Md Saimoom Ferdous

11 September 2020

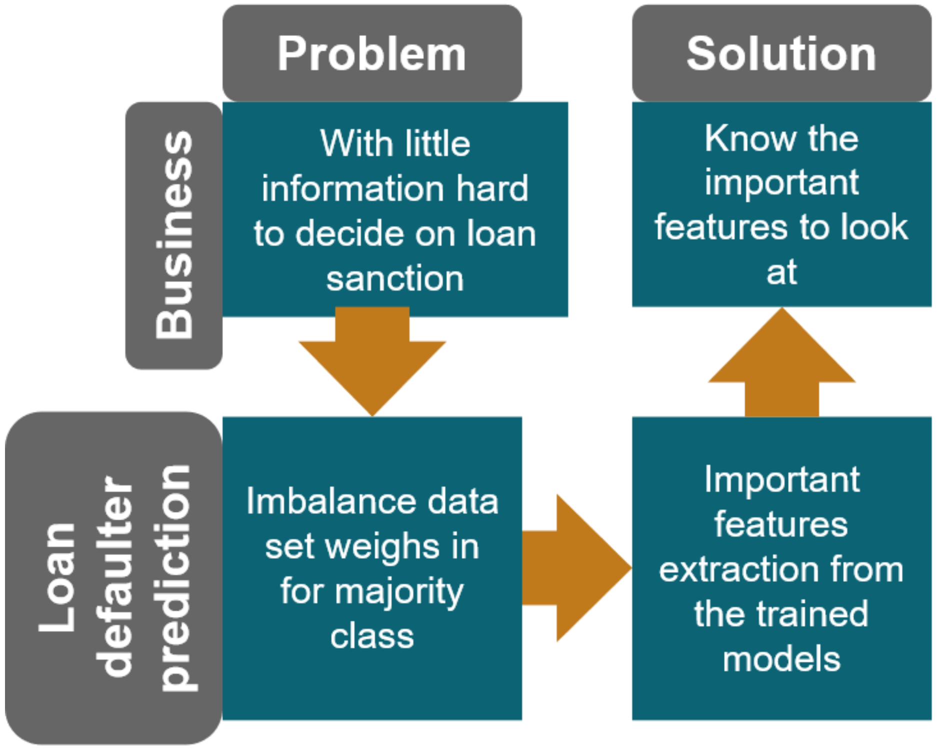
Home Credit Loan Defaulter Prediction: Final Report

GitHub Link

Table of Contents

Problem Statement

Project Flow


Problem
Solution
Business
With little information hard to decide on loan sanction
Know the important features to look at
Loan defaulter prediction
Imbalance data set weighs in for majority class
Important features extraction from the trained models

1. **Start with the business problem**
   Home Credit Group aims to serve millions of underserved population with little to no background information. Without knowing much, it becomes too risky to disburse loans who might not be able to repay. On the other hand too much control may lead to not sanctioning loans while they might be able to repay.
2. **Convert business problem into binary classification problem**
   Majority population is likely to repay the loan whereas minority of the population would be loan defaulter. Given the existing data with imbalanced data, this can be framed as binary classification problem with imbalanced data.
3. **Solve binary classification problem**
   Train machine learning models can be trained and get to know the important features for the classification problems.
4. **Convert binary classification solution into business solution**
   Knowing most important parameters for prediction can certainly help the Home Credit group to look at closely before approving any loans.

Business Understanding

Home Credit Group is one of the largest non-banking financial institution headquartered in Netherlands. It focuses on handing out credits to the population with little or no credit history. Majority of the population living in remote communities needs micro credit, but they do not have enough credit history that will build confidence in lending the credit.

Other than the credit history, what are the other available characteristics (social, demographical) can provide insights into the client groups who would be able to repay the credit. Does repaying credit correlates with age group, occupation, shopping habit or any other unseen traits which can be discovered by data analysis?

Once the features that lead to repaying credits are known, which are the dominant features in determining credit repayment?

Binary Classification

Machine learning algorithms can be trained to predict default vs non-default clients based on the very little information available. These come with two problems. Clients who really in need of the credit may often be misclassified as defaulter in advance. This phenomenon is called false positive. For Home Credit Group this would derail their purpose if anyone deserving does not get the credit. On the other hand, clients with real tendency for loan defaulter may be given green light for the credit as the algorithm would misclassify as non-defaulter. These phenomena are called false negative. This is also highly discouraging as registering loan defaulters in disguise would cripple organizations financial health in the medium to longer terms. It is essential to design machine learning algorithm for loan defaulter in such a way to minimize false positive/negative whereas maximizing true positive/negative rate.

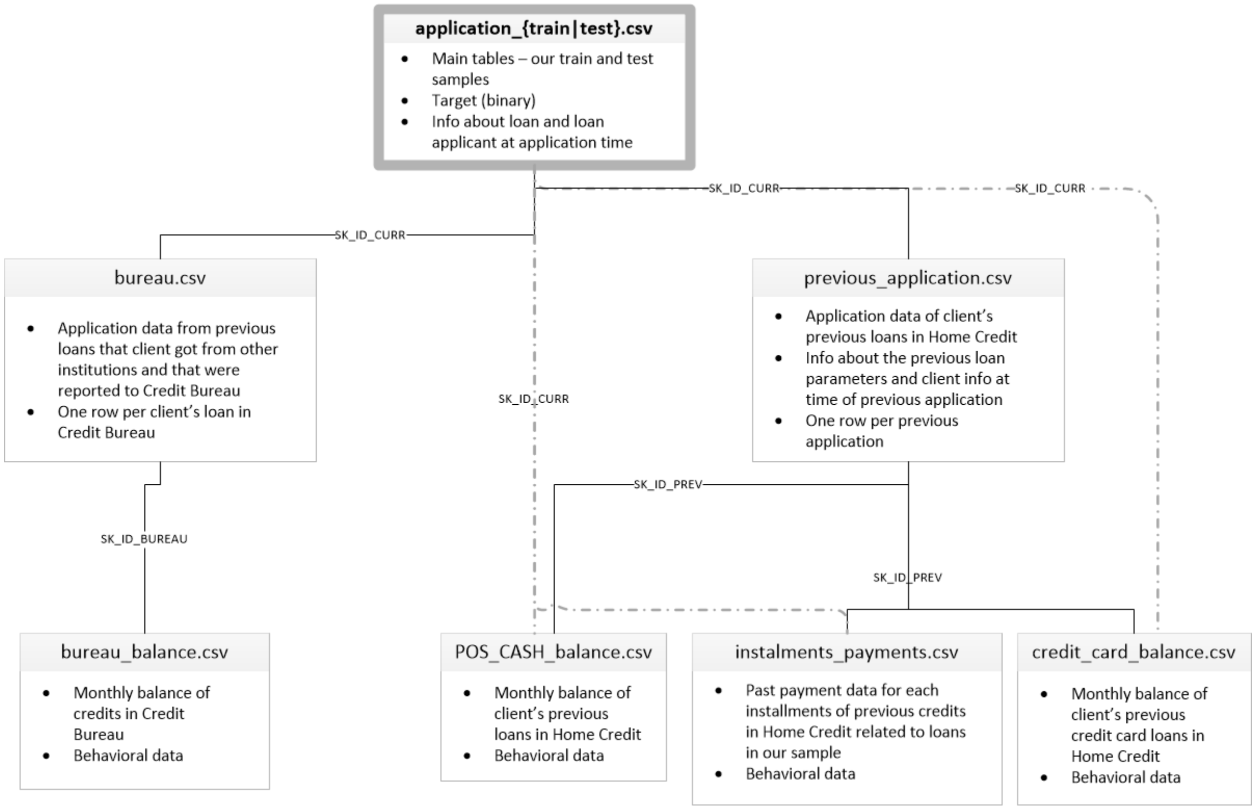
Data Collection

Data Source

Data has been sourced from a Kaggle competition, consisting of thousands of client home loan, credit records Kaggle website provided by Home Credit Group.

Getting to Know around the Data Set

The data package consisted of seven files: application, bureau, bureau_balance, previous_application, POSH_CASH_balance, instalment_payments and credit_card_balance. The files are related to each other by the following flow graph, taken from the Kaggle competition website:


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_PREV
SK_ID_BUREAU
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

A short description of each files:

- **Application:** Main file that contains information about each loans at Home Credit. Every row is unique and identified by `SK_ID_CURR` feature. `TARGET` variable has value of 0 means loans repaid, 1 for loans was not repaid. Although, the file contains training data, it will be split into train/test data.
- **Bureau:** Contains client's previous credits from other institutions. One creditors in 'application' can have multiple records in 'bureau'
- **bureau_balance:** Client's previous credits for every months.
- **credit_card_balance:** Monthly credit card balance of client's at Home Credit.
- **installments_payments:** Previous payments made at Home Credit. One row for payment and another is for missed payment.
- **POS_CASH_balance:** Monthly point of sale (POS) or cash loans each client had with Home Credit. One row is one months POS or cash loans data.
- **previous_applications:** Previous applications made at Home Credit. One row for one previous application.

The 'application' file itself contains 307.5 K number of client record. For this project we will stick to the 'application' file.

Data Wrangling

Data wrangling consisted of cleaning up the data by removing NaN in the dataset and looking for duplicates.

Removing NaN

There are 67 columns in the ‘application’ file which had NaN values ranging from 69 % to 0.3 %. Missing values were fixed by seeing the distribution and deciding whether mean, median, or randomly assigning numbers over 50% distribution. There were many columns with closely correlated values such as AVERAGE, MEDIAN and MODE. Only AVERAGE columns were kept avoiding redundancy. Some categorical columns were reasonably filled by forward fill method. OCCUPATION column had large numbers of classes. So, the missing values were filled by assigning majority class expecting less error.

Duplicates

No duplicated rows were found.

Cleaned dataset was saved for use in the EDA stage.

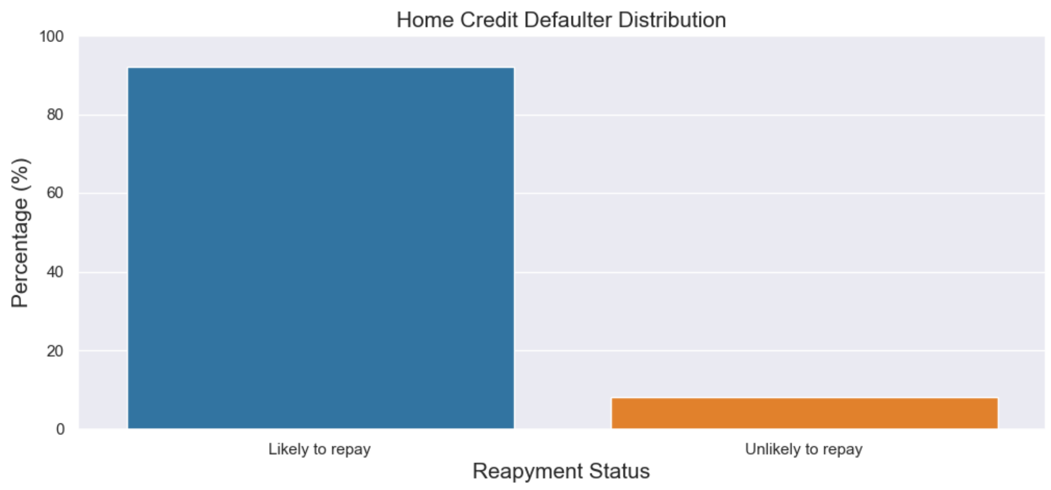
Exploratory Data Analysis

In this stage, demographic, socio-economic population distribution of the data will be explored. Correlation between variables will also be determined. Manual feature engineering will be done to add critical features for the prediction.

Insights into Categorical Variables

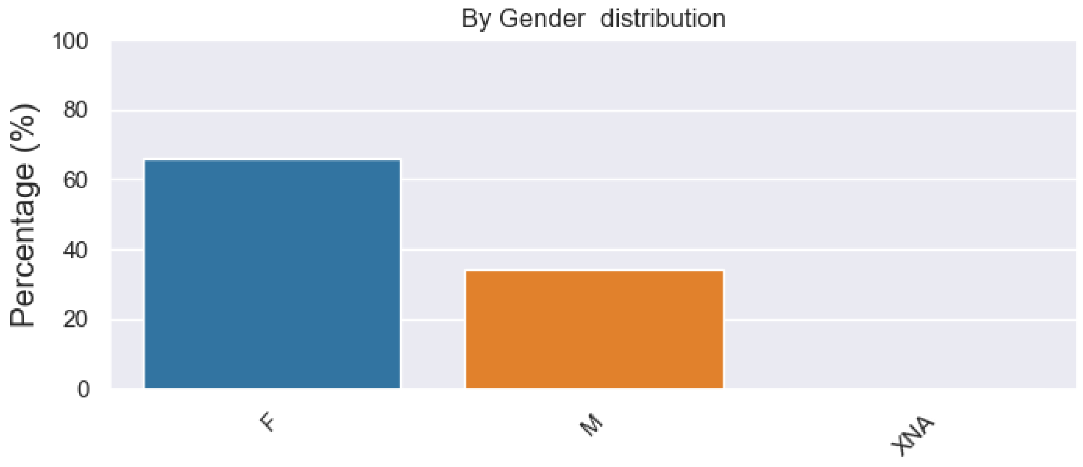
Data distribution was explored for every categorical variables. Here, only six selective variables will be included for succinct.

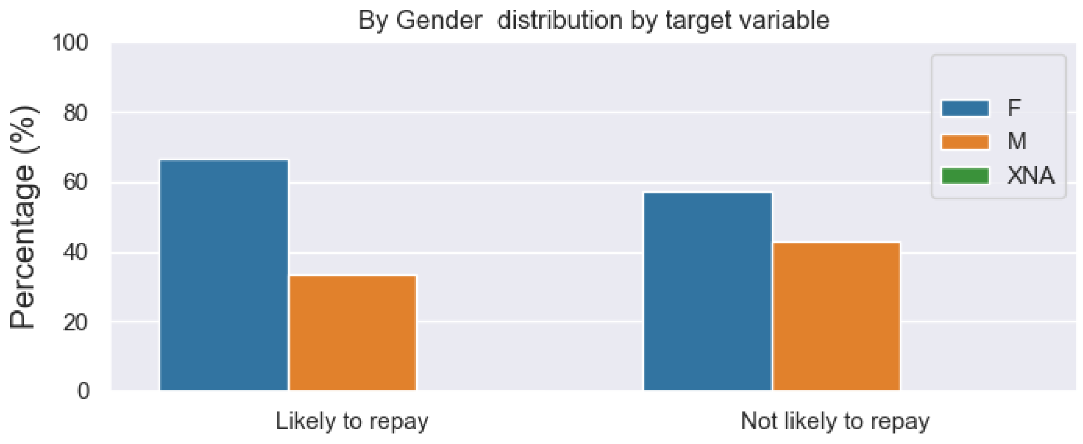
Target Variable Distribution



Among the population in hand, 8 % are unlikely to repay the loan whereas large percentage (~92%) are predicted to repay the loan. The data is highly imbalanced.

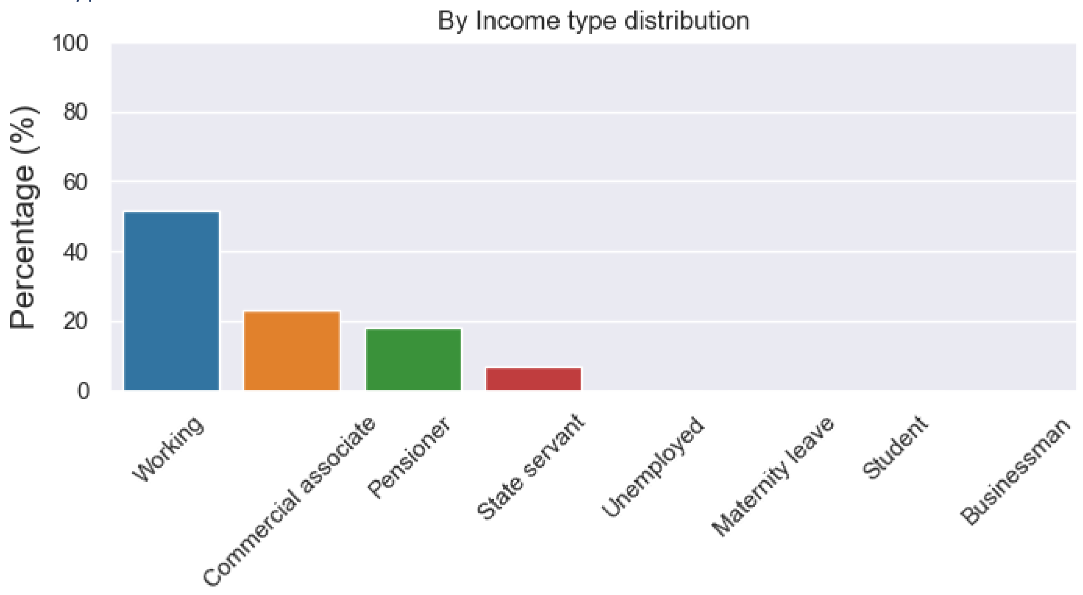
Which Gender Group is Likely to Repay Loan?

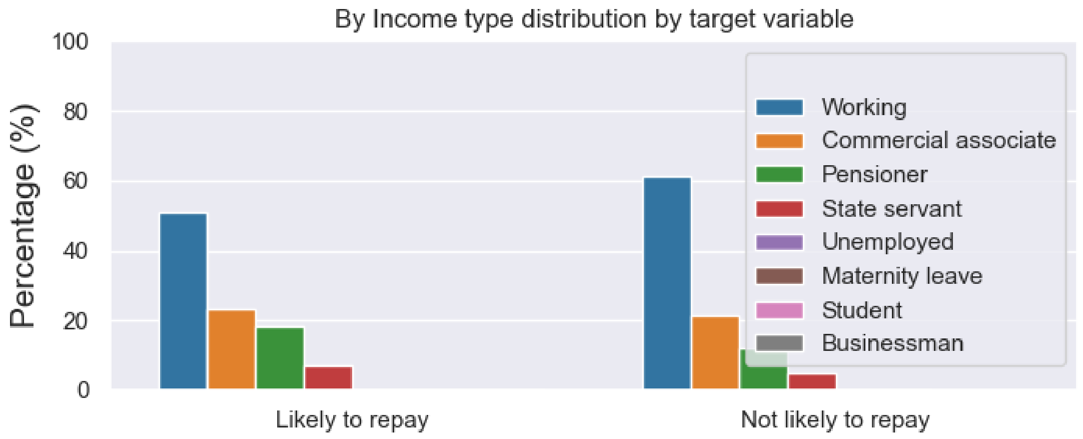




Most loans were distributed to the women. Male population increased in Defaulters list compared to non-defaulter list.

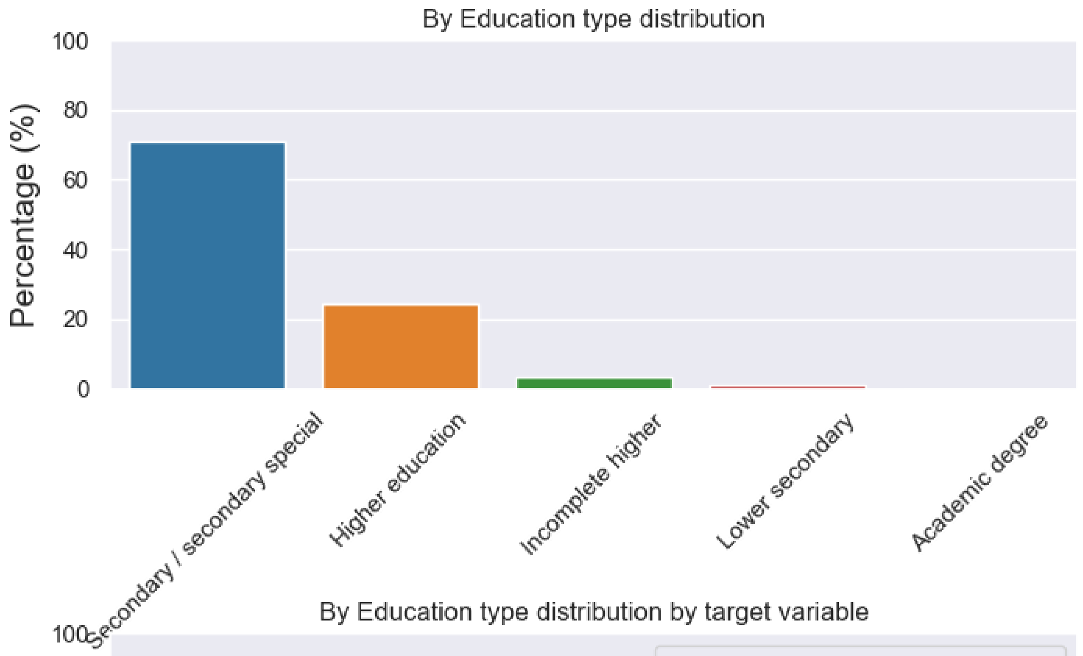
Income Types

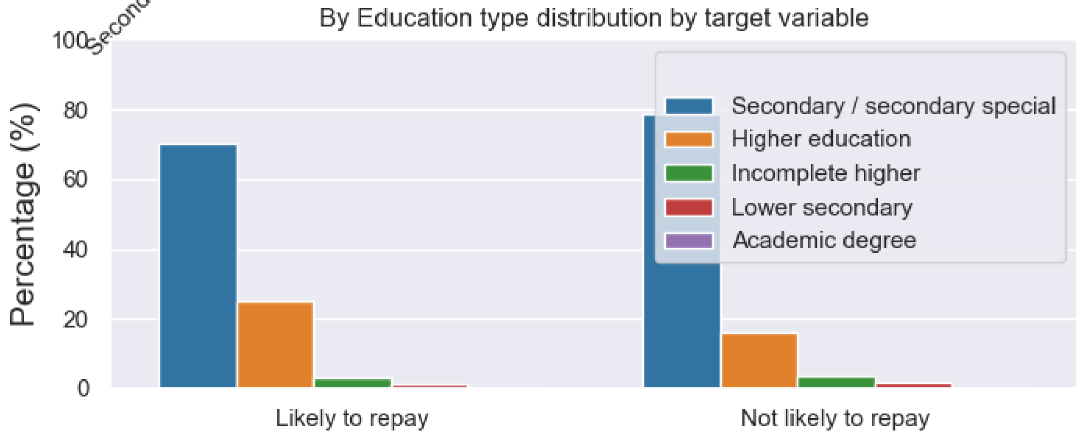




Most clients are from 'working class'. This number slightly increased for loan defaulter class

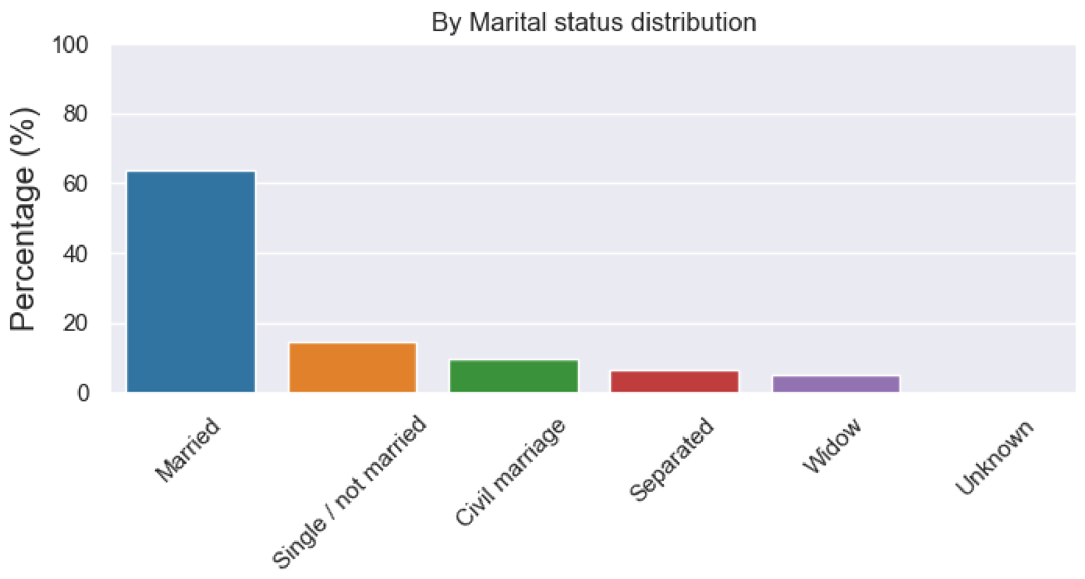
What Educational Background the Clients Come From?

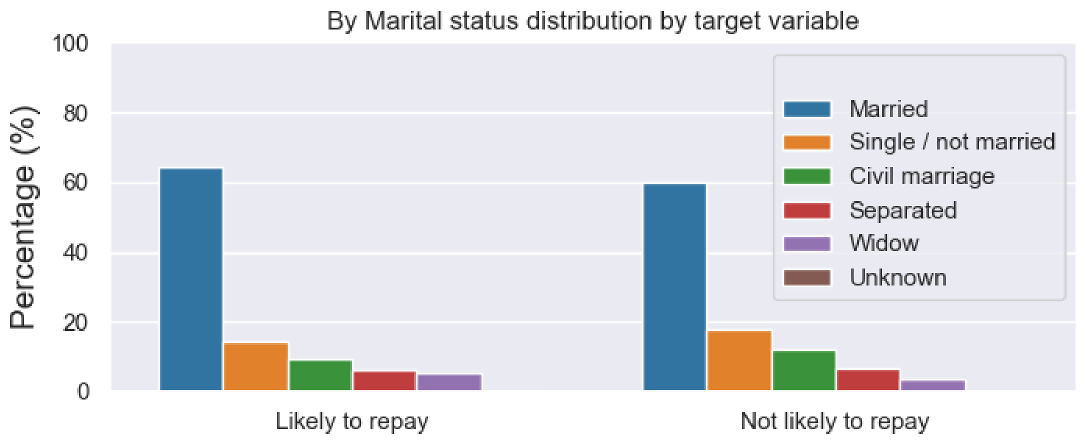




Most clients have 'secondary' education. This number increased for loan defaulter class.

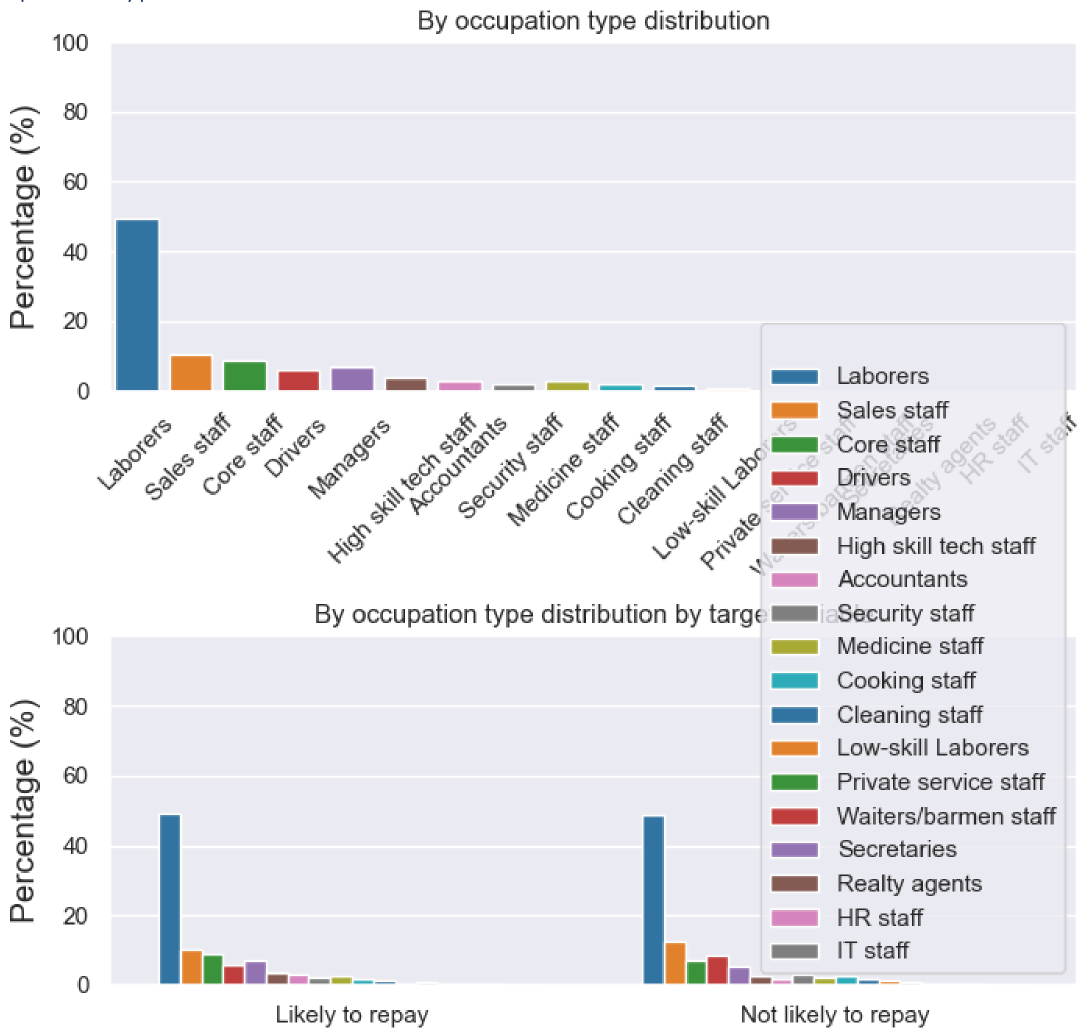
What are Client's Marital Status?





'Married' people applied for maximum loans. In the defaulter class, 'single' people segment increased than non-defaulter class

Occupation Types

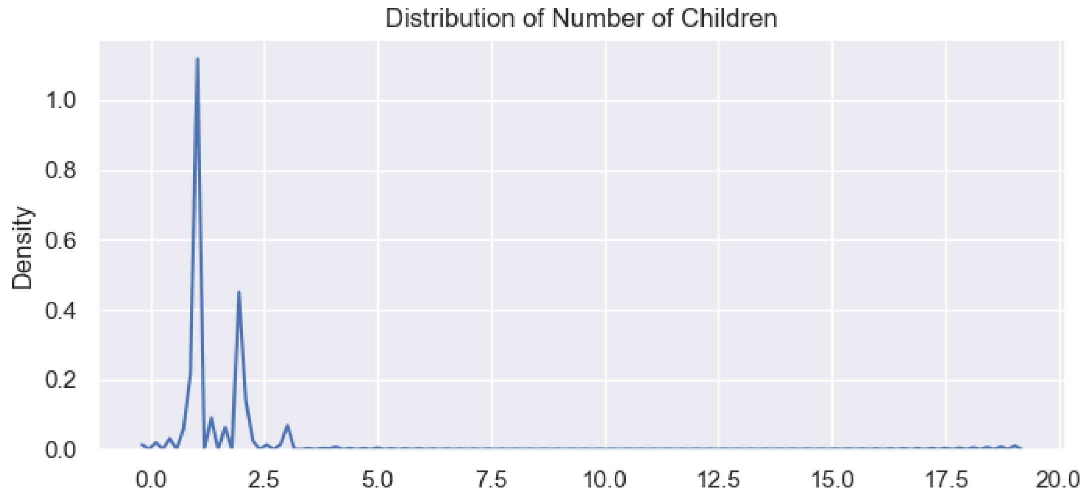


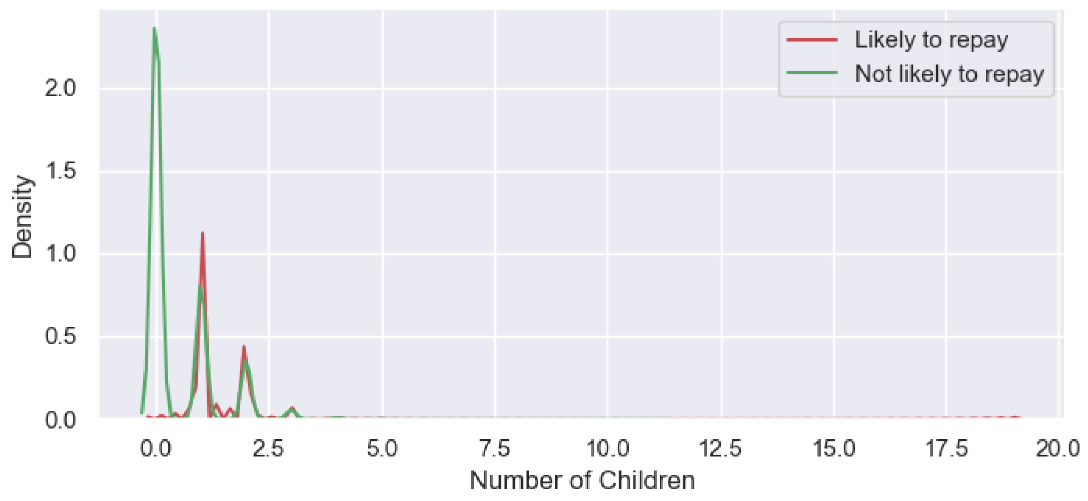
'Laborers' applied for most loans and they are dominant class in both defaulter and non-defaulter categories.

Insights into Numerical Variables

A detailed numerical variables distribution was done for every variables. For being succinct, six selective variables will be included here.

Any Variability in Loan Repayment over Number of Children?





Majority applicants have no children. Interestingly, the distribution extended till 20, which apparently seems suspicious and will be explored further. We will explore if the anomalous distribution is similar or very dissimilar with the non-anomalous trend. Now the challenge is what is the cut off number of children we will be appropriate to pick. In determining, we will pick a range and then find the maximum correlated number with target variable withing the range.

'Loan defaulter' percentage with more than 6 children jumped drastically from general figure of 8%. We will check if the 'anomolous' and 'non-anomolous' distribution is similar or significantly different using 'hypothesis' testing. As the data is highly un-symmetric, we will be using t-distribution.

Hypothesis are:

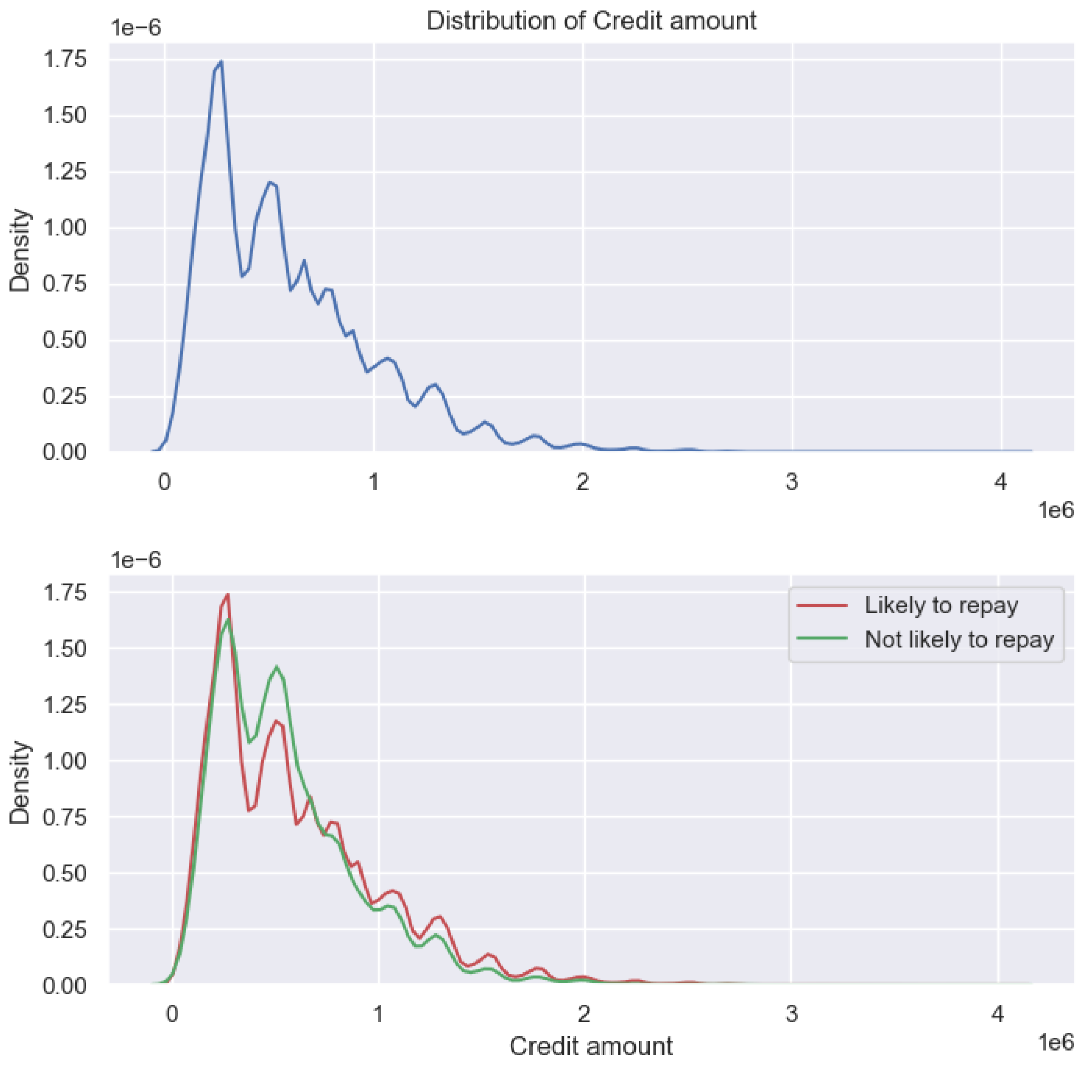
H0: µ1=µ2 (mean of 'target' between anomolous/non-anomolous group are equal)

H1: µ1!=µ2 (mean of 'target' between anomolous/non-anomolous group are different)

Target variable is highly imbalanced so, the distribution would not follow symmetrical shape. t-test will be used, to verify hypothesis set earlier.

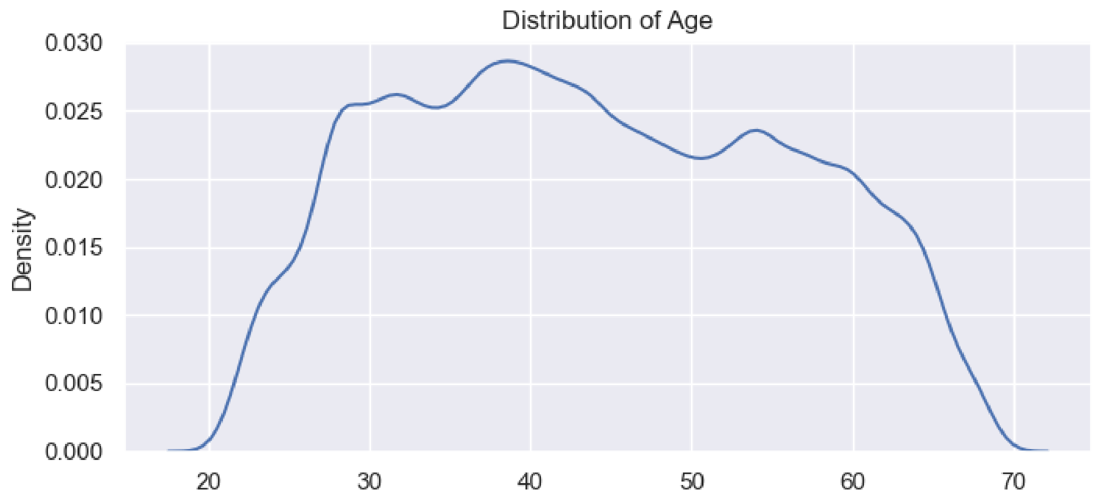
p-value is 2.08 which is way greater than 0.05. So we accept the null hypothesis, that the distributions are similar. Although hypothesis testing suggests anomolous and non-anomolous data distribution are similar, we saw big jump in % of loan defaulter when assuming cut off number of children is 6. We will keep this for future exploration whether adding this as new feature would make any difference by applying in model training.

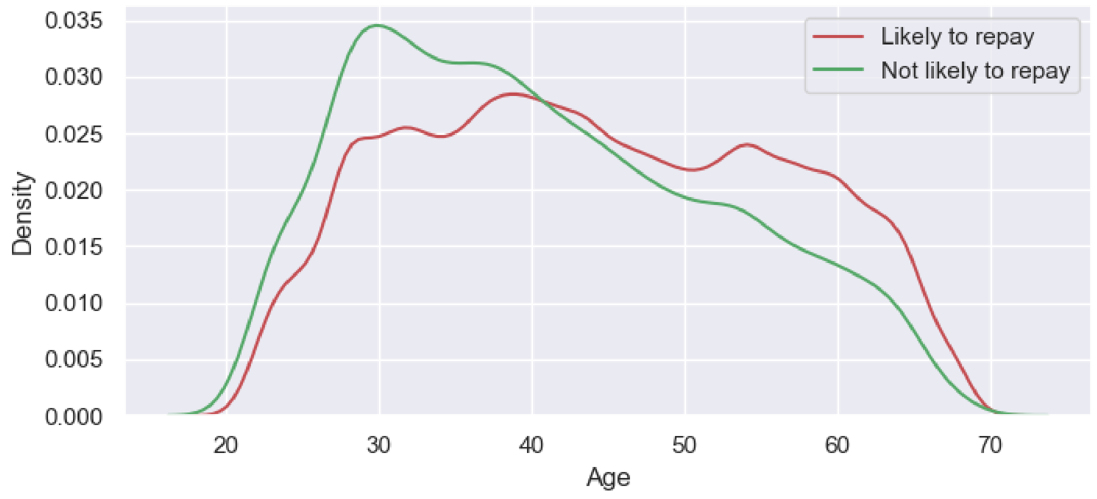
Credit Amount



There are certain credit range Home Credit disburses to the clients. Range between 0 to 1 million. The credit distribution looks normal.

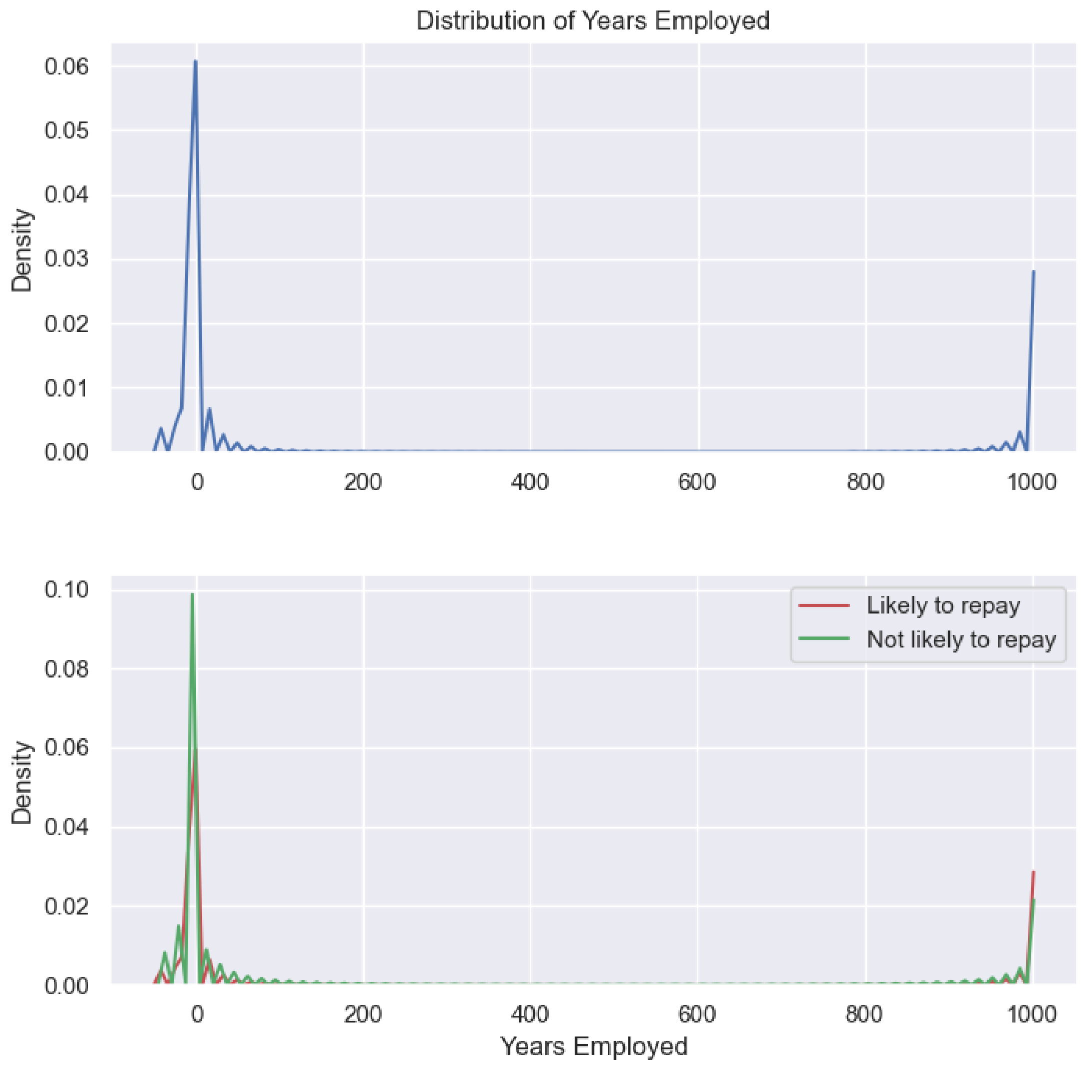
Age Distribution





The age data is well distributed over repayments. Clear distinction between likely to repay/not likely to repay over age. The distribution is skewed to the younger ages. That means, people around 30 years are less likely to repay loans. Relatively older population (>60) are highly likely to repay loans. This is expected to be an important feature.

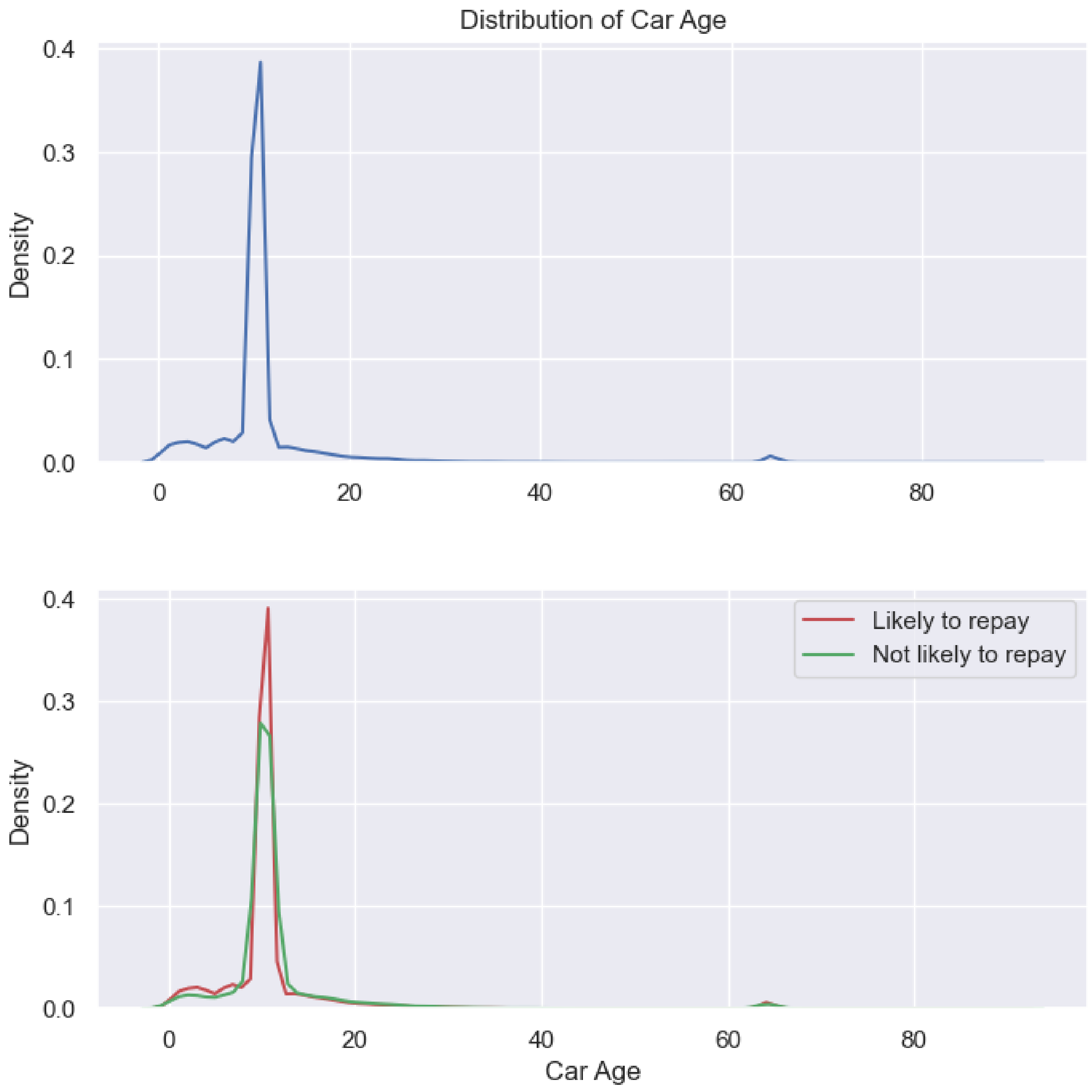
Years Employed



There is a bump around 1000 years, which is unrealistic for number of years employed. Lets determine optimum cut off years employed number for maximum correlation with target

Loan defaulter in the anomolous population is: 5.39%. This percentage is way lower than the overall percantage (~8%). We will mark these numbers as anomolous data. Replace YEARS_EMPLOYED rows greater than 201 with 0, as majority numbers are around 0 and the true values are hard to guess.

Car Age Distribution



Most car age ranged between 0 to 20 years. There is a cluster around 65 year which is suspicious. Cars age with more than 65 years is highly unlikely. Lets determine cut off car age for which maximum correlation will be found between anomolous data and target.

Although anomolous population distribution is closer to the original distribution (~8%), we will pass it on to the t-test to determine if anomolous/non-amolous have different distribution

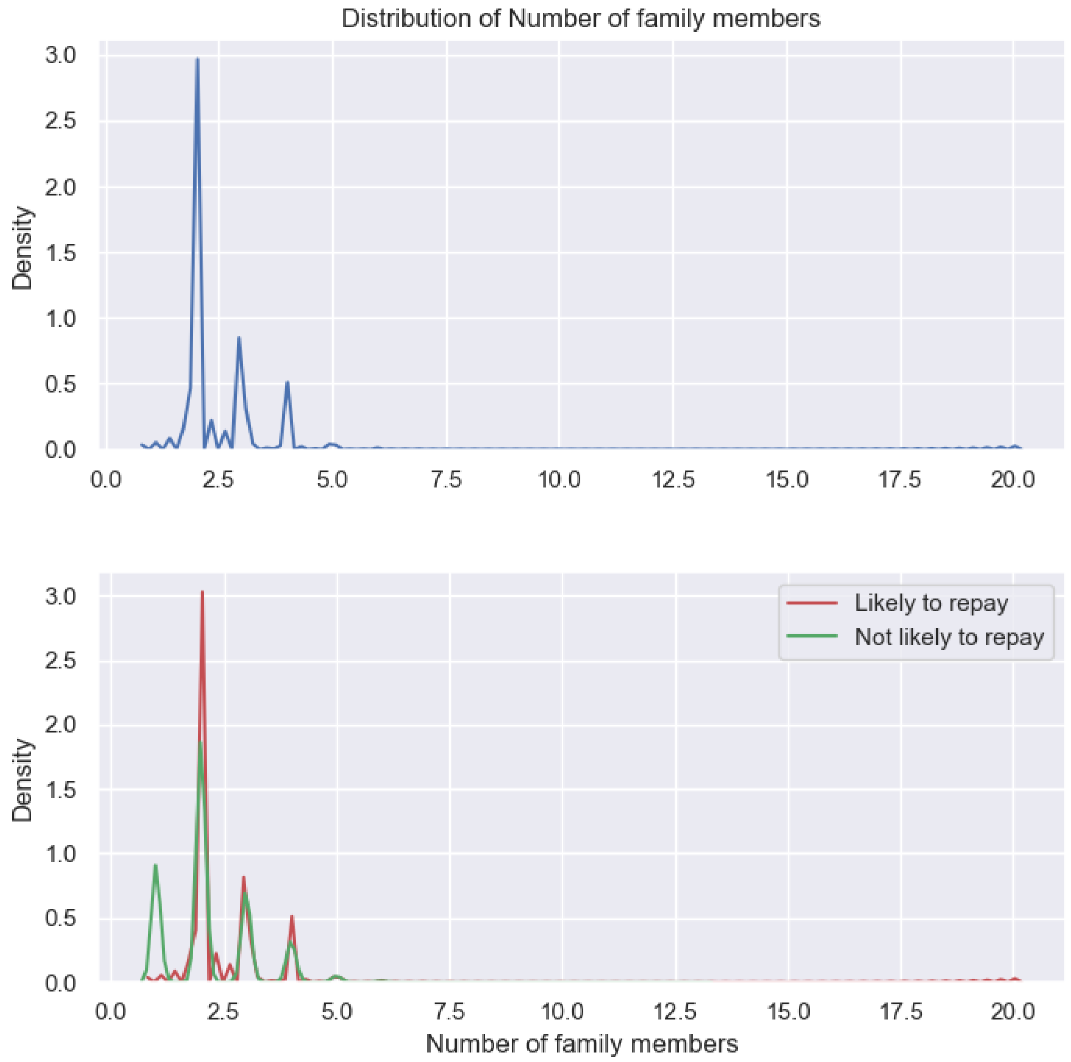
Hypothesis are:

H0: µ1=µ2 (mean of 'target' between anomolous/non-anomolous group are equal)

H1: µ1!=µ2 (mean of 'target' between anomolous/non-anomolous group are different)

p-value is 0.705 which is below 0.05 which suggests the amomolous/non-anomolous distribution are similar.

How Many Family Members Clients have?

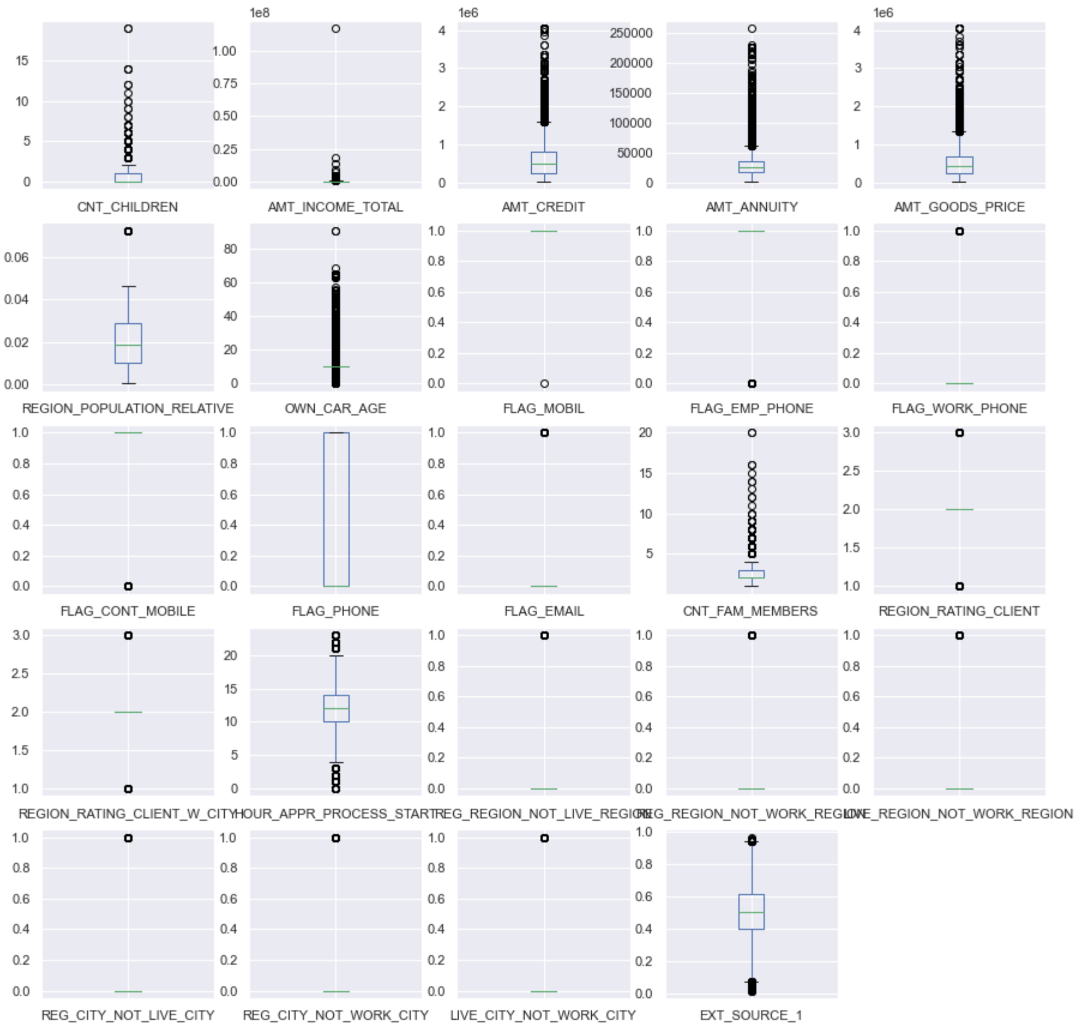


Note that, 2 family members have the majority class, i.e. couple applied for most loans. Lets find out cut off family number for maximum correlation between anomolous and target variable.

Loan defaulter for anomolous family member numbers are very high compared to general distribution (~8%). This make sense, as having high family members have high operating cost and less likely to return any loans. Extra attention will be paid for the anomolous numbers.

Anomalies and Outliers

Lets have an initial look into the data distribution by boxplot if any outliers are spotted. There are 78 columns which will be computationally heavy. We will be picking 25 columns and boxplot. This will give first indication if there is any outliers.



Many columns apparently fall outside of interquantile range (IQR) in the boxplots. Calculating IQR, no anomolous data was found.

Explore Data Relationship

Many column variables could be similar in patterns, which is redundant for modelling purpose. Based on high degree of similarities, variables will be removed for cleaner dataset.

Drop off Highly Correlated Variables

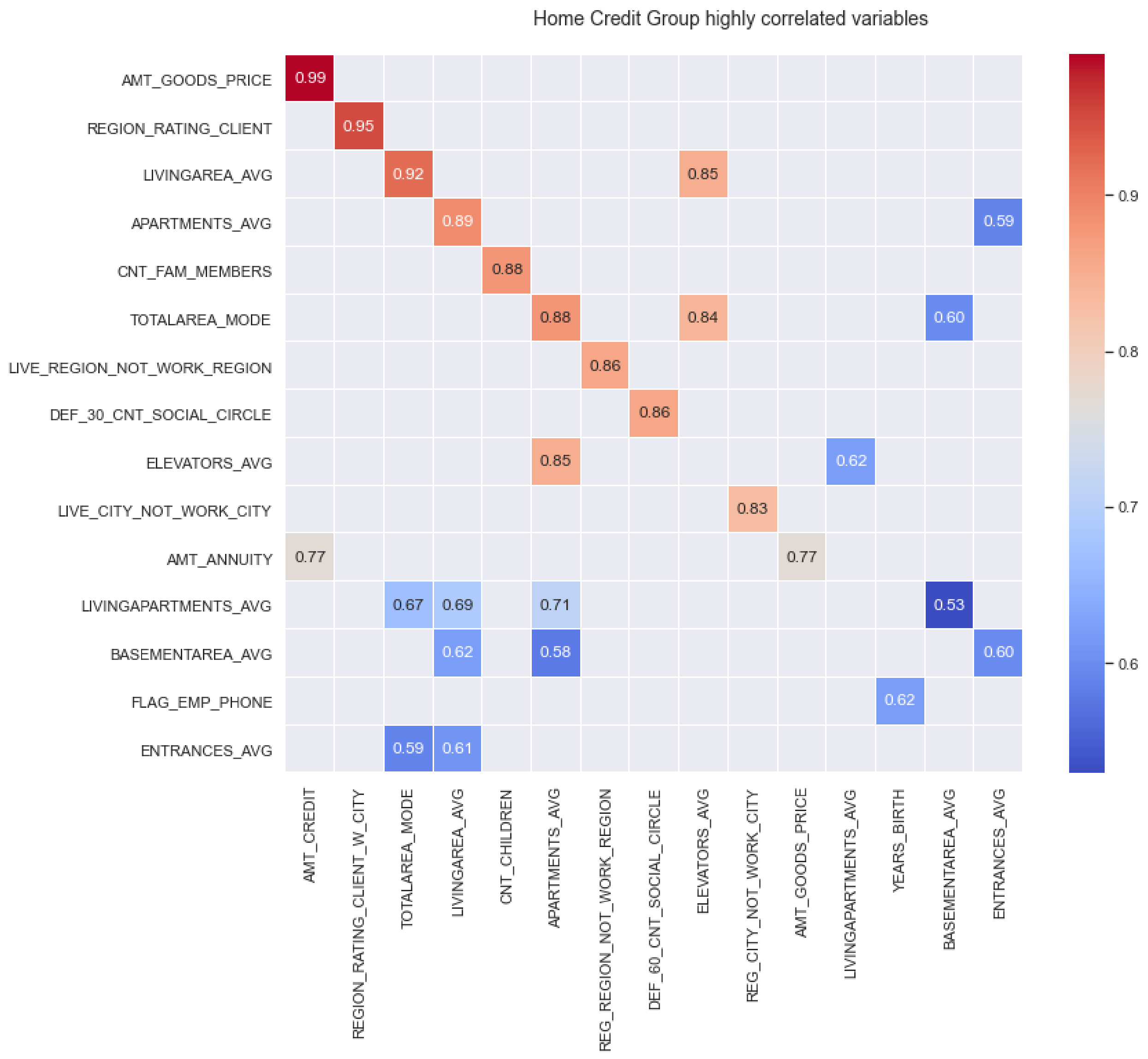
Based on Pearson's correlation coefficient, the variables will be grouped into high, moderate and low similarities

High Degree : >0.5 with maximum of 1.0

Moderate Degree : 0.3-0.49

Low Degree : <0.29

Highly Correlated Variables



Top highly correlated variables found to be:

- AMT_GOODS_PRICE----AMT_CREDIT (0.99). It is the price of the good for which credit is given, they are highly correlated
- REGION_RATING_CLIENT----REGION_RATING_CLIENT_W_CITY (0.95)
- LIVING_AREA----TOTALAREA_AVG (0.92). Bigger the total area, larger the living area would be

Highly correlated variables add very little information in the modelling and hence considered redundant. In this section, highly correlated variables (correlation coefficient >0.8) will be dropped.

Feature Creation

Feature engineering refers to adding additional features to the existing features to increase information content in the training data. It is very crucial to implement a meaningful machine learning modelling. Feature engineeing could be categorized into two types.

- Automatic feature selection (featuretools library comes handy for this)
- Manual feature engineering (i.e. anomolous features, observations) For this work, we will stick to manual feature engineering which will be developed from anomolous features and observations.

Manual features will be added from anomolous features and observations. Earlier, we identified anomolous features which can add extra information to the dataset. In addition, we can create new features from combinations of other features which will be called features from observation.

Anomolous Features

The anomolous columns identified earlier indicate that, the anomolous/non-anomolous data can be used as extra information while training the model. In this section, we will create features based on the anomolous data. Anomolous rows will be indicated by 1 whereas non-anomolous by 0. Created anomolous features are: 'CNT_CHILDREN_ANOM', 'YEARS_EMPLOYED_ANOM', 'OWN_CAR_AGE', 'CNT_FAM_MEMBERS'.

Observed Features

From intuition, we can form some features which might be influential on determining if the client will be repaying loans or not. Individual numerical features without the Boolean features were watched to check if meaningful new features could be formed.

Following observations were found:

- Percentage of annuity over total income, could be a measure of ability to repay the loan.
- Amount of credit taken over total income could be another measure. Because income shortage could lead to taking credit
- How much annuity money payment clients made over credit amount. Again, annuity is a form of payment and can lead people toward taking credit
- How many years employed compared to their age. The ratio between these two may tell peoples ability to repay loans. Higher the number better is expected repaying loans
- Apartment area over the income amount. If the house is bigger compared to income level, people may turn to taking credit more
- Car age in peoples life span. Longer people keep their cars, high chance to make monthly installment and turn to credit
- House age in peoples life span. Longer the house people keep, high chance to make monthly installment and turn to credit We will make new features based on the observations

Based on the above observations following new features were created: 'ANNUNITY_OVER_INCOME', 'CREDIT_OVER_INCOME','ANNUNITY_OVER_CREDIT','EMPLOYED_OVER_AGE','APARTMENT_OVER_INCOME', 'CARAGE_OVER_AGE', 'BUILD_OVER_AGE'.

Top three correlation coefficient for the new features with target were found to be:

FEATURES	CORRELATION COEFFICIENT
BUILD_OVER_AGE	0.070009
ANNUNITY_OVER_INCOME	0.014267
ANNUNITY_OVER_CREDIT	0.012695

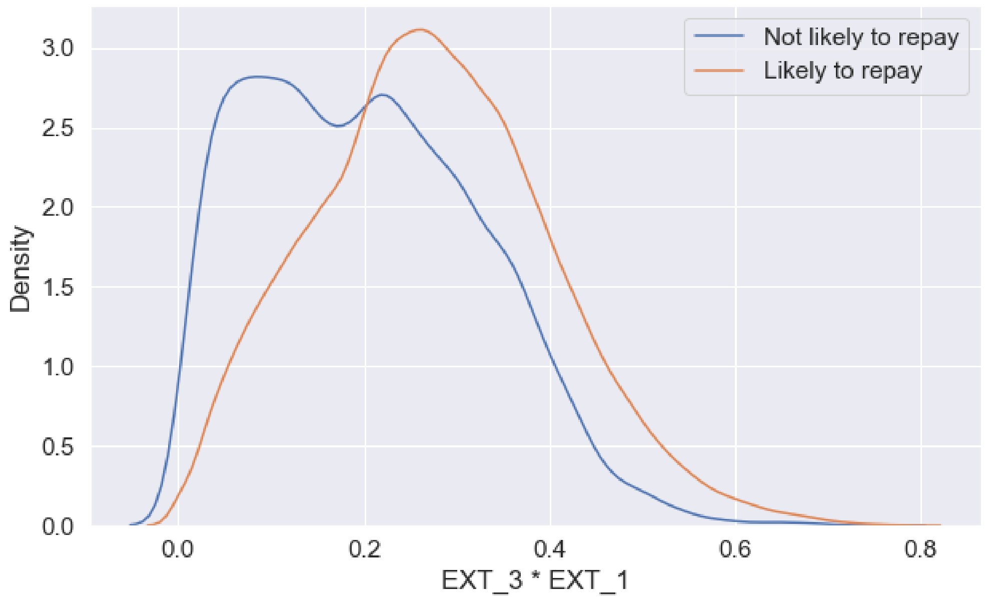
Multiplicative Terms

Highly correlated features with target variable were sorted out. They will be used for additional feature generation, as they can drive up the target variable. The top three correlated features with target variables are:

FEATURES	CORRELATION COEFFICIENT
EXT_SOURCE_2	0.160249
EXT_SOURCE_3	0.148701
EXT_SOURCE_1	0.086761

- Top 3 correlated features are EXT_SOURCE_3, 2 and 1
- Division and multiplication between these variables were tried and found that multiplication terms produced distinctive distribution for the target variable
- Additional features would be: 'EXT_3_2', 'EXT_3_1', 'EXT_2_1'

As a illustration distribution of EXT_3_1 was shown below:



The distribution for EXT_2*EXT_1 is distinct for loan defaulter/non-defaulter which is expected to be a better predictor for target variable.

At this stage, the highly correlated variables removed, and newly added variables were saved as a dataframe for the use in pre-processing stage.

A Detailed data Wrangling and EDA for this project can be found in this link.

Data Pre-processing

Convert Categorical Variables into Dummy Variables

Many machine learning models can not handle categorial variables during training. For example, neural network. We will also run algorithms that can handle categorical variables. It is important to systematically convert categorical variables into meaningful numeric variables. First, we will determine for every categorical column how many unique categories are there. Two of the most popular techniques are label encoding and one hot encoding. Label encoder is used where subcategories needed weighting. For example, 'low', 'medium', 'high'. Whereas one hot encoding splits the categories and put equal weight. We will use one hot encoding instead of label encoding as none of the categorical variables needed weighting according to the sub-categories. Too many subcategories are also undesirable. It will cause 'curse of dimensionality' problem. Which means too many features reduces algorithms ability to cluster similar things together. Features with more than 10 subcategories will be deleted as a cardinal to avoid 'curse of dimensionality' situation.

The whole dataframe was parsed into the following groups to have better maneuverability: [additional features] [original categorical features] [one hot encoded features] [numerical features] [target variable]

The number of column breakdown for the rearranged dataframe is:

Number of columns in df_additional_features	**13**
Number of columns in df_categorical	14
Number of columns in df_one_hot_code	64
Number of columns in df_num	80

Standardize the Magnitude of Numerical Variables

This was applied to avoid bias when there are differences in magnitude among the numerical variables. In this case, we have seen stark differences between numeric variables. For example, annual income, annuity and credits differ sharply in magnitude from other variables. Every numeric variables were scaled except the booleans one.

Balance the Data

From EDA step, we have seen deafaulter and non-defaulter percentage is around 92% vs 8% which is highly imbalanced. Imbalance data set introduces high percentage of biases towards majority class during training. Three popuar methods to combat imbalanced data:

- Over-sampling

- Under-sampling
- Synthetic data creation

We will be using under-sampling to balance the data to 50-50 ratio as it will keep the integrity of the data.

Shuffle the Data

In many cases data is collected in orderly fashion. This becomes particularly problematic when running batch processing for model training. We will shuffle the data to be in the safe side to spread out target variables as much as possible.

Split Data into Train, Validation and Test Set

Split the scaled, balanced dataset into train, validation and test dataset. We will split by 80-10-10 ratio.

The data was made 50-50 balanced. After splitting into train, validation and test set, it was important to check the individual sets are still balanced.

Loan defaulter percentage in train set	**50.09 %**
Loan defaulter percentage in validation set	49.71 %
Loan defaulter percentage in test set	49.57 %

The scaled, balanced, shuffled and split train, validation, test dataset was saved for the use in modelling stage.

Machine Learning Modelling

Four algorithms were trained. Every algorithm was hyperparameter optimized. A combination of features such as categorical vs one-hot-encoded, with/without additional features were tried to see which one yielded better performance.

Deep Neural Net with TensorFlow 2.0

We will be using TensorFlow 2.0 library to build deep learning model for loan defaulter classification. A brief discussion about different optimization algorithms, loss functions, metrices are below:

Optimizers

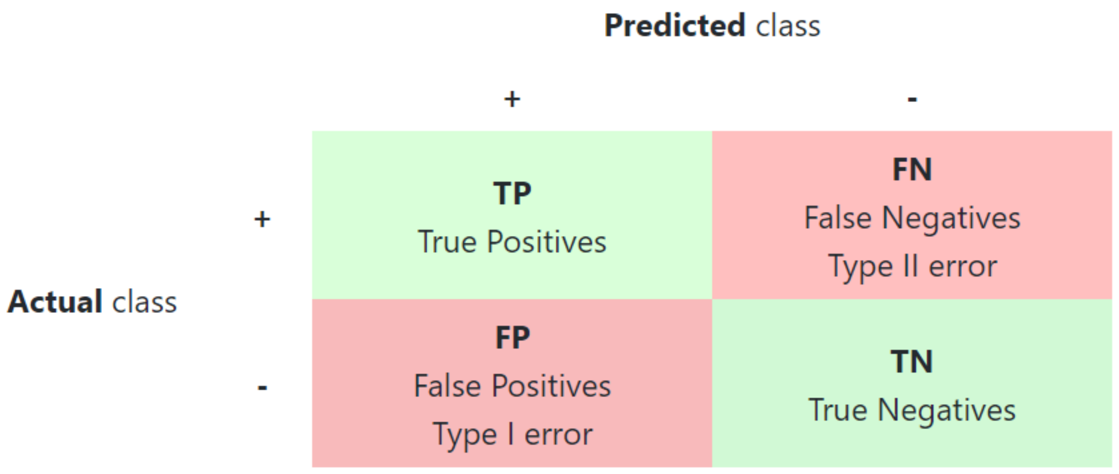
- **Gradient Descent (GD):** Iterates through whole training set. Updates once in an epoch. Slow in speed.
- **Stochastic Gradient Descent (SGD):** Updates weights multiple times in an epoch defined by batch size. Faster in speed.
- **Adaptive Momentum (Adam):** Adaptive learning rate coupled with momentum help the algorithm overcome any local peak and ensure reaching the minimum global peak. We will choose Adam for best performance.

Loss Function

Cross-entropy would be the choice for classification problem. There are three options in TensorFlow 2.0. They are: binary, categorical and sparse categorical cross entropy. Binary expects the data is binary encoded. Categorical expects the data is one hot coded. Sparse can one hot code data during the training. To be in safe side we will apply sparse categorical cross entropy.

Metrics

Confusion matrix is used to get a full picture when assessing the performance of a classification model. The components of a confusion matrix have been shown below:



		Predicted class	
		+	-
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

Image taken from: Stanford.edu

Some other useful metrics can be derived based on the elements of the confusion matrix

- **Accuracy**: Measures the fraction of correctly classified samples from every category

$$\frac{TP + TN}{TP + FP + TN + FN}$$

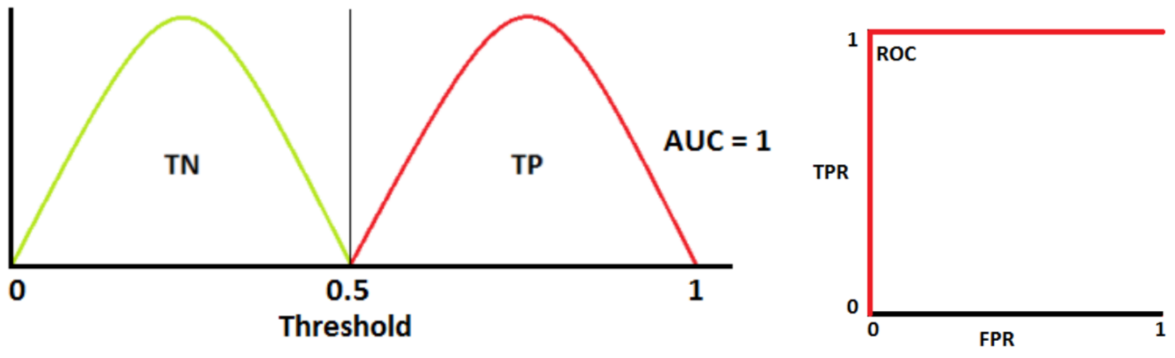
TP = Predicting actual positive samples as positive FP = Predicting a sample positive but actually negative e.g. predicting a client loan defaulter whereas he is able to repay loans FN = Predicting a sample negative but actually positive e.g. predicting a client non-defaulter whereas he would not be able to repay TN = Predicting actual negative samples as negative

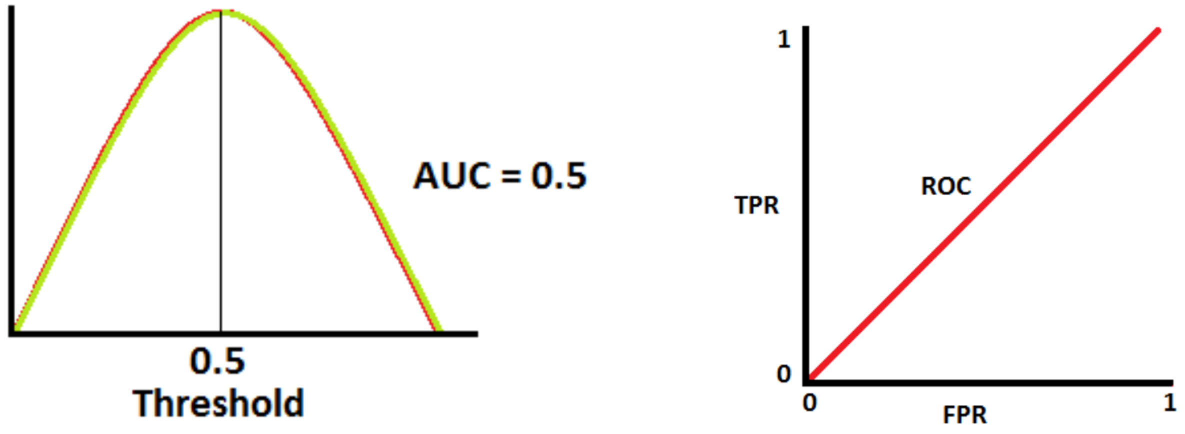
- **ROC curve**: Measures the ability of a model to correctly classify samples at different threshold levels. It plots two parameters, FPR vs TPR

$$FPR = \frac{FP}{FP + TN}$$
$$TPR = \frac{TP}{TP + TN}$$

For a binary classifier, ROC curve shows the ability to correctly classify between 0's and 1's at different threshold levels.

- **AUC score**: Is a single number, that measures the total integral area under the ROC curve which is a measure of separibility between classes. Higher the AUC score better is the classifiers at distinguising 0's as 0's and 1's and 1's. For or case, higher is the AUC better the model to distinguish between loan defaulters and non defaulters. An excellent illustration has been provided in the following link: [https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5]





In the above figure, the two distribution completely overlap. The chances of separability between two classes is 50% (AUC score). This is the worst possible performance a classifier can have.

We will be using mainly AUC score for assessing our models, in addition to Accuracy and ROC curve.

Dataset Preparation

In addition to training data, following dataset are required and their reasoning.

- **Validation Set:** To make sure the model parameters (weights, biases) do not overfit
- **Test Set**: To make sure the model hyperparameters (width, height, batch size etc) do not overfit

Base Model Development

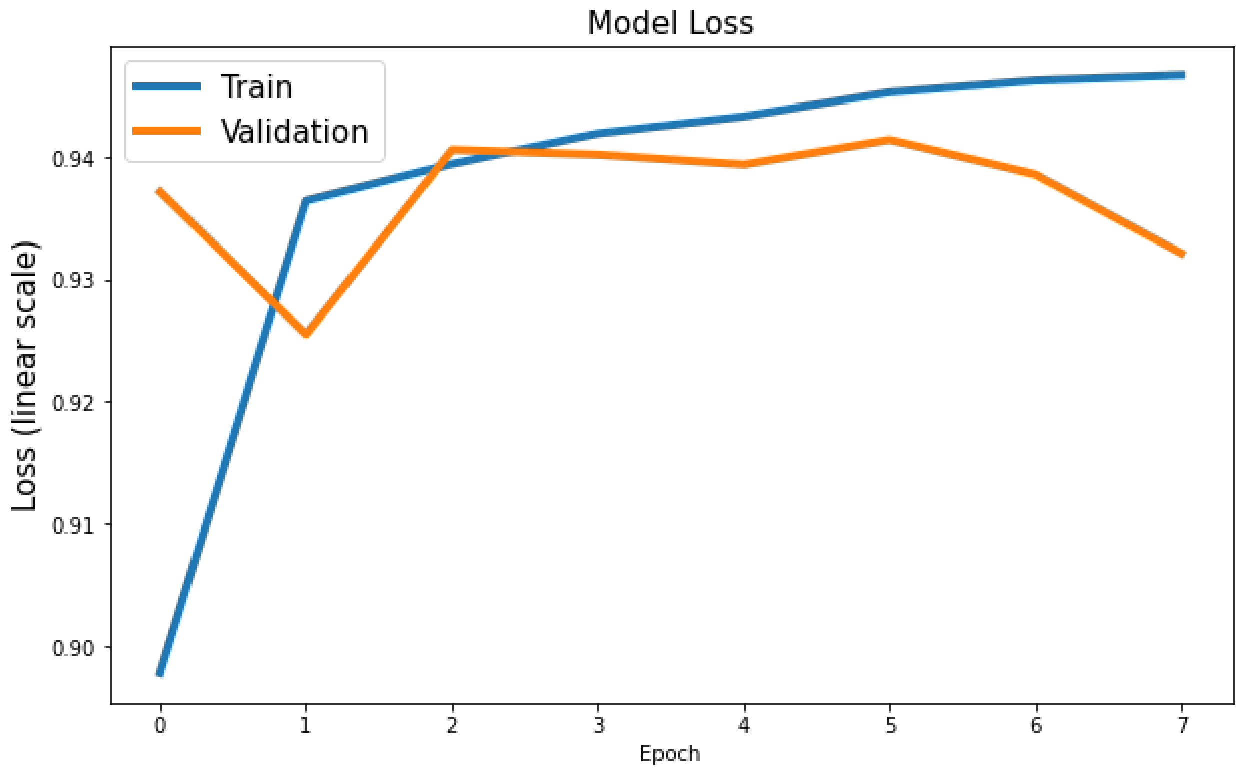
A baseline deep learning model was developed with two hidden layers. 'ReLu' activation function was used in the hidden layers. In the output layer, 'softmax' activation function was used to get the probabilities of binary classes. Different activation functions were tweaked in the later part of this section.

Metrics

For the binary classification problem, best metric would be AUC score along with accuracy measure.

Accuracy

A validation accuracy of 93.21% was found with 'early stopping' of 2 which means that when the validation accuracy drops two consecutive time while the train accuracy keeps increasing, the model will stop training.

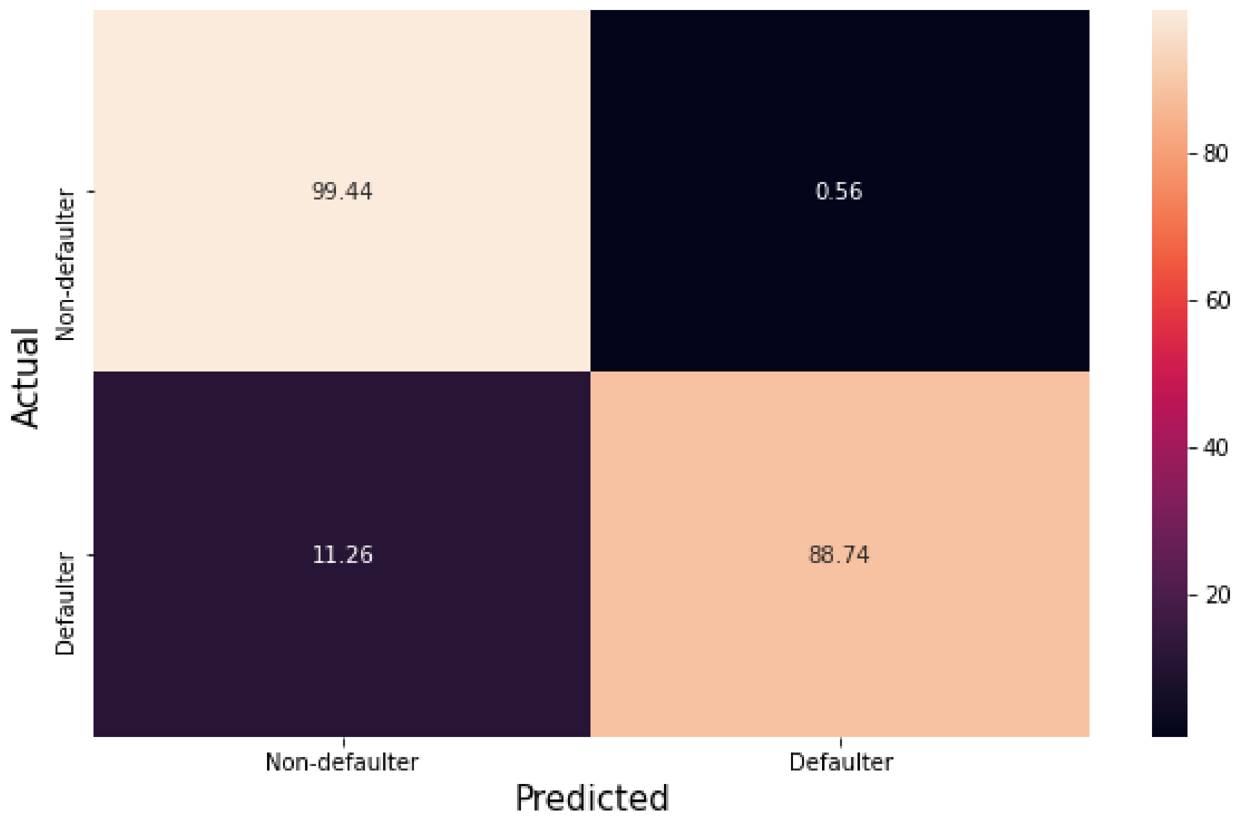


A test accuracy of 93.82% was found. As the measure of overfitting is the difference between validation and test accuracy, very less over fitting is likely.

AUC Score

An AUC score of 0.97 was calculated which is a very good number and tells that the model has 97% separability between the two classes.

Confusion Matrix

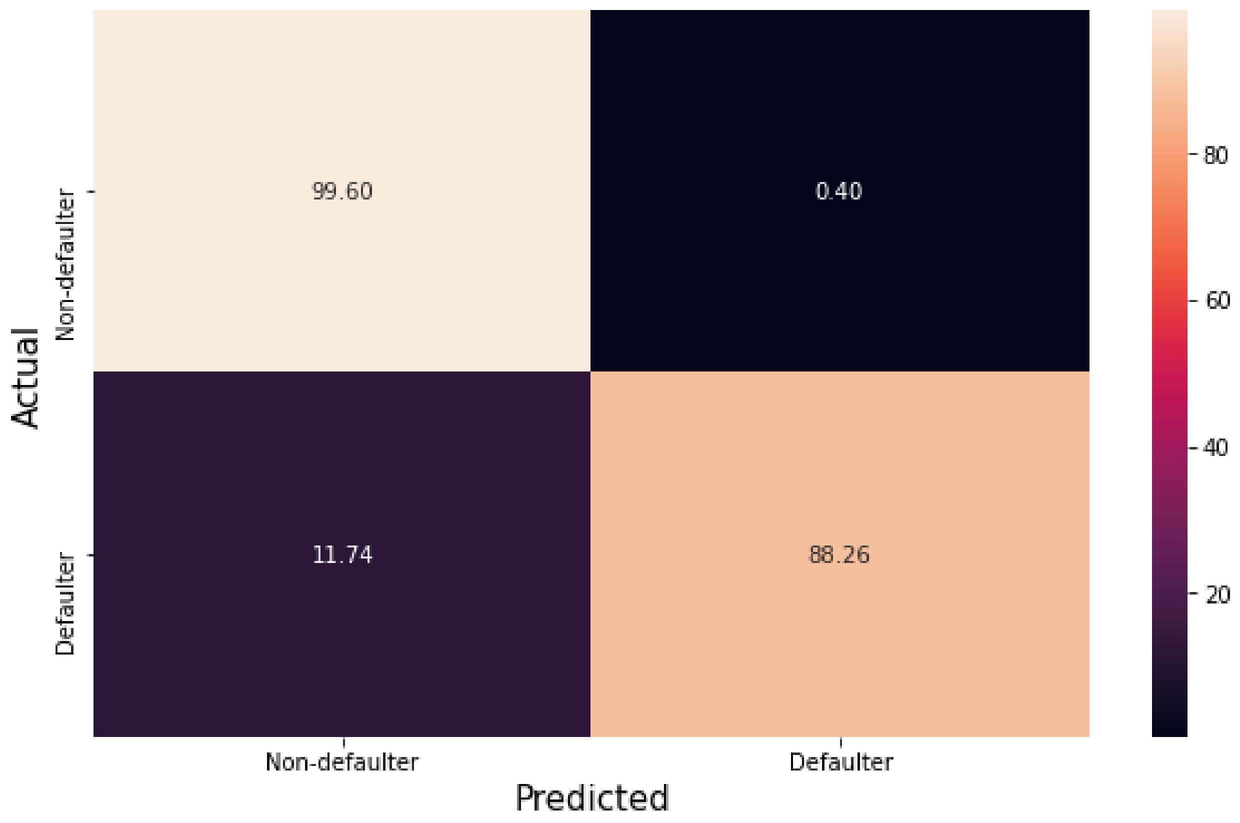


The model has high ratio between FN (1%) and FP (~11%) which would predict more actual loan defaulter as non-defaulter. Desirable is to get relatively small FP and FN in a balanced proportion.

Effect of Additional Features on Model Performance

In this section, we will explore the effect of additional features manually added during EDA stages on model performance.

Confusion matrix



Added features slightly decreased AUC score by 0.03 whereas keeping the confusion matrix scores almost same. We will not include additional features for future analysis

Dataset without Non-standardized Variables

Non-scaled features were trained instead of scaled one to see the effect on model performances. The observations are:

- Non standardized dataset worsened model performance
- We will apply one-hot encoded and standardized variables for subsequent analysis

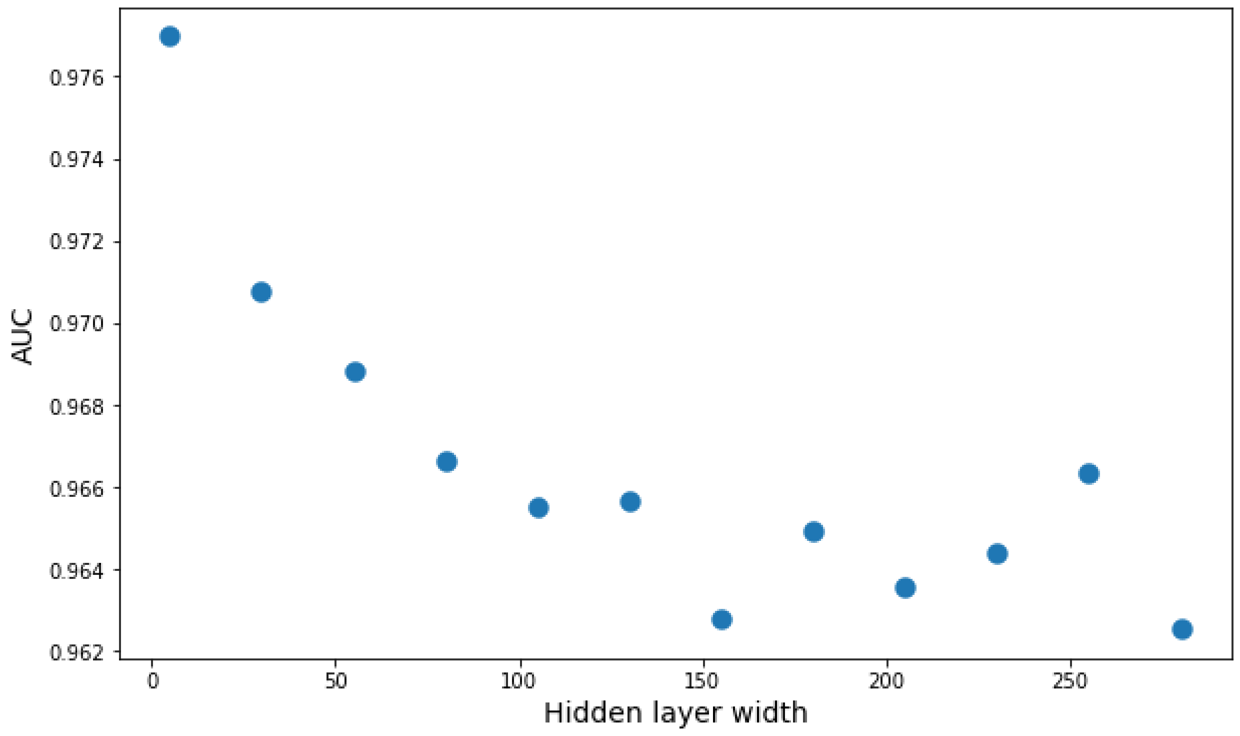
Hyperparameter Optimization Model

Hyperparameters for Deep Neural Net model are:

- Number of hidden units (width)
- Number of hidden layers (height)
- Combinations of width and height
- Activation function (Relu, tanh, leaky Relu, sigmoid)
- Batch size (in a range between 1=SGD and 10,000)
- Learning rates (high at beginning low at end)
- Dropout

Width

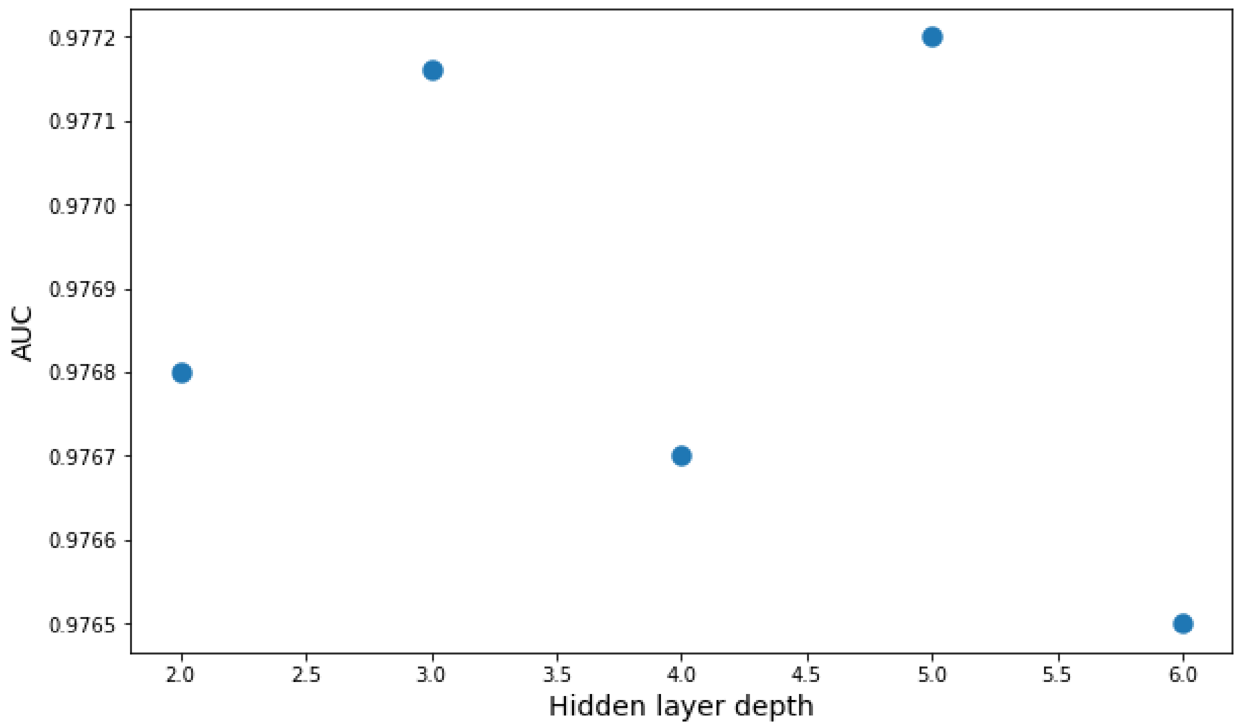
Width of the hidden layer was varied over 5 to 300 in increments of 25. For a particular model parameter we ran the model 10 times and took the average metrics to avoid getting random scores.



We saw a decreasing trend of AUC score with the increase of width. It appears the significant information content are limited to first couple of dominant features signalled by low number of hidden layer width (5). For the subsequent analysis, a hidden layer width of 5 will be used

Depth

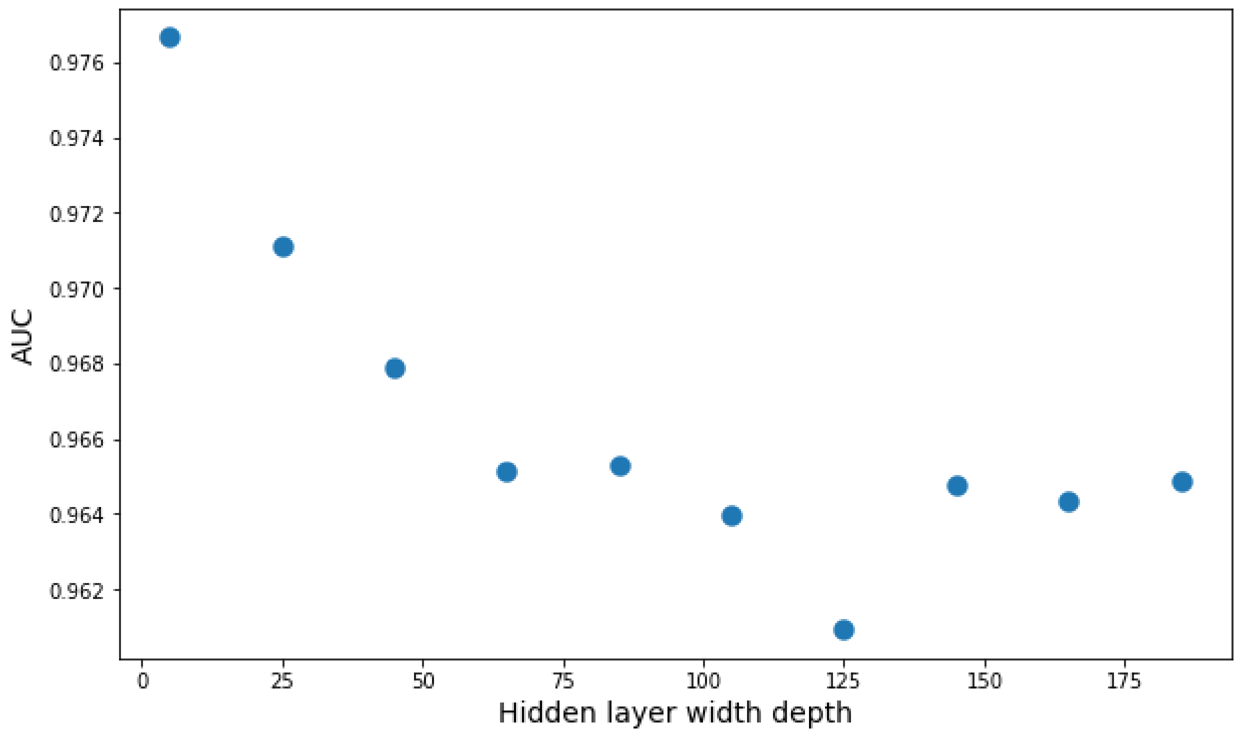
Number of hidden layers were varied for up to 6 layers starting from 2. For each parameter 50 runs were passed and took the average score to get consistency in the metrics.



A clear upward-downward trend was seen for the AUC vs number of depth layer plot. The maximum AUC was found for 5 hidden layers.

Width and Depth

Keeping the depth at 5 the width again to see if the performance improves for another width or remains the same.

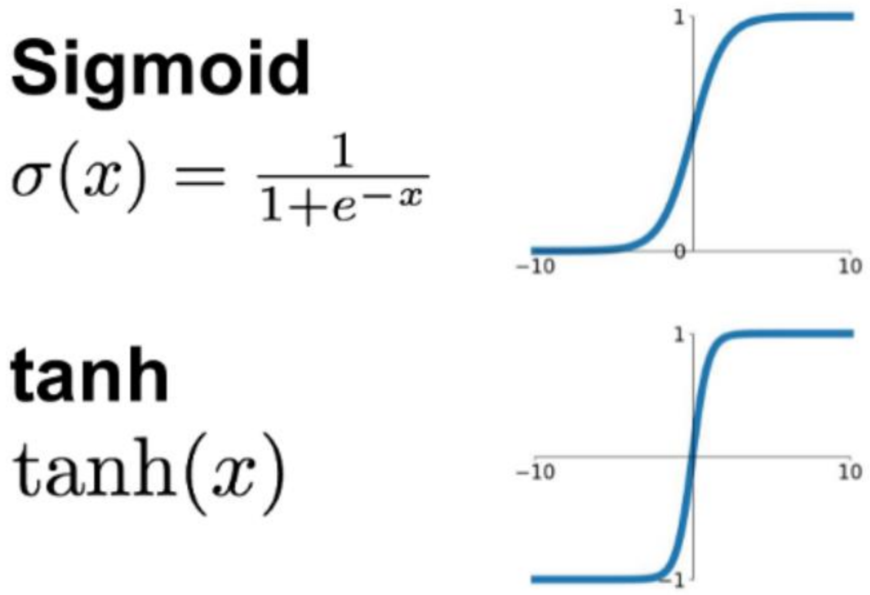


With increase in hidden layer depth in general decrease trend has been seen for AUC metric. With a depth of 5 layers, best model performance would be with hidden layer size of 5.

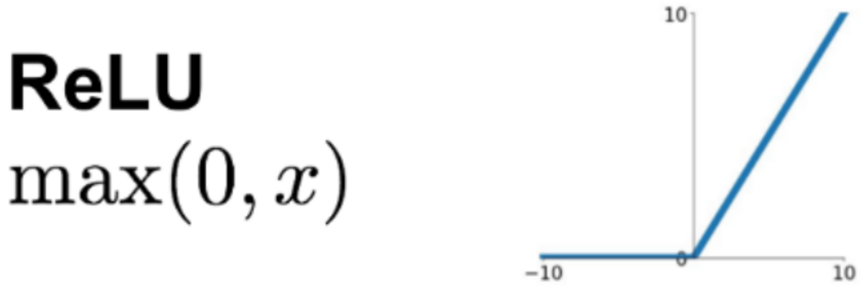
Activation Function

The pros and cons of commonly used activation functions have been summarized below:

Sigmoid and tanh: Comes with the problem of vanishing gradient, updates final layers faster than initial layers.

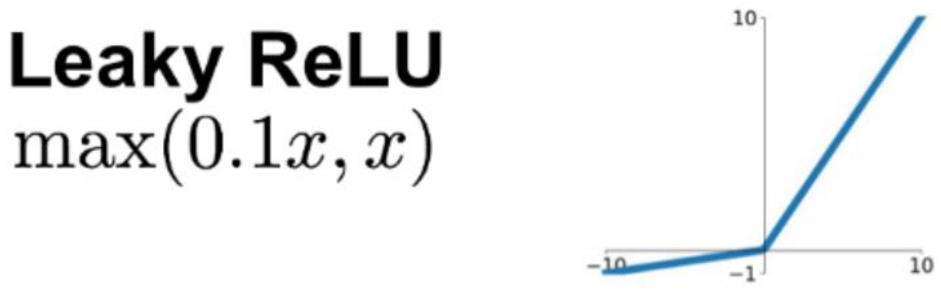


ReLu:



Faster, easy convergence, free from vanishing gradient problem. But it suffers from Dying ReLu problem, once a node gets negative number it becomes zero for the left ReLu curve and unlikely to produce any number.

Leaky ReLu: Fixes the dying ReLu problem.



Softmax: Provides probabilities of numbers, suitable for output layer for classification problem Images were taken from: [https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044]

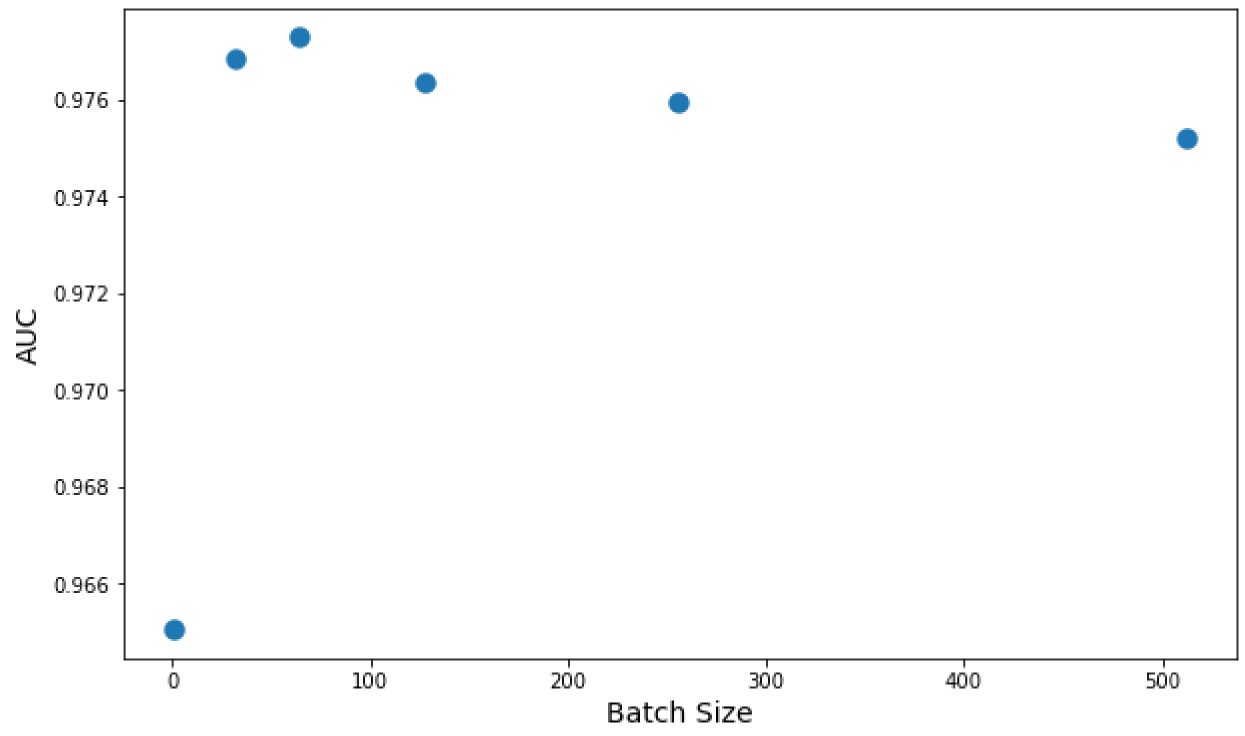
In this part, we will keep the last layer activation function as 'softmax' and all but the first hidden layers as 'ReLu'. We will keep changing the first hidden layer activation function and see the performances.

ACTIVATION FUNCTION	AUC SCORE
RELU	0.9766
SIGMOID	0.9726
TANH	0.9653
LEAKY RELU	0.9639

It has been observed that, all 'ReLu' in the hidden layers performed best for this dataset

Batch Size

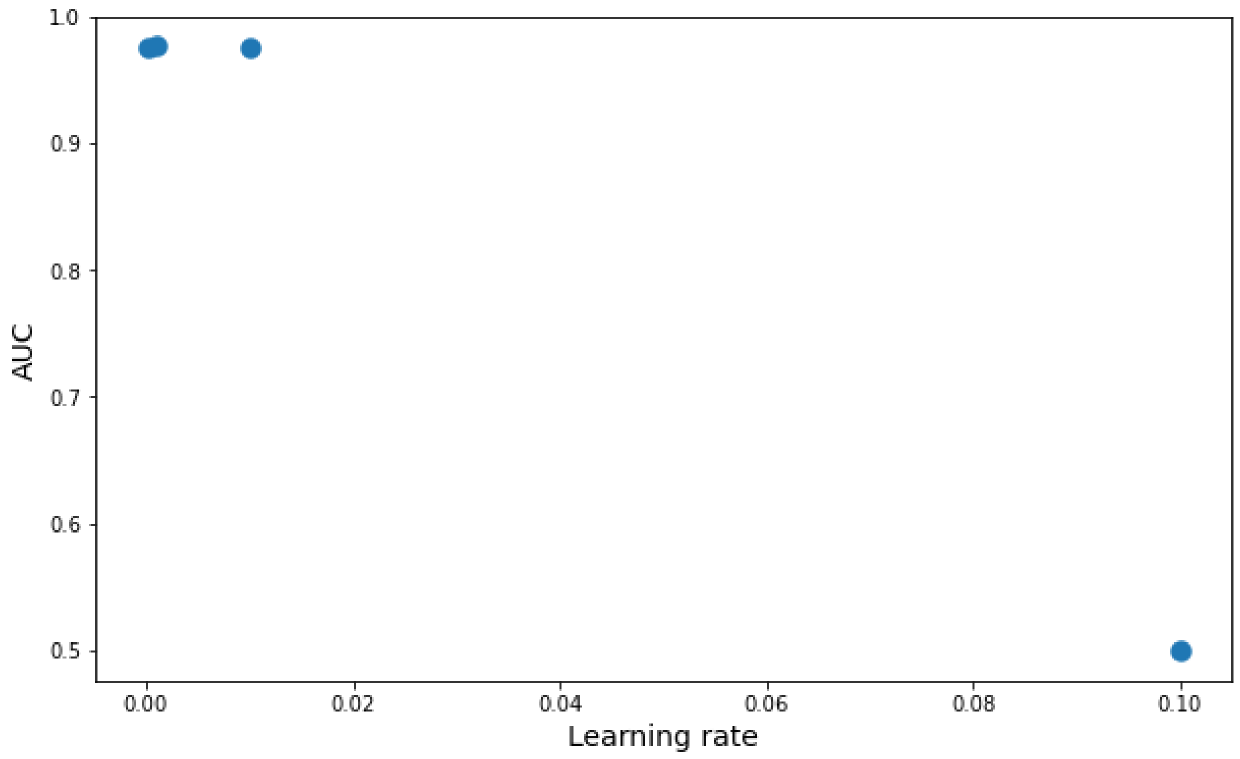
For, Stochastic Gradient Descent (SGD) batch size is 1 which is known for faster training time. Gradient Descent (GD) has batch size equal to the sample size which is the slowest to train. Minibatch GD has batch size between SGD and GD and a trade off between SGD and GD. We tried ranges of 'batch size' and compared which one yielded best results.



For a batch size of 64 maximum AUC score was found to be 0.9773.

Learning Rate

Model was trained with learning rates of 0.0001, 0.001, 0.01, 0.1.

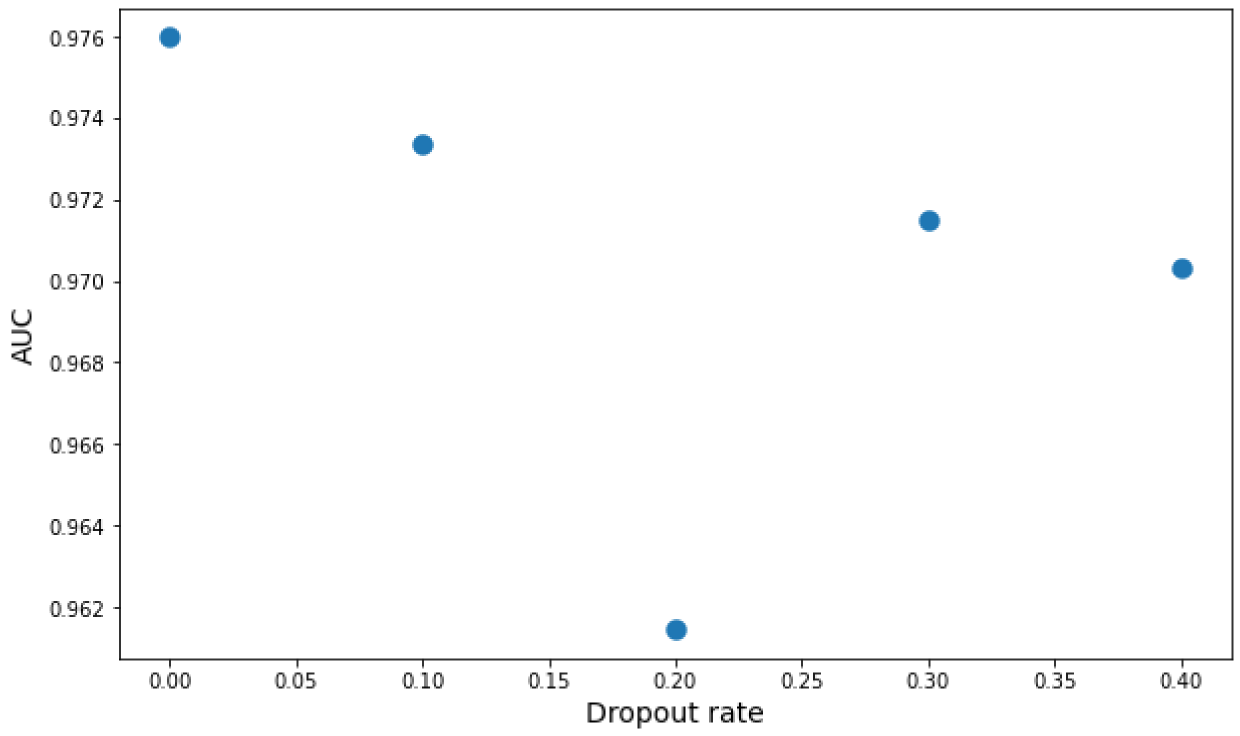


Maximum AUC is achieved for the default learning rate of 0.001.

Dropout

Dropout is a technique to reduce overfitting during training in the model. By specifying dropout, we are telling the model to ignore specified number of nodes, so that the other nodes can take on more responsibility to learn. Dropout can be added to the input layer as well as to the hidden layers. A maximum dropout rate of 0.5 provides maximum regularization also may results in under learning. Too much low drop out rate on the other hand may be too insensitive to learning. We will explore the following scenarios:

1. Dropout at the input layer vs no dropout
2. For hidden layers, vary dropout rate between 0.0 (without dropout) to 0.4



Following observations were made:

- Adding dropout layer at the input worsened the model performance
- Learning rate also increased but 0.001 produced best performances

For this dataset, performance is better without any dropout layer.

So, the optimized hyperparameters for the Deep Neural Net were found to be:

HYPERPARAMETERS	OPTIMIZED NUMBERS/PARAMETERS
WIDTH	5
DEPTH	5
ACTIVATION FUNCTION	ReLu
BATCH SIZE	64
LEARNING RATE	0.001
DROPOUT	No drop out is recommended

Metrics

With the optimized hyperparameters model was trained and following metrics were found:

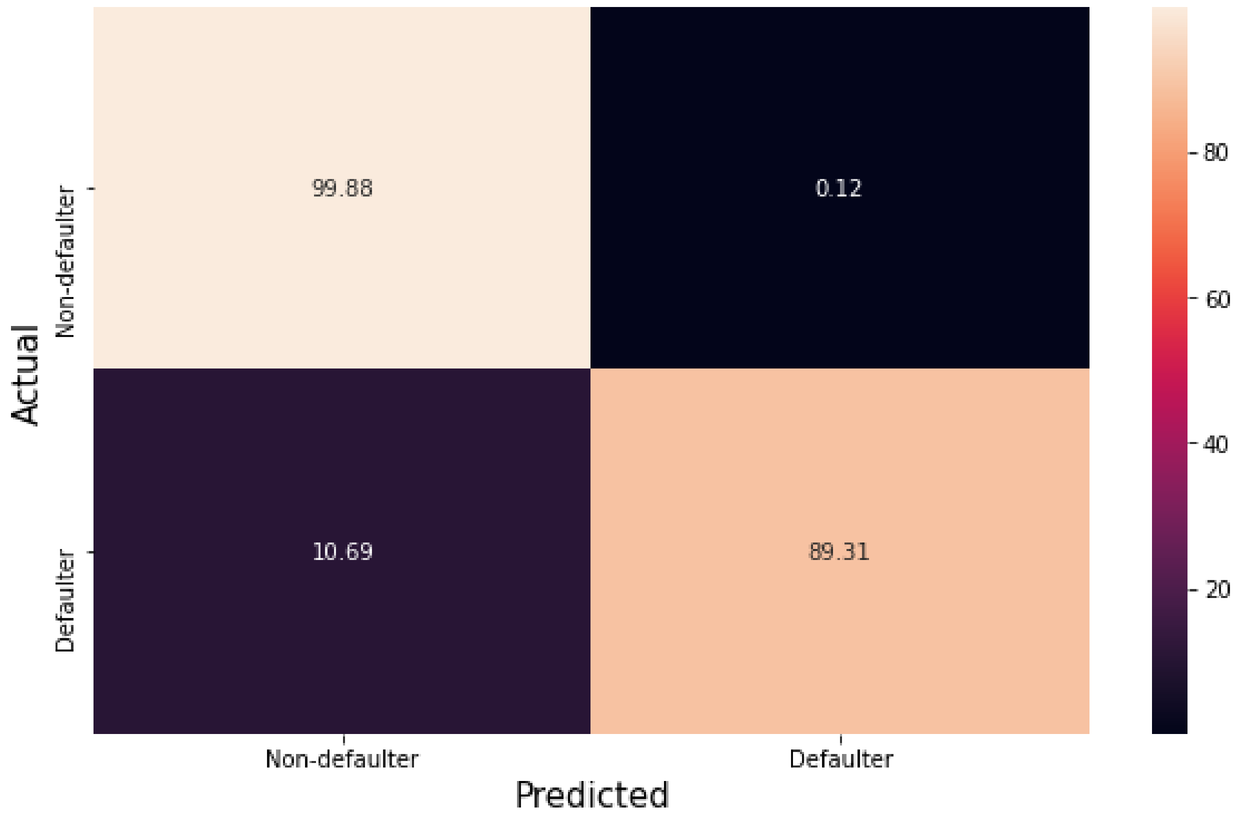
Accuracy

The hyperparameter optimized TensorFlow model yielded a small improvement in test accuracy (94.49%).

AUC Score

Small improvement in two class separability (AUC of 0.9756).

Confusion Matrix



A little lower false negative (FN) (10.69%) although it is still higher than false positive (FP). That means the classifier will allow more loan defaulter to get loan relative to blocking a non-defaulter to get loans

As a loan provider organization, it is desirable to minimize FN as it would like to allow less actual defaulter people to take loans. Reducing FN will cost the organization by increased amount of FP. We need to find balance between FN and FP numbers.

Random Forest

Random Forest is a popular algorithm which build trees and combines the results learned from each tree into one at the end of the training. This algorithm is suitable for noisy and multiclass data. It also has advantage of less over fitting.

Most important hyper-parameters for Random Forest model are number of total number of trees (n_estimators) and number of features considered for splitting at each node (max_features).

Hyper-parameter optimization for Random Forest has been discussed here: [https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76]

Data Preparation

Data was tweaked for standardized vs non-standardized version for simple random forest classifier. Non standardized version yielded better AUC score, so we will be using non-standard data for this section.

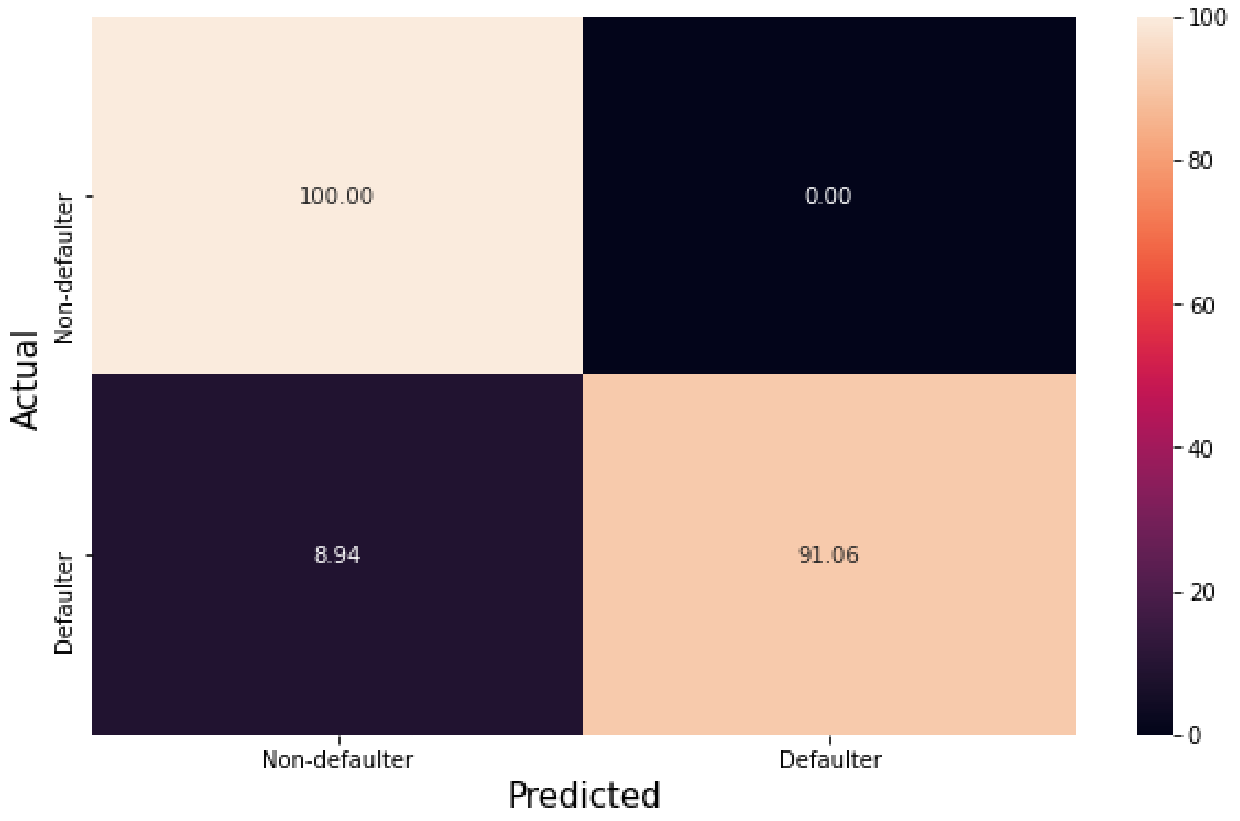
Hyperparameters

Hyperparameters were optimized and following parameters were used for training Random Forest classifier:

HYPERPARAMETRS	NUMBERS/PARAMETERS
N_ESTIMATORS	184
MIN_SAMPLES_SPLIT	2
MIN_SAMPLES_LEAF	1
MAX_FEATURES	sqrt
MAX_DEPTH	8
BOOTSTRAP	true

Metrics

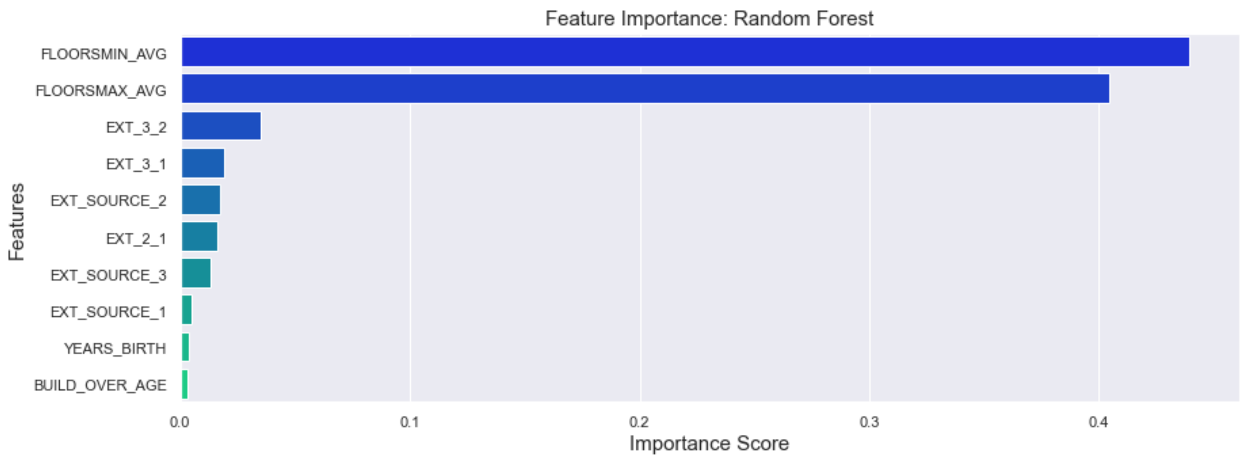
With the hyperparameters a test accuracy of 95.57% was obtained with an AUC-ROC value of 0.9718 which is a slight decrease in AUC score compared to TensorFLow model.



Improvements have been seen for the classification of defaulter vs non-defaulter class (100% for non-defaulter and 91% for deafulter). The number of falsely classifying a defaulter as non-defaulter (9%) and classifying a non-defaulter as defaulter is 0% which is a 1% improvement from TensorFlow model.

Feature Importance

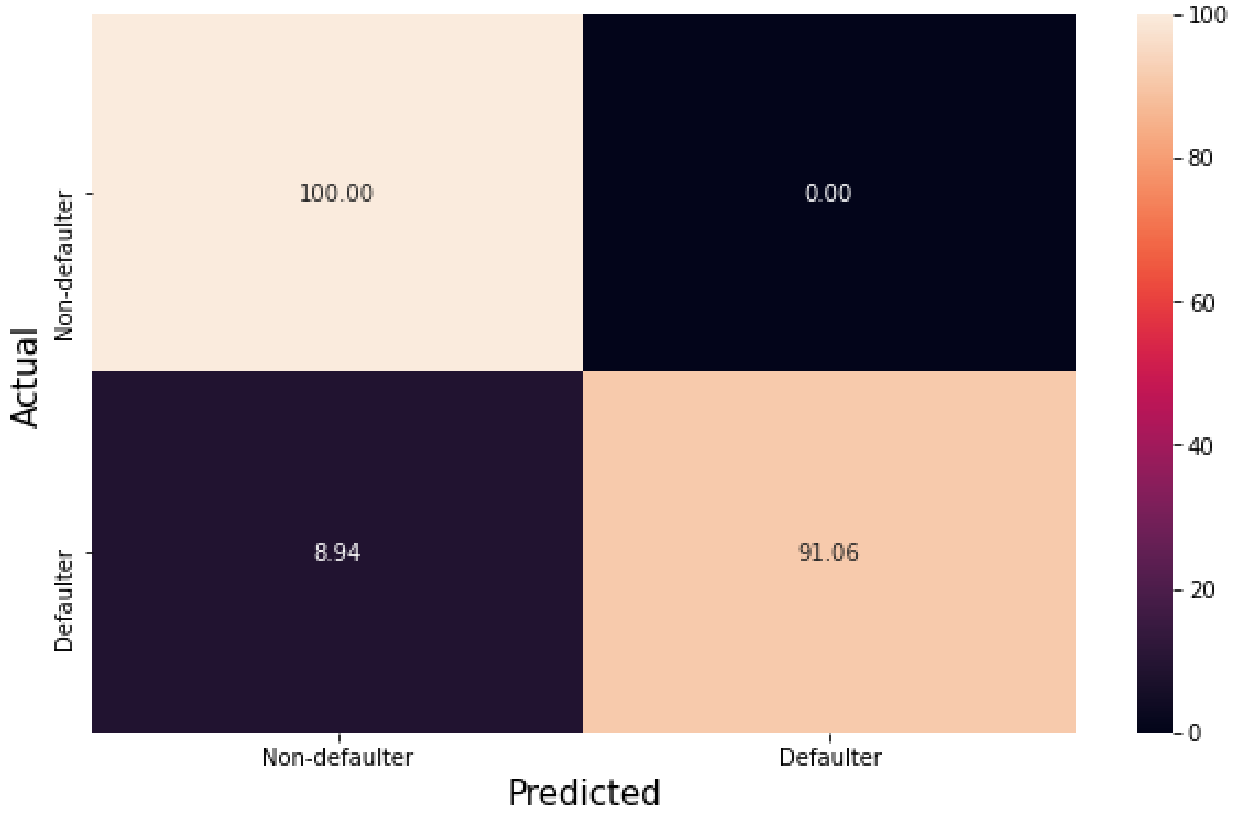
Features important for determining target class can be known from the 'feature importance' option.



Most important features found out by Random Forest model is surprisingly 'Floor area' of the house. Second most important feature is the product of the unnamed features EXT 2 and 3. This was seen in the EDA step, where we found high correlation with target variable.

Model Performance with Top Five Features

From the 'feature importance' top five features were extracted and trained to see classification performance.



The model with top 5 features yielded slightly better AUC score of 0.9772 but the FP and FN scores remained exactly same. For the subsequent models we will still be suing full set of features as other algorithms can alter the top features.

GBM

Gradient Boosting Method (GBM) builds trees at a time. It learns from the weakness from previous tree and builds better trees along the way. Good for unbalanced data as the minority class gets weighted as the training goes on.

There are many parameters to tune to get a optimized model. Many data scientist agree that number of trees, depth and learning are the three most important parameters. We will start with these three and then move to find other tree specific parameters and subsamples.

[https://www.datacareer.de/blog/parameter-tuning-in-gradient-boosting-gbm/]

Data Preparation

Non standardized version yielded better AUC score, so we will be using non-standard data for this section.

Hyperparameters Optimization

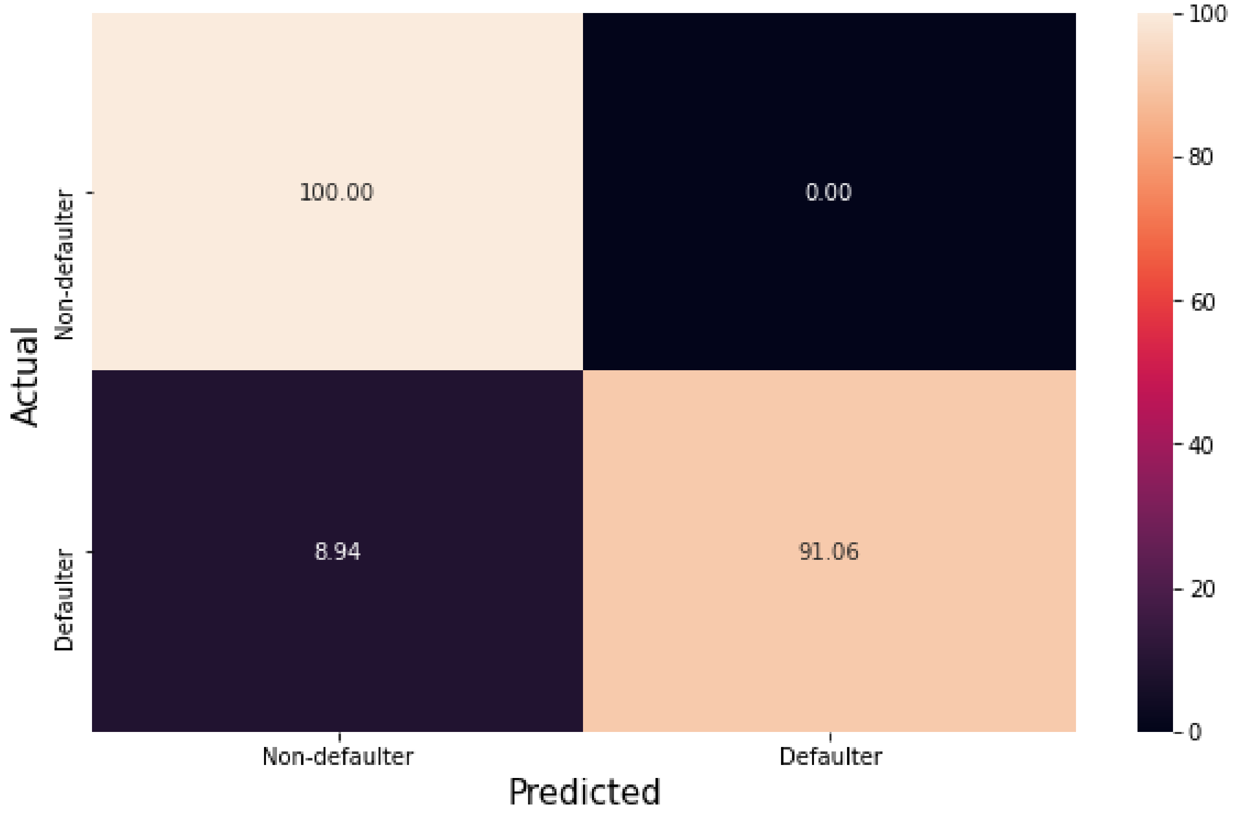
Hyperparameters will be optimized in steps. To save huge computation time hyperparameters were optimized in steps. At first, 'learning rate' and 'n_estimators' was optimized. Based on the optimized learning rate and number of total trees, 'maximum depth' was optimized.

HYPERPARAMETERS	NUMBERS/PARAMETER
LEARNING_RATE	0.01
N_ESTIMATORS	750
MAX_DEPTH	5

Metrics

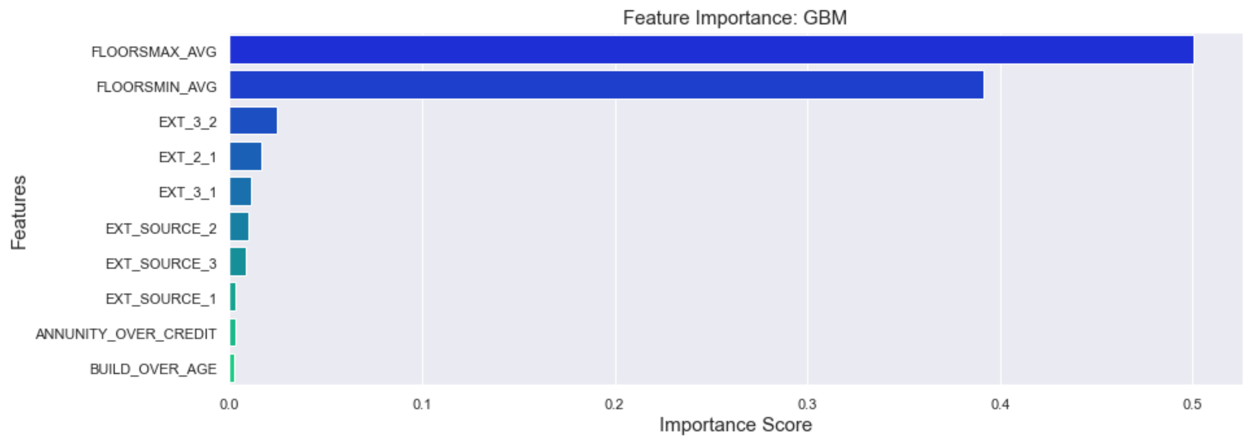
AUC

With the hyperparameters a test accuracy of 95.57% was obtained with an AUC-ROC value of 0.9811 which is a slight increase in AUC score compared to Random Forest model.



The FN/FP ratio remained same at 8.9 /0.0 % which is essentially same as of Random Forest model.

Feature Importance



Like Random Forest, there are many overlapping between top features. Floors area tops the list. In the EDA stage's Feature Creation section, it was indicated that more the spaces a client holds more likely to incline for taking loans to repay mortgage. The next series of top features are multiplicative terms of 'EXT_' variables. In the EDA stage it was found that 'EXT_' variables have high correlations with target variables

Amount of annuity has also affect on loan repay ability, as annuity payment could pressure a client in taking loans. House age over people's life span is also an important factor. Longer the house age, higher the duration people may tend to be paying monthly mortgage.

XGBoost

XGBoost and GBM follow the same gradient boosting technique along the way. Only difference is XGBoost uses a more regularized model formalization to control overfitting.

Like GBM, step by step hyper parameter tuning will be done to cut huge computation time and resources. Instructions followed from here: [https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/]

Data Preparation

Like GBM, one-hot-encoded and non-standard features were used.

Hyperparameter Optimization

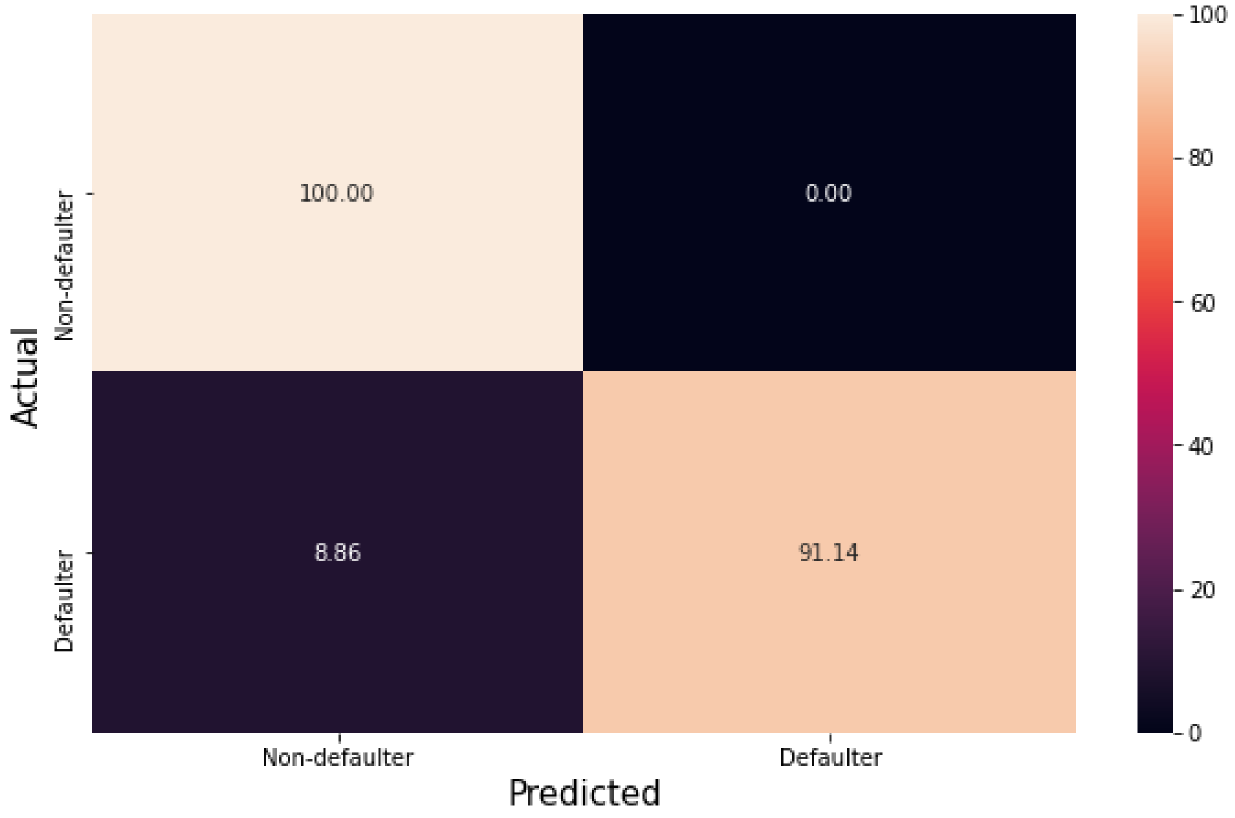
Hyperparameters were optimized in steps. The optimized hyperparameters are:

HYPERPARAMETERS	NUMBERS/PARAMETERS
LEARNING_RATE	0.05
N_ESTIMATORS	150
MAX_DEPTH	4
MIN_CHILD_WEIGHT	1
GAMMA	0.1
NUM_LEAVES	50
COLSAMPLE_BYTREE	0.9

SUBSAMPLE	0.6

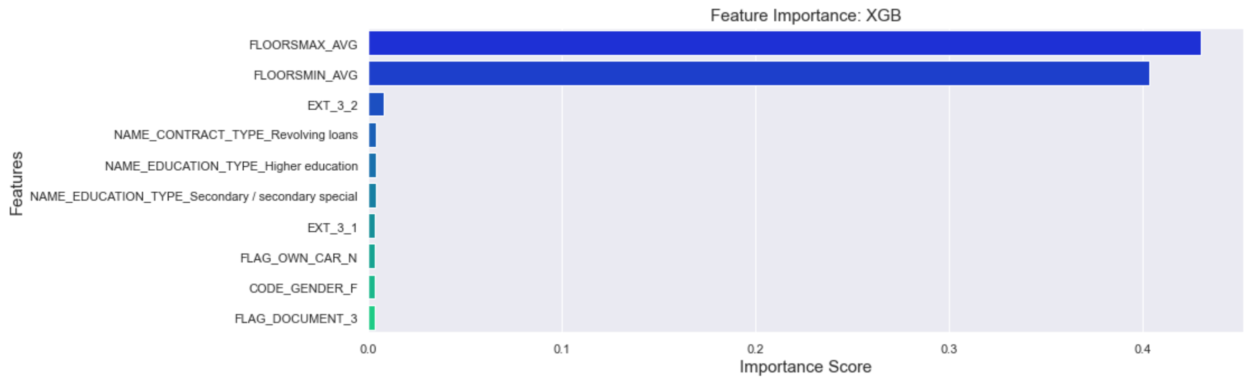
Metrics

The hyperparameter optimized model yielded a test accuracy of 95.61% with an AUC score of 0.9812.



The FN/FP ratio remained same at 8.86/0.0 % although this the lowest we got for the trained model.

Feature Importance



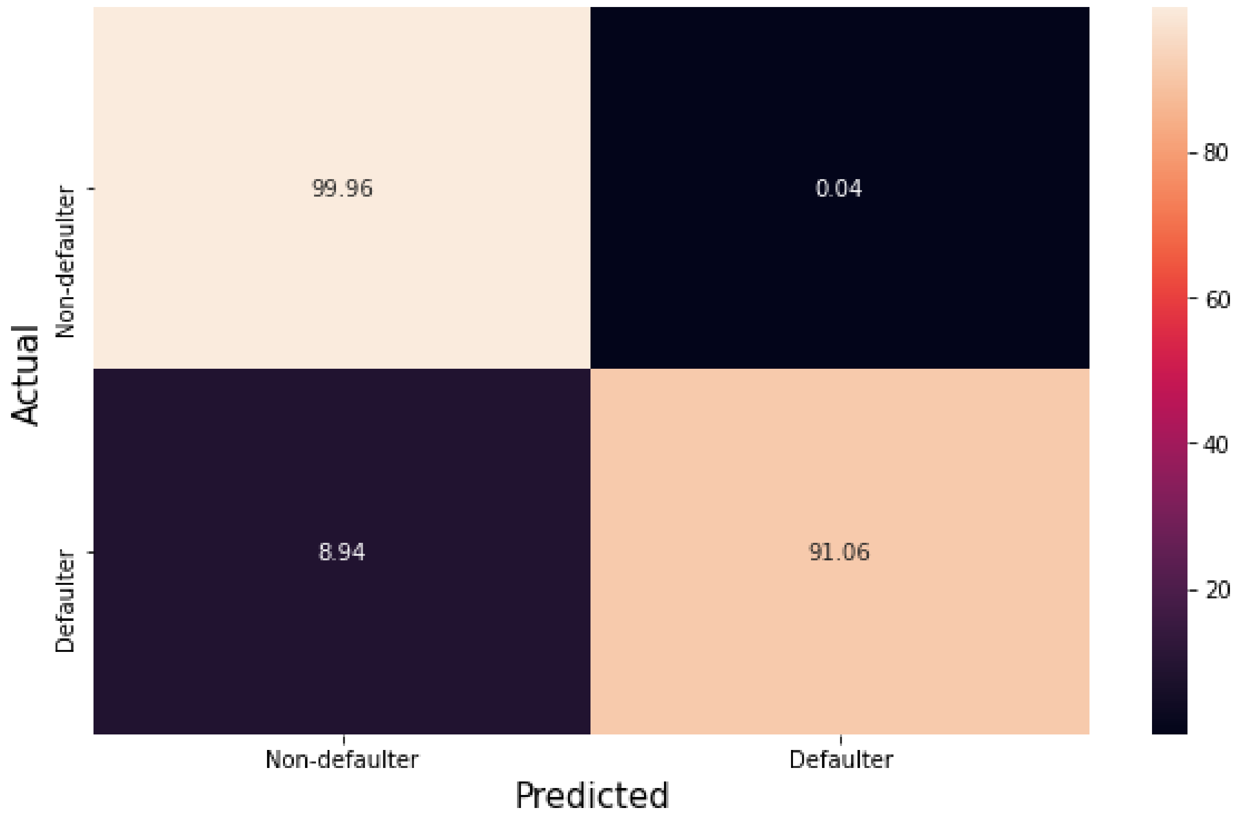
Like Random Forest and GBM models 'Floors area' tops the list. In the EDA stage's Feature Creation section, it was indicated that more the spaces a client holds more likely to incline for taking loans to repay mortgage. Multiplicative terms of 'EXT_' variables were top features. In case of revolving loan type, XGB model is suggesting giving importance. In terms of education level, higher education and secondary education holders need to be put extra attention. Additionally, clients who do not own a car and female in gender also suggested by XGB model to pay extra attentions.

Ensemble Models

We took the average probabilities of TensorFlow, Random Forest, GBM and XGB models to calculate AUC score

Metrics

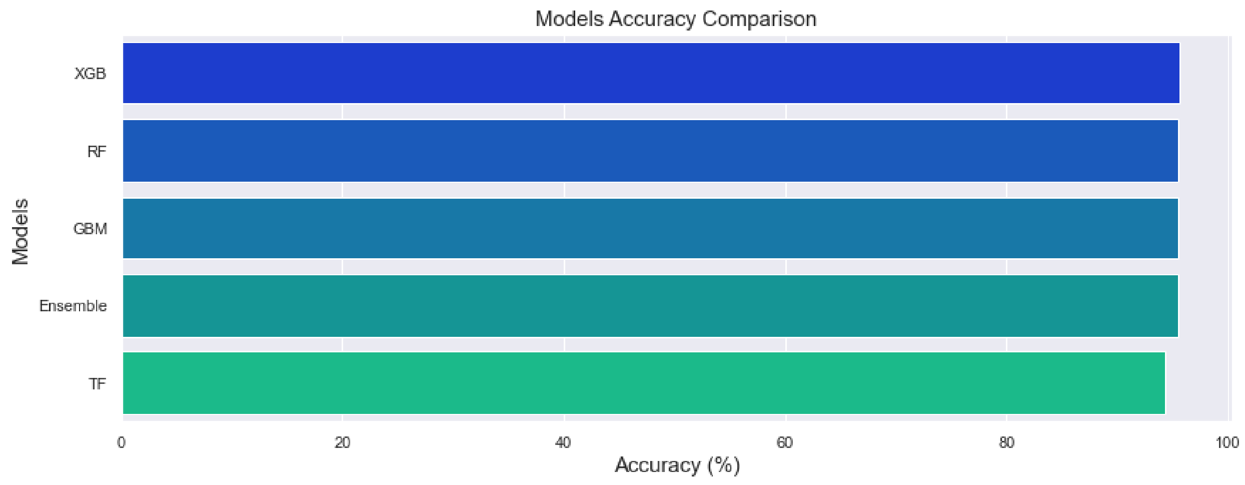
Test accuracy was found to be 95.55% with a AUC score of 0.9809.



The FN/FP and TP/TN ratio remained similar to Random Forest, GBM and XGB models.

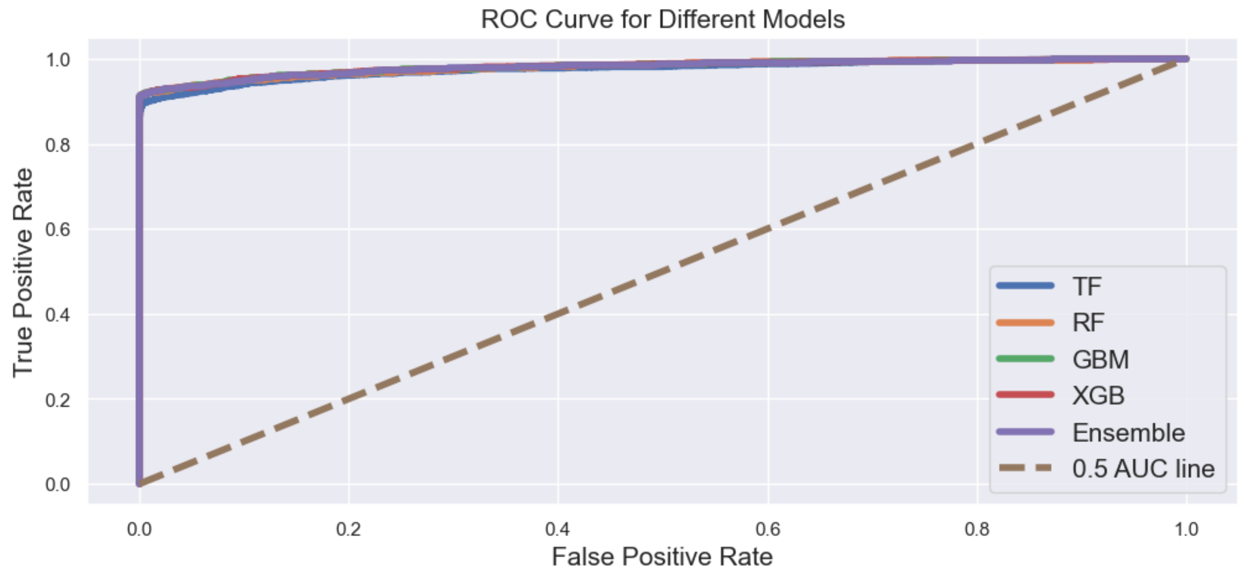
Performance Comparison

Accuracy



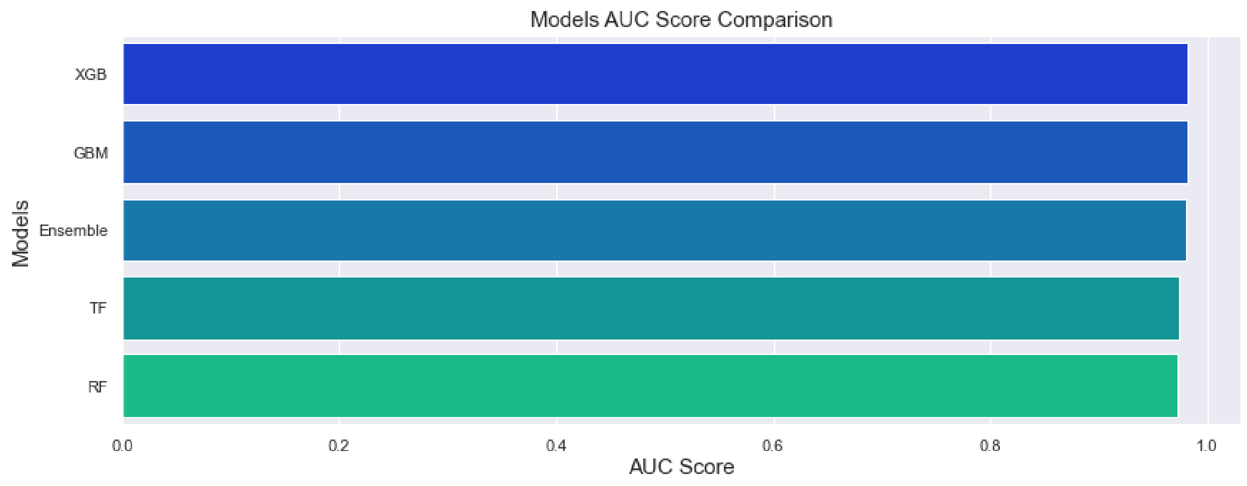
Among all the trained models, maximum accuracy of 95.61% was found for XGB model.

ROC Curve



The ROC scores for all the hyperparameter optimized models are in the well above of the 0.5 AUC baseline.

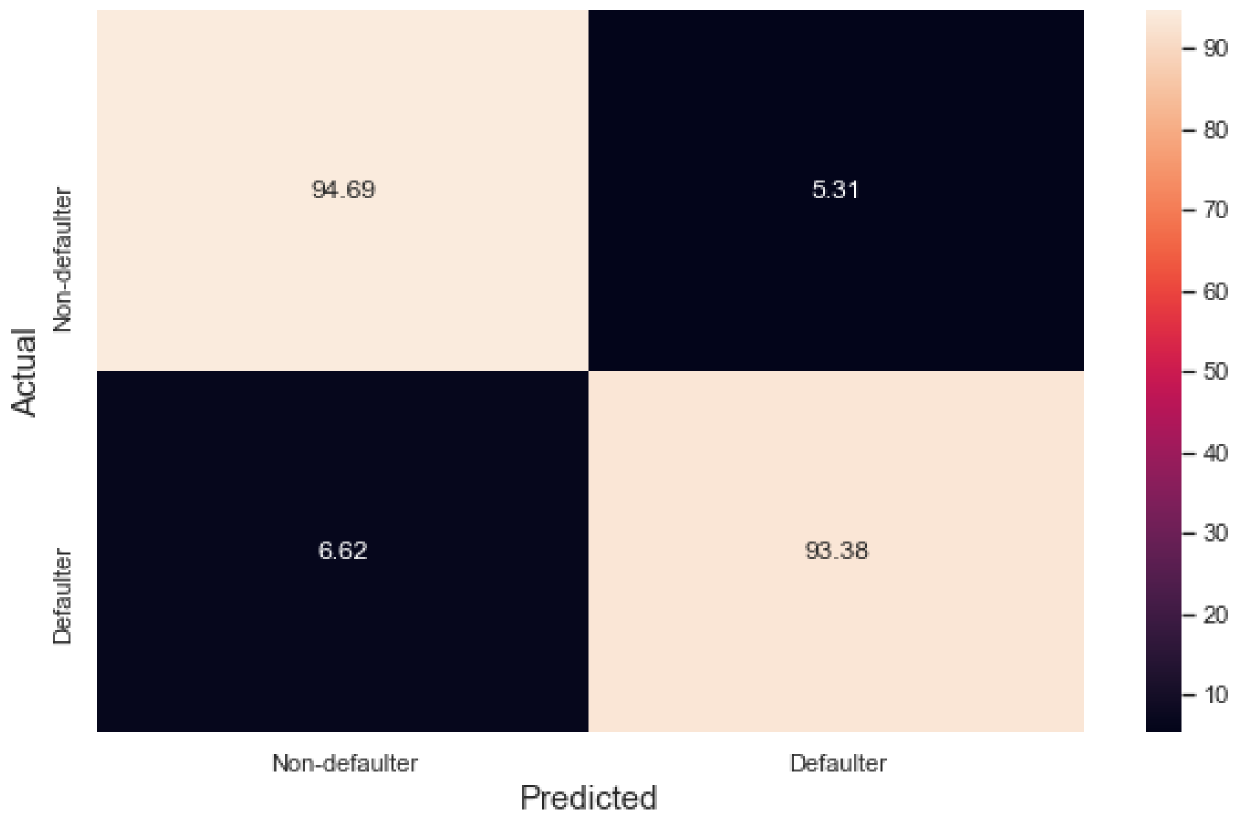
AUC Score



Maximum AUC score of 0.9812 was found for XGB model. The AUC scores for all the hyperparameter optimized models are in the well above of excellent range (>0.9).

Minimizing False Negative/False Positive ratio

One effective way to balance between FN/FP is to tweak the threshold level by which a model predicts binary classes. Here, we ran ranges of threshold values and checked if the confusion matrix improved. The best performed AUC scorer is XGB model. We ran thresholding tests on the XGB model.



A threshold of 0.2 yields best balance between False positive and False negative ratio and True positive and True negative numbers (94/93 %).

A detailed Jupyter notebook version of preprocessing and modelling steps can be found here.

Conclusion

Home Credit Group's loan defaulter prediction was modelled with thousands of existing data. Four algorithms were tried:

- TensorFlow 2.0 (TF)
- Random Forest (RF)
- Gradient Boosting method (GBM)
- Extreme Gradient Boosting (XGBoost)

Dataset was made 'balanced' by under-sampling and 'shuffled' for making training ready. Training, validation, and testing set were split into 80-10-10 ratio. Mainly, Area under curve (AUC) metric was used to measure the performance of binary classification in addition to Accuracy and Receiver Operating Characteristics (ROC) curve for better visualization.

TensorFLow:

Different feature combinations were tried (category vs one-hot-encode, standard vs non-standard and manually created features). The combinations with 'one-hot-encoded' and 'standard' features yielded best performances. A base model was trained and then 'hyperparameters' were optimized

Base Model:

Made sure the model did not overfit by looking into validation (93.2%) and test accuracy (93.8%). An AUC score of 0.9718 was achieved with FN and FP ratio of 11 and 1 %. Although the AUC score is very good, FN is higher compared to FP. This means, the algorithm will allow more real loan defaulters to get loans than blocking real non-defaulters from getting loans.

Hyperparameter Optimized Model:

AUC score of 0.9732 was achieved with accuracy of 94.43 %. This model yielded a slight decrease in FN/FP ratio of 10.69/0.12 %.

Random Forest:

Model was hyperparameter optimized and got the following results:

Hyperparameter Optimized Model:

A slight increase in accuracy score was found: 95.57 %. An AUC score of 0.9718 was achieved. FN/FP ratio is 8.9/0.0 %.

Feature Importance:

Top important features suggested by the model are: 'Floor area', 'EXT_3_2', 'EXT_3_1', 'AGE' and 'BUILD_OVER_AGE'.

GBM

Model was optimized in steps and got the following results:

Hyperparameter Optimized Model:

Same 'accuracy' score of 95.57 % was found as of Random Forest. A slight increase of AUC score was seen compared to Random Forest (0.9811). FN/FP ratio is same as of Random Forest model.

Feature Importance:

Top important features suggested by the model are: 'Floor area', 'EXT_3_2', 'EXT_3_1', 'ANNUITY_OVER_CREDIT', 'BUILD_OVER_AGE'. Noticeably most features overlap with Random Forest

XGBoost

Model was optimized in steps and got the following results:

Hyperparameter Optimized Model:

We got a slight increase in 'accuracy' of 95.61% with associated AUC score of 0.9812. Slight improvement in FN/FP ratio was found 8.8/0.0 % compared to GBM model.

Feature Importance:

Top important features influencing the loan defaulter decision suggested by the model are: 'Floor area', 'EXT_3_2', 'EXT_3_1','Revolving loan' in addition to 'non car owners' and 'female population'

Ensemble

Here, probabilities of the output classes from the above four models were averaged and prediction calculated. We got an 'accuracy' score of 95.55 % with AUC score of 0.9809. The FN/FP ratio found to be 8.9/0.04 %.

Which Model has Got Best Metrics?

XGBoost model has the highest AUC score of 0.9812 with 'accuracy' of 95.61 %. This means the model will be successful in rightly separating the two classes with 98.12 % probability which is an excellent score.

Can We Get a Balanced Ratio between False Negative (FN) and False Positive (FP)?

The best AUC score was found to be 0.9812 for XGBoost model. We tuned the 'threshold' of the classifier and observed with 0.2 threshold value, FN/FP ratio came to at 6.6/5.3 % and TN/TP ratio came to at 94/93 %.

This is an excellent result, as the classifier could reduce the number of real loan defaulters getting loan access with the cost of increased number of non-defaulters denying loan access. This way Home Credit Group can make a fine balance between these two opposing forces which is the best interest for their objective.

Future Directions

Rooms for future exploration were identified as:

- Over sampling and synthetic data creation methodologies can be applied to balance the data and see if they improve performance
- Automatic feature engineering can be tried with additional data provided in the Kaggle competition
- Keras tuner can be used to see if it can improve TF model performance
- An app can be developed which can benefit similar organizations like Home Credit Group to provide automated and more accurate services to their clients