greatlearning

Final report on

Credit Risk Analysis Using Machine Learning Models

Submitted by

Group No. 7 Batch: Online March 21

Mentored by Mr Ankush Bansal

Group Members

1. **Shaik Lal Zaheer Hussain**
2. **Himanshu Singh Sankhala**
3. **Abhishek Anand**
4. **Meenakshi**
5. **Tangudu Saikiran**

Great Lakes Institute of Management

GREAT LAKES

INSTITUTE OF MANAGEMENT

Global Mindset - Indian Roots

Acknowledgment

We would take this opportunity to express our sincere thanks and appreciation to our project mentor Mr. Ankush Bansal. His willingness to help students is unparalleled. His guidance and feedback throughout the process of the project was pivotal to the completion and success of this project.

Final Report

Contents

1) Summary of problem statement, data and findings

Overview: Credit risk refers to the threat associated with a borrower's capacity to repay a loan as well as the interest rate charged. The borrower can be a person or business. Banks can maximize profits by giving credit to just those borrowers who are most likely to repay their debts while reducing losses by not providing credit to those who are likely to fail on their loans.

Problem Statement: Building a model to estimate how capable each applicant is of repaying a loan, with the goal of only approving loans for those who are most likely to repay them. We want to be very certain when we categorize someone as a non-defaulter as the cost of making a mistake might be quite costly to organizations.

Approach**:** We have retained following algorithms

- Logistic Regression
- Decision Tree
- Random Forest
- Extreme Gradient Boosting Model

The project also mainly highlights that it's critical to check data quality (by excluding redundant variables during the preparation and cleaning phase) and to deal with an imbalanced training dataset to avoid bias in majority class. We shall also indicate that the features chosen to meet the business objective (Can we make this decision with only a few features to save time?) and the algorithm applied to make the decision (whether the borrower defaults or not) are two important keys in the decision management processing when issuing a loan (in this case, the bank).

2) Overview of final process

Data Description and Descriptive Statistics

The original dataset "application_train.csv" obtained from "Home Credit Group", a data source available at Kaggle had 307511 rows and 122 columns. There are some columns with negative values, among which for some columns having a negative value was illogical, so negative values for such columns were converted to positive. Columns having false negative values were- DAYS_LAST_PHONE_CHANGE, DAYS_BIRTH, DAYS_REGISTRATION, DAYS_ID_PUBLISH and DAYS_EMPLOYED. Casting correct data types was performed. It is observed that there are certain features which contains huge values or values at a different scale. Data contains some columns with high percentage of missing values and outliers.

Overview of methodology

- **Knowing our Data:** -Getting to know our data by studying the descriptive statistics, plotting appropriate graphs to know relationships between variables and distribution of the data, performing statistical tests by creating hypothesis and verifying them.
- **Data cleaning:** -this involved cleaning our data by detecting and treating missing values and outliers, treating target class imbalance, dropping redundant columns.
- **Feature engineering:** -involves logical imputation for some columns, creating new columns based of domain knowledge, reducing and introducing new levels for some categorical features. Transforming data and performing feature encoding for categorical features.
- **Feature Selection**: - Did multicollinearity check using VIF and dropped features with high multicollinearity. %. Selected features using RFE and Variance threshold methods.

- **Modelling**: - Built four models -Logistic regression, Decision tree, Random Forest, and XGBoost. Scaling of the data was done according to the requirement of the algorithm.
- **Model performance and evaluation**: - Various performance metrics (Recall, precision, accuracy, f1-score, Cohen-kappa score, AUC score, ROC curve, classification report, AIC) were used in order to know model performance on test data. Model evaluation was done taking mean of Cross validation scores for all the splits.

Data Dictionary

Feature	Description
SK_ID_CURR	ID of loan in our sample
TARGET	Target variable (1 - client with payment difficulties: he/she had late payment more than X days on at least one of the first Y instalments of the loan in our sample, 0 - all other cases)
NAME_CONTRACT_TYPE	Identification if loan is cash or revolving
CODE_GENDER	Gender of the client
FLAG_OWN_CAR	Flag if the client owns a car
FLAG_OWN_REALTY	Flag if client owns a house or flat
CNT_CHILDREN	Number of children the client has
AMT_INCOME_TOTAL	Income of the client
AMT_CREDIT	Credit amount of the loan
AMT_ANNUITY	Loan annuity
AMT_GOODS_PRICE	For consumer loans it is the price of the goods for which the loan is given
NAME_TYPE_SUITE	Who was accompanying client when he was applying for the loan
NAME_INCOME_TYPE	Clients' income type (businessman, working, maternity leave,)
NAME_EDUCATION_TYPE	Level of highest education the client achieved
NAME_FAMILY_STATUS	Family status of the client
NAME_HOUSING_TYPE	What is the housing situation of the client (renting, living with parents? ...)
REGION_POPULATION_RELATIVE	Normalized population of region where client lives (higher number means the client lives in more populated region)
DAYS_BIRTH	Client's age in days at the time of application
DAYS_EMPLOYED	How many days before the application the person started current employment
DAYS_REGISTRATION	How many days before the application did client change his registration
DAYS_ID_PUBLISH	How many days before the application did client change the identity document with which he applied for the loan
OWN_CAR_AGE	Age of client's car
FLAG_MOBIL	Did client provide mobile phone (1=YES, 0=NO)
FLAG_EMP_PHONE	Did client provide work phone (1=YES, 0=NO)
FLAG_WORK_PHONE	Did client provide home phone (1=YES, 0=NO)
FLAG_CONT_MOBILE	Was mobile phone reachable (1=YES, 0=NO)
FLAG_PHONE	Did client provide home phone (1=YES, 0=NO)
FLAG_EMAIL	Did client provide email (1=YES, 0=NO)

OCCUPATION_TYPE	What kind of occupation does the client have
CNT_FAM_MEMBERS	How many family members does client have
REGION_RATING_CLIENT	Our rating of the region where client lives (1,2,3)
REGION_RATING_CLIENT_W_CITY	Our rating of the region where client lives with taking city into account (1,2,3)
WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply for the loan
HOUR_APPR_PROCESS_START	Approximately at what hour did the client apply for the loan
REG_REGION_NOT_LIVE_REGION	Flag if client's permanent address does not match contact address (1=different, 0=same, at region level)
REG_REGION_NOT_WORK_REGION	Flag if client's permanent address does not match work address (1=different, 0=same, at region level)
LIVE_REGION_NOT_WORK_REGION	Flag if client's contact address does not match work address (1=different, 0=same, at region level)
REG_CITY_NOT_LIVE_CITY	Flag if client's permanent address does not match contact address (1=different, 0=same, at city level)
REG_CITY_NOT_WORK_CITY	Flag if client's permanent address does not match work address (1=different, 0=same, at city level)
LIVE_CITY_NOT_WORK_CITY	Flag if client's contact address does not match work address (1=different, 0=same, at city level)
ORGANIZATION_TYPE	Type of organization where client works
EXT_SOURCE_1	Normalized score from external data source
EXT_SOURCE_2	Normalized score from external data source
EXT_SOURCE_3	Normalized score from external data source

APARTMENTS_AVG, BASEMENTAREA_AVG, YEARS_BEGINEXPLUATATION_AVG, YEARS_BUILD_AVG, COMMONAREA_AVG, ENTRANCES_AVG, FLOORSMAX_AVG, FLOORSMIN_AVG, LANDAREA_AVG, LIVINGAPARTMENTS_AVG, LIVINGAREA_AVG, NONLIVINGAPARTMENTS_AVG, NONLIVINGAREA_AVG, APARTMENTS_MODE, BASEMENTAREA_MODE, YEARS_BEGINEXPLUATATION_MODE , YEARS_BUILD_MODE, COMMONAREA_MODE, ELEVATORS_MODE, ENTRANCES_MODE, FLOORSMAX_MODE, FLOORSMIN_MODE, LANDAREA_MODE, LIVINGAPARTMENTS_MODE, LIVINGAREA_MODE, NONLIVINGAPARTMENTS_MODE, NONLIVINGAREA_MODE, APARTMENTS_MEDI, BASEMENTAREA_MEDI, YEARS_BEGINEXPLUATATION_MEDI, YEARS_BUILD_MEDI, COMMONAREA_MEDI, ELEVATORS_MEDI, ENTRANCES_MEDI, FLOORSMAX_MEDI, FLOORSMIN_MEDI, LANDAREA_MEDI, LIVINGAPARTMENTS_MEDI, LIVINGAREA_MEDI, NONLIVINGAPARTMENTS_MEDI, NONLIVINGAREA_MEDI, FONDKAPREMONT_MODE, HOUSETYPE_MODE, TOTALAREA_MODE, WALLSMATERIAL_MODE, EMERGENCYSTATE_MODE	Normalized information about building where the client lives, what is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floors etc…

OBS_30_CNT_SOCIAL_CIRCLE	How many observations of client's social surroundings with observable 30 DPD (days past due) default
DEF_30_CNT_SOCIAL_CIRCLE	How many observations of client's social surroundings defaulted on 30 DPD (days past due)
OBS_60_CNT_SOCIAL_CIRCLE	How many observations of client's social surroundings with observable 60 DPD (days past due) default
DEF_60_CNT_SOCIAL_CIRCLE	How many observations of client's social surroundings defaulted on 60 (days past due) DPD
DAYS_LAST_PHONE_CHANGE	How many days before application did client change phone
FLAG_DOCUMENT_2	Did client provide document 2
FLAG_DOCUMENT_3	Did client provide document 3
FLAG_DOCUMENT_4	Did client provide document 4
FLAG_DOCUMENT_5	Did client provide document 5
FLAG_DOCUMENT_6	Did client provide document 6
FLAG_DOCUMENT_7	Did client provide document 7
FLAG_DOCUMENT_8	Did client provide document 8
FLAG_DOCUMENT_9	Did client provide document 9
FLAG_DOCUMENT_10	Did client provide document 10
FLAG_DOCUMENT_11	Did client provide document 11
FLAG_DOCUMENT_12	Did client provide document 12
FLAG_DOCUMENT_13	Did client provide document 13
FLAG_DOCUMENT_14	Did client provide document 14
FLAG_DOCUMENT_15	Did client provide document 15
FLAG_DOCUMENT_16	Did client provide document 16
FLAG_DOCUMENT_17	Did client provide document 17
FLAG_DOCUMENT_18	Did client provide document 18
FLAG_DOCUMENT_19	Did client provide document 19
FLAG_DOCUMENT_20	Did client provide document 20
FLAG_DOCUMENT_21	Did client provide document 21
AMT_REQ_CREDIT_BUREAU_HOUR	Number of enquiries to Credit Bureau about the client one hour before application
AMT_REQ_CREDIT_BUREAU_DAY	Number of enquiries to Credit Bureau about the client one day before application (excluding one hour before application)
AMT_REQ_CREDIT_BUREAU_WEEK	Number of enquiries to Credit Bureau about the client one week before application (excluding one day before application)
AMT_REQ_CREDIT_BUREAU_MON	Number of enquiries to Credit Bureau about the client one month before application (excluding one week before application)
AMT_REQ_CREDIT_BUREAU_QRT	Number of enquiries to Credit Bureau about the client 3 month before application (excluding one month before application)
AMT_REQ_CREDIT_BUREAU_YEAR	Number of enquiries to Credit Bureau about the client one day year (excluding last 3 months before application)

3) Step-by-step walk through of the solution

We have employed Cross Industry Standard Process for Data Mining (CRISP-DM), a standard process used for data mining. CRISP-DM breaks data mining into six phases:

- Business Understanding
- Data Understanding
- Data Preparation
- Modelling
- Evaluation
- Deployment

In order to solve this problem, we follow the following steps for this particular dataset:

- Firstly, the project begins with Exploratory Data Analysis (EDA) and Hypothesis testing using various statistical techniques.
- Clean the dataset using the user-defined function **cleanse_dataset(),** which subsequently drops the features having high percentage of null values, based on our understanding and observation that imputing them might introduce bias in the dataset
- Moving on to rectifying the features and binning their classes by calling **rectify_features()**
- Imputation of the null values with the function call **treat_null_values()** for median imputation if the feature is skewed, mode imputation on categorical variables, and constant (zero) imputation if a pattern of standard missing value percentage is observed among a set of variables.
- Transformation using the function **transform_data()** to reduce skewness and make the distribution normal to the best possible extent
- Invoke **treat_outliers()** to apply flooring and capping of the outliers
- The function call **encode_variables()** will perform label encoding, ordinal encoding, and dummy encoding on appropriate features
- We apply Synthetic Minority Oversampling Technique by calling **apply_SMOTE()** to balance the imbalanced target variable.
- **scale_data()** to apply Standardization (Standard Scaling) so that the features with higher value range do not start dominating.
- Select top 30 features that are important for each model
- Train models from Logistic Regression to XGBModel using the selected features
- Finally, measure the accuracy of the models and compare their results based on roc-auc score.

Exploratory Data Analysis

Detecting and treating missing values

Here is the list of all the columns with any missing values and the corresponding percentage of missing values.

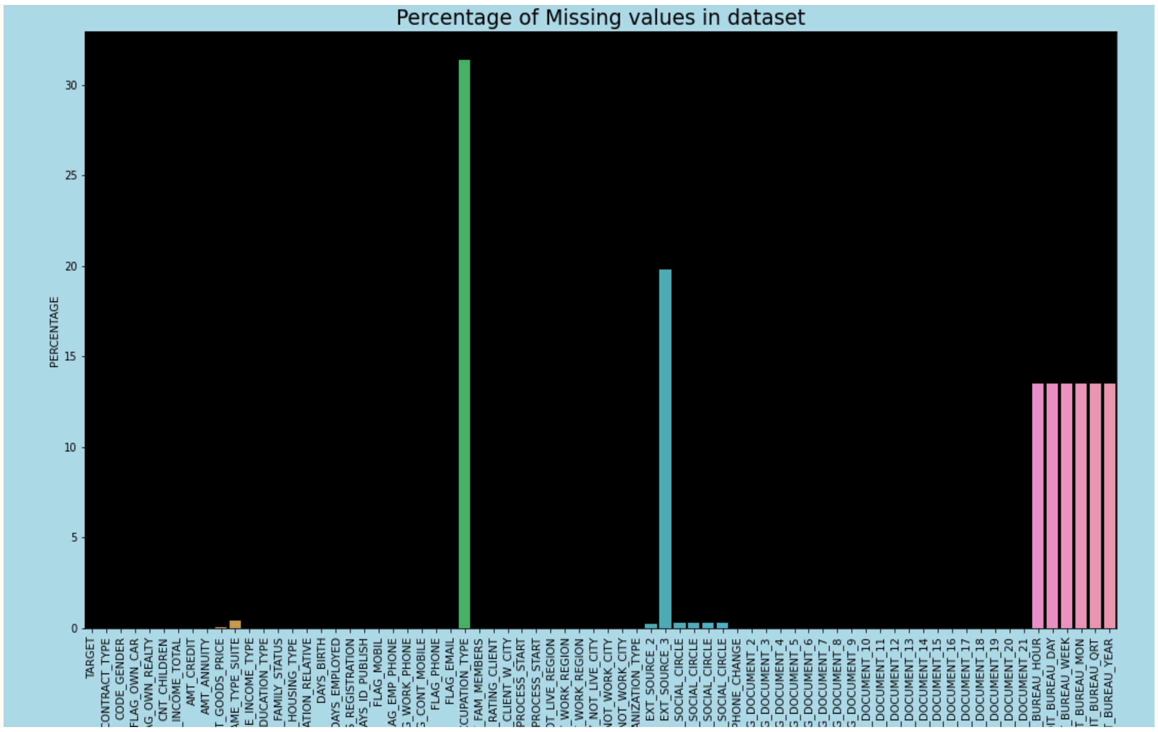
```
COMMONAREA_MEDI                  69.872297
COMMONAREA_AVG                   69.872297
COMMONAREA_MODE                  69.872297
NONLIVINGAPARTMENTS_MODE         69.432963
NONLIVINGAPARTMENTS_AVG          69.432963
NONLIVINGAPARTMENTS_MEDI         69.432963
FONDKAPREMONT_MODE               68.386172
LIVINGAPARTMENTS_MODE            68.354953
LIVINGAPARTMENTS_AVG             68.354953
```

```
LIVINGAPARTMENTS_MEDI            68.354953
FLOORSMIN_AVG                    67.848630
FLOORSMIN_MODE                   67.848630
FLOORSMIN_MEDI                   67.848630
YEARS_BUILD_MEDI                 66.497784
YEARS_BUILD_MODE                 66.497784
YEARS_BUILD_AVG                  66.497784
OWN_CAR_AGE                      65.990810
LANDAREA_MEDI                    59.376738
LANDAREA_MODE                    59.376738
LANDAREA_AVG                     59.376738
BASEMENTAREA_MEDI                58.515956
BASEMENTAREA_AVG                 58.515956
BASEMENTAREA_MODE                58.515956
EXT_SOURCE_1                     56.381073
NONLIVINGAREA_MODE               55.179164
NONLIVINGAREA_AVG                55.179164
NONLIVINGAREA_MEDI               55.179164
ELEVATORS_MEDI                   53.295980
ELEVATORS_AVG                    53.295980
ELEVATORS_MODE                   53.295980
WALLSMATERIAL_MODE               50.840783
APARTMENTS_MEDI                  50.749729
PARTMENTS_AVG                    50.749729
APARTMENTS_MODE                  50.749729
ENTRANCES_MEDI                   50.348768
ENTRANCES_AVG                    50.348768
ENTRANCES_MODE                   50.348768
LIVINGAREA_AVG                   50.193326
LIVINGAREA_MODE                  50.193326
LIVINGAREA_MEDI                  50.193326
HOUSETYPE_MODE                   50.176091
FLOORSMAX_MODE                   49.760822
FLOORSMAX_MEDI                   49.760822
FLOORSMAX_AVG                    49.760822
YEARS_BEGINEXPLUATATION_MODE     48.781019
YEARS_BEGINEXPLUATATION_MEDI     48.781019
YEARS_BEGINEXPLUATATION_AVG      48.781019
TOTALAREA_MODE                   48.268517
EMERGENCYSTATE_MODE              47.398304
OCCUPATION_TYPE                  31.345545
EXT_SOURCE_3                     19.825307
AMT_REQ_CREDIT_BUREAU_HOUR       13.501631
AMT_REQ_CREDIT_BUREAU_DAY        13.501631
AMT_REQ_CREDIT_BUREAU_WEEK       13.501631
AMT_REQ_CREDIT_BUREAU_MON        13.501631
AMT_REQ_CREDIT_BUREAU_QRT        13.501631
AMT_REQ_CREDIT_BUREAU_YEAR       13.501631
NAME_TYPE_SUITE                   0.420148
OBS_30_CNT_SOCIAL_CIRCLE          0.332021
DEF_30_CNT_SOCIAL_CIRCLE          0.332021
OBS_60_CNT_SOCIAL_CIRCLE          0.332021
DEF_60_CNT_SOCIAL_CIRCLE          0.332021
```

```
EXT_SOURCE_2                     0.214626
AMT_GOODS_PRICE                  0.090403
AMT_ANNUITY                      0.003902
CNT_FAM_MEMBERS                  0.000650
DAYS_LAST_PHONE_CHANGE           0.000325
```

For treating these missing values, we used a 40% cut-off margin to eliminate all the columns having more than 40% data as null. The shape of data frame changes from (307511,122) to (307511,72). Now for the remaining columns we used standard procedure for numerical columns by replacing null values with mean in case data is normal and by median in case data is not normal, and for categorical data we replaced the data with the most frequent category in each column. For OCCUPATION_TYPE we checked for the null values and found that a large percentage of null values were for the rows which has NAME_INCOME_TYPE as pensioner, so we replaced all these null values with Retired thus imputing a new category into the column.

The resultant dataset after dropping the columns with more than 40% as threshold can be observed in the below figure.



Statistical tests

- We ran the test to check the normality for some columns- AMT_CREDIT, AMT_INCOME_TOTAL, AMT_ANNUITY,'CNT_CHILDREN, REGION_RATING_CLIENT.

  Shapiro-Wilk test: -

  H0: 'data is Normally distributed'

  H1: 'data is not Normally distributed'

  We got following results for the tests: -

  stat=0.685, p1=0.000, p2=0.000, p3=0.000, p4=0.000, p5=0.000
  We are rejecting null hypothesis, hence AMT_CREDIT is not normally distributed.

We are rejecting null hypothesis, hence AMT_INCOME_TOTAL is not normally distributed.
We are rejecting null hypothesis, hence AMT_ANNUITY is not normally distributed.
We are rejecting null hypothesis, hence CNT_CHILDREN is not normally distributed.
We are rejecting null hypothesis, hence REGION_RATING_CLIENT is normally distributed.

D'Agostino's K-square Normality Test:

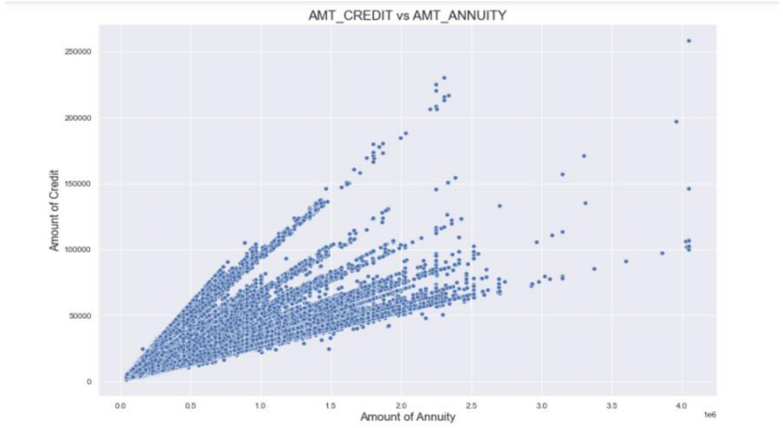
H0: data is Normally distributed

H1: data is not Normally distributed

We got following results for the tests: -
stat=4706.031, p1=0.000, p2=0.000, p3=0.000, p4=0.000, p5=0.000
AMT_CREDIT-Probably not Normally Distributed
AMT_INCOME_TOTAL-Probably not Normally Distributed
AMT_ANNUITY-Probably not Normally Distributed
CNT_CHILDREN-Probably not Normally Distributed
REGION_RATING_CLIENT-Probably not Normally Distributed

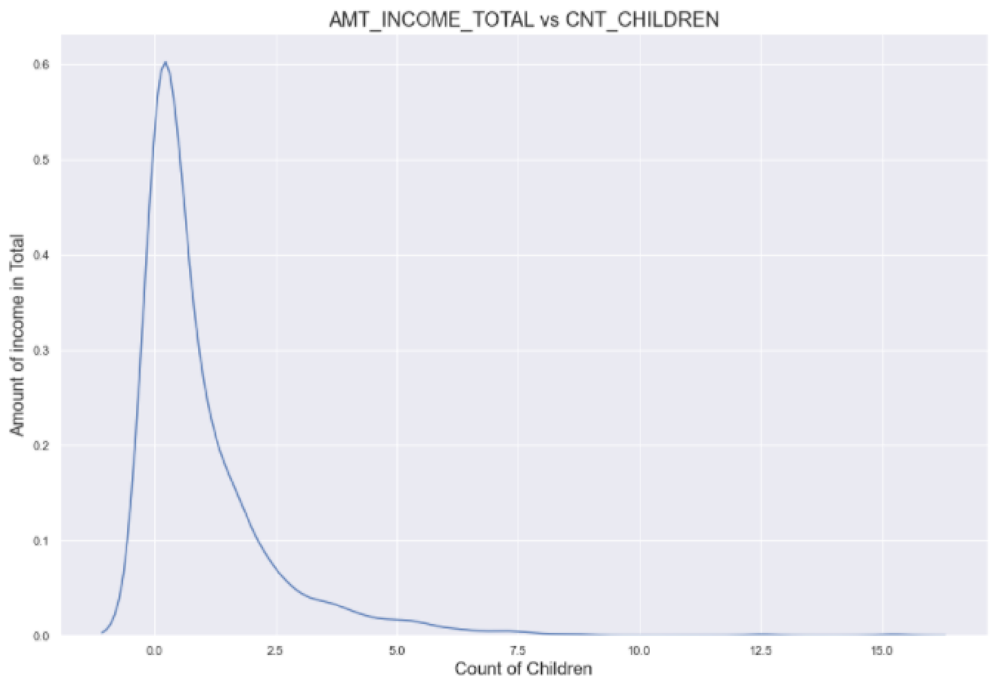
- Correlation Tests
  Pearson's Correlation Coefficient: -

  Defining Null and Alternative Hypotheses
  H0: -correlation coefficient IS NOT significantly different from 0. There is not a significant r elationship between the features
  H1: - correlation coefficient IS significantly different from 0. There is a significant relationsh ip between the variables
  alpha = 0.05
  The test was performed for 2 columns- AMT_CREDIT, AMT_ANNUITY



  p-value: -0.07, which means we accept the null hypothesis.
  We can see that the relationship is not linear between the two columns**.**

- Chi square test
  Chi-Squared Test of independence for columns
  Null and alternate hypothesis
  H0: -there is not a significant relationship between the columns
  H1: -there is a significant relationship between the columns
  Tests performed for columns AMT_INCOME_TOTAL and CNT_CHILDREN
  Outputs: - stat=514939.431, p=0.000
  Probably dependent



Observation:
The p value obtained from chi-square test for independence is significant ($p < 0.05$), and therefore, we conclude that there is a significant association between Columns CNT_CHILDREN and AMT_INCOME_TOTAL

Test for columns- NAME_CONTRACT_TYPE and WEEKDAY_APPR_PROCESS_START

Stats: 67.99048638705025 P-value: 1.0553487110442855e-12
Both the categorical attributes are dependent (Reject H0).

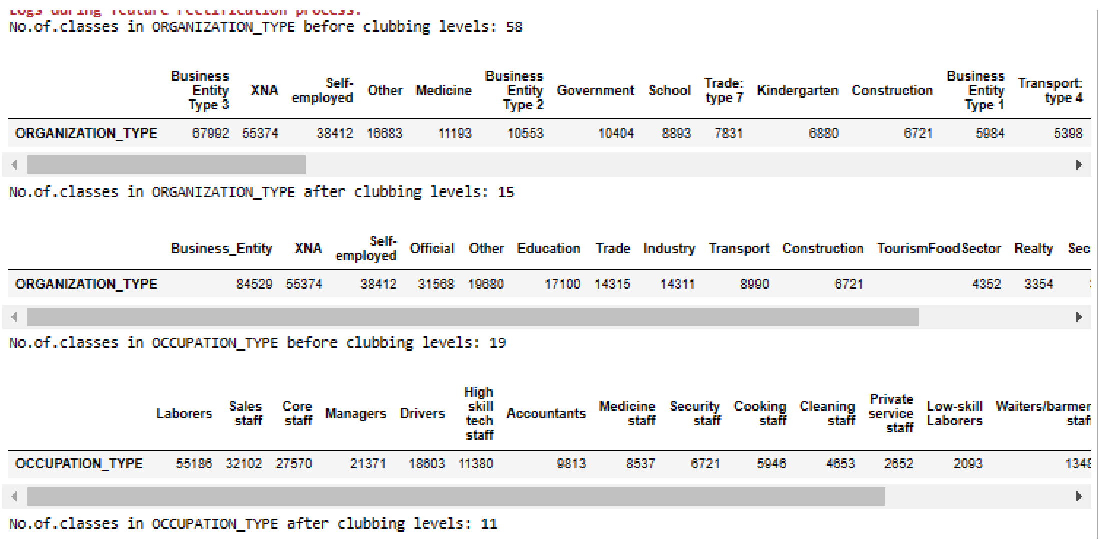
Test for columns-NAME_INCOME_TYPE and OCCUPATION_TYPE

Stats: 98842.68043941665 P-value: 0.0
Both the categorical attributes are dependent (Reject H0).

To train and build the models we will predominantly use the NumPy, Pandas, Scikit-Learn open-source ML packages. For the Boosted tree models, we will use xgboost library. And we will use Variance Threshold, Variance Inflation Factor and Recursive Feature Elimination techniques to help us with feature selection so we create the best possible set of features to train. After that, we will train a few different ML models and measure the AUC (Area Under the Curve), Precision and Recall scores. All the experiments will be conducted in Python and Jupyter Notebook will be used for visualization.

Feature Engineering

- Reduced the levels for some columns which had high number of levels (>20). ORGANIZATION_TYPE and OCCUPATION_TYPE we reduced the levels for both the columns by logically combining levels of same kind or which had some logical relation.



```
Logs during feature rectification process:
No.of.classes in ORGANIZATION_TYPE before clubbing levels: 58
```

	Business Entity Type 3	XNA	Self-employed	Other	Medicine	Business Entity Type 2	Government	School	Trade: type 7	Kindergarten	Construction	Business Entity Type 1	Transport: type 4
ORGANIZATION_TYPE	67992	55374	38412	16683	11193	10553	10404	8893	7831	6880	6721	5984	5398

```
No.of.classes in ORGANIZATION_TYPE after clubbing levels: 15
```

	Business_Entity	XNA	Self-employed	Official	Other	Education	Trade	Industry	Transport	Construction	TourismFoodSector	Realty	Sec
ORGANIZATION_TYPE	84529	55374	38412	31568	19680	17100	14315	14311	8990	6721	4352	3354	

```
No.of.classes in OCCUPATION_TYPE before clubbing levels: 19
```

	Laborers	Sales staff	Core staff	Managers	Drivers	High skill tech staff	Accountants	Medicine staff	Security staff	Cooking staff	Cleaning staff	Private service staff	Low-skill Laborers	Waiters/barmer staf
OCCUPATION_TYPE	55186	32102	27570	21371	18603	11380	9813	8537	6721	5946	4653	2652	2093	134

```
No.of.classes in OCCUPATION_TYPE after clubbing levels: 11
```

- Reducing levels for columns which had some levels which form low % of data for the overall column example less than 2%.
  NAME_INCOME_TYPE

```
print("No.of.classes in NAME_INCOME_TYPE before clubbing levels:",len(df.NAME_INCOME_TYPE.unique()))
display(pd.DataFrame(df.NAME_INCOME_TYPE.value_counts()).T)
df['NAME_INCOME_TYPE'].replace('Businessman','Commercial associate', inplace=True)
df['NAME_INCOME_TYPE'].replace('Maternity leave','Pensioner', inplace=True)
df['NAME_INCOME_TYPE'].replace('Student','State servant', inplace=True)
df['NAME_INCOME_TYPE'].replace('Unemployed','Pensioner', inplace=True)
print("No.of.classes in NAME_INCOME_TYPE after clubbing levels:",len(df.NAME_INCOME_TYPE.unique()))
display(pd.DataFrame(df.NAME_INCOME_TYPE.value_counts()).T)
```

```
No.of.classes in NAME_INCOME_TYPE before clubbing levels: 8
```

	Working	Commercial associate	Pensioner	State servant	Unemployed	Student	Businessman	Maternity leave
NAME_INCOME_TYPE	158774	71617	55362	21703	22	18	10	5

```
No.of.classes in NAME_INCOME_TYPE after clubbing levels: 4
```

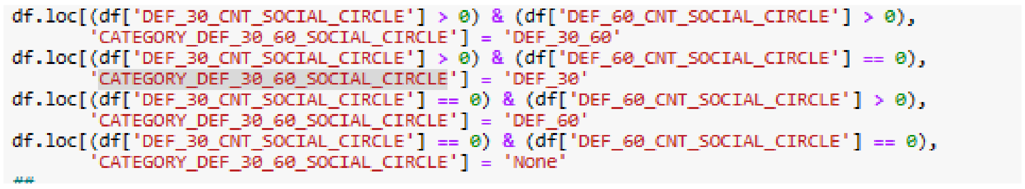
	Working	Commercial associate	Pensioner	State servant
NAME_INCOME_TYPE	158774	71627	55389	21721

- Removing some features based on very low variance, and clubbing features of one type i.e FLAG_DOCUMENT into a single feature representing the document submission score.

```
drop_flags = ['FLAG_DOCUMENT_12','FLAG_DOCUMENT_11','FLAG_DOCUMENT_19','FLAG_DOCUMENT_5','FLAG_DOCUMENT_20',
              'FLAG_DOCUMENT_15','FLAG_DOCUMENT_7','FLAG_DOCUMENT_4','FLAG_DOCUMENT_10','FLAG_DOCUMENT_18',
              'FLAG_DOCUMENT_2','FLAG_DOCUMENT_21','FLAG_DOCUMENT_13','FLAG_DOCUMENT_16','FLAG_DOCUMENT_14',
              'FLAG_DOCUMENT_9','FLAG_DOCUMENT_17']
# drop columns in drop_flags
for feature in drop_flags:
    if feature in df.columns:
        df.drop(feature, axis = 1, inplace = True)
# select features which show some percentage to club them
sum_flags = ['FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_8']
df['SCORE_DOCUMENT_SUBMISSION'] = df[sum_flags].sum(axis=1)
df.drop(sum_flags, axis=1, inplace=True)
print("Correlation between the feature SCORE_DOCUMENT_SUBMISSION and TARGET",df.SCORE_DOCUMENT_SUBMISSION.corr(df.TARGET))
```

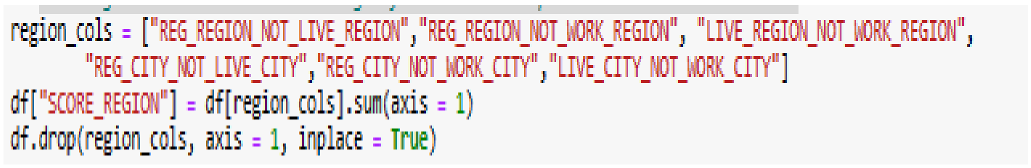
Correlation between the feature SCORE_DOCUMENT_SUBMISSION and TARGET 0.03015619952371004

- Making a new feature named CATEGORY_DEF_30_60_SOCIAL_CIRCLE by clubbing classes into relevant category for the features DEF_30_CNT_SOCIAL_CIRCLE and DEF_60_CNT_SOCIAL_CIRCLE.



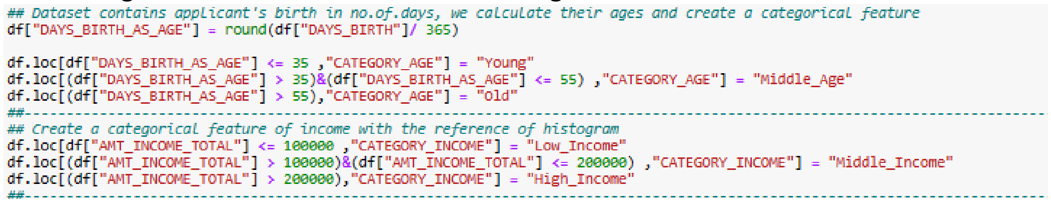
```
df.loc[(df['DEF_30_CNT_SOCIAL_CIRCLE'] > 0) & (df['DEF_60_CNT_SOCIAL_CIRCLE'] > 0),
           'CATEGORY_DEF_30_60_SOCIAL_CIRCLE'] = 'DEF_30_60'
df.loc[(df['DEF_30_CNT_SOCIAL_CIRCLE'] > 0) & (df['DEF_60_CNT_SOCIAL_CIRCLE'] == 0),
           'CATEGORY_DEF_30_60_SOCIAL_CIRCLE'] = 'DEF_30'
df.loc[(df['DEF_30_CNT_SOCIAL_CIRCLE'] == 0) & (df['DEF_60_CNT_SOCIAL_CIRCLE'] > 0),
           'CATEGORY_DEF_30_60_SOCIAL_CIRCLE'] = 'DEF_60'
df.loc[(df['DEF_30_CNT_SOCIAL_CIRCLE'] == 0) & (df['DEF_60_CNT_SOCIAL_CIRCLE'] == 0),
           'CATEGORY_DEF_30_60_SOCIAL_CIRCLE'] = 'None'
##
```

- Club Region based columns into single feature to interpret their residential score



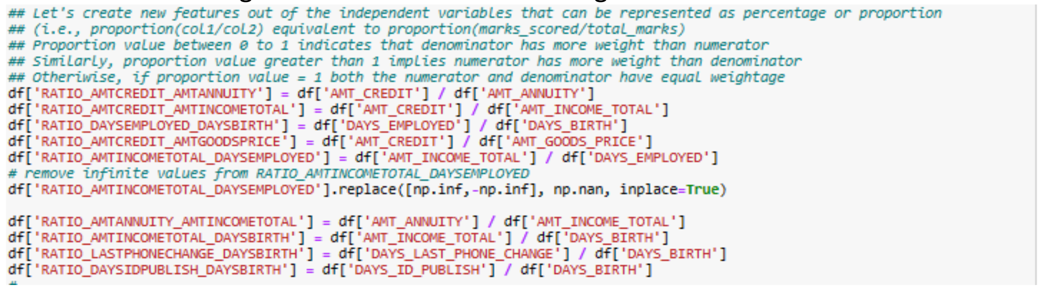
```
region_cols = ["REG_REGION_NOT_LIVE_REGION","REG_REGION_NOT_WORK_REGION", "LIVE_REGION_NOT_WORK_REGION",
        "REG_CITY_NOT_LIVE_CITY","REG_CITY_NOT_WORK_CITY","LIVE_CITY_NOT_WORK_CITY"]
df["SCORE_REGION"] = df[region_cols].sum(axis = 1)
df.drop(region_cols, axis = 1, inplace = True)
```

- Club EXT_SOURCE related columns into single feature as their average score.
- From histogram and boxplot, DAYS_EMPLOYED contains values > 300000 days (i.e., more tha n 800 years), which doesn't hold true in real world. So, we consider them as Nans.
- Gender distribution pie chart shows that negligible no.of applicant's gender information isn't available.
- Bar plot of NAME_FAMILY_STATUS shows 'Unknown' with count around 0. So, we excluded t he unknown rows
- Converting some numerical features into categorical features: -



```
## Dataset contains applicant's birth in no.of.days, we calculate their ages and create a categorical feature
df["DAYS_BIRTH_AS_AGE"] = round(df["DAYS_BIRTH"]/ 365)

df.loc[df["DAYS_BIRTH_AS_AGE"] <= 35 ,"CATEGORY_AGE"] = "Young"
df.loc[(df["DAYS_BIRTH_AS_AGE"] > 35)&(df["DAYS_BIRTH_AS_AGE"] <= 55) ,"CATEGORY_AGE"] = "Middle_Age"
df.loc[(df["DAYS_BIRTH_AS_AGE"] > 55),"CATEGORY_AGE"] = "Old"
##----------------------------------------------------------------------------------------------------
## Create a categorical feature of income with the reference of histogram
df.loc[df["AMT_INCOME_TOTAL"] <= 100000 ,"CATEGORY_INCOME"] = "Low_Income"
df.loc[(df["AMT_INCOME_TOTAL"] > 100000)&(df["AMT_INCOME_TOTAL"] <= 200000) ,"CATEGORY_INCOME"] = "Middle_Income"
df.loc[(df["AMT_INCOME_TOTAL"] > 200000),"CATEGORY_INCOME"] = "High_Income"
##----------------------------------------------------------------------------------------------------
```

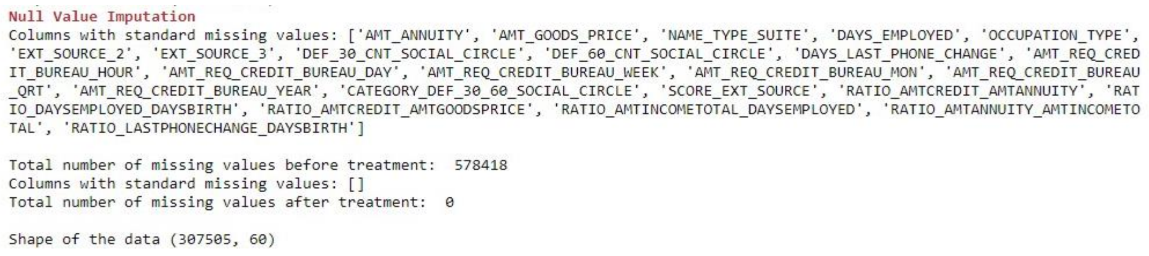
- Logically creating a few new columns based on the relationships within the columns and base d on domain knowledge and business understanding.



```
## Let's create new features out of the independent variables that can be represented as percentage or proportion
## (i.e., proportion(col1/col2) equivalent to proportion(marks_scored/total_marks)
## Proportion value between 0 to 1 indicates that denominator has more weight than numerator
## Similarly, proportion value greater than 1 implies numerator has more weight than denominator
## Otherwise, if proportion value = 1 both the numerator and denominator have equal weightage
df['RATIO_AMTCREDIT_AMTANNUITY'] = df['AMT_CREDIT'] / df['AMT_ANNUITY']
df['RATIO_AMTCREDIT_AMTINCOMETOTAL'] = df['AMT_CREDIT'] / df['AMT_INCOME_TOTAL']
df['RATIO_DAYSEMPLOYED_DAYSBIRTH'] = df['DAYS_EMPLOYED'] / df['DAYS_BIRTH']
df['RATIO_AMTCREDIT_AMTGOODSPRICE'] = df['AMT_CREDIT'] / df['AMT_GOODS_PRICE']
df['RATIO_AMTINCOMETOTAL_DAYSEMPLOYED'] = df['AMT_INCOME_TOTAL'] / df['DAYS_EMPLOYED']
# remove infinite values from RATIO_AMTINCOMETOTAL_DAYSEMPLOYED
df['RATIO_AMTINCOMETOTAL_DAYSEMPLOYED'].replace([np.inf,-np.inf], np.nan, inplace=True)

df['RATIO_AMTANNUITY_AMTINCOMETOTAL'] = df['AMT_ANNUITY'] / df['AMT_INCOME_TOTAL']
df['RATIO_AMTINCOMETOTAL_DAYSBIRTH'] = df['AMT_INCOME_TOTAL'] / df['DAYS_BIRTH']
df['RATIO_LASTPHONECHANGE_DAYSBIRTH'] = df['DAYS_LAST_PHONE_CHANGE'] / df['DAYS_BIRTH']
df['RATIO_DAYSIDPUBLISH_DAYSBIRTH'] = df['DAYS_ID_PUBLISH'] / df['DAYS_BIRTH']
```

Null Value Imputation

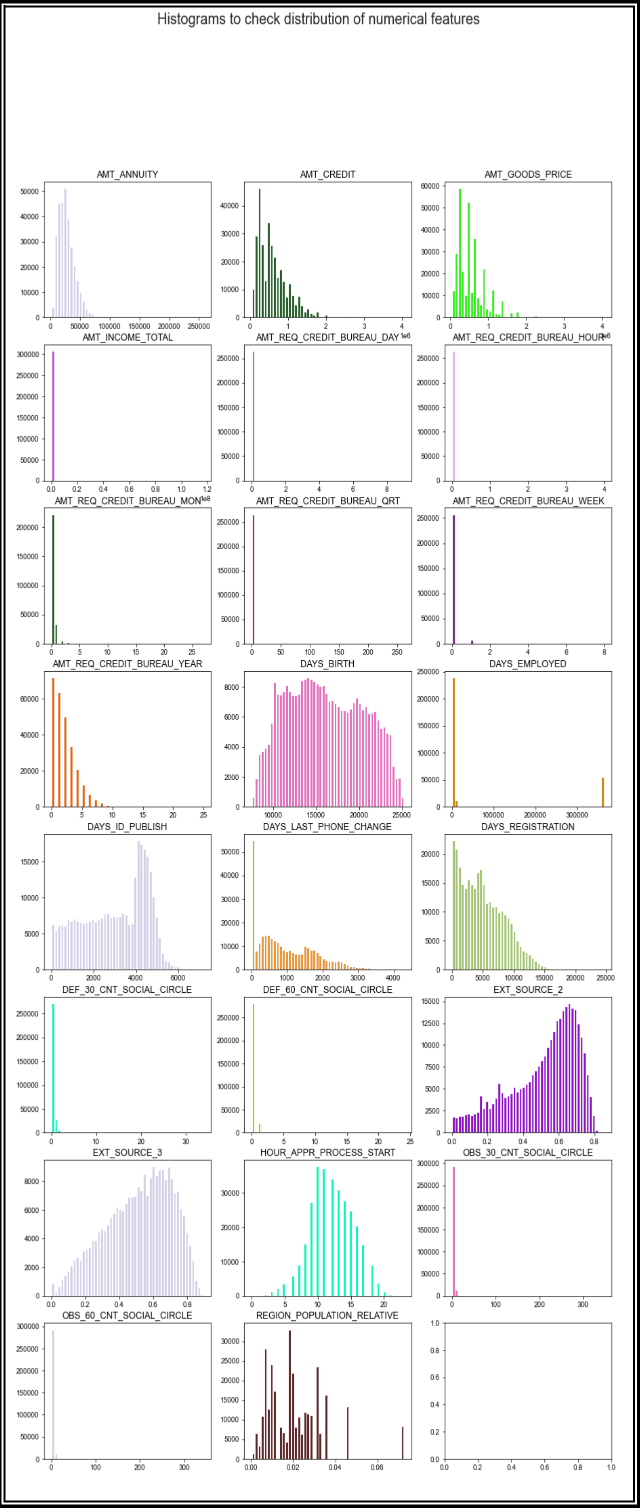
Data can have missing values for a number of reasons such as observations that were not recorded and data corruption. Handling missing data is important as many machine learning algorithms do not support data with missing values. Following are the result after median, mode imputations on skewed numerical features and categorical features respectively. Additionally, we fill 0 into null values wherever we can observe equivalent trend in null percentage among the variables in order to avoid bias due to other imputation techniques.

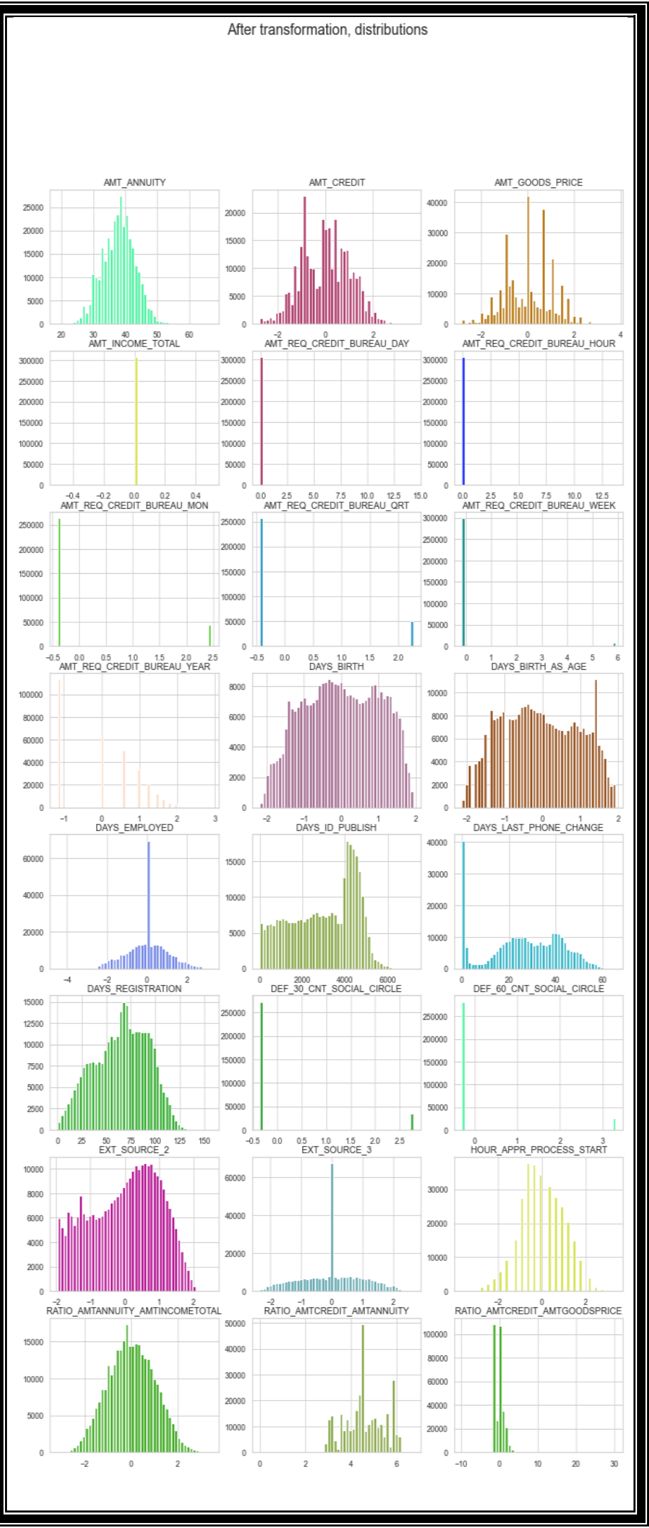


```
Null Value Imputation
Columns with standard missing values: ['AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'DAYS_EMPLOYED', 'OCCUPATION_TYPE',
'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DEF_30_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CRED
IT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU
_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR', 'CATEGORY_DEF_30_60_SOCIAL_CIRCLE', 'SCORE_EXT_SOURCE', 'RATIO_AMTCREDIT_AMTANNUITY', 'RAT
IO_DAYSEMPLOYED_DAYSBIRTH', 'RATIO_AMTCREDIT_AMTGOODSPRICE', 'RATIO_AMTINCOMETOTAL_DAYSEMPLOYED', 'RATIO_AMTANNUITY_AMTINCOMETO
TAL', 'RATIO_LASTPHONECHANGE_DAYSBIRTH']

Total number of missing values before treatment:  578418
Columns with standard missing values: []
Total number of missing values after treatment:  0

Shape of the data (307505, 60)
```

Transformation

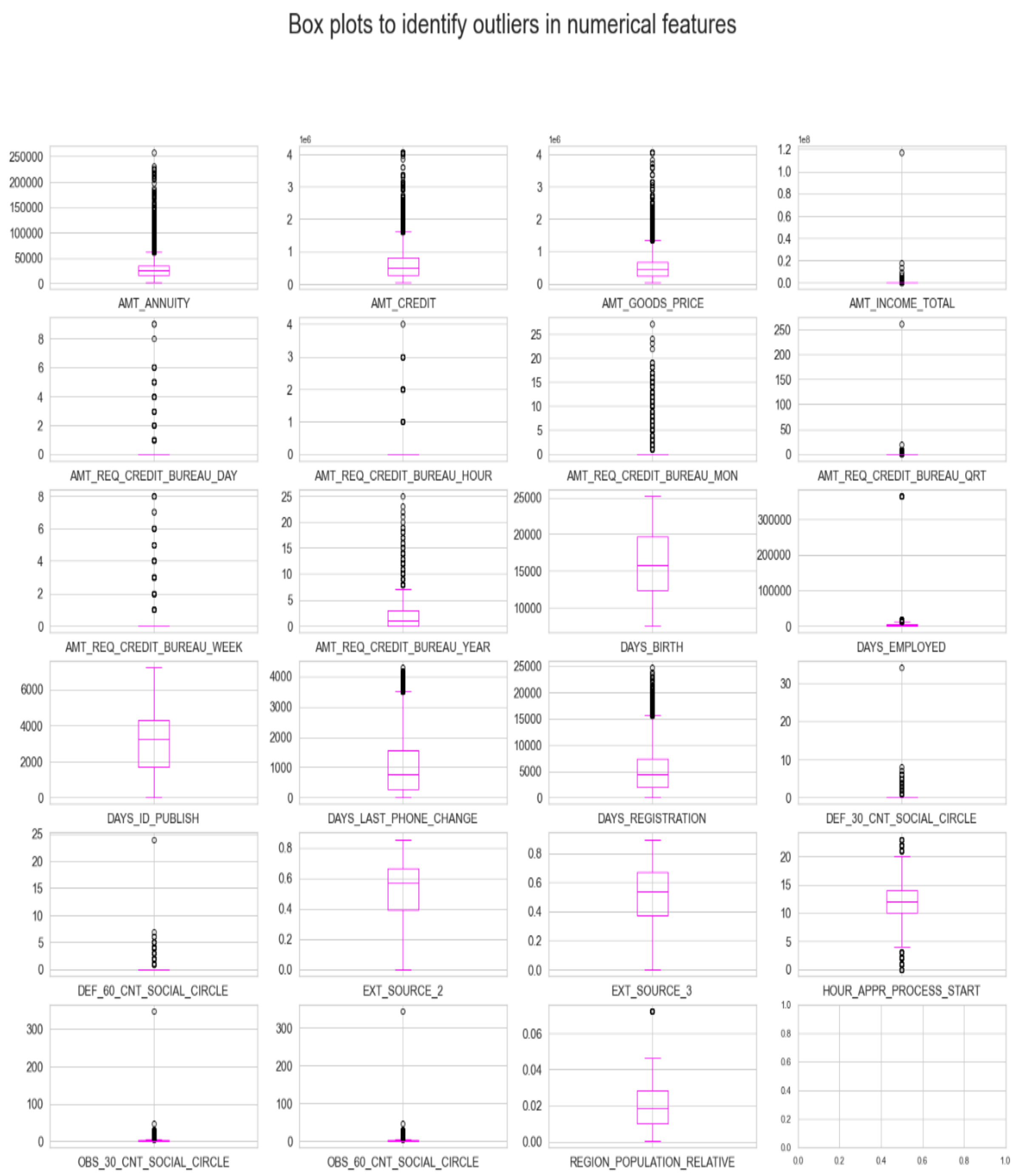
In data analysis, transformation is the replacement of a variable by a function of that variable: for example, replacing a variable x by the square root of x or the logarithm of x. In a stronger sense, a transformation is a replacement that changes the shape of a distribution or relationship. We can use different transformations to make the original data as normal as possible so that the normality assumption in various machine learning algorithms becomes valid. The features in the dataset has been applied with the transformation which reduces the skewness to the least possible extent by using a user-defined function transform_data(). Below are the results of before and after transformation





Outlier detection and treatment

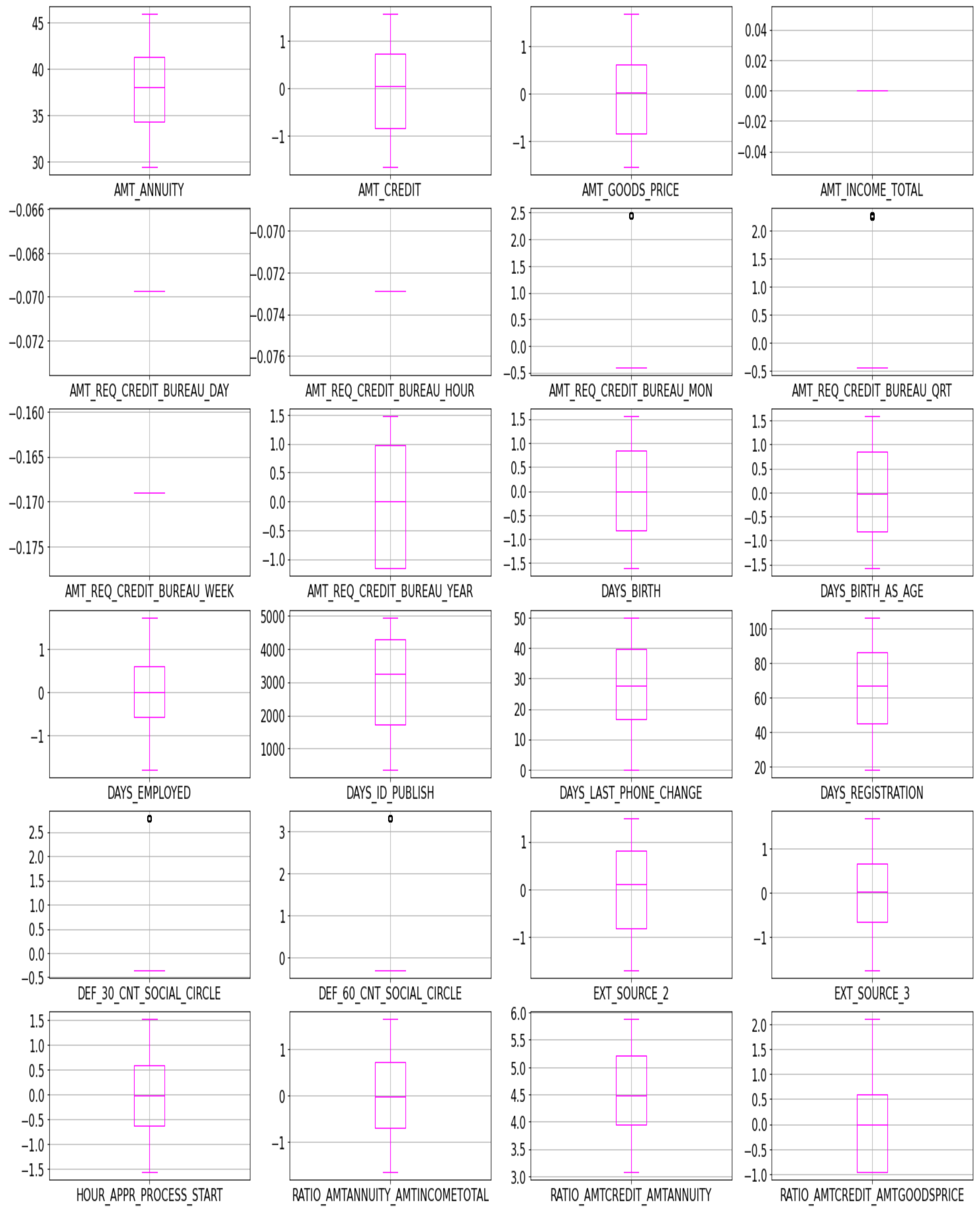
To detect the outliers, we used box-plot, by which we could see that almost all the numerical features have outliers present in them.



Observations: The above boxplots show the presence of outliers except the EXT_SOURCE_2 and EXT_SOURCE_3 feature.

In order to treat outliers, we applied flooring and capping. This technique includes replacing values beyond upper and lower viscous with the values at viscous. We considered quantile value if 0.05 as our lower viscous and a quantile value of 0.95 as upper viscous. And after replacement we again plotted the box-plots for the numerical features and found that no outliers could be detected.

Box plots to identify outliers in numerical features



We can observe that the above result shows most of the outliers have been reduced after outlier treatment.

Feature Encoding

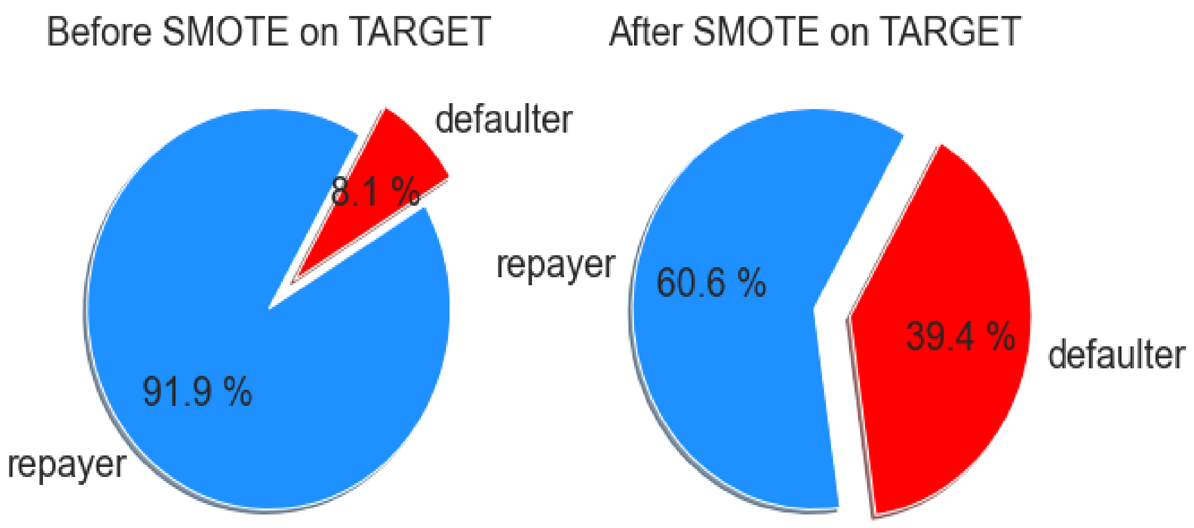
Machines can only understand numbers and hence any categorical feature has to be converted into numerical format before training a model. Although few ML algorithms such as Tree-based (Decision Tree, Random Forest) do a better job in handling categorical variables, it is best practice is to transform categorical data to a numeric value. For the same reason, we used different encoding techniques such as label, ordinal and n-1 dummy encoding on the features depending on their values.

Resultant shape of the dataset after feature encoding (307505, 103)

Class imbalance and its treatment

The dataset which we are using was highly imbalanced with 91.92% values of target column being '0' and only 8.07% values as '1'. For a classification problem treatment of such an imbalanced data set is necessary otherwise our models won't learn much about one kind of TARGET category and would still give high values of performance metrics on measuring. For example, in our case if the model goes on to predict all the results as '0' still it will have 92% accuracy,

whereas we know that the model is not predicting a single correct value for '1' category. We have used Synthetic Minority Oversampling Technique (SMOTE) for balancing the data. We are experimenting for different ratios of value of dependent feature example -0:1=50:50,0:1=60:40, because increasing the percentage of '1' category from 8 to 50 may lead to overfitting, thus experimentation with different ratios is necessary. Shape of data frame after this step is (466422, 103)

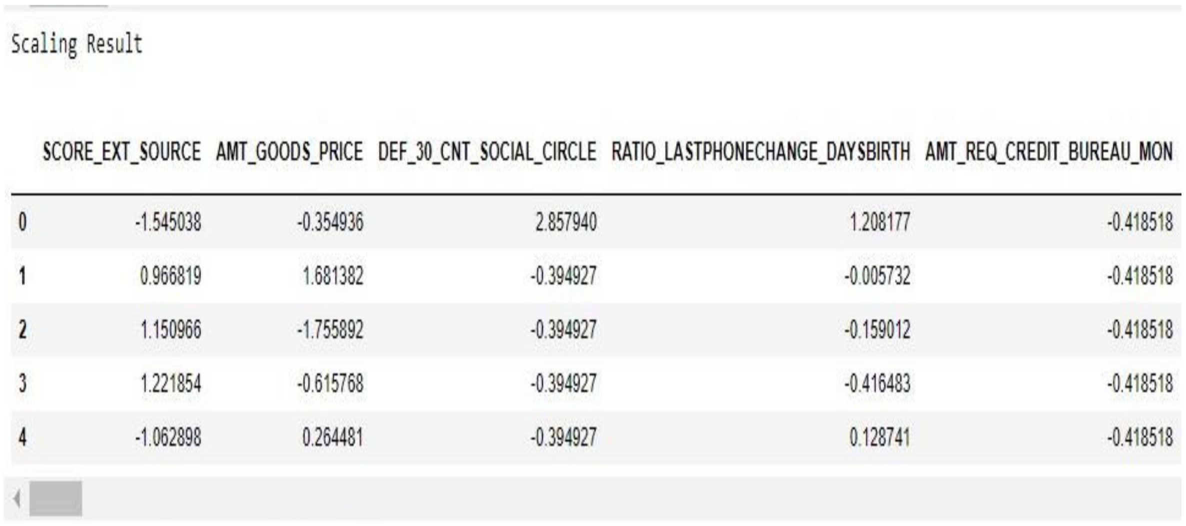


Data Scaling

Feature scaling also known as data normalization is a technique used to transform the data into a common scale. Since the features have various ranges, it becomes a necessary step in data pre-processing while using machine learning algorithms. This step is essential for machine learning methods that calculate distances between data like KNN, K-Means. When computing distances, if the feature with the higher value range is not scaled, the feature with the higher value range starts dominating. Therefore, all features should have their ranges adjusted such that each contributes about proportionally to the final distance. In addition, when we desire faster convergence for algorithms that utilize gradient descent technique, normalization is must. Scaling can also be performed when the ML algorithm expects some scale or to preserve linear relationships between the features, for example linear and logistic regression that require the correlation between the two variables unaltered after a linear transformation.

Algorithms that do not require scaling are the ones which are built on tree structure since they would not be affected by any monotonic transformations of the variables. For example, CART, Random Forests, Gradient Boosted Decision Trees. And classifiers such as Naive Bayes are built to deal with this and assign weights to the features accordingly. Hence, in these algorithms, doing features scaling may not have much of an impact.

In this project, since logistic regression provides the basis for feature importance score based on the coefficients, and assumes that the input variables have the same scale or have been scaled prior to fitting a model, appropriate scaling is applied. However, other algorithms like Decision Tree, Random Forest, Naive Bayes and Gradient Boosted Decision Trees do not provide significant improvement in the results and the same can be verified by comparing the model performance before and after normalization. Few records from the dimension of (466422, 103) of scaled data are as given below:



Scaling Result

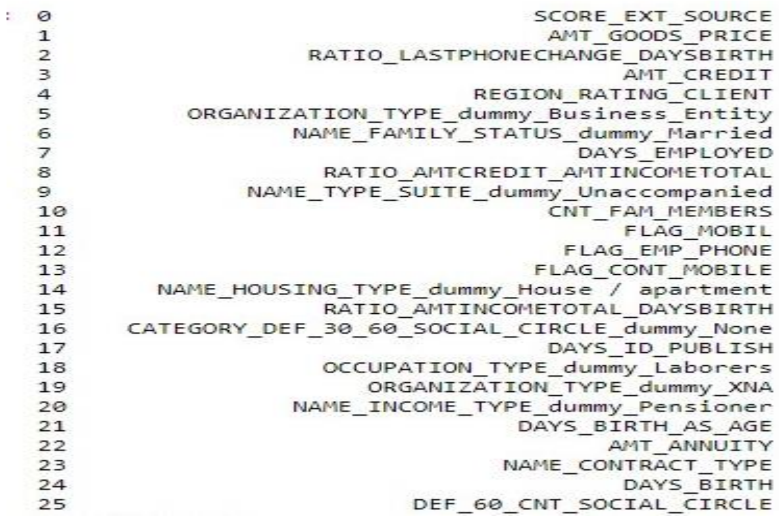
	SCORE_EXT_SOURCE	AMT_GOODS_PRICE	DEF_30_CNT_SOCIAL_CIRCLE	RATIO_LASTPHONECHANGE_DAYSBIRTH	AMT_REQ_CREDIT_BUREAU_MON
0	-1.545038	-0.354936	2.857940	1.208177	-0.418518
1	0.966819	1.681382	-0.394927	-0.005732	-0.418518
2	1.150966	-1.755892	-0.394927	-0.159012	-0.418518
3	1.221854	-0.615768	-0.394927	-0.416483	-0.418518
4	-1.062898	0.264481	-0.394927	0.128741	-0.418518

Check for multi-collinearity

To learn the severity of multi-collinearity we used Variance Inflation factor (VIF). It is the score of an independent variable representing the degree to which other independent variables explain the variable. The variance inflation factor not only pinpoints correlation between independent variables but the strength of the correlation. IF has a range that signifies various levels of multicollinearity. A VIF value of 1 is non-collinear. We consider it to be negligible. If values ranging between 1 and 5 are moderate. They represent a medium level of collinearity. Values higher than 5 are considered highly collinear.

In our project we have used a threshold of 10 and not five because our dataset is large and setting a threshold of 5 is leaving with a very few variables to make a model with. The point to be taken care here is that VIF can be performed only with numerical variables, and we performed VIF before doing feature encoding. Which means that we performed VIF for all the features which were by default numerical and not encoded from categorical to numerical. We didn't perform VIF after feature encoding because that would lead to removal of many encoded columns from one categorical feature, as columns that we get after dummy encoding can have multi-collinearity but removing a few columns and keeping other columns won't make any sense and may also lead to data loss that may be essential for the model. We first drop features with low variance via sklearn's **VarianceThreshold(VT)** which brings the dimensions of the dataset to (466422, 99), and then applying **VIF** results to (466422, 73).

Following are set of features which got dropped after VIF: -

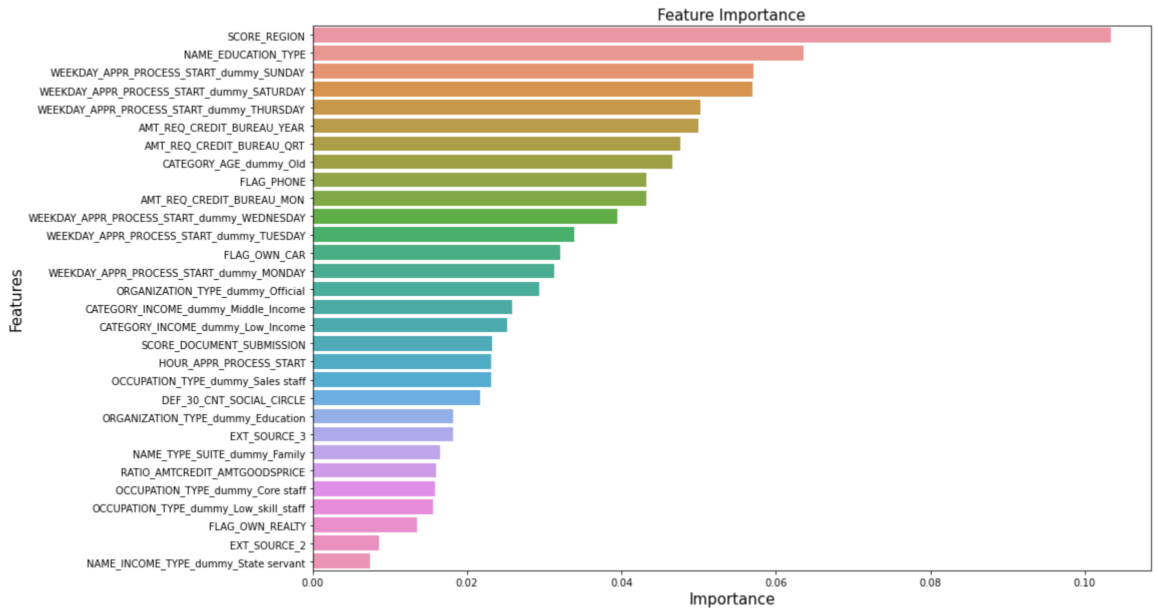


```
:  0                                 SCORE_EXT_SOURCE
   1                                 AMT_GOODS_PRICE
   2                   RATIO_LASTPHONECHANGE_DAYSBIRTH
   3                                 AMT_CREDIT
   4                              REGION_RATING_CLIENT
   5           ORGANIZATION_TYPE_dummy_Business Entity
   6                NAME_FAMILY_STATUS_dummy_Married
   7                                 DAYS_EMPLOYED
   8                    RATIO_AMTCREDIT_AMTINCOMETOTAL
   9              NAME_TYPE_SUITE_dummy_Unaccompanied
   10                                CNT_FAM_MEMBERS
   11                                FLAG_MOBIL
   12                                FLAG_EMP_PHONE
   13                                FLAG_CONT_MOBILE
   14      NAME_HOUSING_TYPE_dummy_House / apartment
   15                   RATIO_AMTINCOMETOTAL_DAYSBIRTH
   16    CATEGORY_DEF_30_60_SOCIAL_CIRCLE_dummy_None
   17                                DAYS_ID_PUBLISH
   18                 OCCUPATION_TYPE_dummy_Laborers
   19                     ORGANIZATION_TYPE_dummy_XNA
   20              NAME_INCOME_TYPE_dummy_Pensioner
   21                                DAYS_BIRTH_AS_AGE
   22                                AMT_ANNUITY
   23                                NAME_CONTRACT_TYPE
   24                                DAYS_BIRTH
   25                         DEF_60_CNT_SOCIAL_CIRCLE
```

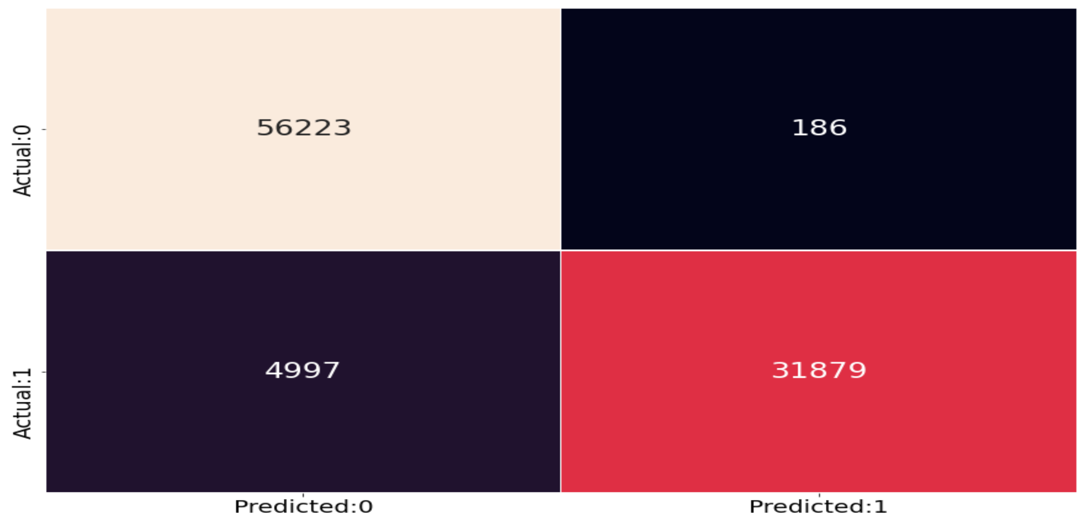
Feature Selection and Dimensionality Reduction

Feature selection process assists the algorithm to feed only those variables which are most important instead of inputting all the available features, especially when the dataset contains too many features. This helps to avoid cardinality problems and overfitting the model. For this dataset, it can be observed that, in an attempt to remove multi-collinearity, Variance Inflation Factor (VIF) brought down the number of features to 73 and the dimension has become sufficient enough for feeding the models. Furthermore, to select and compare top 'k' features Recursive Feature Elimination can be utilized for each model.

Output Generation**:** We will generate outputs in Jupyter Notebook plots to visualize global and local explanations for a given model. For example, below is a sample feature importance plot of a XGBClassifier model created for credit risk.



Similarly, sklearn provides the ability to look into individual prediction explanations like below. For example, when we take a look at a classification report and study with heatmap to observe the model's performance in terms of true and false predictions, we can understand how many observations have been correctly classified. Furthermore, it also helps us to calculate various metrics like accuracy, recall, specificity and f1-score.



4) Model performance and evaluation

Logistic Regression: -

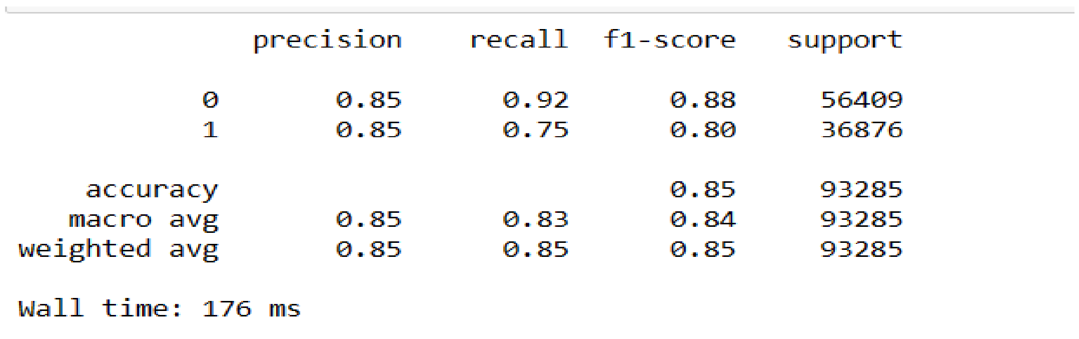
A base model was built using best 30 features obtained from RFE (Recursive feature elimination) and variance threshold method. In order to find the best threshold value for prediction a data frame with columns Probability Cutoff, AUC Score, Precision Score, Recall Score, Accuracy Score, Kappa Score, f1-Score was built.

	Probability Cutoff	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	0.100000	0.715557	0.548152	0.934917	0.669625	0.384182	0.691103
1	0.200000	0.806215	0.676012	0.891854	0.788283	0.580348	0.769076
2	0.300000	0.816673	0.696946	0.884884	0.802391	0.605096	0.779751
3	0.400000	0.824403	0.722270	0.866661	0.815554	0.627100	0.787905
4	0.500000	0.831586	0.852357	0.747858	0.849118	0.677555	0.796695
5	0.600000	0.825132	0.902336	0.699778	0.851380	0.676560	0.788252
6	0.700000	0.820298	0.928089	0.674775	0.850769	0.672370	0.781415
7	0.800000	0.816071	0.942508	0.658396	0.849086	0.666950	0.775241
8	0.900000	0.500000	0.000000	0.000000	0.604695	0.000000	0.000000

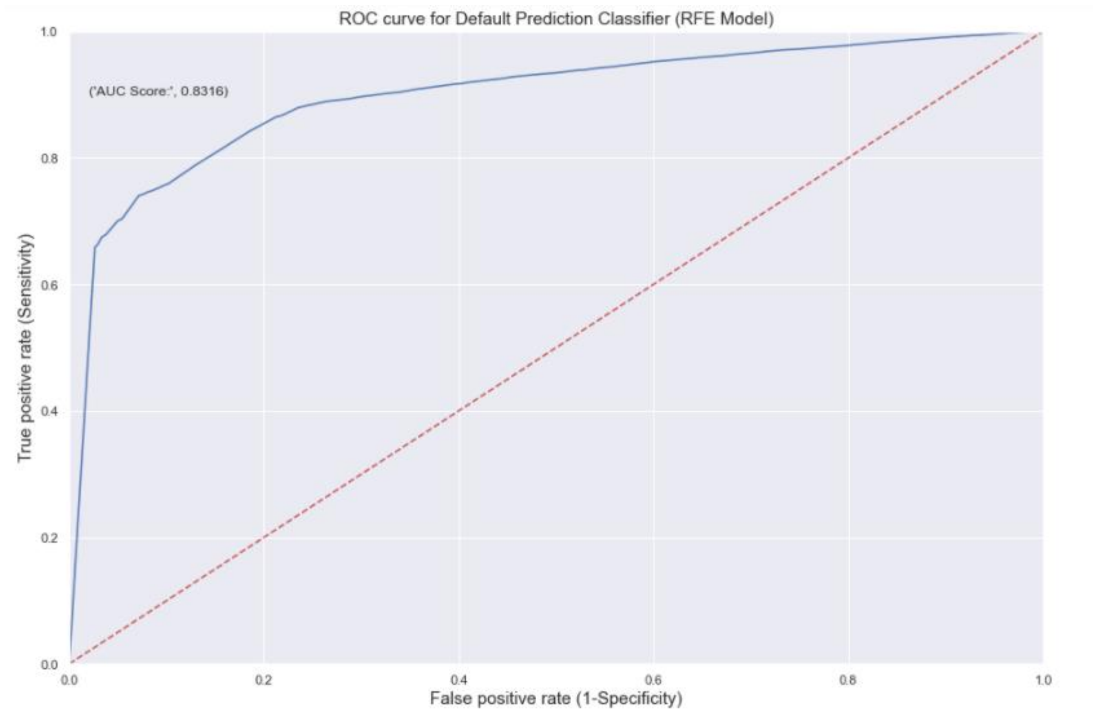
The business objective demands our model to have a high recall and f1-score, because our focus is to detect as many correct defaulters as possible. So, from the above outputs we can see that a cutoff value of 0.1 is having highest recall score of 0.93 but it has a low f1-score. We need to take a cut-off value with good recall and a good f1-score, a balance of both. So, we took the cut-off value of 0.5.

The pseudo rsquare value of the model built is 0.4391, and the LLR p-value being 0.0 thus rejecting the null hypothesis, and the model turns out be significant. FLAG_EMAIL, NAME_TYPE_SUITE_dummy_FAMILY,NAME_TYPE_SUITE_dummy_Other_A,NAME_TYPE_SUITE_dummy_Spouse/partner has the heights values of coefficients. It is observed that for all the 30 features the values of the coefficient were negative. The corresponding values of the coefficients suggest that the odds of an applicant being defaulter increases by a factor of e^(coefficient) due to one unit increase in the feature value, keeping other features constant.

Below we can see at the results of classification report, suggesting a high recall and f1 score for loan payers, and a descent recall and f1 score for defaulters, which means that these values can be improved.



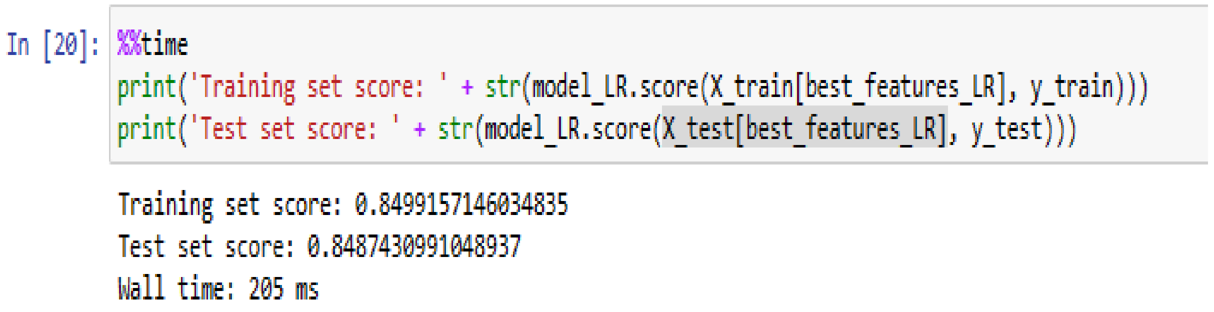
```
              precision    recall  f1-score   support

           0       0.85      0.92      0.88     56409
           1       0.85      0.75      0.80     36876

    accuracy                           0.85     93285
   macro avg       0.85      0.83      0.84     93285
weighted avg       0.85      0.85      0.85     93285

Wall time: 176 ms
```



AUC score=0.8316, tells us that our auc curve is far away from 0.5, and can be considered as a descent score. But we would be focusing on improving this AUC score for other models.

In order to improve this model we used Logistic Regression from sklearn.linear_model , so that we can implement L1 and L2 regularization. We implemented GridSearchCV from sklearn.model_selection in order to perform hyperparameter tuning. Inputs to GridSearchCV for penalty term were L1 and L2, showing that the among the 2 regularization methods would be chosen, and for C log0, log4, log10 were given. The best parameters returned were-C:1.0, penalty: L2. But the model used to use these hyper-parameters didn't have any significant improvement in recall or precision values.

For the model evaluation process Repeated Stratified k-fold cross validation was used in order to get a cross-val-score. KFold process can lead to estimates which can be noisy. An alternate approach is to repeat the k-fold cross-validation process multiple times and report the mean performance across all the folds and all repeats. The cross-validation score turned out to be0.899 which is a good score. The train accuracy and test accuracy are almost equal suggesting that there is no overfitting or under fitting.



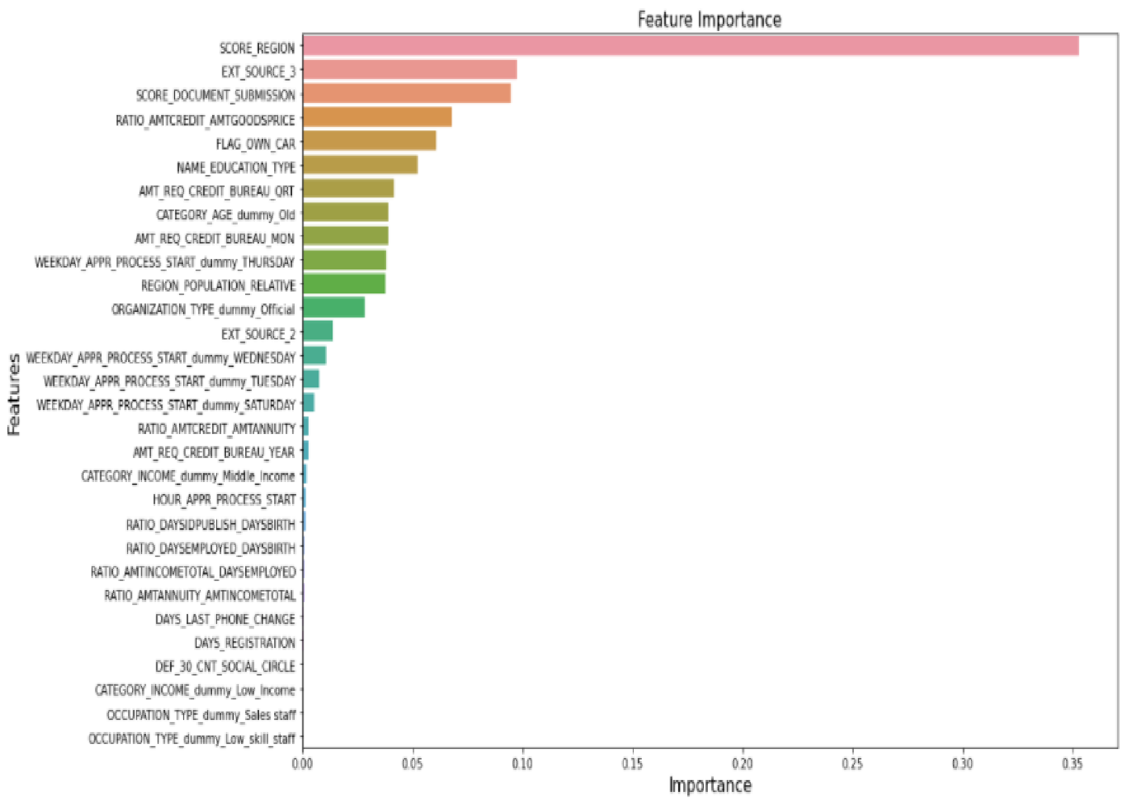
```
In [20]: %%time
         print('Training set score: ' + str(model_LR.score(X_train[best_features_LR], y_train)))
         print('Test set score: ' + str(model_LR.score(X_test[best_features_LR], y_test)))

         Training set score: 0.8499157146034835
         Test set score: 0.8487430991048937
         Wall time: 205 ms
```

Decision tree: -

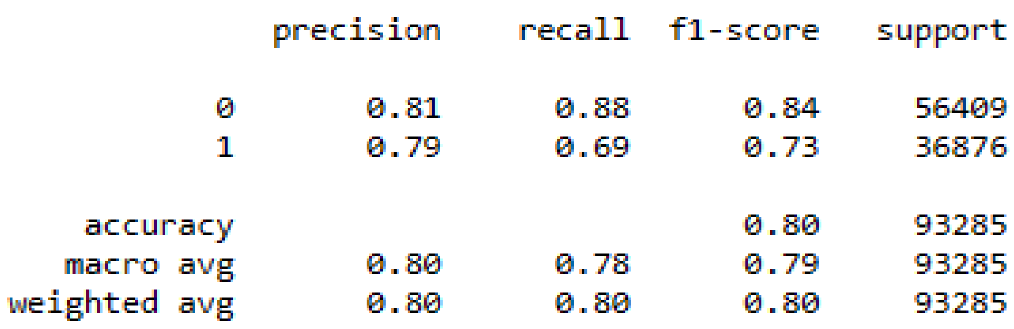
Same procedure of first dealing with multicollinearity and then selecting features using RFE and variance threshold was used for this algorithm too. For tuning the hyper parameters, the inputs to the algorithm included-

{'criterion': ['entropy', 'gini'], 'max_depth': range (2, 10),'max_features': ["sqrt", "log2"]}

Best parameters turned out to be - {'criterion': 'gini', 'max_depth': 9, 'max_features': 'sqrt'}.
After building the decision tree using the hyper-parameters, we visualised the importance of the feat ures using a bar graph.

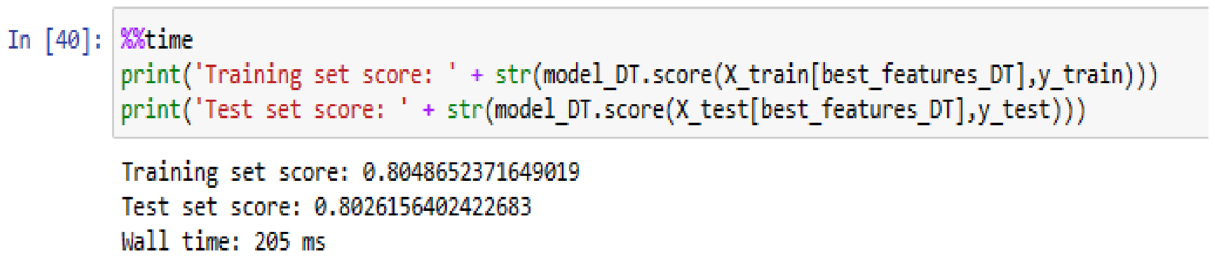


From the graph we can see that SCORE_REGION, EXT_SCORE_3, SCORE_DOCUMENT_SUBMISSION is the top most import features.



```
              precision    recall  f1-score   support

           0       0.81      0.88      0.84     56409
           1       0.79      0.69      0.73     36876

    accuracy                           0.80     93285
   macro avg       0.80      0.78      0.79     93285
weighted avg       0.80      0.80      0.80     93285
```

The classification report for decision tree shows us that logistic regression performed better as compared to the decision tree, as he had a higher recall and f1 score for both repayers and defaulters for logistic regression. We got a Cohen kappa score of 0.577 which suggests that we have moderate agreement between actual and predicted values. The train accuracy and test accuracy are almost equal suggesting that there is no overfitting or under fitting.



```
In [40]: %%time
         print('Training set score: ' + str(model_DT.score(X_train[best_features_DT],y_train)))
         print('Test set score: ' + str(model_DT.score(X_test[best_features_DT],y_test)))

         Training set score: 0.8048652371649019
         Test set score: 0.8026156402422683
         Wall time: 205 ms
```

The cross-validation score for decision tree is equal to 0.868, suggesting the model to be a descent one.

Random Forest: -

Did the VIF and feature selection for Random Forest, followed by grid search CV for tuning hyper parameter tuning. Following set of inputs were provided to grid search CV: -

tuned_paramaters = [{'criterion': ['gini'],

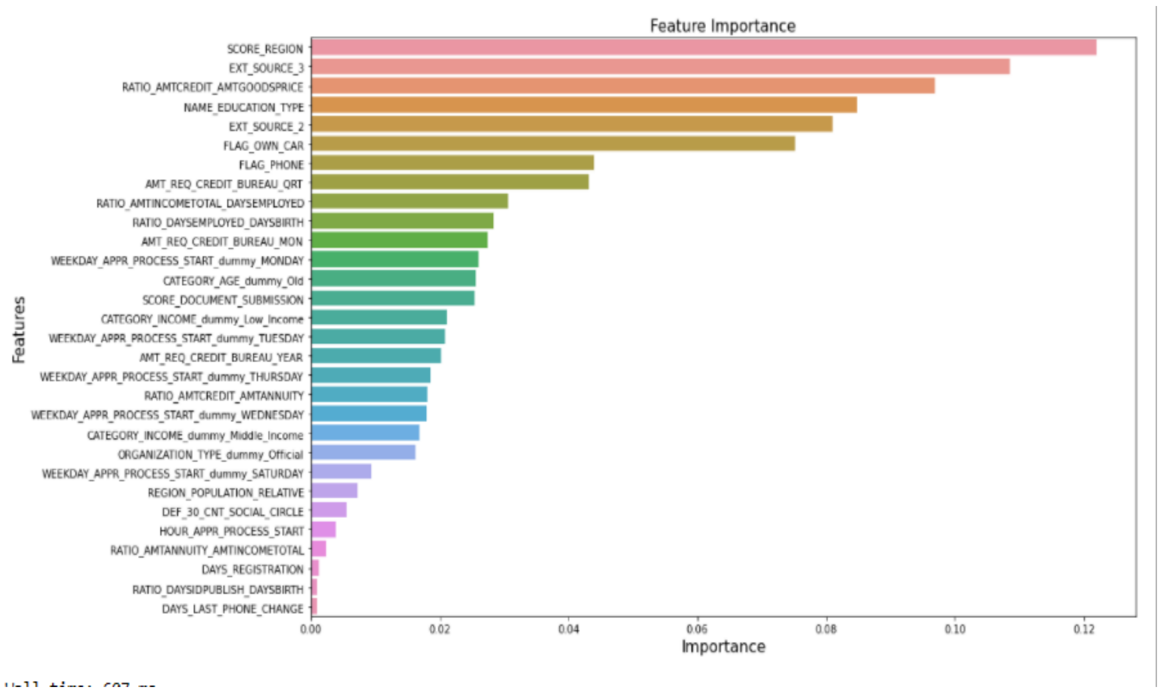
'n_estimators': [10, 30, 50, 70, 90],

'max_depth': range (2, 10),

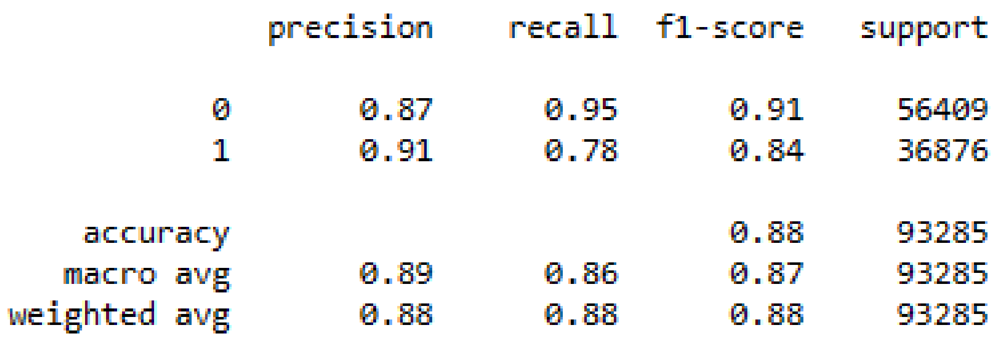
'max_features': ["sqrt", "log2"]}]

The best parameteres obtained were: -

Best parameters for random forest classifier: {'criterion': 'gini', 'max_depth': 9, 'max_features': 'log2', 'n_estimators': 70}
Then built a model using the best parameter values and plotted the feature importance on bar graph.

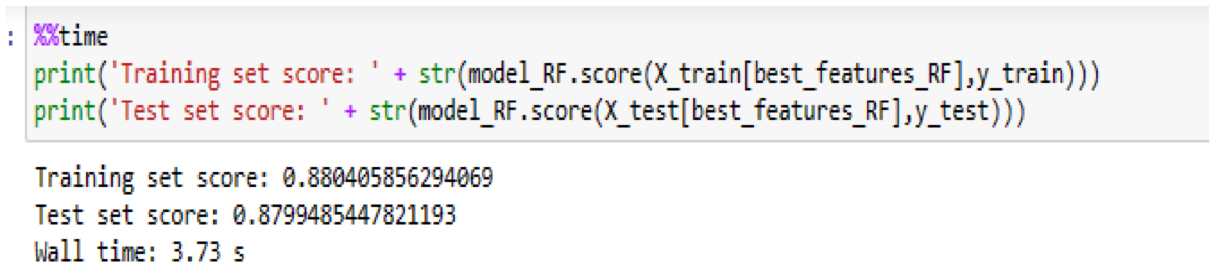


From the classification report we can see that precision score, recall score and f1-score is better for both 1 and 0 as compared to decision tree and logistic regression.

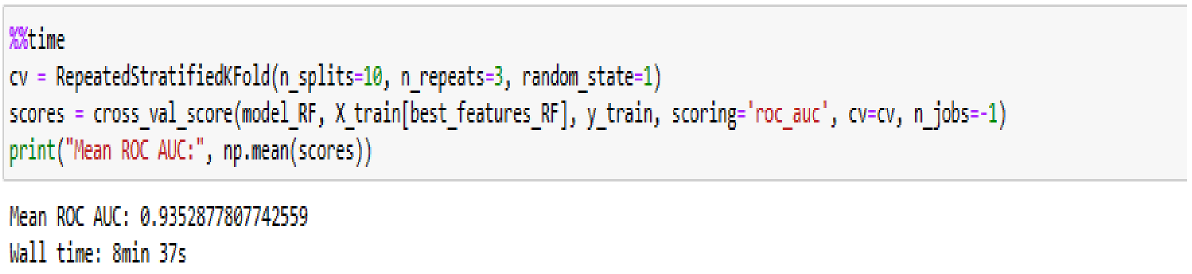


```
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     56409
           1       0.91      0.78      0.84     36876

    accuracy                           0.88     93285
   macro avg       0.89      0.86      0.87     93285
weighted avg       0.88      0.88      0.88     93285
```

The accuracy score is almost similar to logistic regression but better than decision tree. The Cohen kappa score is equal to 0.74, which shows substantial agreement between predicted and actual values. The accuracy scores for training and test data are almost similar showing us that the model doesn't suffer from over-fitting.



```
%%time
print('Training set score: ' + str(model_RF.score(X_train[best_features_RF],y_train)))
print('Test set score: ' + str(model_RF.score(X_test[best_features_RF],y_test)))

Training set score: 0.880405856294069
Test set score: 0.8799485447821193
Wall time: 3.73 s
```



```
%%time
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(model_RF, X_train[best_features_RF], y_train, scoring='roc_auc', cv=cv, n_jobs=-1)
print("Mean ROC AUC:", np.mean(scores))
```

```
Mean ROC AUC: 0.9352877807742559
Wall time: 8min 37s
```

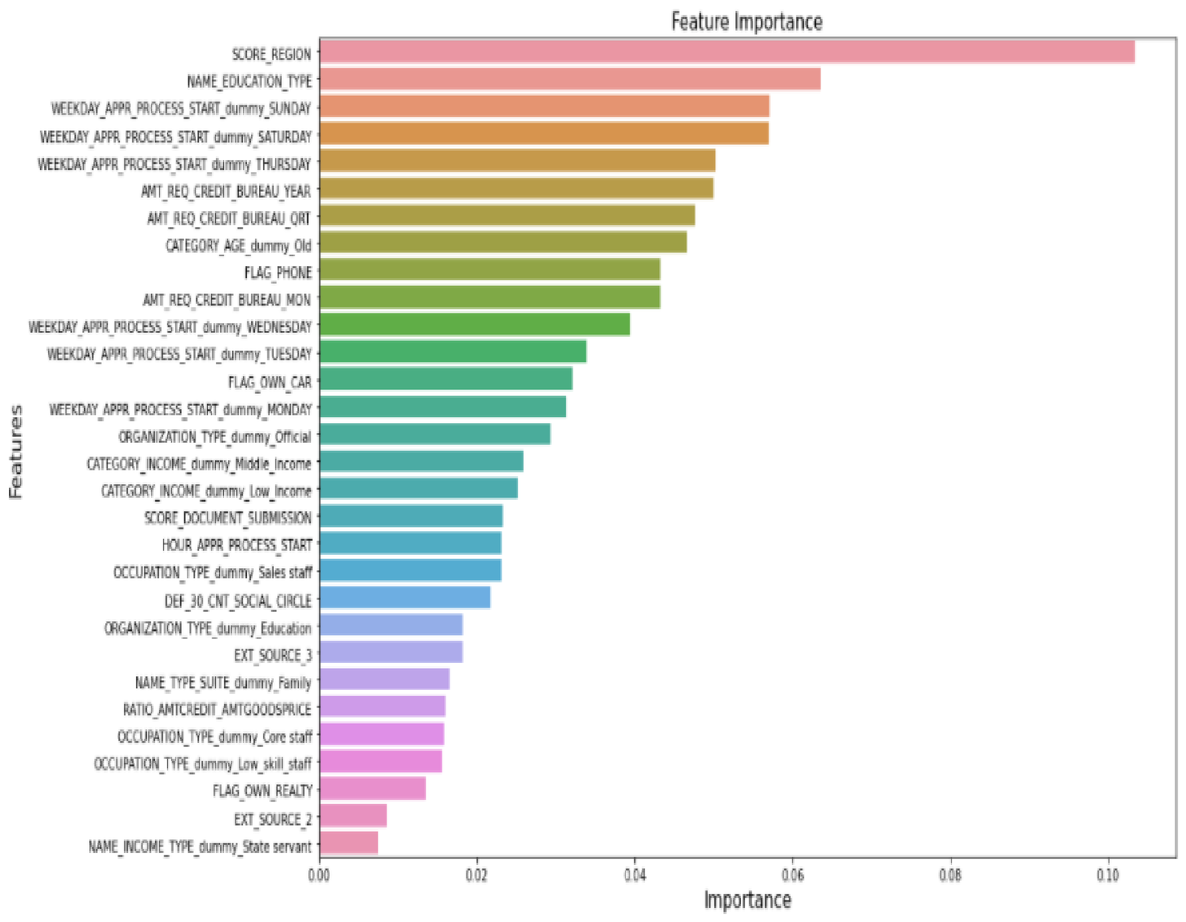
From above output we can see that the average ROC AUC score over all the CVs, is 0.93 which is far away from 0.5 and is a large value showing that the model built is good.

XGBoost: -

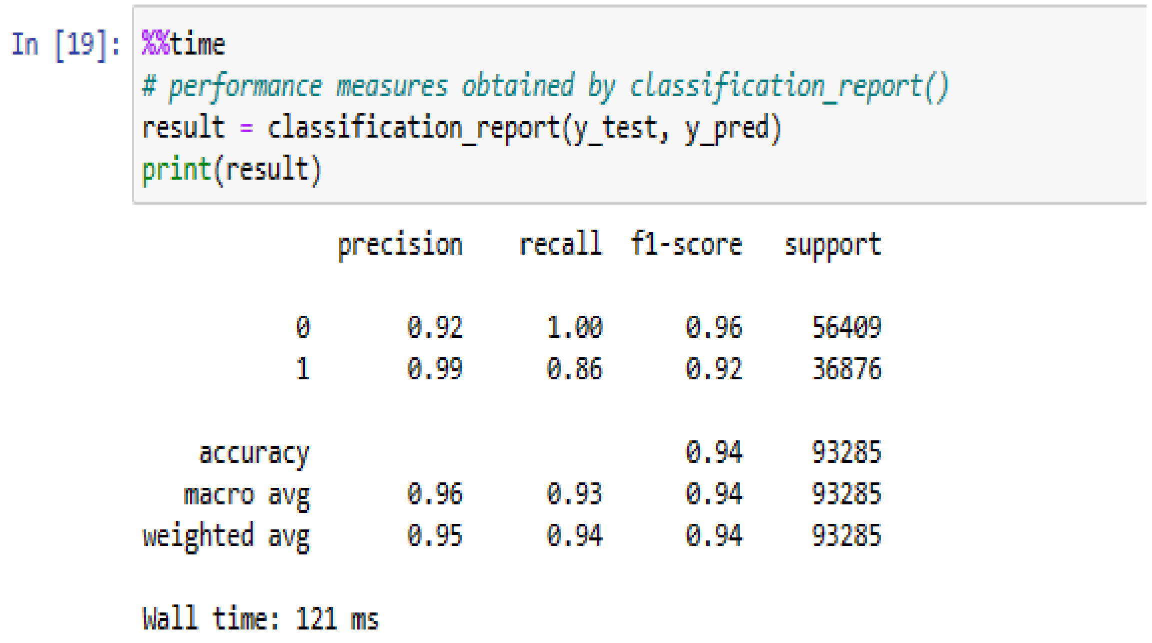
Input parameters to GridSearchCV: - Best parameters for XGBoost classifier: {'learning_rate': [0.1, 0.0 1, 0.05], 'max_depth': range (3,10,2),'gamma': [0, 1, 2, 3, 4]}.

Best parameters for XGBoost classifier: {'gamma': 4, 'learning_rate': 0.1, 'max_depth': 9}.

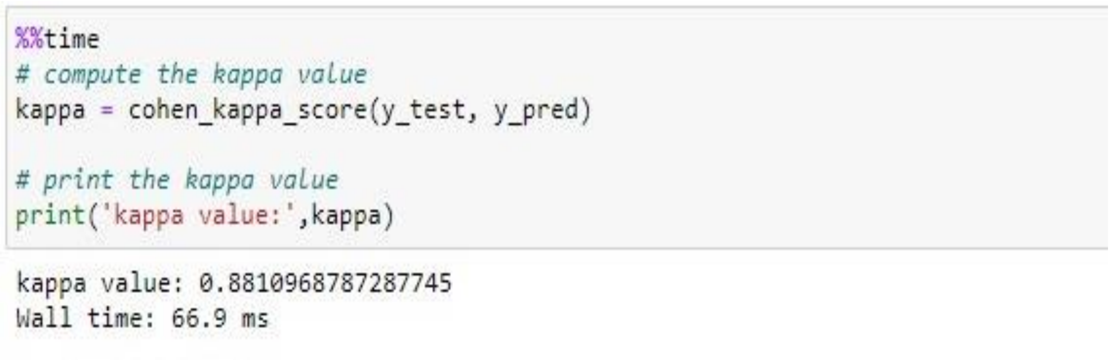
Feature importance output: -



Output of classification report: -

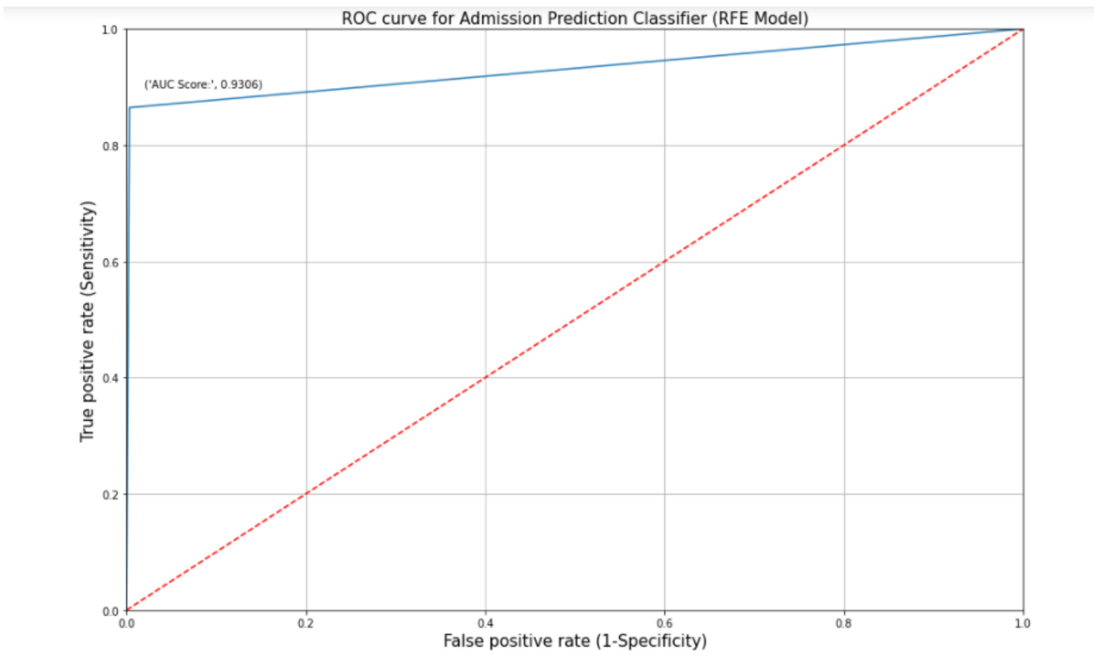


```
In [19]: %%time
         # performance measures obtained by classification_report()
         result = classification_report(y_test, y_pred)
         print(result)

                      precision    recall  f1-score   support

                   0       0.92      1.00      0.96     56409
                   1       0.99      0.86      0.92     36876

            accuracy                           0.94     93285
           macro avg       0.96      0.93      0.94     93285
        weighted avg       0.95      0.94      0.94     93285

         Wall time: 121 ms
```



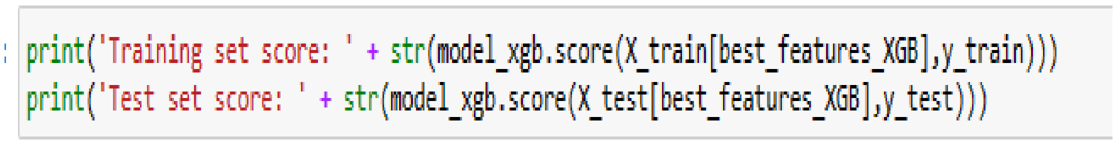
```
%%time
# compute the kappa value
kappa = cohen_kappa_score(y_test, y_pred)

# print the kappa value
print('kappa value:',kappa)

kappa value: 0.8810968787287745
Wall time: 66.9 ms
```

The Cohen kappa score is 0.88 showing almost perfect agreement between actual and predicted valu es.

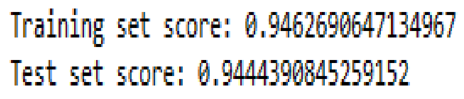


The ROC AUC score is also far away from 0.5 and is very high indicating a good model.

The train and test score are almost similar showing that there is no overfitting.

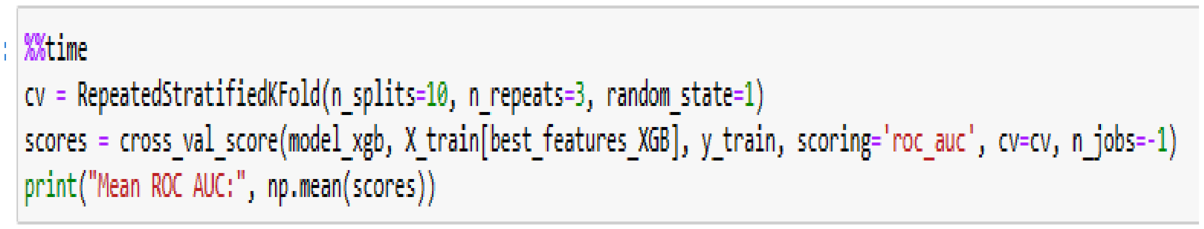


```
print('Training set score: ' + str(model_xgb.score(X_train[best_features_XGB],y_train)))
print('Test set score: ' + str(model_xgb.score(X_test[best_features_XGB],y_test)))
```

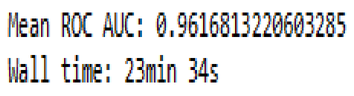


```
Training set score: 0.9462690647134967
Test set score: 0.9444390845259152
```

Cross val score is also very high showing no case of overfitting.



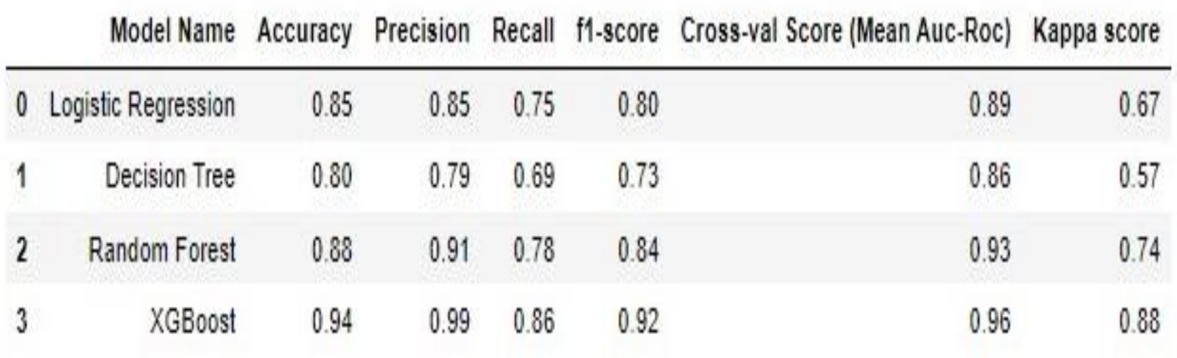
```
%%time
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(model_xgb, X_train[best_features_XGB], y_train, scoring='roc_auc', cv=cv, n_jobs=-1)
print("Mean ROC AUC:", np.mean(scores))
```



```
Mean ROC AUC: 0.9616813220603285
Wall time: 23min 34s
```

5) Comparison to benchmark

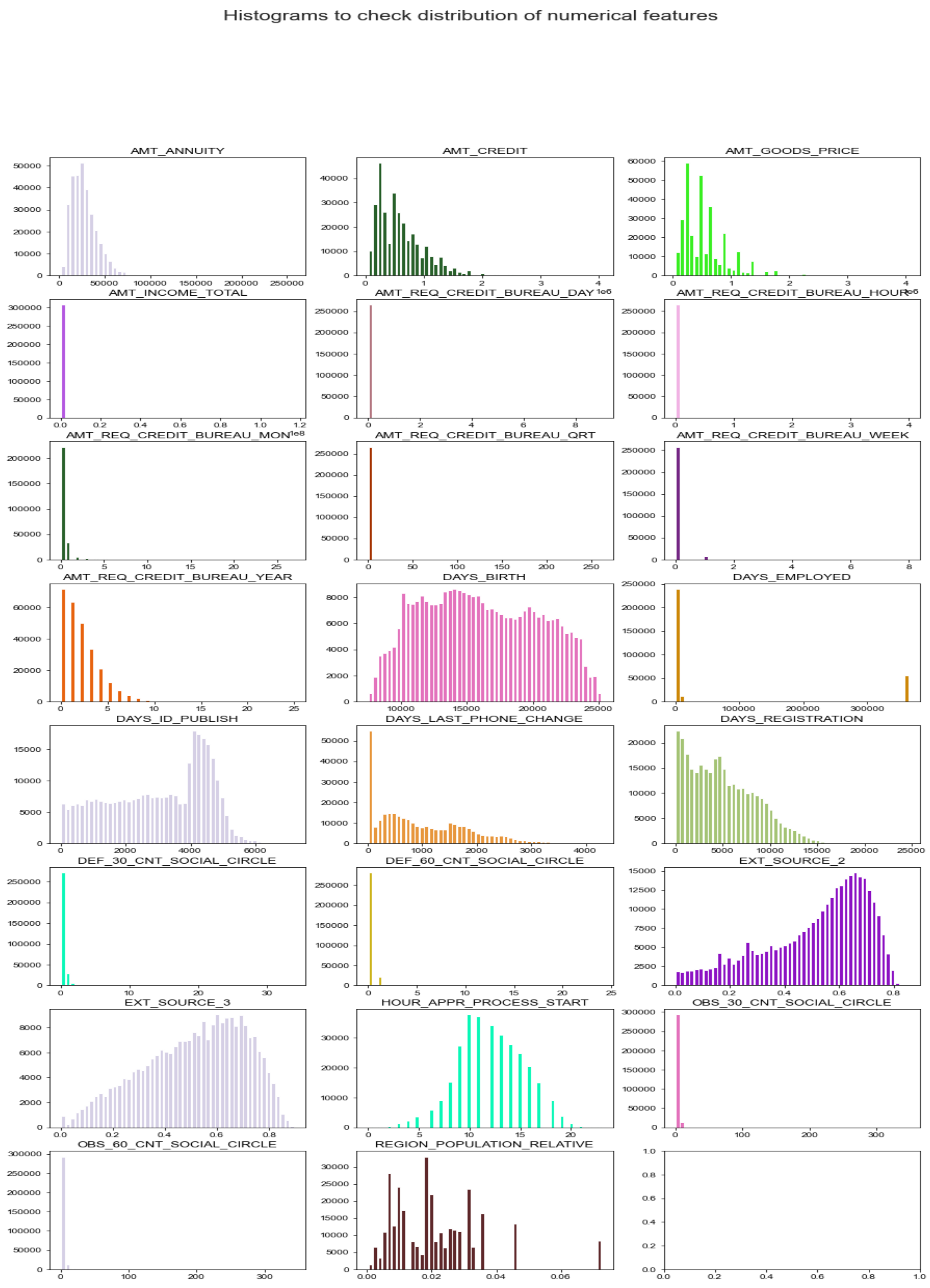
Our benchmark was to obtain a minimum of 0.75 Roc-Auc Score for each model. So, we started off by considering Logistic Regression as base model, and comparing the performance of other models with respect to a common metric, in our case the cross-val-score. The following table illustrates that the Decision Tree had low performance compared to peer models, for which we assume the reason could be the dataset is too large to tune it's hyperparameters, however, Random Forest might have given better results because of randomness and bagging nature. Obvious that, eXtreme gradient boosted model outperformed all the models as improves upon the base gradient boosting framework through systems optimization and algorithmic enhancements, such as: hardware optimization, parallelized tree building, efficient handling of missing data, tree pruning using 'depth-first' approach, built-in cross-validation capability (at each iteration), regularization through both LASSO (L1) and Ridge (L2) for avoiding overfitting.



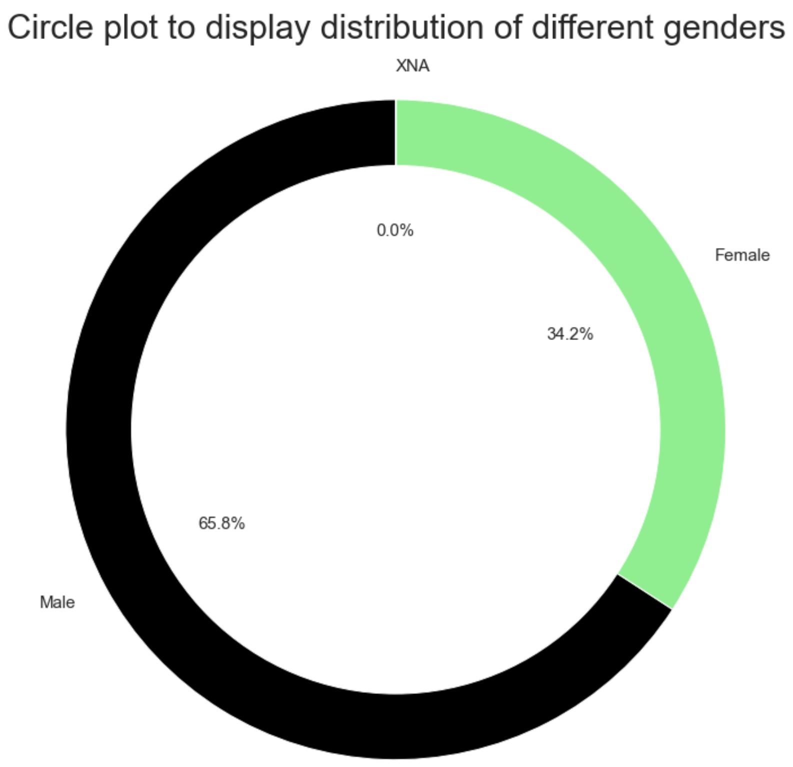
	Model Name	Accuracy	Precision	Recall	f1-score	Cross-val Score (Mean Auc-Roc)	Kappa score
0	Logistic Regression	0.85	0.85	0.75	0.80	0.89	0.67
1	Decision Tree	0.80	0.79	0.69	0.73	0.86	0.57
2	Random Forest	0.88	0.91	0.78	0.84	0.93	0.74
3	XGBoost	0.94	0.99	0.86	0.92	0.96	0.88

6) Visualizations

Distribution of variables

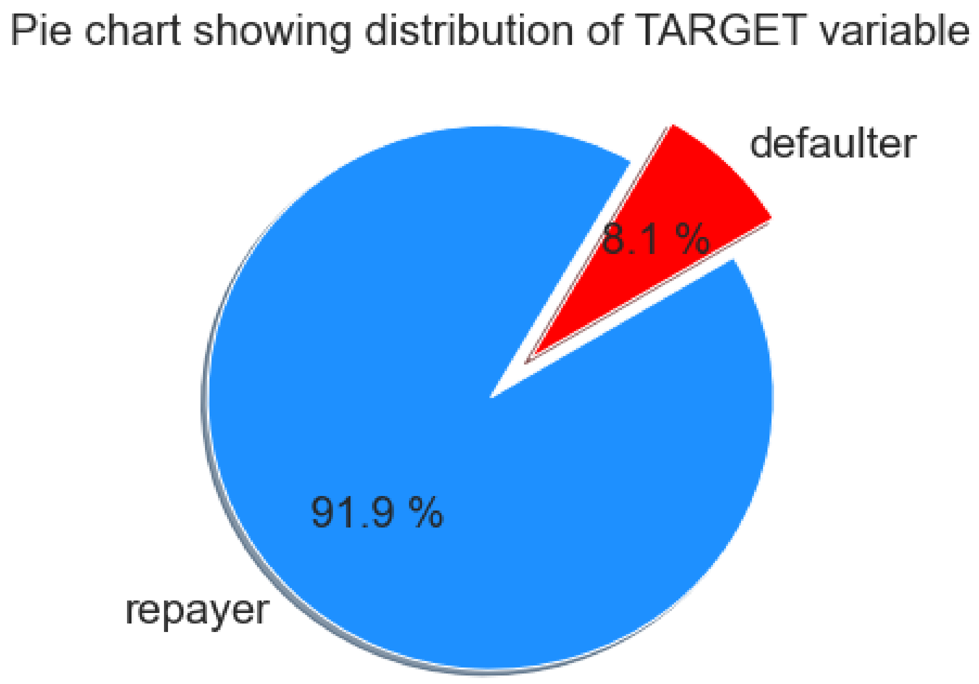


Observations: The above visualization represents the Histograms to check distribution of numerical features. From the visualization it can be inferred that most of the numerical data are skewed in nature. Majority of the skewness is right skewness.



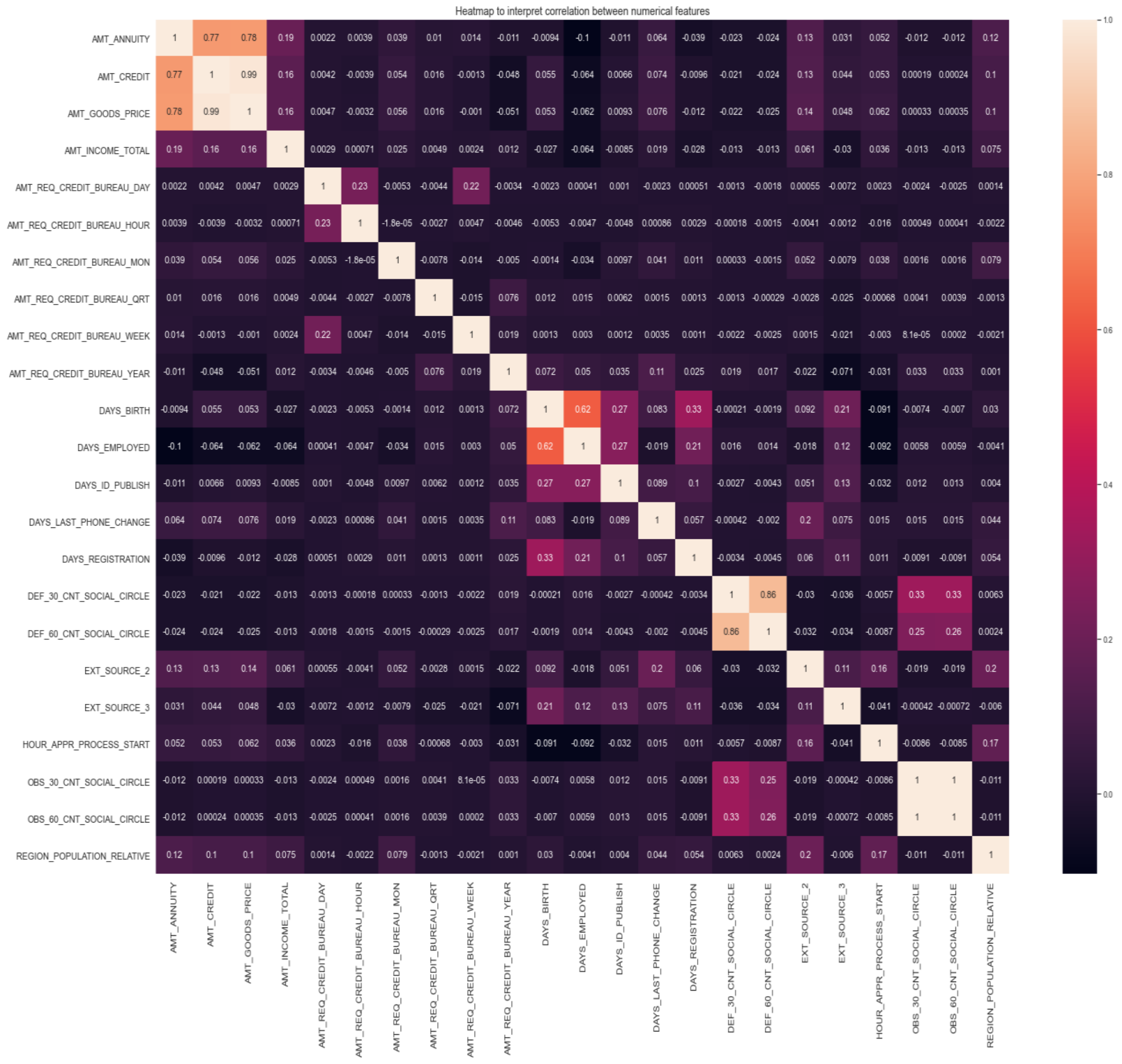
Observations: The above pie chart tells us that number of Males in applying for loans are greater than number of females that apply for loans. The number of males is approximately double the female

Check for Imbalance



Observation: We can conclude from the pie-chart that the dependent variable is imbalanced.

Relationship between variables



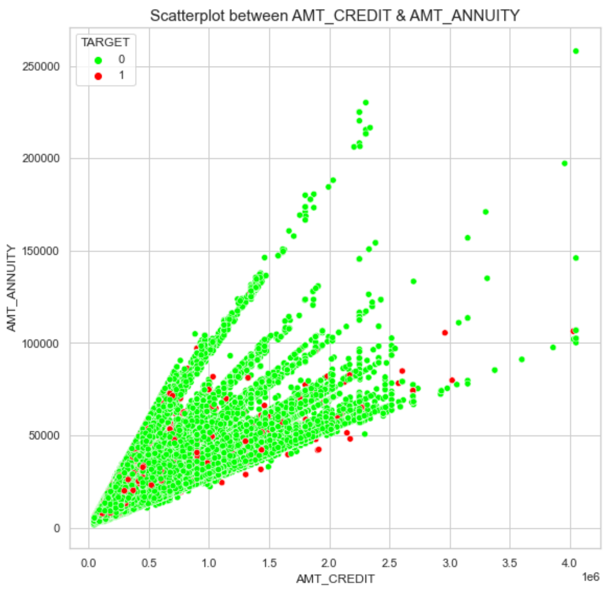
Following pair of features exhibit strong positive correlation: -

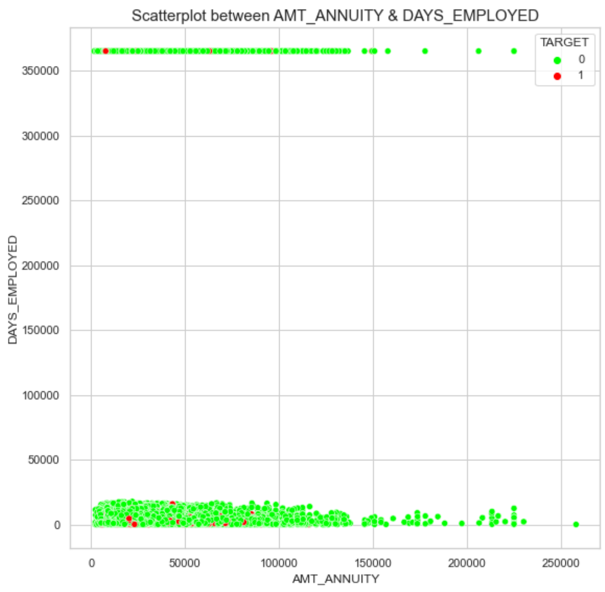
OBS_60_CNT_SOCIAL_CIRCLE	-	OBS_30_CNT_SOCIAL_CIRCLE
FLAG_EMP_PHONE	-	DAYS_EMPLOYED
AMT_GOODS_PRICE	-	AMT_CREDIT
REGION_RATING_CLIENT_W_CITY	-	REGION_RATING_CLIENT
CNT_FAM_MEMBERS	-	CNT_CHILDREN
DEF_60_CNT_SOCIAL_CIRCLE	-	DEF_30_CNT_SOCIAL_CIRCLE
LIVE_REGION_NOT_WORK_REGION	-	REG_REGION_NOT_WORK_REGION
LIVE_CITY_NOT_WORK_CITY	-	REG_CITY_NOT_WORK_CITY
AMT_ANNUITY	-	AMT_CREDIT
AMT_GOODS_PRICE	-	AMT_ANNUITY

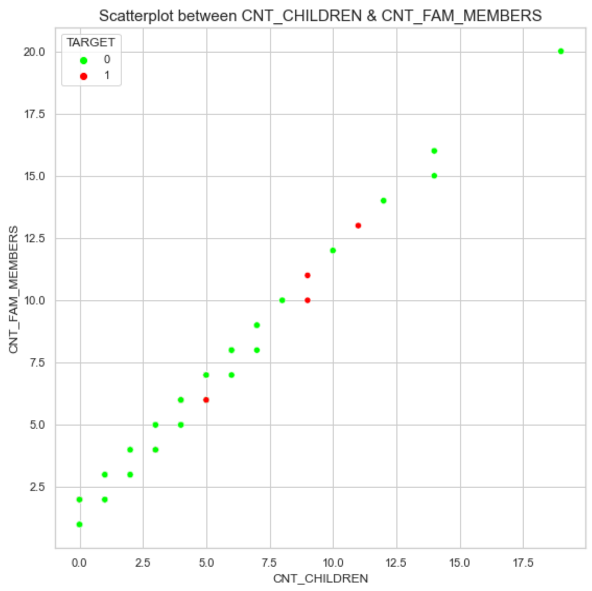
Following pair of features exhibit strong negative correlation: -

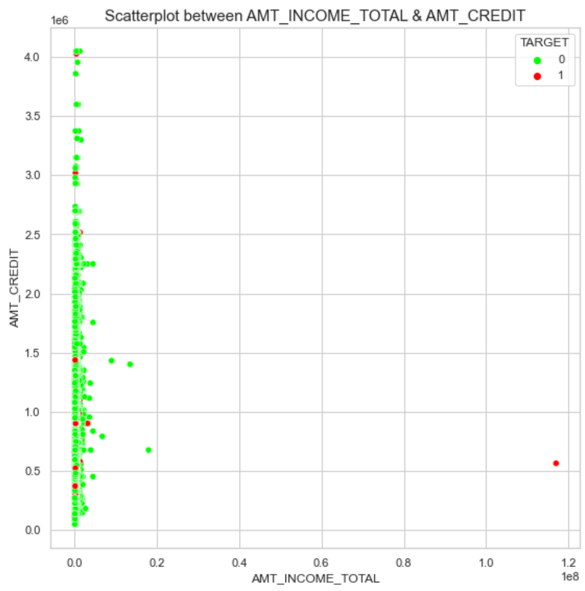
```
REGION_RATING_CLIENT_W_CITY      -       REGION_POPULATION_RELATIVE

REGION_RATING_CLIENT_CLIENT      -       REGION_POPULATION_RELATIVE
```

Multivariate Analysis:

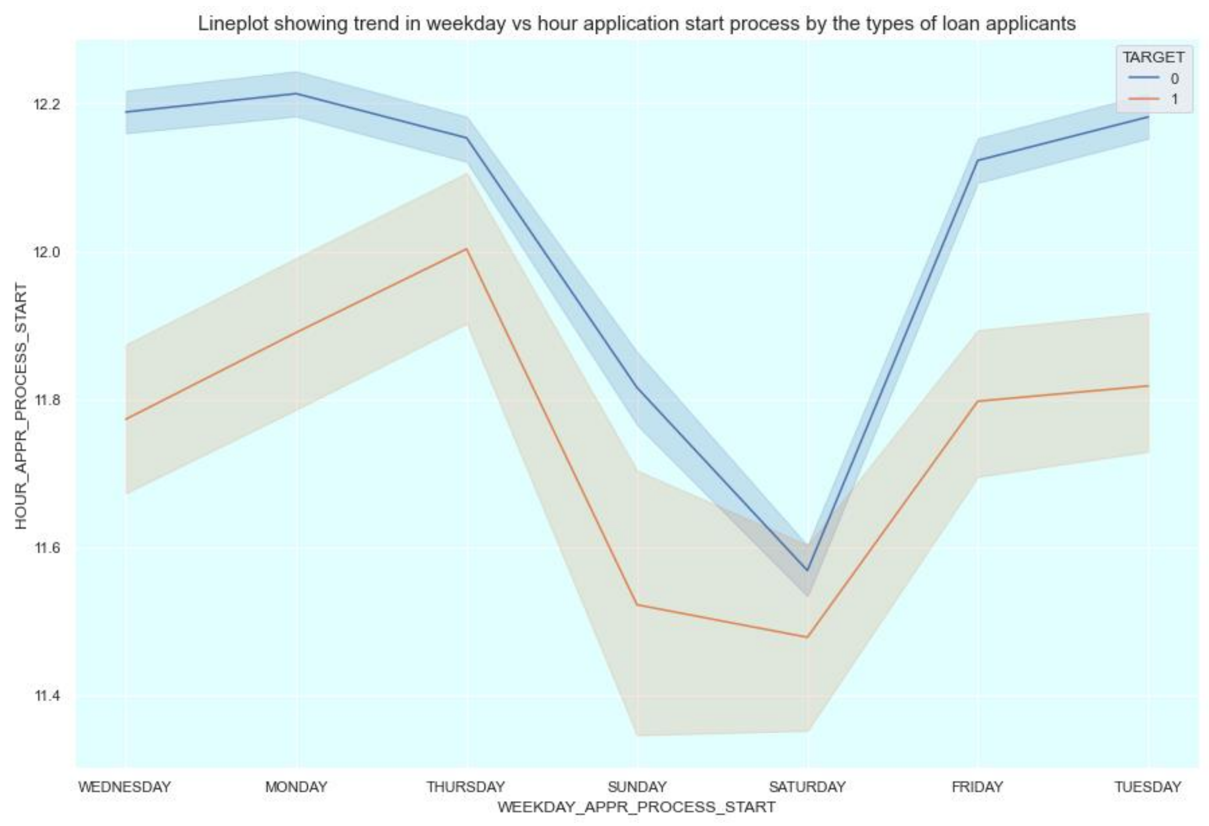








Observations: Family Count is directly proportional to the Count of children in a house. With increase in credit amount, annuity increases.



Observations: Clients applied for loan mostly on Thursday and weekdays overall and least on Weekends

7) Implications

We have shown that the selection of the 30 top variables, based on the variable importance of models, does not necessarily yield stable results given the underlying model. Our strategy of re-checking model performance on these top variables can help data experts validate the choice of models and variables selected. As demonstrated in the project, requesting a few lists of variables from clients can help speed up the time to deliver a decision on a loan request thereby improving the profitability and certainty to the financial businesses. Additionally, we have shown that it is important to consider a pool of models that match the data and business problem.

To improve the transparency of the modelling process, we recommend that performance metrics not be confined to only one criterion like the AUC and it is important to note that standard criteria like the AIC, Bayesian information criterion (BIC) and R2 are not always suitable for the comparison of model performance, given the types of model classes (regression based, classification based, etc.).

8) Limitations

It The logistic regression and decision tree models is having a low recall and f1-score as compared to other two models, meaning that these models will produce more errors in prediction as compared to the other two models when provided with real world data. The logistic regression model is not showing any performance improvement even after providing penalty using L2 regularization. As per our goal of precisely predicting as many correct defaulters as possible using a smaller number of features, we couldn’t provide too many input parameters to decision tree as the grid search CV is taking too long if we give more input parameters to get the best from. The tree-based models also have a limitation of giving less feature importance to dummy encoded features as the column may contain a large number of 1’s or 0’s as compared to the other level, thus making the column less important for model. This issue normally happens when we have a column with large number of levels and dummy encoding returning us a large number of new columns. For this reason, we reduced the levels of some columns to below 15 which were around 50. But we may could have got a better model performance for decision tree and random forest if we would have reduced the levels further down to 7 or 8. Another thing that could have helped is using binary or target encoding, this could have also returned better model performance. In case of XGBoost we sticked with the default value of gamma=1 which could have been changed to any other value, but the performance of XGBoost is already very good, anything more could have led to overfitting. While another thing that could have improved model performance is that selecting some features for transformation before outlier treatment, rather than transforming all the features at once. The reason behind transforming the features before outlier treatment is that there may be some columns for which the outliers shown by boxplot were too many, leading us to think if they were actually outliers or a new category of data in itself. The other possible things that could have improved model performances are trying forward and backward feature selection, CA for dimensionality reduction.

9) Closing Reflections

The process involved a lot of new learning regarding many new techniques for every step. A summary of the learning is below: -

- Learning to use power transformers
- How to use business and domain knowledge to impute new columns or reduce redundant columns.
- Getting insights about missing values and not simply replacing missing values with mean, median or modes.
- Dealing with outliers using new techniques like flooring and caping, and getting to know reasons about a value being outlier before just treating it.
- Logically reducing the number of levels for categorical columns.
- Using pipelines for getting to know best scaling methods for every algorithm.
- Learning and using Repeated stratified cross validation.
- Deep understanding of all the classification algorithms and resolving the possible run time errors.

Assumptions

Regression – Logistic Regression

1) The Response Variable is Binary: Logistic regression assumes that the response variable only takes on two possible outcomes.
2) The Observations are Independent: Logistic regression assumes that the observations in the dataset are independent of each other. The easiest way to check this assumption is to create a plot of residuals against time (i.e., the order of the observations) and observe whether or not there is a random pattern. If there is not a random pattern, then this assumption may be violated.
3) There is No Multicollinearity Among Explanatory Variables: Logistic regression assumes that there is no severe multicollinearity among the explanatory variables.
4) There are No Extreme Outliers: Logistic regression assumes that there are no extreme outliers or influential observations in the dataset.

5) There is a Linear Relationship Between Explanatory Variables and the Logit of the Response Variable: Logistic regression assumes that there exists a linear relationship between each explanatory variable and the logit of the response variable.

6) The Sample Size is Sufficiently Large: Logistic regression assumes that the sample size of the dataset if large enough to draw valid conclusions from the fitted logistic regression model.

Classification –

Decision Tree

1) Discretization of continuous variables is required
2) The data taken for training should be wholly considered as root
3) Distribution of records is done in a recursive manner on the basis of attribute values.

Random Forest

Assumption of no formal distributions. Being a non-parametric model, it can handle skewed and multi-modal data.

Gradient Boosted models

It may have an assumption that encoded integer value for each variable has ordinal relation.

Naïve Bayes

The biggest and only assumption is the assumption of conditional independence.

Sample Reference for Dataset(s)

Original owner of data	Kaggle
Data set information	https://www.kaggle.com/c/home-credit-default-risk
Any past relevant articles using the dataset	https://www.record-evolution.de/en/credit-default-risk-prediction/
Reference	Credit Risk Analysis Using Machine and Deep Learning Models
Link to web page	https://www.mdpi.com/2227-9091/6/2/38

-- **End of Final Report** --