**Machine Learning Engineer Nanodegree**

**Capstone Project- Identify Home Credit Default Risk**

Ravi Mittal  
May 29th, 2018

**I. Definition**

**Project Overview**

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.

[Home Credit](http://www.homecredit.net/) 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.

**Kaggle Competition path**

<https://www.kaggle.com/c/home-credit-default-risk>

**Problem Statement**

We have been provided with 7 files containing Home Credit data of various customers of an organization. This includes the application\_train.csv file as well, which would be used to train the model.

An application\_test.csv file is also been provided which would be used as input test data to make predictions from the model. We need to make predictions on each of the customer data record in application\_test.csv to find out that a particular customer is a potential defaulter of Home credit provided to him or not. Below would be format of output.

SK\_ID\_CURR, TARGET

100001,1

100002,0

100003,1

...

**Metrics**

Since in application\_train.csv file, we have received labelled data with Target label values 0 and 1, it would be intuitive to evaluate model performance based on number of correctly identified labels in test validation data. So, evaluation is done based on Accuracy score and F-Beta score.

For benchmark model, we have used a model that always predicts '1' (i.e. the individual would always be a defaulter) and take it as benchmark model. Benchmark model will have no True Negatives (TN) or False Negatives (FN) as we are not making any negative ('0' value) predictions. Therefore our Accuracy in this case becomes the same as our Precision(True Positives/(True Positives + False Positives)) as every prediction that we have made with value '1' that should have '0' becomes a False Positive; therefore our denominator in this case is the total number of records we have in total.

Our Recall score (True Positives / (True Positives + False Negatives)) in this setting becomes 1 as we have no False Negatives.

**II. Analysis**

**Data Exploration**

We have been provided with the following files containing Home Credit data of various customers of an organization.

* 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 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.

**Exploratory Visualization**

Some of excel file provided as part of dataset had more than 120 columns and some had more than 1 million records. So I had to do data cleanup activity.

Because of the high volume of data in these files and processing limitations of my laptop (Windows 10 + 16 GB RAM + Intel i-7 processor), I had to reduce amount of data in each file. Data reduction in each file was also not straight forward as there were many records with primary key data in application\_{train|test}.csv files that were not having corresponding foreign key relationships in other tables. So to reduce the data I had to find all those records in application\_{train|test}.csv files which were having at least one foreign key relationship in all other files.

**Below is statistics of primary key data in data\_bureau which is available as foreign key in data\_bureau\_balance**

False 942074

True 774354

Name: SK\_ID\_BUREAU, dtype: int64

**Below is statistics of primary key data in data\_train which is available as foreign key in data\_bureau**

True 263491

False 44020

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_train which is available as foreign key in data\_credit\_card\_balance**

False 220606

True 86905

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_train which is available as foreign key in data\_installments\_payments**

True 291643

False 15868

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_train which is available as foreign key in data\_POS\_CASH\_balance**

True 289444

False 18067

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_train which is available as foreign key in data\_previous\_application**

True 291057

False 16454

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_test which is available as foreign key in data\_bureau**

True 42320

False 6424

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_test which is available as foreign key in data\_credit\_card\_balance**

False 32091

True 16653

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_test which is available as foreign key in data\_installments\_payments**

True 47944

False 800

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_test which is available as foreign key in data\_POS\_CASH\_balance**

True 47808

False 936

Name: SK\_ID\_CURR, dtype: int64

**Below is statistics of primary key data in data\_test which is available as foreign key in data\_previous\_application**

True 47800

False 944

Name: SK\_ID\_CURR, dtype: int64

Once I had those records having relationship with other files in both of application\_{train|test}.csv files, I took top 100 records from both application\_{train|test}.csv files. After that, I had to revisit all other files and delete all data in those files except the matching primary key data in both application\_{train|test}.csv files.

**Below is Shape of data\_bureau before filtering**

(1716428, 17)

**Below is Shape of data\_bureau after filtering to retain only that data where SK\_ID\_BUREAU is present in data\_bureau\_balance**

(774354, 17)

**Below is Shape of data\_train before filtering**

(307511, 122)

**Below is final Shape of data\_train after filtering to retain only that data where pk SK\_ID\_CURR is available as foreign key in all other tables and this will be used as final training dataset**

(100, 122)

**Below is Shape of data\_test before filtering**

(14462, 121)

**Below is final Shape of data\_test after filtering to retain only that data where pk SK\_ID\_CURR is available as foreign key in all other tables and this will be used as final test dataset**

(14462, 121)

**Algorithms and Techniques**

From multiple files provided as input, we need to make predictions on each of the customer data in application\_test.csv to find out that a particular customer is a potential defaulter of Home credit provided to him or not.

Since it is a binary classification problem and to reduce dimensionality of data we would use PCA and Logistics Regression pipeline to create model after reducing dimensionality.

In Logistics Regression, model complexity is low, overfitting is not a big issue and with less complexity, it is easy to train model using memory constraints of laptop.

**Benchmark**

For benchmark model, we have used a model that always predicts '1' (i.e. the individual would always be a defaulter) and take it as benchmark model. Benchmark model will have no True Negatives (TN) or False Negatives (FN) as we are not making any negative ('0' value) predictions. Therefore our Accuracy in this case becomes the same as our Precision(True Positives/(True Positives + False Positives)) as every prediction that we have made with value '1' that should have '0' becomes a False Positive; therefore our denominator in this case is the total number of records we have in total.

Our Recall score (True Positives / (True Positives + False Negatives)) in this setting becomes 1 as we have no False Negatives.

**III. Methodology**

**Data Preprocessing**

Some of excel files provided as part of dataset had more than 120 columns and some had more than 1 million records. So I had to do data cleanup activity.

Because of the high volume of data in these files and processing limitations of my laptop (Windows 10 + 16 GB RAM + Intel i-7 processor), I had to reduce amount of data in each file. Data reduction in each file was also not straight forward as there were many records with primary key data in application\_{train|test}.csv files that were not having corresponding foreign key relationships in other tables. So to reduce the data I had to find all those records in application\_{train|test}.csv files which were having at least one foreign key relationship in all other files.

**Preprocessing step 1:**

Data cleaning and exploration

For prediction and to reduce the amount of data, we would like to consider only that data in application\_{train|test}.csv, whose reference data (data with matching primary key SK\_ID\_CURR) is available in all other files (bureau.csv, bureau\_balance.csv e.t.c) and take top 100 rows from it.

In each of the supporting (bureau.csv, bureau\_balance.csv e.t.c) data files, we would like to take only that data which is having foreign key reference available both in reduced to top 100 application\_{train|test}.csv.

**Preprocessing step 2:**

Do One-hot encoding of concatenated training data without target label and testing data using pandas.get\_dummies(). Also, we have to fill empty train and test data with 0 using fillna function.

**Implementation**

From multiple files provided as input, we need to make predictions on each of the customer data in application\_test.csv to find out that a particular customer is a potential defaulter of Home credit provided to him or not

Because of the high volume of data in these files and processing limitations of my laptop (Windows 10 + 16 GB RAM + Intel i-7 processor), I would use only below files to merge data for final predictions and would need to perform data cleaning to reduce the size of input data.

We would use PCA and Logistics Regression pipeline to create model after reducing dimensionality.

* + application\_{train|test}.csv
  + bureau.csv
  + bureau\_balance.csv
  + credit\_card\_balance.csv

To achieve this, I would follow below steps

**Step 1:**

Data cleaning and exploration

For prediction and to reduce the amount of data, we would like to consider only that data in application\_{train|test}.csv, whose reference data (data with matching primary key SK\_ID\_CURR) is available in all other files (bureau.csv, bureau\_balance.csv e.t.c) and take top 100 rows from it.

In each of the supporting (bureau.csv, bureau\_balance.csv e.t.c) data files, we would like to take only that data which is having foreign key reference available both in reduced to top 100 application\_{train|test}.csv.

**Step 2:**

Do One-hot encoding of concatenated training data without target label and testing data using pandas.get\_dummies()

**Step 3:**

Split the features and labels data into training and testing sets

**Step 4:**

Calculate Naive Model Accuracy and F-Score, which would be used as bench mark.

**Step 5:**

Use PCA to logistics regression pineline to reduce dimensionality of data train the model and Use GridSearchCSV to optimize the parameters \ hyper-parameters of model.

**Step 6:**

Use PCA to logistics regression pineline to reduce dimensionality of data train the model and Use GridSearchCSV to optimize the parameters \ hyper-parameters of model. Predict and save test data.

**Refinement**

I have used sklearn.model\_selection.GridSearch.csv method to tune the number of components of PCA Logistictics Regression Pipeline.

**Initial Results**

**Final Results**

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

* *Has an initial solution been found and clearly reported?*
* *Is the process of improvement clearly documented, such as what techniques were used?*
* *Are intermediate and final solutions clearly reported as the process is improved?*

**IV. Results**

*(approx. 2-3 pages)*

**Model Evaluation and Validation**

From multiple files provided as input, we need to make predictions on each of the customer data in application\_test.csv to find out that a particular customer is a potential defaulter of Home credit provided to him or not.

Since it is a binary classification problem and to reduce dimensionality of data we would use PCA and Logistics Regression pipeline to create model after reducing dimensionality.

In Logistics Regression, model complexity is low, overfitting is not a big issue and with less complexity, it is easy to train model using memory constraints of laptop.

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

* *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
* *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
* *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
* *Can results found from the model be trusted?*

**Justification**

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

* *Are the final results found stronger than the benchmark result reported earlier?*
* *Have you thoroughly analyzed and discussed the final solution?*
* *Is the final solution significant enough to have solved the problem?*

**V. Conclusion**

*(approx. 1-2 pages)*

**Free-Form Visualization**

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

* *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
* *Is the visualization thoroughly analyzed and discussed?*
* *If a plot is provided, are the axes, title, and datum clearly defined?*

**Reflection**

From multiple files provided as input, I needed to make predictions on each of the customer data in application\_test.csv to find out that a particular customer is a potential defaulter of Home credit provided to him or not.

One of the interesting and somewhat difficult to handle aspect of the problem is volume of data in the form of excel files provided. Some of excel file had more than 120 columns and some had more than 1 million records. So I had to do data cleanup activity as mentioned in step 1 below.

Because of the high volume of data in these files and processing limitations of my laptop (Windows 10 + 16 GB RAM + Intel i-7 processor), I had to reduce amount of data in each file. Data reduction in each file was also not straight forward as there were many records with primary key data in application\_{train|test}.csv files that were not having corresponding foreign key relationships in other tables. So to reduce the data I had to find all those records in application\_{train|test}.csv files which were having at least one foreign key relationship in all other files. Once I had those records having relationship with other files in both of application\_{train|test}.csv files, I took top 100 records from both application\_{train|test}.csv files. After that, I had to revisit all other files and delete all data in those files except the matching primary key data in both application\_{train|test}.csv files.

Even after this comprehensive data reduction, I was still getting out of memory errors while merging the data of all files to create final data.

So I had to use only below files to merge data for final predictions. But I have provided commented code to merge all other files.

* + bureau.csv
  + bureau\_balance.csv
  + credit\_card\_balance.csv

Below are overall workflow steps of project.

**Step 1:**

Data cleaning and exploration

For prediction and to reduce the amount of data, we would like to consider only that data in application\_{train|test}.csv, whose reference data (data with matching primary key SK\_ID\_CURR) is available in all other files (bureau.csv, bureau\_balance.csv e.t.c) and take top 100 rows from it.

In each of the supporting (bureau.csv, bureau\_balance.csv e.t.c) data files, we would like to take only that data which is having foreign key reference available both in reduced to top 100 application\_{train|test}.csv.

***Pseudo Code***

Statistics of primary key data in data\_train which is available as foreign key in data\_bureau

display(data\_train['SK\_ID\_CURR'].isin(data\_bureau['SK\_ID\_CURR']).value\_counts())

Filter data bureau with only data where SK\_ID\_BUREAU is present in data\_bureau\_balance

data\_bureau=data\_bureau.loc[(data\_bureau['SK\_ID\_BUREAU'].isin(data\_bureau\_balance['SK\_ID\_BUREAU']))]

To reduce memory requirements populate data in application\_train.csv only with those primary key SK\_ID\_CURR which are available as foreign key in all other tables

data\_train=data\_train.loc[(data\_train['SK\_ID\_CURR'].isin(data\_bureau['SK\_ID\_CURR']))

& (data\_train['SK\_ID\_CURR'].isin(data\_credit\_card\_balance['SK\_ID\_CURR']))

& (data\_train['SK\_ID\_CURR'].isin(data\_installments\_payments['SK\_ID\_CURR']))

& (data\_train['SK\_ID\_CURR'].isin(data\_POS\_CASH\_balance['SK\_ID\_CURR']))

& (data\_train['SK\_ID\_CURR'].isin(data\_previous\_application['SK\_ID\_CURR']

Take only top 100 records for final training data

data\_train =data\_train.head(100)

To reduce memory requirements populate data in application\_test.csv only with those primary key SK\_ID\_CURR which are available as foreign key in all other tables

data\_test=data\_test.loc[(data\_test['SK\_ID\_CURR'].isin(data\_bureau['SK\_ID\_CURR']))

& (data\_test['SK\_ID\_CURR'].isin(data\_credit\_card\_balance['SK\_ID\_CURR']))

& (data\_test['SK\_ID\_CURR'].isin(data\_installments\_payments['SK\_ID\_CURR']))

& (data\_test['SK\_ID\_CURR'].isin(data\_POS\_CASH\_balance['SK\_ID\_CURR']))

& (data\_test['SK\_ID\_CURR'].isin(data\_previous\_application['SK\_ID\_CURR']))]

Take only top 100 records for final test data

data\_test =data\_test.head(100)

Filter data-frame (e.g. data\_credit\_card\_balance) containing each supporting file (e.g. credit\_card\_balance.csv) file to contain only that data which which matches primary key 'SK\_ID\_CURR' of both reduced size training and test data. Please refer to OR (|) in below code.

data\_credit\_card\_balance=data\_credit\_card\_balance.loc[(data\_credit\_card\_balance['SK\_ID\_CURR'].isin(data\_train['SK\_ID\_CURR'])) | (data\_credit\_card\_balance['SK\_ID\_CURR'].isin(data\_test['SK\_ID\_CURR']))]

Display the final shape of each filtered datasets

display(data\_credit\_card\_balance.shape)

Merge final training data (top 100 records) with each of the reduced data sets (e.g. data\_bureau). To avoid memory issues I am merging only below files (reduced data) with training data in this step. But I have provided commented code to merge all other files.

bureau.csv

bureau\_balance.csv

credit\_card\_balance.csv

data\_train=data\_train.merge(data\_bureau, on='SK\_ID\_CURR', how='inner')

Merge final test data (top 100 records) with each of the reduced data sets (e.g. data\_bureau). To avoid memory issues I am merging only below files (reduced data) with training data in this step. But I will provide commented code to merge all other files.

bureau.csv

bureau\_balance.csv

credit\_card\_balance.csv

data\_test=data\_test.merge(data\_bureau, on='SK\_ID\_CURR', how='inner')

**Step 2:**

Do One-hot encoding of concatenated training data without target label and testing data using pandas.get\_dummies()

***Pseudo Code***

features\_final = pd.get\_dummies(pd.concat([features\_without\_labels,data\_test],keys=[0,1]))

**Step 3:**

Split the features and labels data into training and testing sets

***Pseudo Code***

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features\_final\_train, application\_labels\_train,

test\_size = 0.2, random\_state = 0)

**Step 4:**

Calculate Naive Model Accuracy and F-Score, which would be used as bench mark.

***Pseudo Code***

accuracy = np.sum(application\_labels\_train) / len(application\_labels\_train)

fscore = ((1 + np.square(.5))\*precision\*recall) / ((np.square(.5)\*precision)+recall)

**Step 5:**

Use PCA to logistics regression pineline to reduce dimensionality of data train the model and Use GridSearchCSV to optimize the parameters \ hyper-parameters of model.

***Pseudo Code***

pca = skdc.PCA()

logreg = LogisticRegression()

pca\_logistic\_regressor\_pipe = skpl.Pipeline([('pca', pca), ('logistic', logreg)])

grid\_obj = GridSearchCV(pca\_logistic\_regressor\_pipe,

dict(pca\_\_n\_components=n\_components,logistic\_\_C=Cs))

**Step 6:**

***Pseudo Code***

Use PCA to logistics regression pineline to reduce dimensionality of data train the model and Use GridSearchCSV to optimize the parameters \ hyper-parameters of model. Predict and save test data.

results['acc\_test'] = accuracy\_score(y\_test,predictions\_test)

results['f\_test'] = fbeta\_score(y\_test,predictions\_test,beta=0.5)

**Improvement**

Since there are data processing limitations in my laptop machine, I could use the whole dataset to train the model and had to do some data cleanup activity to decrease the size of data, one obvious improvement would be to use machine which higher processing capability so that model can be better trained and this would mitigate any risk of under-fitting.

Also, I could have considered the use of popular [ensemble methods](https://www.dataquest.io/blog/introduction-to-ensembles/) such as [xgboost](https://xgboost.readthedocs.io/en/latest/model.html) and [LightGBM](https://blogs.technet.microsoft.com/machinelearning/2017/07/25/lessons-learned-benchmarking-fast-machine-learning-algorithms/) and that may have given better predictions.

**Before submitting, ask yourself. . .**

* Does the project report you’ve written follow a well-organized structure similar to that of the project template?
* Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
* Would the intended audience of your project be able to understand your analysis, methods, and results?
* Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
* Are all the resources used for this project correctly cited and referenced?
* Is the code that implements your solution easily readable and properly commented?
* Does the code execute without error and produce results similar to those reported?