set



```
Confusion matrix:

Predicted      0      1
Actual
0          59189  25617
1           2373   5075

Results

    ROC_AUC    Recall  Precision  F1-Score  Accuracy
1  0.754154  0.681391   0.165353  0.266125  0.696599
```

Table 19: Summary of results from test set and cross-validation

	Testing_Source	Ratio_Attributes	ROC_AUC	Recall	Precision	F1-Score	Accuracy
AdaBoost	Test_Set	Yes	0.754154	0.681391	0.165353	0.266125	0.696599
AdaBoost	Cross_Validation	Yes	0.756327	0.683641	0.166426	0.267679	0.698035

The results from the test set are not quite as good as the cross-validation results but the difference is small. The model maintained consistent performance with both the test set and cross-validation.

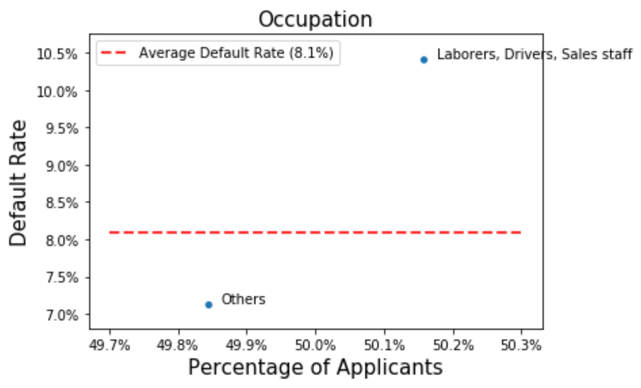
5. Discussion of Results

Initial Exploratory Data Analysis

Assuming the dataset is recent, Home Credit B.V is experiencing an average default rate of around 8.1%, which appears high by current international standards. For example, as of Oct. 2018, the default rate in the U.S. for consumer bank cards was 3.1%[2]. The higher default rate is to be expected given the business model and target client base of Home Credit B.V. However, based on the initial EDA, the opportunity exists for Home Credit B.V. to reduce its average default rate by changing the mix of its existing client base.

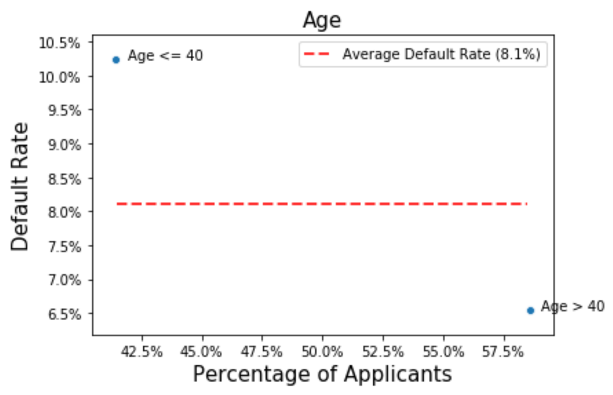
For example, the business should attempt to reduce the proportion of applicants with the occupations 'Laborers', 'Sales Staff' and 'Drivers', which currently make up over 50% of clients and have higher than average default rates around 10-11%. See Figure 18. Occupations to target could include 'IT staff' and 'HR staff', which have lower default rates and are a small proportion of applicants.

Figure 18: Default rate and proportion of Laborers, Drivers and Sales Staff



In addition, the business should further focus on clients above 40 years of age, who have a much lower than average default rate around 6.5%. See Figure 19.

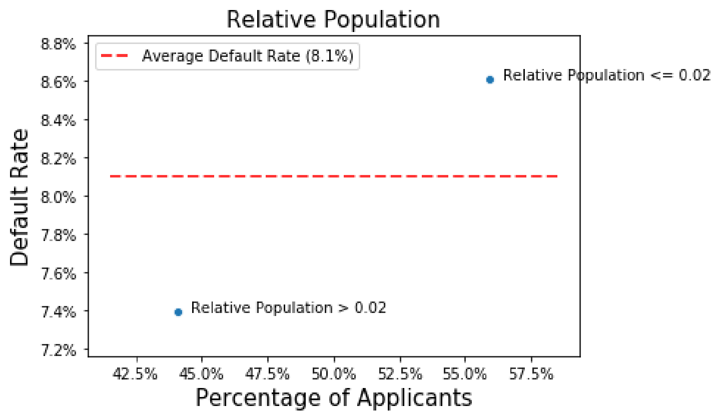
Figure 19: Default rate and proportion of Age > 40



[2] Source: https://us.spindices.com/indices/indicators/sp-experian-bankcard-default-index

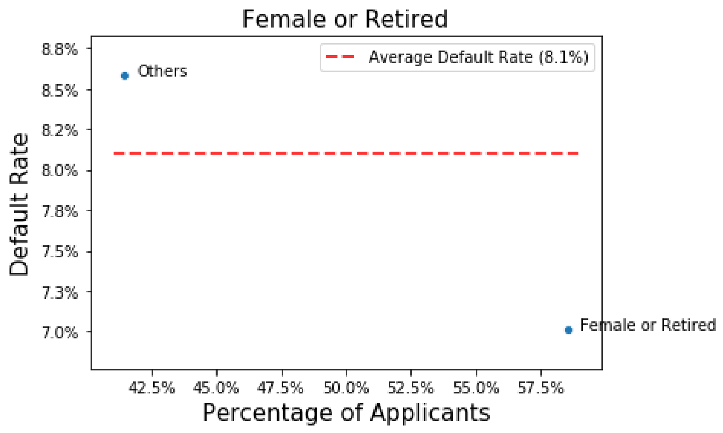
It also appears Home Credit B.V. is doing most of its business in lower population areas - perhaps small towns and rural areas. Going forward, the business should consider expanding into more urban regions with higher relative populations and lower default rates. See Figure 20.

Figure 20: Default rate and proportion of relative population



Lastly, areas where the business is doing well include focusing on women and retired applicants. Women make up two thirds of all applicants and have a lower default rate around 7%. Retired applicants represent about 18% of all clients and have a low default rate around 5%. See Figure 21. The business should continue to focus on such applicants.

Figure 21: Default rate and proportion of female or retired applicants



Predictive Modelling

Overview

Three models suitable for binary classification were explored: Logistic Regression, Random Forest and AdaBoost.

The performance metrics investigated were: Recall, Precision, F1-Score and Accuracy. The area under the ROC curve was also calculated for reference. For initial training and testing, all metrics were calculated as the mean result from ten-fold cross-validation.

For Home Credit B.V., incorrectly classifying an applicant as non-defaulting (a false negative) is a more expensive mistake than incorrectly classifying an applicant as defaulting (a false positive). Therefore, the choice was made to maximize Recall (i.e. minimize false negatives) while maintaining the F1-Score as high as possible.

Baseline performance metrics were established with the Logistic Regression model using default hyperparameters and base data (i.e. using only the basic application data before new attributes were derived from the bureau data).

A number of experiments were performed to test and improve the models. The experiments included different sampling techniques to cope with the imbalanced target variable, hyperparameter tuning, sub-setting attributes based on principal component analysis and the most important features from Random Forest, attribute engineering and building an ensemble voting classifier from the three models.

Finally, a best performing model was chosen and tested on the unseen test set.

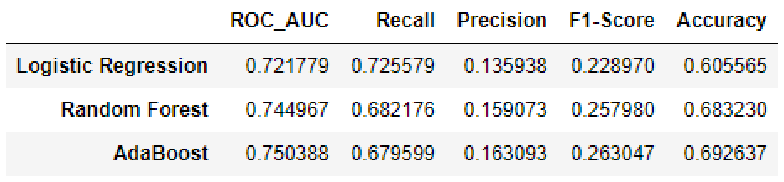
Results

Baseline performance metrics from Logistic Regression were weak with Recall at just 1.3% and the F1-Score at 2.4%. The imbalanced target variable was causing problems for the model with over 99% of observations simply classified as 0 (non-default). Adding new attributes derived from the bureau data helped marginally (moving Recall to 1.5%) but the overall results remained poor.

Both Random Forest and AdaBoost also suffered from the imbalanced target variable with initial Recall and F1-Scores of 1.1% / 2.0% and 2.2% / 4.2% respectively.

Over-sampling the defaulting observations during training helped greatly but the best initial results were achieved with under-sampling the non-defaulting observations. Recall for all three models was further improved by tuning hyperparameters. The best initial results are summarized below.

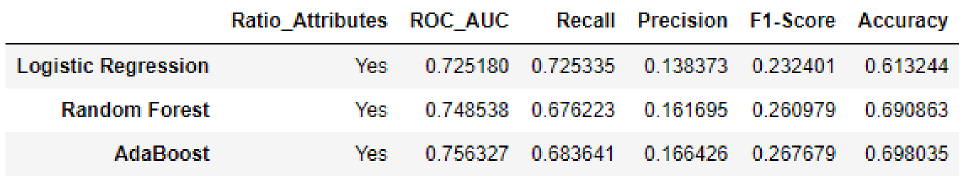
Table 20: Best initial results from under-sampling and hyperparameter tuning



	ROC_AUC	Recall	Precision	F1-Score	Accuracy
Logistic Regression	0.721779	0.725579	0.135938	0.228970	0.605565
Random Forest	0.744967	0.682176	0.159073	0.257980	0.683230
AdaBoost	0.750388	0.679599	0.163093	0.263047	0.692637

Following further experiments, the only other improvement in results came from engineering 22 new attributes based on ratios of existing attributes. The ratios were chosen to extract new relevant information from the data.

Table 21: Best results achieved after adding new ratio attributes



	Ratio_Attributes	ROC_AUC	Recall	Precision	F1-Score	Accuracy
Logistic Regression	Yes	0.725180	0.725335	0.138373	0.232401	0.613244
Random Forest	Yes	0.748538	0.676223	0.161695	0.260979	0.690863
AdaBoost	Yes	0.756327	0.683641	0.166426	0.267679	0.698035

For AdaBoost, the improvement in results (after adding the ratio attributes) was tested and found to be statistically significant for all metrics with the exception of Recall which did not significantly change (up or down). This improvement demonstrates the value that careful attribute engineering can add to a prediction task.

The F1-Score was used to determine a best performing model as it reflects performance in both Recall and Precision. Following hypothesis testing, the F1-Scores from AdaBoost were found to be significantly higher than the F1-Scores from Random Forest and Logistic Regression.

The AdaBoost model continued to produce consistent results (although very slightly worse) when tested on the unseen test set. This indicates that the model did not suffer from overfitting to the training data and should maintain consistent performance on further unseen data.

In summary, the results were substantially improved from the baseline but still remained somewhat average. For the three models, Recall settled in a useful range of 68-73% but Precision and the F1-Score remained low around 14-17% and 23-27% respectively.

6. Conclusion

This report has shown that the opportunity exists for Home Credit B.V. to bring down its average default rate in the ‘under-banked’ segment by changing the mix of its existing client base.

Further, it was shown that a model can be built to provide a worthwhile level of default prediction. The best performing algorithm, AdaBoost, can correctly identify over two thirds of defaulting applicants albeit with a high level of incorrectly classified non-defaulting applicants. In its current state, the model would be a useful addition to existing decision-making tools but further work is needed to reduce the number of incorrectly classified non-defaulting applicants.

7. Further Work

There are a number of areas that could be investigated in an attempt to improve results and reduce the false positives:

- ***Incorporating the remaining data files that were excluded from this analysis***. The additional files contain further information on the past repayment performance of applicants and is likely to improve predictions.
- ***Further attribute engineering especially following the inclusion of the unused data files.*** It was demonstrated in this report that good attribute engineering can help improve results.
- ***More granular missing value imputation.*** Missing values were imputed using median values calculated across each entire attribute. A more granular approach where observations are first clustered with other 'like' observations and missing values imputed within each cluster may help performance.
- ***Analyze the false positives.*** Examine the characteristics of the incorrectly classified non-defaulting applicants with the aim of understanding why they were misclassified.
- ***Optimize hyperparameters using custom cost function.*** Hyperparameters were optimized based on Recall only. Using a custom cost function that takes both false negatives and false positives into account may help find hyperparameters than can reduce the false positives.

8. References

[1] Turkson, R.E., Baagyere, E.Y. and Wenya, G.E., 2016, September. A machine learning approach for predicting bank credit worthiness. In *Artificial Intelligence and Pattern Recognition (AIPR), International Conference on* (pp. 1-7). IEEE.

[2] Liu, M.A., 2018. A Comparison of Machine Learning Algorithms for Prediction of Past Due Service in Commercial Credit. Grey Literature for Ph.D. Candidates, Kennesaw State University.

[3] Yu, X., 2017. Machine learning application in online lending risk prediction. *arXiv preprint arXiv:1707.04831*.

[4] Islam, S.R., Eberle, W. and Ghafoor, S.K., 2018. Credit Default Mining Using Combined Machine Learning and Heuristic Approach. *arXiv preprint arXiv:1807.01176*.

[5] Xiong, T., Wang, S., Mayers, A. and Monga, E., 2013. Personal bankruptcy prediction by mining credit card data. *Expert systems with applications*, *40*(2), pp.665-676..