Feature selection

Featurization produced 1770 features, however all of these features can not be used in the machine learning model. It is necessary to understand the most important features among these 1770 features which will be useful for better predictions.

This script describes the steps in feature selection such as data loading, data cleaning, statistical methods (Weight of evidence and Information Value) for feature selection and final feature selection using votes from multiple algorithms.

Importing libraries, modules and data

In [ ]:
```
# Importing the required libraries and modules
import os
import zipfile
import numpy as np
import pandas as pd
from datetime import datetime
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
from sklearn.preprocessing import MinMaxScaler
import imblearn
from collections import Counter
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler

import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
# Mounting google drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

Defining the functions

In [ ]:
```
#Function for reducing the memory usage of dataframe
def reduce_mem_usage(data, verbose = True):
    #Reference: https://www.kaggle.com/gemartin/load-data-reduce-memory-usage
    '''
    This function is used to reduce the memory usage by converting the datatypes of a pandas
    DataFrame withing required limits.

    Inputs=
    data= name of the dataframe

    Outputs=
    Memory reduced dataframe

    '''

    start_mem = data.memory_usage().sum() / 1024**2
    if verbose:
        print('-'*100)
        print('Memory usage of dataframe: {:.2f} MB'.format(start_mem))

    for col in data.columns:
        col_type = data[col].dtype

        if col_type != object:
            c_min = data[col].min()
            c_max = data[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    data[col] = data[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    data[col] = data[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    data[col] = data[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    data[col] = data[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    data[col] = data[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    data[col] = data[col].astype(np.float32)
                else:
                    data[col] = data[col].astype(np.float64)

    end_mem = data.memory_usage().sum() / 1024**2
    if verbose:
        print('Memory usage after optimization: {:.2f} MB'.format(end_mem))
        print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))
        print('-'*100)

    return data
```

In [ ]:
```
# Function to plot bar plot of percentage NaN values in a dataframe
def plot_nan_pct(dataframe,title):
  '''
  Function plots bar plot representing percentage of NaN values in a dataframe.
  This function first computes all the variable/columns of dataframe consisting of NaN
  values and then computes the corresponding percentage and plots that percentage values.

  Inputs=
  dataframe= DataFrame name
  title= data frame name entered as string (This will be used in bar plot title)

  Outputs=
  1. Bar plot ot percentage of NaN values in a dataframe
    '''

  nan_col_name=[]
  nan_val_count=[]
  nan_value_dict={}

  # prepering a dictionary of columns and correponding NaN percentage
  for i in range(dataframe.shape[1]):
    count=round(dataframe[dataframe[dataframe.columns[i]].isna()].shape[0]/dataframe.shape[0]*100,2)
    if count!=0:
      nan_value_dict[dataframe.columns[i]]=count
  # sorting the dict in reverse order and storing the column name and NaN percentage in lists
  for w in sorted(nan_value_dict, key=nan_value_dict.get, reverse=True):
    nan_val_count.append(nan_value_dict[w])
    nan_col_name.append(w)

  if len(nan_val_count)>0:
    print("Number of variables having NaN samples are ",len(nan_col_name))
    # generating the plot
    fig = plt.figure(figsize = (25, 5))

    # creating the bar plot
    plt.bar(nan_col_name, nan_val_count, color ='maroon')

    plt.xlabel("Variable Name")
    plt.ylabel("Percentage (%)")
    plt.title("Percentage of NaN values in "+title)
    plt.xticks(rotation = 90)
    plt.show()

  else:
    print("Dataframe {} does not have any NaN variable".format(title))
```

In [ ]:
```
# Lets remove the columns which has >=99% same column values
def drop_constant_column(dataf,freq_ratio=0.99):
  most_freq_col_df=pd.DataFrame(columns=["Column_Name","Item_Freq"])
  for col in range(0,dataf.columns.size):
    item_freq=dataf.iloc[:,col].value_counts().max()/dataf.shape[0]
    most_freq_col_df=most_freq_col_df.append(pd.DataFrame([[dataf.columns[col],item_freq]],
                                                          columns=["Column_Name","Item_Freq"]))
  remove_columns=most_freq_col_df[(most_freq_col_df["Item_Freq"]>freq_ratio)]["Column_Name"].values
  if remove_columns.tolist():
    print("Folowing columns are removed from the data: ")
    print(sorted(remove_columns.tolist()))
  else:
    print("All columns were keptas no columns with frequent value > {} is found".format(freq_ratio))

  # lets delete the most repeated column from the dataf
  dataf.drop(remove_columns,axis=1,inplace=True)

  return(dataf)
```

In [ ]:
```
# https://github.com/Sundar0989/WOE-and-IV?source=post_page-----6f05072e83eb--------------------------------
from numpy.lib import quantile
import pandas.core.algorithms as algos
from pandas import Series
import scipy.stats.stats as stats
import re
import traceback
import string

max_bin = 20
force_bin = 16

# define a binning function
def mono_bin(Y, X, n = max_bin):
    df1 = pd.DataFrame({"X": X, "Y": Y})
    justmiss = df1[['X','Y']][df1.X.isnull()]
    notmiss = df1[['X','Y']][df1.X.notnull()]
    r = 0
    while np.abs(r) < 1:
        try:
            d1 = pd.DataFrame({"X": notmiss.X, "Y": notmiss.Y, "Bucket": pd.qcut(notmiss.X, n)})
            d2 = d1.groupby('Bucket', as_index=True)
            r, p = stats.spearmanr(d2.mean().X, d2.mean().Y)
            n = n - 1
        except Exception as e:
            n = n - 1
    if len(d2) == 1:
        n = force_bin
        # bins = algos.quantile(notmiss.X, np.linspace(0, 1, n))
        bins = notmiss.X.quantile(np.linspace(0, 1, n))
        if len(np.unique(bins)) == 2:
```

```python
            # bins = np.insert(bins, 0, 1)
            bins=np.insert(np.asarray(bins), 0, 1)
            bins[1] = bins[1]-(bins[1]/2)
        d1 = pd.DataFrame({"X": notmiss.X, "Y": notmiss.Y, "Bucket": pd.cut(notmiss.X, np.unique(bins),include_lowest=True)})
        d2 = d1.groupby('Bucket', as_index=True)
    d3 = pd.DataFrame({},index=[])
    d3["MIN_VALUE"] = d2.min().X
    d3["MAX_VALUE"] = d2.max().X
    d3["COUNT"] = d2.count().Y
    d3["EVENT"] = d2.sum().Y
    d3["NONEVENT"] = d2.count().Y - d2.sum().Y
    d3=d3.reset_index(drop=True)

    if len(justmiss.index) > 0:
        d4 = pd.DataFrame({'MIN_VALUE':np.nan},index=[0])
        d4["MAX_VALUE"] = np.nan
        d4["COUNT"] = justmiss.count().Y
        d4["EVENT"] = justmiss.sum().Y
        d4["NONEVENT"] = justmiss.count().Y - justmiss.sum().Y
        d3 = d3.append(d4,ignore_index=True)

    d3["EVENT_RATE"] = d3.EVENT/d3.COUNT
    d3["NON_EVENT_RATE"] = d3.NONEVENT/d3.COUNT
    d3["DIST_EVENT"] = d3.EVENT/d3.sum().EVENT
    d3["DIST_NON_EVENT"] = d3.NONEVENT/d3.sum().NONEVENT
    d3["WOE"] = np.log(d3.DIST_EVENT/d3.DIST_NON_EVENT)
    d3["IV"] = (d3.DIST_EVENT-d3.DIST_NON_EVENT)*np.log(d3.DIST_EVENT/d3.DIST_NON_EVENT)
    d3["VAR_NAME"] = "VAR"
    d3 = d3[['VAR_NAME','MIN_VALUE', 'MAX_VALUE', 'COUNT', 'EVENT', 'EVENT_RATE', 'NONEVENT', 'NON_EVENT_RATE', 'DIST_EVENT','DIST_NON_EVEN
    d3 = d3.replace([np.inf, -np.inf], 0)
    d3.IV = d3.IV.sum()
    return(d3)

def char_bin(Y, X):

    df1 = pd.DataFrame({"X": X, "Y": Y})
    justmiss = df1[['X','Y']][df1.X.isnull()]
    notmiss = df1[['X','Y']][df1.X.notnull()]
    df2 = notmiss.groupby('X',as_index=True)

    d3 = pd.DataFrame({},index=[])
    d3["COUNT"] = df2.count().Y
    d3["MIN_VALUE"] = df2.sum().Y.index
    d3["MAX_VALUE"] = d3["MIN_VALUE"]
    d3["EVENT"] = df2.sum().Y
    d3["NONEVENT"] = df2.count().Y - df2.sum().Y

    if len(justmiss.index) > 0:
        d4 = pd.DataFrame({'MIN_VALUE':np.nan},index=[0])
        d4["MAX_VALUE"] = np.nan
        d4["COUNT"] = justmiss.count().Y
        d4["EVENT"] = justmiss.sum().Y
        d4["NONEVENT"] = justmiss.count().Y - justmiss.sum().Y
        d3 = d3.append(d4,ignore_index=True)

    d3["EVENT_RATE"] = d3.EVENT/d3.COUNT
    d3["NON_EVENT_RATE"] = d3.NONEVENT/d3.COUNT
    d3["DIST_EVENT"] = d3.EVENT/d3.sum().EVENT
    d3["DIST_NON_EVENT"] = d3.NONEVENT/d3.sum().NONEVENT
    d3["WOE"] = np.log(d3.DIST_EVENT/d3.DIST_NON_EVENT)
    d3["IV"] = (d3.DIST_EVENT-d3.DIST_NON_EVENT)*np.log(d3.DIST_EVENT/d3.DIST_NON_EVENT)
    d3["VAR_NAME"] = "VAR"
    d3 = d3[['VAR_NAME','MIN_VALUE', 'MAX_VALUE', 'COUNT', 'EVENT', 'EVENT_RATE', 'NONEVENT', 'NON_EVENT_RATE', 'DIST_EVENT','DIST_NON_EVEN
    d3 = d3.replace([np.inf, -np.inf], 0)
    d3.IV = d3.IV.sum()
    d3 = d3.reset_index(drop=True)

    return(d3)

def data_vars(df1, target):

    stack = traceback.extract_stack()
    filename, lineno, function_name, code = stack[-2]
    vars_name = re.compile(r'\((.*?)\).*$').search(code).groups()[0]
    final = (re.findall(r"[\w']+", vars_name))[-1]

    x = df1.columns
    count = -1

    for i in x:
        print('processing ',i)
        if i.upper() not in (final.upper()):
            if np.issubdtype(df1[i], np.number) and len(Series.unique(df1[i])) > 2:
                conv = mono_bin(target, df1[i])
                conv["VAR_NAME"] = i
                count = count + 1
            else:
                conv = char_bin(target, df1[i])
                conv["VAR_NAME"] = i
                count = count + 1

            if count == 0:
                iv_df = conv
            else:
                iv_df = iv_df.append(conv,ignore_index=True)

    iv = pd.DataFrame({'IV':iv_df.groupby('VAR_NAME').IV.max()})
```

```
    iv = iv.reset_index()
    return(iv_df,iv)
```

Loading and cleaning of merged dataframe

In [ ]:
```
# Importing the featurized dataframes
pickle_path = '/content/drive/MyDrive/Career/DS/Case Studies/Home Credit Default Risk/'

#1. Merged Application train dataframe
pickle_in=open(pickle_path+"application_train_merged.pickle","rb")
application_train_merged=pickle.load(pickle_in)
pickle_in.close()
print("Shape of featurized Application train merged dataframe is",application_train_merged.shape)
```

```
Shape of featurized Application train merged dataframe is (307511, 1772)
```

In [ ]:
```
# Lets check for missing values in the merged dataframe
plot_nan_pct(application_train_merged,"application_train_merged")
```

```
Dataframe application_train_merged does not have any NaN variable
```

In [ ]:
```
# Lets remove the columns from merged dataframe which has >99% frequency of a single value
# For the 8% default rate we can keep 99% threshold for constant column drop
print("Shape of the application_train_merged dataframe is ",application_train_merged.shape)
application_train_merged=drop_constant_column(application_train_merged,0.99)
print("Shape of the application_train_merged after droping constant columns is ",application_train_merged.shape)
```

```
Shape of the application_train_merged dataframe is  (307511, 1772)
Folowing columns are removed from the data:
['AMT_BALANCE_MAXCompleted', 'AMT_BALANCE_MAXRest', 'AMT_BALANCE_MEANCompleted', 'AMT_BALANCE_MEANRest', 'AMT_BALANCE_SUMCompleted', 'AMT_B
ALANCE_SUMRest', 'AMT_DRAWINGS_ATM_CURRENT_MAXCompleted', 'AMT_DRAWINGS_ATM_CURRENT_MAXRest', 'AMT_DRAWINGS_ATM_CURRENT_SUMCompleted', 'AMT
_DRAWINGS_ATM_CURRENT_SUMRest', 'AMT_DRAWINGS_CURRENT_MAXCompleted', 'AMT_DRAWINGS_CURRENT_MAXRest', 'AMT_DRAWINGS_CURRENT_SUMCompleted',
'AMT_DRAWINGS_CURRENT_SUMRest', 'AMT_DRAWINGS_OTHER_CURRENT_MAXCompleted', 'AMT_DRAWINGS_OTHER_CURRENT_MAXRest', 'AMT_DRAWINGS_OTHER_CURREN
T_MAX_Latest_year', 'AMT_DRAWINGS_OTHER_CURRENT_SUMCompleted', 'AMT_DRAWINGS_OTHER_CURRENT_SUMRest', 'AMT_DRAWINGS_OTHER_CURRENT_SUM_Latest
_year', 'AMT_DRAWINGS_POS_CURRENT_MAXCompleted', 'AMT_DRAWINGS_POS_CURRENT_MAXRest', 'AMT_DRAWINGS_POS_CURRENT_SUMCompleted', 'AMT_DRAWINGS
_POS_CURRENT_SUMRest', 'AMT_DRAWING_SUM_MAXCompleted', 'AMT_DRAWING_SUM_MAXRest', 'AMT_DRAWING_SUM_SUMCompleted', 'AMT_DRAWING_SUM_SUMRes
t', 'AMT_INST_MIN_REGULARITY_MAXCompleted', 'AMT_INST_MIN_REGULARITY_MAXRest', 'AMT_INST_MIN_REGULARITY_MEANCompleted', 'AMT_INST_MIN_REGUL
ARITY_MEANRest', 'AMT_INST_MIN_REGULARITY_MINCompleted', 'AMT_INST_MIN_REGULARITY_MINRest', 'AMT_INTEREST_RECEIVABLE_MEANCompleted', 'AMT_I
NTEREST_RECEIVABLE_MEANRest', 'AMT_INTEREST_RECEIVABLE_MINCompleted', 'AMT_INTEREST_RECEIVABLE_MINRest', 'AMT_OVERDUE_DURATION_LEFT_RATIO_M
AX', 'AMT_OVERDUE_DURATION_LEFT_RATIO_MEAN', 'AMT_PAYMENT_CURRENT_MAXCompleted', 'AMT_PAYMENT_CURRENT_MAXRest', 'AMT_PAYMENT_CURRENT_MEANCo
mpleted', 'AMT_PAYMENT_CURRENT_MEANRest', 'AMT_PAYMENT_CURRENT_MINCompleted', 'AMT_PAYMENT_CURRENT_MINRest', 'AMT_PAYMENT_TOTAL_CURRENT_MAX
Completed', 'AMT_PAYMENT_TOTAL_CURRENT_MAXRest', 'AMT_PAYMENT_TOTAL_CURRENT_MEANCompleted', 'AMT_PAYMENT_TOTAL_CURRENT_MEANRest', 'AMT_PAYM
ENT_TOTAL_CURRENT_MIN', 'AMT_PAYMENT_TOTAL_CURRENT_MINCompleted', 'AMT_PAYMENT_TOTAL_CURRENT_MINRest', 'AMT_RECEIVABLE_PRINCIPAL_MAXComplet
ed', 'AMT_RECEIVABLE_PRINCIPAL_MAXRest', 'AMT_RECEIVABLE_PRINCIPAL_MEANCompleted', 'AMT_RECEIVABLE_PRINCIPAL_MEANRest', 'AMT_RECEIVABLE_PRI
NCIPAL_SUMCompleted', 'AMT_RECEIVABLE_PRINCIPAL_SUMRest', 'AMT_RECIVABLE_MAXCompleted', 'AMT_RECIVABLE_MAXRest', 'AMT_RECIVABLE_MEANComplet
ed', 'AMT_RECIVABLE_MEANRest', 'AMT_RECIVABLE_SUMCompleted', 'AMT_RECIVABLE_SUMRest', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_H
OUR', 'AMT_TOTAL_RECEIVABLE_MAXCompleted', 'AMT_TOTAL_RECEIVABLE_MAXRest', 'AMT_TOTAL_RECEIVABLE_MEANCompleted', 'AMT_TOTAL_RECEIVABLE_MEAN
Rest', 'AMT_TOTAL_RECEIVABLE_SUMCompleted', 'AMT_TOTAL_RECEIVABLE_SUMRest', 'BALANCE_LIMIT_RATIO_MAXCompleted', 'BALANCE_LIMIT_RATIO_MAXRes
t', 'BALANCE_LIMIT_RATIO_MEANCompleted', 'BALANCE_LIMIT_RATIO_MEANRest', 'BALANCE_LIMIT_RATIO_MINCompleted', 'BALANCE_LIMIT_RATIO_MINRest',
'CNT_DRAWINGS_ATM_CURRENT_MAXCompleted', 'CNT_DRAWINGS_ATM_CURRENT_MAXRest', 'CNT_DRAWINGS_ATM_CURRENT_SUMCompleted', 'CNT_DRAWINGS_ATM_CUR
RENT_SUMRest', 'CNT_DRAWINGS_CURRENT_MAXCompleted', 'CNT_DRAWINGS_CURRENT_MAXRest', 'CNT_DRAWINGS_CURRENT_SUMCompleted', 'CNT_DRAWINGS_CURR
ENT_SUMRest', 'CNT_DRAWINGS_OTHER_CURRENT_MAXCompleted', 'CNT_DRAWINGS_OTHER_CURRENT_MAXRest', 'CNT_DRAWINGS_OTHER_CURRENT_MAX_Latest_yea
r', 'CNT_DRAWINGS_OTHER_CURRENT_SUMCompleted', 'CNT_DRAWINGS_OTHER_CURRENT_SUMRest', 'CNT_DRAWINGS_OTHER_CURRENT_SUM_Latest_year', 'CNT_DRA
WINGS_POS_CURRENT_MAXCompleted', 'CNT_DRAWINGS_POS_CURRENT_MAXRest', 'CNT_DRAWINGS_POS_CURRENT_SUMCompleted', 'CNT_DRAWINGS_POS_CURRENT_SUM
Rest', 'CNT_DRAWING_SUM_MAXRest', 'CNT_DRAWING_SUM_SUMRest', 'CNT_INSTALMENT_FUTURE_MAXCOMPLETED_MEAN', 'CNT_INSTALMENT_FUTURE_MEANCOMPLETE
D_MAX', 'CNT_INSTALMENT_FUTURE_MEANCOMPLETED_MEAN', 'CNT_INSTALMENT_FUTURE_MEANCOMPLETED_SUM', 'CNT_INSTALMENT_FUTURE_MINCOMPLETED_MEAN',
'CNT_INSTALMENT_MATURE_CUM_MAXRest', 'CNT_INSTALMENT_MATURE_CUM_MINRest', 'CNT_INSTALMENT_MATURE_CUM_SUMRest', 'CNT_PROLONGED_DURATION_RATI
O_MAX', 'CNT_PROLONGED_DURATION_RATIO_MEAN', 'CREDIT_CURRENCY_LOW_COUNT_MEAN', 'CREDIT_DAY_OVERDUE_MIN', 'CREDIT_DURATION_MAX', 'CREDIT_DUR
ATION_MEAN', 'CREDIT_ENDDATE_UPDATE_DIFF_MAX', 'CREDIT_ENDDATE_UPDATE_DIFF_MIN', 'CREDIT_TYPE_ANOTHER TYPE OF LOAN_MEAN', 'CREDIT_TYPE_LOAN
FOR BUSINESS DEVELOPMENT_MEAN', 'CREDIT_TYPE_LOW_COUNT_MEAN', 'CURRENT_AMT_OVERDUE_DURATION_RATIO_MAX', 'CURRENT_AMT_OVERDUE_DURATION_RATIO
_SUM', 'DAYS_CREDIT_ENDDATE_MAX', 'DAYS_CREDIT_ENDDATE_MEAN', 'DAYS_CREDIT_ENDDATE_MIN', 'DAYS_CREDIT_UPDATE_MEAN', 'DAYS_CREDIT_UPDATE_MI
N', 'DAYS_ENDDATE_FACT_MEAN', 'DAYS_ENDDATE_FACT_MIN', 'EMA_AMT_BALANCE_LASTRest', 'EMA_AMT_DRAWING_SUM_LASTRest', 'EMA_AMT_INTEREST_RECEIV
ABLE_LASTRest', 'EMA_AMT_RECEIVABLE_PRINCIPAL_LASTRest', 'EMA_AMT_RECIVABLE_LASTRest', 'EMA_AMT_TOTAL_RECEIVABLE_LASTRest', 'EMA_BALANCE_LI
MIT_RATIO_LASTRest', 'EMA_CNT_DRAWING_SUM_LASTRest', 'EMA_MIN_PAYMENT_RATIO_LASTRest', 'EMA_MIN_PAYMENT_TOTAL_RATIO_LASTRest', 'EMA_PAYMENT
_MIN_DIFF_LASTRest', 'EMA_SK_DPD_RATIO_LASTCompleted', 'EMA_SK_DPD_RATIO_LASTRest', 'FLAG_CONT_MOBILE', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_
11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT
_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_
9', 'FLAG_MOBIL', 'HOUSETYPE_MODE_specific housing', 'HOUSETYPE_MODE_terraced house', 'INTEREST_CREDIT_MAX_FIRST_2', 'INTEREST_CREDIT_MEAN_
FIRST_2', 'INTEREST_CREDIT_PRIVILEGED_MAX_FIRST_2', 'INTEREST_CREDIT_PRIVILEGED_MEAN_FIRST_2', 'INTEREST_CREDIT_PRIVILEGED_SUM_FIRST_2', 'I
NTEREST_CREDIT_SUM_FIRST_2', 'MAX_AMT_OVERDUE_DURATION_RATIO_MAX', 'MAX_AMT_OVERDUE_DURATION_RATIO_SUM', 'MIN_PAYMENT_RATIO_MEANCompleted',
'MIN_PAYMENT_RATIO_MEANRest', 'MIN_PAYMENT_RATIO_MINCompleted', 'MIN_PAYMENT_RATIO_MINRest', 'MIN_PAYMENT_TOTAL_RATIO_MEANCompleted', 'MIN_
PAYMENT_TOTAL_RATIO_MEANRest', 'MIN_PAYMENT_TOTAL_RATIO_MINCompleted', 'MIN_PAYMENT_TOTAL_RATIO_MINRest', 'NAME_CONTRACT_STATUS_AMORTIZED D
EBT_MEAN', 'NAME_CONTRACT_STATUS_Approved', 'NAME_CONTRACT_STATUS_CANCELED_MEAN', 'NAME_CONTRACT_STATUS_DEMAND_MEAN', 'NAME_CONTRACT_STATUS
_Demand', 'NAME_CONTRACT_STATUS_Refused', 'NAME_CONTRACT_STATUS_Sent proposal', 'NAME_CONTRACT_STATUS_XNA_MEAN', 'NAME_CONTRACT_TYPE_XNA_LA
ST_ALL', 'NAME_CONTRACT_TYPE_XNA_LAST_FIRST_2', 'NAME_CONTRACT_TYPE_XNA_LAST_LAST_5', 'NAME_CONTRACT_TYPE_XNA_MEAN_ALL', 'NAME_CONTRACT_TYP
E_XNA_MEAN_FIRST_2', 'NAME_CONTRACT_TYPE_XNA_MEAN_LAST_5', 'NAME_CONTRACT_TYPE_XNA_SUM_ALL', 'NAME_CONTRACT_TYPE_XNA_SUM_FIRST_2', 'NAME_CO
NTRACT_TYPE_XNA_SUM_LAST_5', 'NAME_FAMILY_STATUS_Unknown', 'NAME_INCOME_TYPE_Businessman', 'NAME_INCOME_TYPE_Maternity leave', 'NAME_INCOME
_TYPE_Student', 'NAME_INCOME_TYPE_Unemployed', 'NAME_PAYMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLOYER_LAST_ALL', 'NAME_PAYMENT_TYPE_C
ASHLESS FROM THE ACCOUNT OF THE EMPLOYER_LAST_FIRST_2', 'NAME_PAYMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLOYER_LAST_LAST_5', 'NAME_PA
YMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLOYER_MEAN_ALL', 'NAME_PAYMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLOYER_MEAN_FIRST_2',
'NAME_PAYMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLOYER_MEAN_LAST_5', 'NAME_PAYMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLOYER_SUM
_ALL', 'NAME_PAYMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLOYER_SUM_FIRST_2', 'NAME_PAYMENT_TYPE_CASHLESS FROM THE ACCOUNT OF THE EMPLO
YER_SUM_LAST_5', 'NAME_PAYMENT_TYPE_NON-CASH FROM YOUR ACCOUNT_LAST_ALL', 'NAME_PAYMENT_TYPE_NON-CASH FROM YOUR ACCOUNT_LAST_FIRST_2', 'NAM
E_PAYMENT_TYPE_NON-CASH FROM YOUR ACCOUNT_LAST_LAST_5', 'NAME_PAYMENT_TYPE_NON-CASH FROM YOUR ACCOUNT_MEAN_FIRST_2', 'NAME_PAYMENT_TYPE_NON
-CASH FROM YOUR ACCOUNT_SUM_FIRST_2', 'NAME_PORTFOLIO_CARS_LAST_ALL', 'NAME_PORTFOLIO_CARS_LAST_FIRST_2', 'NAME_PORTFOLIO_CARS_LAST_LAST_
5', 'NAME_PORTFOLIO_CARS_MEAN_ALL', 'NAME_PORTFOLIO_CARS_MEAN_FIRST_2', 'NAME_PORTFOLIO_CARS_MEAN_LAST_5', 'NAME_PORTFOLIO_CARS_SUM_ALL',
'NAME_PORTFOLIO_CARS_SUM_FIRST_2', 'NAME_PORTFOLIO_CARS_SUM_LAST_5', 'PAYMENT_MIN_DIFF_MEANCompleted', 'PAYMENT_MIN_DIFF_MEANRest', 'PAYMEN
T_MIN_DIFF_MINCompleted', 'PAYMENT_MIN_DIFF_MINRest', 'RATE_INTEREST_PRIMARY_MAX_FIRST_2', 'RATE_INTEREST_PRIMARY_MEAN_FIRST_2', 'RATE_INTE
REST_PRIVILEGED_MAX_FIRST_2', 'RATE_INTEREST_PRIVILEGED_MEAN_FIRST_2', 'SK_DPD_DEF_MAXCOMPLETED_MEAN', 'SK_DPD_DEF_MAXCompleted', 'SK_DPD_D
EF_MAXREST_MEAN', 'SK_DPD_DEF_MAXRest', 'SK_DPD_DEF_MAX_Latest_year', 'SK_DPD_DEF_SUMCOMPLETED_MEAN', 'SK_DPD_DEF_SUMCompleted', 'SK_DPD_DE
F_SUMREST_MEAN', 'SK_DPD_DEF_SUMRest', 'SK_DPD_DEF_SUM_Latest_year', 'SK_DPD_MAXCOMPLETED_MEAN', 'SK_DPD_MAXCompleted', 'SK_DPD_MAXREST_MEA
N', 'SK_DPD_MAXRest', 'SK_DPD_RATIO_MAXCOMPLETED_MEAN', 'SK_DPD_RATIO_MAXCompleted', 'SK_DPD_RATIO_MAXREST_MEAN', 'SK_DPD_RATIO_MAXRest',
'SK_DPD_RATIO_MEANCOMPLETED_MAX', 'SK_DPD_RATIO_MEANCOMPLETED_MEAN', 'SK_DPD_RATIO_MEANCOMPLETED_SUM', 'SK_DPD_RATIO_MEANCompleted', 'SK_DP
D_RATIO_MEANREST_MAX', 'SK_DPD_RATIO_MEANREST_MEAN', 'SK_DPD_RATIO_MEANREST_SUM', 'SK_DPD_RATIO_MEANRest', 'SK_DPD_SUMCOMPLETED_MEAN', 'SK_
DPD_SUMCompleted', 'SK_DPD_SUMREST_MEAN', 'SK_DPD_SUMRest', 'SMA_AMT_BALANCE_LASTCompleted', 'SMA_AMT_BALANCE_LASTRest', 'SMA_AMT_BALANCE_M
EANCompleted', 'SMA_AMT_BALANCE_MEANRest', 'SMA_AMT_CREDIT_LIMIT_ACTUAL_LASTRest', 'SMA_AMT_CREDIT_LIMIT_ACTUAL_MEANRest', 'SMA_AMT_DRAWING
_SUM_LASTCompleted', 'SMA_AMT_DRAWING_SUM_LASTRest', 'SMA_AMT_DRAWING_SUM_MEANRest', 'SMA_AMT_INTEREST_RECEIVABLE_LASTCompleted', 'SMA_AMT_
INTEREST_RECEIVABLE_LASTRest', 'SMA_AMT_INTEREST_RECEIVABLE_MEANCompleted', 'SMA_AMT_INTEREST_RECEIVABLE_MEANRest', 'SMA_AMT_RECEIVABLE_PRI
NCIPAL_LASTCompleted', 'SMA_AMT_RECEIVABLE_PRINCIPAL_LASTRest', 'SMA_AMT_RECEIVABLE_PRINCIPAL_MEANCompleted', 'SMA_AMT_RECEIVABLE_PRINCIPAL
_MEANRest', 'SMA_AMT_RECIVABLE_LASTCompleted', 'SMA_AMT_RECIVABLE_LASTRest', 'SMA_AMT_RECIVABLE_MEANCompleted', 'SMA_AMT_RECIVABLE_MEANRes
t', 'SMA_AMT_TOTAL_RECEIVABLE_LASTCompleted', 'SMA_AMT_TOTAL_RECEIVABLE_LASTRest', 'SMA_AMT_TOTAL_RECEIVABLE_MEANCompleted', 'SMA_AMT_TOTAL
_RECEIVABLE_MEANRest', 'SMA_BALANCE_LIMIT_RATIO_LASTCompleted', 'SMA_BALANCE_LIMIT_RATIO_LASTRest', 'SMA_BALANCE_LIMIT_RATIO_MEANComplete
d', 'SMA_BALANCE_LIMIT_RATIO_MEANRest', 'SMA_CNT_DRAWING_SUM_LASTRest', 'SMA_CNT_DRAWING_SUM_MEANRest', 'SMA_CNT_INSTALMENT_FUTURE_LASTREST
_MEAN', 'SMA_CNT_INSTALMENT_LASTREST_MEAN', 'SMA_MIN_PAYMENT_RATIO_LASTRest', 'SMA_MIN_PAYMENT_RATIO_MEANRest', 'SMA_MIN_PAYMENT_TOTAL_RATI
O_LASTRest', 'SMA_MIN_PAYMENT_TOTAL_RATIO_MEANRest', 'SMA_PAYMENT_MIN_DIFF_LASTCompleted', 'SMA_PAYMENT_MIN_DIFF_LASTRest', 'SMA_PAYMENT_MI
N_DIFF_MEANRest', 'SMA_SK_DPD_RATIO_LAST', 'SMA_SK_DPD_RATIO_LASTActive', 'SMA_SK_DPD_RATIO_LASTCompleted', 'SMA_SK_DPD_RATIO_LASTRest', 'S
MA_SK_DPD_RATIO_LAST_Latest_year', 'SMA_SK_DPD_RATIO_MEANCompleted', 'SMA_SK_DPD_RATIO_MEANRest', 'WMA_AMT_BALANCE_LASTCompleted', 'WMA_AMT
_BALANCE_LASTRest', 'WMA_AMT_BALANCE_MEANCompleted', 'WMA_AMT_BALANCE_MEANRest', 'WMA_AMT_CREDIT_LIMIT_ACTUAL_LASTRest', 'WMA_AMT_CREDIT_LI
MIT_ACTUAL_MEANRest', 'WMA_AMT_DRAWING_SUM_LASTCompleted', 'WMA_AMT_DRAWING_SUM_LASTRest', 'WMA_AMT_DRAWING_SUM_MEANRest', 'WMA_AMT_INTERES
T_RECEIVABLE_LASTCompleted', 'WMA_AMT_INTEREST_RECEIVABLE_LASTRest', 'WMA_AMT_INTEREST_RECEIVABLE_MEANCompleted', 'WMA_AMT_INTEREST_RECEIVA
BLE_MEANRest', 'WMA_AMT_RECEIVABLE_PRINCIPAL_LASTCompleted', 'WMA_AMT_RECEIVABLE_PRINCIPAL_LASTRest', 'WMA_AMT_RECEIVABLE_PRINCIPAL_MEANCom
pleted', 'WMA_AMT_RECEIVABLE_PRINCIPAL_MEANRest', 'WMA_AMT_RECIVABLE_LASTCompleted', 'WMA_AMT_RECIVABLE_LASTRest', 'WMA_AMT_RECIVABLE_MEANC
ompleted', 'WMA_AMT_RECIVABLE_MEANRest', 'WMA_AMT_TOTAL_RECEIVABLE_LASTCompleted', 'WMA_AMT_TOTAL_RECEIVABLE_LASTRest', 'WMA_AMT_TOTAL_RECE
IVABLE_MEANCompleted', 'WMA_AMT_TOTAL_RECEIVABLE_MEANRest', 'WMA_BALANCE_LIMIT_RATIO_LASTCompleted', 'WMA_BALANCE_LIMIT_RATIO_LASTRest', 'W
MA_BALANCE_LIMIT_RATIO_MEANCompleted', 'WMA_BALANCE_LIMIT_RATIO_MEANRest', 'WMA_CNT_DRAWING_SUM_LASTRest', 'WMA_CNT_DRAWING_SUM_MEANRest',
'WMA_CNT_INSTALMENT_FUTURE_LASTREST_MEAN', 'WMA_CNT_INSTALMENT_LASTREST_MEAN', 'WMA_MIN_PAYMENT_RATIO_LASTRest', 'WMA_MIN_PAYMENT_RATIO_MEA
NRest', 'WMA_MIN_PAYMENT_TOTAL_RATIO_LASTCompleted', 'WMA_MIN_PAYMENT_TOTAL_RATIO_LASTRest', 'WMA_MIN_PAYMENT_TOTAL_RATIO_MEANRest', 'WMA_P
AYMENT_MIN_DIFF_LASTCompleted', 'WMA_PAYMENT_MIN_DIFF_LASTRest', 'WMA_PAYMENT_MIN_DIFF_MEANRest', 'WMA_SK_DPD_RATIO_LAST', 'WMA_SK_DPD_RATI
O_LASTActive', 'WMA_SK_DPD_RATIO_LASTCompleted', 'WMA_SK_DPD_RATIO_LASTRest', 'WMA_SK_DPD_RATIO_LAST_Latest_year', 'WMA_SK_DPD_RATIO_MEANCo
mpleted', 'WMA_SK_DPD_RATIO_MEANRest']
Shape of the application_train_merged after droping constant columns is  (307511, 1418)
```

In [ ]:

```
#2. Merged Application test dataframe
pickle_in=open(pickle_path+"application_test_merged.pickle","rb")
```

```
application_test_merged=pickle.load(pickle_in)
pickle_in.close()
print("Shape of featurized Application test merged dataframe is",application_test_merged.shape)
```

Shape of featurized Application test merged dataframe is (48744, 1771)

In [ ]:
```
# Lets check for missing values in the merged dataframe
plot_nan_pct(application_test_merged,"application_test_merged")
```

Dataframe application_test_merged does not have any NaN variable

Splitting merged dataframe in train, test and oot

In [ ]:
```
train, test, oot = np.split(application_train_merged.sample(frac=1, random_state=42),
                            [int(.8*len(application_train_merged)), int(.9*len(application_train_merged))])

print('Shape of the trin data is ', train.shape)
print('Shape of the test data is ', test.shape)
print('Shape of the oot data is ', oot.shape)
```

```
Shape of the trin data is  (246008, 1418)
Shape of the test data is  (30751, 1418)
Shape of the oot data is  (30752, 1418)
```

Top 40 feature selection using Information Value

Calculating Weight of evidence and Information value

The weight of evidence tells the predictive power of a single feature concerning its independent feature. If any of the categories/bins of a feature has a large proportion of events compared to the proportion of non-events, we will get a high value of WoE which in turn says that that class of the feature separates the events from non-events. https://www.analyticsvidhya.com/blog/2021/06/understand-weight-of-evidence-and-information-value/

In [ ]:
```
final_iv, IV = data_vars(train,train.TARGET)
```

In [ ]:
```
IV.to_csv('/content/drive/MyDrive/Career/DS/Case Studies/Home Credit Default Risk/Information_Value.csv')
IV.sort_values(by=['IV'], ascending=False).head(10)
```

Out[ ]:

	VAR_NAME	IV
623	EXT_SOURCE_2	0.318135
624	EXT_SOURCE_3	0.246297
626	EXT_SOURCE_MEAN	0.199057
628	EXT_SOURCE_MUL	0.198053
1310	WEIGHTED_EXT_SOURCE	0.194873
627	EXT_SOURCE_MIN	0.177928
682	INCOME_EXT_RATIO	0.162892
625	EXT_SOURCE_MAX	0.137515
450	CREDIT_EXT_RATIO	0.130561
622	EXT_SOURCE_1	0.110553

Manual selection of variables based on Information Value:

38 variables are manually selected based on the information value. 0.04-0.5 information value threshold is used for selection.

In [ ]:
```
iv_selected_df=pd.read_csv('/content/drive/MyDrive/Career/DS/Case Studies/Home Credit Default Risk/Information_Value.csv')
iv_selected_df.drop(columns=['Unnamed: 0'],inplace=True)
iv_selected_df=iv_selected_df.reset_index(drop=True)
print("Following are the manually selected 39 variables :")
iv_selected_df[iv_selected_df['DECISION']=='KEEP']
```

Following are the manually selected 39 variables :

Out[ ]:

	VAR_NAME	IV	DECISION
41	EXT_SOURCE_2	0.318135	KEEP
63	EXT_SOURCE_3	0.246297	KEEP
173	EXT_SOURCE_MEAN	0.199057	KEEP
229	WEIGHTED_EXT_SOURCE	0.194873	KEEP
283	INCOME_EXT_RATIO	0.162892	KEEP
286	CREDIT_EXT_RATIO	0.130561	KEEP
316	EXT_SOURCE_1	0.110553	KEEP
318	DAYS_CREDIT_MEAN	0.108673	KEEP
319	CURRENT_DEBT_TO_CREDIT_RATIO_MEAN	0.106510	KEEP
342	DAYS_EMPLOYED	0.100511	KEEP
345	AMT_GOODS_PRICE	0.095468	KEEP
450	CREDIT_ACTIVE_CLOSED_MEAN	0.083690	KEEP
452	DAYS_BIRTH	0.083650	KEEP
459	EMPLOYED_TO_AGE_RATIO	0.083327	KEEP
463	CURRENT_CREDIT_DEBT_DIFF_MEAN	0.081329	KEEP
466	DAYS_PAYMENT_RATIO_MAX_Latest_year	0.076490	KEEP
476	NAME_CONTRACT_STATUS_MEAN_ALL	0.076209	KEEP
512	AMT_CREDIT_GOODS_RATIO_MEAN_ALL	0.064434	KEEP
534	BALANCE_LIMIT_RATIO_MAX_Latest_year	0.060782	KEEP
536	EMA_BALANCE_LIMIT_RATIO_LAST	0.056558	KEEP
558	ANNUITY_GOODS_MAX_ALL	0.056201	KEEP
573	DEF_60_CREDIT_RATIO	0.055447	KEEP
574	INTEREST_SHARE_MEAN_LAST_5	0.053174	KEEP
576	DAYS_DETAILS_CHANGE_MUL	0.052973	KEEP
626	AMT_CREDIT	0.052588	KEEP
628	EMA_AMT_PAYMENT_DIFF_LAST_Latest_year	0.052542	KEEP
707	AMT_PAYMENT_DIFF_MAX_Latest_year	0.050351	KEEP
709	NAME_EDUCATION_TYPE	0.048538	KEEP
715	CODE_REJECT_REASON_FREQ_ENCODE_MEAN_ALL	0.048142	KEEP
721	OCCUPATION_TYPE	0.047905	KEEP
779	INTEREST_RATE_MIN_ALL	0.047360	KEEP
782	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	0.044054	KEEP
826	NAME_INCOME_TYPE_Working	0.043330	KEEP
934	DAYS_LAST_PHONE_CHANGE	0.042783	KEEP
1021	PRODUCT_COMBINATION_FREQ_ENCODE_LAST_ALL	0.042641	KEEP
1023	REGION_POPULATION_RELATIVE	0.042427	KEEP
1042	NAME_PRODUCT_TYPE_WALK-IN_MEAN_ALL	0.041520	KEEP
1310	ORGANIZATION_TYPE	0.040641	KEEP

Train, Test and OOT dataframes consisting of only selected variables from IV:

In [ ]:

```
# List of selected variables based on IV value
iv_selected_var=iv_selected_df[iv_selected_df['DECISION']=='KEEP']['VAR_NAME'].to_list()
print('Total {} variables are selected based on IV value'.format(len(iv_selected_var)))

X_train=train[iv_selected_var]
X_test=test[iv_selected_var]
X_oot=oot[iv_selected_var]
# this is the test dataframe fromed from a test dataframe which home credit has provided
X_test_oot=application_test_merged[iv_selected_var]

y_train=train['TARGET']
y_test=test['TARGET']
y_oot=oot['TARGET']

print('Shape of X_train is {} and of y_train is {} '.format(X_train.shape,y_train.shape))
print('Shape of X_test is {} and of y_test is {} '.format(X_test.shape,y_test.shape))
print('Shape of X_oot is {} and of y_oot is {} '.format(X_oot.shape,y_oot.shape))
print('Shape of X_test_oot is {} '.format(X_test_oot.shape))
```

```
Total 38 variables are selected based on IV value
Shape of X_train is (246008, 38) and of y_train is (246008,)
Shape of X_test is (30751, 38) and of y_test is (30751,)
Shape of X_oot is (30752, 38) and of y_oot is (30752,)
Shape of X_test_oot is (48744, 38)
```

Min Max Scaling and SMOTE train data

In this section min max scaler is used to scale the data. Scaled data converges faster on the solution.

Min max scaling

Before using any algorithm lets use min max scaling to normalize the dataframes.

In [ ]:
```
# defining the minmaxscaler
scaler = MinMaxScaler()
# fit_transform on the train data
X_train=pd.DataFrame(scaler.fit_transform(X_train),columns=X_train.columns, index=X_train.index)
# transform on the test and oot data
X_test=pd.DataFrame(scaler.transform(X_test),columns=X_test.columns, index=X_test.index)
X_oot=pd.DataFrame(scaler.transform(X_oot),columns=X_oot.columns, index=X_oot.index)
X_test_oot=pd.DataFrame(scaler.transform(X_test_oot),columns=X_test_oot.columns, index=X_test_oot.index)
```

SMOTE:

As the data is imbalance lets use SMOTE technique to balance the data first before applying any machine learning algorithm. Lets use SMOTE data for train samples.

In [ ]:
```
# summarize class distribution
counter = Counter(y_train)
print('Existing data target distribution is ',counter)
# define pipeline
over = SMOTE(sampling_strategy=1)
under = RandomUnderSampler(sampling_strategy=1)
steps = [('o', over), ('u', under)]
pipeline = Pipeline(steps=steps)
# transform the dataset
X_train_smote, y_train_smote = pipeline.fit_resample(X_train, y_train)
# summarize the new class distribution
counter = Counter(y_train_smote)
print('After SMOTE data target distribution is ',counter)
```

```
Existing data target distribution is  Counter({0: 226136, 1: 19872})
After SMOTE data target distribution is  Counter({0: 226136, 1: 226136})
```

Lets observe the dataframes distribution

In [ ]:
```
X_train.describe()
```

Out[ ]:

	EXT_SOURCE_2	EXT_SOURCE_3	EXT_SOURCE_MEAN	WEIGHTED_EXT_SOURCE	INCOME_EXT_RATIO	CREDIT_EXT_RATIO	EXT_SOURCE_1	DAYS_CREDIT_MEAN	C
count	246008.000000	246008.000000	246008.000000	246008.000000	246008.000000	246008.000000	246008.000000	246008.000000	
mean	0.600253	0.457216	0.215458	0.215991	0.001046	0.001108	0.227318	0.317555	
std	0.224838	0.299115	0.303893	0.304752	0.014875	0.017125	0.296267	0.220632	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.456882	0.200681	0.000000	0.000000	0.000107	0.000076	0.000000	0.145192	
50%	0.661336	0.513079	0.000000	0.000000	0.000220	0.000220	0.000000	0.317420	
75%	0.775557	0.708447	0.526559	0.527076	0.000385	0.000432	0.473884	0.466461	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 38 columns

In [ ]:
```
X_train_smote.describe()
```

Out[ ]:

	EXT_SOURCE_2	EXT_SOURCE_3	EXT_SOURCE_MEAN	WEIGHTED_EXT_SOURCE	INCOME_EXT_RATIO	CREDIT_EXT_RATIO	EXT_SOURCE_1	DAYS_CREDIT_MEAN	C
count	452272.000000	452272.000000	452272.000000	452272.000000	452272.000000	452272.000000	452272.000000	452272.000000	
mean	0.545419	0.400233	0.185811	0.186203	0.001218	0.001243	0.194695	0.285838	
std	0.236487	0.288659	0.276850	0.277727	0.014507	0.016185	0.269694	0.211655	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.365778	0.126581	0.000000	0.000000	0.000120	0.000090	0.000000	0.112218	
50%	0.600228	0.417439	0.000000	0.000000	0.000250	0.000255	0.000000	0.278405	
75%	0.737794	0.649046	0.436865	0.435257	0.000460	0.000487	0.386156	0.428474	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 38 columns

In [ ]:
```
X_test.describe()
```

Out[ ]:

	EXT_SOURCE_2	EXT_SOURCE_3	EXT_SOURCE_MEAN	WEIGHTED_EXT_SOURCE	INCOME_EXT_RATIO	CREDIT_EXT_RATIO	EXT_SOURCE_1	DAYS_CREDIT_MEAN	C
count	30751.000000	30751.000000	30751.000000	30751.000000	30751.000000	30751.000000	30751.000000	30751.000000	
mean	0.599866	0.455973	0.217253	0.217651	0.001034	0.001053	0.229716	0.316871	
std	0.226577	0.299134	0.304788	0.305522	0.014173	0.015917	0.297475	0.219878	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.456882	0.196049	0.000000	0.000000	0.000108	0.000076	0.000000	0.142668	
50%	0.661907	0.513079	0.000000	0.000000	0.000222	0.000220	0.000000	0.317591	
75%	0.776414	0.708447	0.527425	0.527592	0.000388	0.000435	0.476420	0.466290	
max	0.961165	0.988011	1.009238	1.013925	0.592593	0.856185	0.979716	0.999316	

8 rows × 38 columns

In [ ]:
```
X_oot.describe()
```

Out[ ]:

	EXT_SOURCE_2	EXT_SOURCE_3	EXT_SOURCE_MEAN	WEIGHTED_EXT_SOURCE	INCOME_EXT_RATIO	CREDIT_EXT_RATIO	EXT_SOURCE_1	DAYS_CREDIT_MEAN	C
count	30752.000000	30752.000000	30752.000000	30752.000000	30752.000000	30752.000000	30752.00000	30752.000000	
mean	0.601621	0.457463	0.213945	0.214491	0.000986	0.001091	0.22638	0.318629	
std	0.224285	0.301016	0.303727	0.304704	0.013439	0.017224	0.29610	0.221822	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
25%	0.459452	0.193733	0.000000	0.000000	0.000106	0.000074	0.00000	0.144935	
50%	0.662193	0.513079	0.000000	0.000000	0.000218	0.000215	0.00000	0.317420	
75%	0.777841	0.712262	0.523383	0.524304	0.000378	0.000424	0.47211	0.466119	
max	1.000000	0.984741	0.993649	0.988138	0.666667	0.875000	0.97718	1.000000	

8 rows × 38 columns

Lets store the dataframes that has reduced variables based on IV value, has scaler transformed and also has train SMOTE. Lets also store scaler for future reference.

In [ ]:
```
# Lets store the dataframes to reuse
pickle_path = '/content/drive/MyDrive/Career/DS/Case Studies/Home Credit Default Risk/'
pickle_out=open(pickle_path+'X_train_iv.pickle', 'wb')
pickle.dump(X_train,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'X_train_smote_iv.pickle', 'wb')
pickle.dump(X_train_smote,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'X_test_iv.pickle', 'wb')
pickle.dump(X_test,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'X_oot_iv.pickle', 'wb')
pickle.dump(X_oot,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_train_iv.pickle', 'wb')
pickle.dump(y_train,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_train_smote_iv.pickle', 'wb')
pickle.dump(y_train_smote,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_test_iv.pickle', 'wb')
pickle.dump(y_test,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_oot_iv.pickle', 'wb')
pickle.dump(y_oot,pickle_out)
pickle_out.close()
```

In [ ]:
```
pickle_out=open(pickle_path+'MinMaxscaler.pickle', 'wb')
pickle.dump(scaler,pickle_out)
pickle_out.close()
```

In [ ]:
```
pickle_out=open(pickle_path+'X_test_oot_iv.pickle', 'wb')
pickle.dump(X_test_oot,pickle_out)
pickle_out.close()
```

In [ ]:
```
del X_train
del y_train
del X_train_smote
del y_train_smote
del X_test
```

```
del y_test
del X_oot
del y_oot
del X_test_oot
```

In [ ]:
```
# Importing the train test oot and train_smote dataframes
pickle_path = '/content/drive/MyDrive/Career/DS/Case Studies/Home Credit Default Risk/'

#1. X_train dataframe
pickle_in=open(pickle_path+"X_train_iv.pickle","rb")
X_train=pickle.load(pickle_in)
pickle_in.close()
print("Shape of X_train dataframe is",X_train.shape)

#2. y_train dataframe
pickle_in=open(pickle_path+"y_train_iv.pickle","rb")
y_train=pickle.load(pickle_in)
pickle_in.close()
print("Shape of y_train dataframe is",y_train.shape)

#3. X_train_smote dataframe
pickle_in=open(pickle_path+"X_train_smote_iv.pickle","rb")
X_train_smote=pickle.load(pickle_in)
pickle_in.close()
print("Shape of X_train_smote dataframe is",X_train_smote.shape)

#4. y_train_smote dataframe
pickle_in=open(pickle_path+"y_train_smote_iv.pickle","rb")
y_train_smote=pickle.load(pickle_in)
pickle_in.close()
print("Shape of y_train_smote dataframe is",y_train_smote.shape)

#5. X_test dataframe
pickle_in=open(pickle_path+"X_test_iv.pickle","rb")
X_test=pickle.load(pickle_in)
pickle_in.close()
print("Shape of X_test dataframe is",X_test.shape)

#6. y_test dataframe
pickle_in=open(pickle_path+"y_test_iv.pickle","rb")
y_test=pickle.load(pickle_in)
pickle_in.close()
print("Shape of y_test dataframe is",y_test.shape)

#7. X_oot dataframe
pickle_in=open(pickle_path+"X_oot_iv.pickle","rb")
X_oot=pickle.load(pickle_in)
pickle_in.close()
print("Shape of X_oot dataframe is",X_oot.shape)

#8. y_oot dataframe
pickle_in=open(pickle_path+"y_oot_iv.pickle","rb")
y_oot=pickle.load(pickle_in)
pickle_in.close()
print("Shape of y_oot dataframe is",y_oot.shape)

#9. X_test_oot dataframe
pickle_in=open(pickle_path+"X_test_oot_iv.pickle","rb")
X_test_oot=pickle.load(pickle_in)
pickle_in.close()
print("Shape of X_test_oot dataframe is",X_test_oot.shape)
```

```
Shape of X_train dataframe is (246008, 38)
Shape of y_train dataframe is (246008,)
Shape of X_train_smote dataframe is (452272, 38)
Shape of y_train_smote dataframe is (452272,)
Shape of X_test dataframe is (30751, 38)
Shape of y_test dataframe is (30751,)
Shape of X_oot dataframe is (30752, 38)
Shape of y_oot dataframe is (30752,)
Shape of X_test_oot dataframe is (48744, 38)
```

Selection of top 15 variables by votes count:

Now lets select the top 15 variables from manually selected 39 variables based on the information value. Lets count the votes based on the multiple algorithms.

1. RFE with Logisticregression

In [ ]:
```
def feature_selection_rfe_lr(train_x,train_y,feature_to_select=20,step=3):
  from sklearn.linear_model import LogisticRegression
  from sklearn.feature_selection import RFE
  rfe=RFE(estimator=LogisticRegression(), n_features_to_select=feature_to_select,step=step,verbose=10)
  rfe=rfe.fit(train_x,train_y.values.ravel())
  feature_importance_rfe=pd.DataFrame({'feature':train_x.columns,
                                       'ranking':rfe.ranking_,
                                       'RFE_LR_decisions':rfe.support_})
  return(feature_importance_rfe)
```

In [ ]:
```
fs_1=feature_selection_rfe_lr(X_train_smote,y_train_smote)
print('Top 20 features are:')
fs_1=fs_1.sort_values(by=['ranking'])
fs_1
```

```
Fitting estimator with 38 features.
Fitting estimator with 35 features.
Fitting estimator with 32 features.
Fitting estimator with 29 features.
Fitting estimator with 26 features.
Fitting estimator with 23 features.
Top 20 features are:
```

Out[ ]:

	feature	ranking	RFE_LR_decisions
0	EXT_SOURCE_2	1	True
23	DAYS_DETAILS_CHANGE_MUL	1	True
24	AMT_CREDIT	1	True
16	NAME_CONTRACT_STATUS_MEAN_ALL	1	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True
14	CURRENT_CREDIT_DEBT_DIFF_MEAN	1	True
25	EMA_AMT_PAYMENT_DIFF_LAST_Latest_year	1	True
12	DAYS_BIRTH	1	True
26	AMT_PAYMENT_DIFF_MAX_Latest_year	1	True
22	INTEREST_SHARE_MEAN_LAST_5	1	True
10	AMT_GOODS_PRICE	1	True
8	CURRENT_DEBT_TO_CREDIT_RATIO_MEAN	1	True
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	1	True
6	EXT_SOURCE_1	1	True
5	CREDIT_EXT_RATIO	1	True
3	WEIGHTED_EXT_SOURCE	1	True
2	EXT_SOURCE_MEAN	1	True
1	EXT_SOURCE_3	1	True
27	NAME_EDUCATION_TYPE	1	True
21	DEF_60_CREDIT_RATIO	1	True
35	REGION_POPULATION_RELATIVE	2	False
37	ORGANIZATION_TYPE	2	False
17	AMT_CREDIT_GOODS_RATIO_MEAN_ALL	2	False
7	DAYS_CREDIT_MEAN	3	False
33	DAYS_LAST_PHONE_CHANGE	3	False
20	ANNUITY_GOODS_MAX_ALL	3	False
13	EMPLOYED_TO_AGE_RATIO	4	False
9	DAYS_EMPLOYED	4	False
30	INTEREST_RATE_MIN_ALL	4	False
36	NAME_PRODUCT_TYPE_WALK-IN_MEAN_ALL	5	False
29	OCCUPATION_TYPE	5	False
32	NAME_INCOME_TYPE_Working	5	False
28	CODE_REJECT_REASON_FREQ_ENCODE_MEAN_ALL	6	False
4	INCOME_EXT_RATIO	6	False
18	BALANCE_LIMIT_RATIO_MAX_Latest_year	6	False
19	EMA_BALANCE_LIMIT_RATIO_LAST	7	False
11	CREDIT_ACTIVE_CLOSED_MEAN	7	False
34	PRODUCT_COMBINATION_FREQ_ENCODE_LAST_ALL	7	False

In [ ]:
```
# Lets store the fs_1
pickle_out=open(pickle_path+'fs_1.pickle', 'wb')
pickle.dump(fs_1,pickle_out)
pickle_out.close()
```

2. Correlation

In [ ]:
```
def feature_selection_stats_corr(train_x,train_y):
  return(train_x.corrwith(train_y).reset_index().rename(columns={'index':'feature',0:'correlation'}))
```

In [ ]:
```
fs_2=feature_selection_stats_corr(X_train_smote,y_train_smote)
print("Top features are:")
fs_2['Correlation_decision']=np.where(abs(fs_2['correlation'])>=0.071,True,False)
fs_2['correlation']=fs_2['correlation'].abs()
fs_2=fs_2.sort_values(by=['correlation'],ascending=False)
fs_2[fs_2['Correlation_decision']==True]
```

```
Top features are:
```

Out[ ]:

	feature	correlation	Correlation_decision
0	EXT_SOURCE_2	0.276816	True
1	EXT_SOURCE_3	0.234374	True
7	DAYS_CREDIT_MEAN	0.175907	True
12	DAYS_BIRTH	0.154611	True
11	CREDIT_ACTIVE_CLOSED_MEAN	0.148378	True
6	EXT_SOURCE_1	0.142267	True
23	DAYS_DETAILS_CHANGE_MUL	0.128967	True
3	WEIGHTED_EXT_SOURCE	0.127342	True
2	EXT_SOURCE_MEAN	0.127105	True
33	DAYS_LAST_PHONE_CHANGE	0.123175	True
16	NAME_CONTRACT_STATUS_MEAN_ALL	0.113167	True
27	NAME_EDUCATION_TYPE	0.107837	True
32	NAME_INCOME_TYPE_Working	0.103941	True
10	AMT_GOODS_PRICE	0.102654	True
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	0.099438	True
35	REGION_POPULATION_RELATIVE	0.093559	True
21	DEF_60_CREDIT_RATIO	0.087042	True
9	DAYS_EMPLOYED	0.085235	True
24	AMT_CREDIT	0.083425	True
13	EMPLOYED_TO_AGE_RATIO	0.080600	True
36	NAME_PRODUCT_TYPE_WALK-IN_MEAN_ALL	0.079963	True

In [ ]:

```
# Lets store the fs_2
pickle_out=open(pickle_path+'fs_2.pickle', 'wb')
pickle.dump(fs_2,pickle_out)
pickle_out.close()
```

3. Lasso

In [ ]:

```
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
L1_test = SelectFromModel(LogisticRegression(penalty='l1',solver='liblinear'))
L1_selector=L1_test.fit(X_train,y_train)
np.set_printoptions(precision=2)
L1_coef=np.round(L1_selector.estimator_.coef_,2).ravel()
L1_support=L1_selector.get_support()
L1_feature=X_train_smote.loc[:,L1_support].columns.tolist()
print(str(len(L1_feature)),'selected feature')
```

```
34 selected feature
```

In [ ]:

```
fs_3=pd.DataFrame({'feature':X_train_smote.columns,'Coeff_Lasso':L1_coef})
fs_3['Lasso_decision']=np.where(abs(fs_3['Coeff_Lasso'])<0.59,False,True)
print('selected features are',len(fs_3[fs_3['Lasso_decision']==True]['feature']))
fs_3[fs_3['Lasso_decision']==True]
```

```
selected features are 21
```

Out[ ]:

	feature	Coeff_Lasso	Lasso_decision
0	EXT_SOURCE_2	-1.94	True
1	EXT_SOURCE_3	-1.24	True
2	EXT_SOURCE_MEAN	5.60	True
3	WEIGHTED_EXT_SOURCE	-4.23	True
4	INCOME_EXT_RATIO	0.90	True
6	EXT_SOURCE_1	-1.77	True
8	CURRENT_DEBT_TO_CREDIT_RATIO_MEAN	-1.47	True
10	AMT_GOODS_PRICE	-10.45	True
12	DAYS_BIRTH	-0.60	True
13	EMPLOYED_TO_AGE_RATIO	-1.12	True
14	CURRENT_CREDIT_DEBT_DIFF_MEAN	-10.96	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	-2.67	True
16	NAME_CONTRACT_STATUS_MEAN_ALL	0.59	True
17	AMT_CREDIT_GOODS_RATIO_MEAN_ALL	-1.28	True
21	DEF_60_CREDIT_RATIO	-0.97	True
22	INTEREST_SHARE_MEAN_LAST_5	1.17	True
23	DAYS_DETAILS_CHANGE_MUL	0.59	True
24	AMT_CREDIT	10.47	True
25	EMA_AMT_PAYMENT_DIFF_LAST_Latest_year	1.90	True
26	AMT_PAYMENT_DIFF_MAX_Latest_year	8.53	True
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	3.45	True

In [ ]:

```
# Lets store the fs_3
pickle_out=open(pickle_path+'fs_3.pickle', 'wb')
pickle.dump(fs_3,pickle_out)
pickle_out.close()
```

4. RFE with Decisiontree

In [ ]:

```
def feature_selection_rfe_dt(train_x,train_y,feature_to_select=20,step=3):
  from sklearn import tree
  from sklearn.feature_selection import RFE
  from sklearn.tree import DecisionTreeClassifier
  rfe=RFE(estimator=DecisionTreeClassifier(), n_features_to_select=feature_to_select,step=step,verbose=10)
  rfe=rfe.fit(train_x,train_y.values.ravel())
  feature_importance_rfe=pd.DataFrame({'feature':train_x.columns,
                                       'ranking':rfe.ranking_,
                                       'RFE_DT_decisions':rfe.support_})
  return(feature_importance_rfe)
```

In [ ]:

```
fs_4=feature_selection_rfe_dt(X_train_smote,y_train_smote)
print('Top 20 features are:')
fs_4=fs_4.sort_values(by=['ranking'])
fs_4[fs_4['RFE_DT_decisions']==True]
```

```
Fitting estimator with 38 features.
Fitting estimator with 35 features.
Fitting estimator with 32 features.
Fitting estimator with 29 features.
Fitting estimator with 26 features.
Fitting estimator with 23 features.
Top 20 features are:
```

Out[ ]:

	feature	ranking	RFE_DT_decisions
0	EXT_SOURCE_2	1	True
35	REGION_POPULATION_RELATIVE	1	True
34	PRODUCT_COMBINATION_FREQ_ENCODE_LAST_ALL	1	True
33	DAYS_LAST_PHONE_CHANGE	1	True
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	1	True
30	INTEREST_RATE_MIN_ALL	1	True
29	OCCUPATION_TYPE	1	True
26	AMT_PAYMENT_DIFF_MAX_Latest_year	1	True
23	DAYS_DETAILS_CHANGE_MUL	1	True
22	INTEREST_SHARE_MEAN_LAST_5	1	True
21	DEF_60_CREDIT_RATIO	1	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True
13	EMPLOYED_TO_AGE_RATIO	1	True
37	ORGANIZATION_TYPE	1	True
1	EXT_SOURCE_3	1	True
5	CREDIT_EXT_RATIO	1	True
6	EXT_SOURCE_1	1	True
10	AMT_GOODS_PRICE	1	True
12	DAYS_BIRTH	1	True
7	DAYS_CREDIT_MEAN	1	True

In [ ]:

```
# Lets store the fs_4
pickle_out=open(pickle_path+'fs_4.pickle', 'wb')
pickle.dump(fs_4,pickle_out)
pickle_out.close()
```

5. RFE with Randomforest

In [ ]:

```
def feature_selection_rfe_rf(train_x,train_y,feature_to_select=20,step=3):
  from sklearn import tree
  from sklearn.feature_selection import RFE
  from sklearn.ensemble import RandomForestClassifier
  rfe=RFE(estimator=RandomForestClassifier(), n_features_to_select=feature_to_select,step=step,verbose=10)
  rfe=rfe.fit(train_x,train_y.values.ravel())
  feature_importance_rfe=pd.DataFrame({'feature':train_x.columns,
                                       'ranking':rfe.ranking_,
                                       'RFE_RF_decisions':rfe.support_})
  return(feature_importance_rfe)
```

In [ ]:

```
fs_5=feature_selection_rfe_rf(X_train_smote,y_train_smote)
print('Top 20 features are:')
fs_5=fs_5.sort_values(by=['ranking'])
fs_5[fs_5['RFE_RF_decisions']==True]
```

```
Fitting estimator with 38 features.
Fitting estimator with 35 features.
Fitting estimator with 32 features.
Fitting estimator with 29 features.
Fitting estimator with 26 features.
Fitting estimator with 23 features.
Top 20 features are:
```

Out[ ]:

	feature	ranking	RFE_RF_decisions
0	EXT_SOURCE_2	1	True
35	REGION_POPULATION_RELATIVE	1	True
34	PRODUCT_COMBINATION_FREQ_ENCODE_LAST_ALL	1	True
33	DAYS_LAST_PHONE_CHANGE	1	True
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	1	True
26	AMT_PAYMENT_DIFF_MAX_Latest_year	1	True
25	EMA_AMT_PAYMENT_DIFF_LAST_Latest_year	1	True
24	AMT_CREDIT	1	True
23	DAYS_DETAILS_CHANGE_MUL	1	True
22	INTEREST_SHARE_MEAN_LAST_5	1	True
21	DEF_60_CREDIT_RATIO	1	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True
13	EMPLOYED_TO_AGE_RATIO	1	True
37	ORGANIZATION_TYPE	1	True
11	CREDIT_ACTIVE_CLOSED_MEAN	1	True
1	EXT_SOURCE_3	1	True
10	AMT_GOODS_PRICE	1	True
7	DAYS_CREDIT_MEAN	1	True
12	DAYS_BIRTH	1	True
5	CREDIT_EXT_RATIO	1	True

In [ ]:
```
# Lets store the fs_5
pickle_out=open(pickle_path+'fs_5.pickle', 'wb')
pickle.dump(fs_5,pickle_out)
pickle_out.close()
```

6. RFE with GradientBoostingClassifier

In [ ]:
```
def feature_selection_rfe_gb(train_x,train_y,feature_to_select=20,step=3):
  from sklearn import tree
  from sklearn.feature_selection import RFE
  from sklearn.ensemble import GradientBoostingClassifier
  rfe=RFE(estimator=GradientBoostingClassifier(), n_features_to_select=feature_to_select,step=step,verbose=10)
  rfe=rfe.fit(train_x,train_y.values.ravel())
  feature_importance_rfe=pd.DataFrame({'feature':train_x.columns,
                                       'ranking':rfe.ranking_,
                                       'RFE_GB_decisions':rfe.support_})
  return(feature_importance_rfe)
```

In [ ]:
```
fs_6=feature_selection_rfe_gb(X_train_smote,y_train_smote)
print('Top 20 features are:')
fs_6=fs_6.sort_values(by=['ranking'])
fs_6[fs_6['RFE_GB_decisions']==True]
```

```
Fitting estimator with 38 features.
Fitting estimator with 35 features.
Fitting estimator with 32 features.
Fitting estimator with 29 features.
Fitting estimator with 26 features.
Fitting estimator with 23 features.
Top 20 features are:
```

Out[ ]:

	feature	ranking	RFE_GB_decisions
0	EXT_SOURCE_2	1	True
35	REGION_POPULATION_RELATIVE	1	True
34	PRODUCT_COMBINATION_FREQ_ENCODE_LAST_ALL	1	True
33	DAYS_LAST_PHONE_CHANGE	1	True
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	1	True
30	INTEREST_RATE_MIN_ALL	1	True
29	OCCUPATION_TYPE	1	True
28	CODE_REJECT_REASON_FREQ_ENCODE_MEAN_ALL	1	True
27	NAME_EDUCATION_TYPE	1	True
26	AMT_PAYMENT_DIFF_MAX_Latest_year	1	True
24	AMT_CREDIT	1	True
21	DEF_60_CREDIT_RATIO	1	True
36	NAME_PRODUCT_TYPE_WALK-IN_MEAN_ALL	1	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True
37	ORGANIZATION_TYPE	1	True
13	EMPLOYED_TO_AGE_RATIO	1	True
1	EXT_SOURCE_3	1	True
11	CREDIT_ACTIVE_CLOSED_MEAN	1	True
6	EXT_SOURCE_1	1	True
10	AMT_GOODS_PRICE	1	True

In [ ]:

```
# Lets store the fs_6
pickle_out=open(pickle_path+'fs_6.pickle', 'wb')
pickle.dump(fs_6,pickle_out)
pickle_out.close()
```

7. RFE with Perceptron

In [ ]:

```
def feature_selection_rfe_perceptron(train_x,train_y,feature_to_select=20,step=3):
  from sklearn import tree
  from sklearn.feature_selection import RFE
  from sklearn.linear_model import Perceptron
  rfe=RFE(estimator=Perceptron(), n_features_to_select=feature_to_select,step=step,verbose=10)
  rfe=rfe.fit(train_x,train_y.values.ravel())
  feature_importance_rfe=pd.DataFrame({'feature':train_x.columns,
                                       'ranking':rfe.ranking_,
                                       'RFE_Pe_decisions':rfe.support_})
  return(feature_importance_rfe)
```

In [ ]:

```
fs_7=feature_selection_rfe_perceptron(X_train_smote,y_train_smote)
print('Top 20 features are:')
fs_7=fs_7.sort_values(by=['ranking'])
fs_7[fs_7['RFE_Pe_decisions']==True]
```

```
Fitting estimator with 38 features.
Fitting estimator with 35 features.
Fitting estimator with 32 features.
Fitting estimator with 29 features.
Fitting estimator with 26 features.
Fitting estimator with 23 features.
Top 20 features are:
```

Out[ ]:

	feature	ranking	RFE_Pe_decisions
0	EXT_SOURCE_2	1	True
32	NAME_INCOME_TYPE_Working	1	True
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	1	True
26	AMT_PAYMENT_DIFF_MAX_Latest_year	1	True
25	EMA_AMT_PAYMENT_DIFF_LAST_Latest_year	1	True
24	AMT_CREDIT	1	True
22	INTEREST_SHARE_MEAN_LAST_5	1	True
20	ANNUITY_GOODS_MAX_ALL	1	True
19	EMA_BALANCE_LIMIT_RATIO_LAST	1	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True
14	CURRENT_CREDIT_DEBT_DIFF_MEAN	1	True
18	BALANCE_LIMIT_RATIO_MAX_Latest_year	1	True
5	CREDIT_EXT_RATIO	1	True
1	EXT_SOURCE_3	1	True
10	AMT_GOODS_PRICE	1	True
2	EXT_SOURCE_MEAN	1	True
3	WEIGHTED_EXT_SOURCE	1	True
4	INCOME_EXT_RATIO	1	True
8	CURRENT_DEBT_TO_CREDIT_RATIO_MEAN	1	True
6	EXT_SOURCE_1	1	True

In [ ]:

```
# Lets store the fs_7
pickle_out=open(pickle_path+'fs_7.pickle', 'wb')
pickle.dump(fs_7,pickle_out)
pickle_out.close()
```

8. ExtraTreeClassifier

In [ ]:

```
from sklearn.ensemble import ExtraTreesClassifier
model=ExtraTreesClassifier()
model.fit(X_train_smote,y_train_smote)
```

Out[ ]:

```
▾ ExtraTreesClassifier
ExtraTreesClassifier()
```

In [ ]:

```
fs_8= model
fs_8= pd.DataFrame({'feature':X_train_smote.columns,'F_imp':fs_8.feature_importances_})
fs_8['ExtraTreeClf_decision']=np.where(abs(fs_8['F_imp'])<0.026,False,True)
print('Selected variables are ',len(fs_8[fs_8['ExtraTreeClf_decision']==True]))
fs_8=fs_8.sort_values(by=['F_imp'])
fs_8[fs_8['ExtraTreeClf_decision']==True]
```

```
Selected variables are  20
```

Out[ ]:

	feature	F_imp	ExtraTreeClf_decision
22	INTEREST_SHARE_MEAN_LAST_5	0.026880	True
9	DAYS_EMPLOYED	0.027108	True
29	OCCUPATION_TYPE	0.027347	True
37	ORGANIZATION_TYPE	0.027674	True
16	NAME_CONTRACT_STATUS_MEAN_ALL	0.028811	True
23	DAYS_DETAILS_CHANGE_MUL	0.028854	True
6	EXT_SOURCE_1	0.029141	True
13	EMPLOYED_TO_AGE_RATIO	0.029663	True
21	DEF_60_CREDIT_RATIO	0.030872	True
7	DAYS_CREDIT_MEAN	0.032443	True
24	AMT_CREDIT	0.032450	True
33	DAYS_LAST_PHONE_CHANGE	0.033197	True
10	AMT_GOODS_PRICE	0.034004	True
34	PRODUCT_COMBINATION_FREQ_ENCODE_LAST_ALL	0.034419	True
35	REGION_POPULATION_RELATIVE	0.035913	True
11	CREDIT_ACTIVE_CLOSED_MEAN	0.037011	True
12	DAYS_BIRTH	0.037276	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	0.038571	True
1	EXT_SOURCE_3	0.058580	True
0	EXT_SOURCE_2	0.070455	True

In [ ]:

```
# Lets store the fs_8
pickle_out=open(pickle_path+'fs_8.pickle', 'wb')
pickle.dump(fs_8,pickle_out)
pickle_out.close()
```

9. Permutation Importance

In [ ]:

```
from sklearn.inspection import permutation_importance
fs_9=permutation_importance(model,X_train_smote,y_train_smote,n_repeats=5, random_state=0)
```

In [ ]:

```
fs_9= pd.DataFrame({'feature':X_train_smote.columns,'F_imp':fs_9.importances_mean})
fs_9['Permutation_decision']=np.where(abs(fs_9['F_imp'])<0.00035,False,True)
print('Selected variables are ',len(fs_9[fs_9['Permutation_decision']==True]))
fs_9=fs_9.sort_values(by=['F_imp'])
fs_9[fs_9['Permutation_decision']==True]
```

```
Selected variables are  21
```

Out[ ]:

	feature	F_imp	Permutation_decision
31	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	0.000367	True
36	NAME_PRODUCT_TYPE_WALK-IN_MEAN_ALL	0.000382	True
2	EXT_SOURCE_MEAN	0.000550	True
3	WEIGHTED_EXT_SOURCE	0.000651	True
15	DAYS_PAYMENT_RATIO_MAX_Latest_year	0.000790	True
23	DAYS_DETAILS_CHANGE_MUL	0.001102	True
34	PRODUCT_COMBINATION_FREQ_ENCODE_LAST_ALL	0.001141	True
28	CODE_REJECT_REASON_FREQ_ENCODE_MEAN_ALL	0.001566	True
16	NAME_CONTRACT_STATUS_MEAN_ALL	0.002141	True
35	REGION_POPULATION_RELATIVE	0.002818	True
9	DAYS_EMPLOYED	0.003610	True
33	DAYS_LAST_PHONE_CHANGE	0.005153	True
7	DAYS_CREDIT_MEAN	0.006074	True
30	INTEREST_RATE_MIN_ALL	0.007408	True
12	DAYS_BIRTH	0.016498	True
6	EXT_SOURCE_1	0.017723	True
11	CREDIT_ACTIVE_CLOSED_MEAN	0.024323	True
32	NAME_INCOME_TYPE_Working	0.026994	True
27	NAME_EDUCATION_TYPE	0.035727	True
1	EXT_SOURCE_3	0.078976	True
0	EXT_SOURCE_2	0.080720	True

In [ ]:
```
# Lets store the fs_9
pickle_out=open(pickle_path+'fs_9.pickle', 'wb')
pickle.dump(fs_9,pickle_out)
pickle_out.close()
```

Votes Counting for feature selection

In [ ]:
```
# Lets import all the votes dataframes
#1. fs_1
pickle_in=open(pickle_path+"fs_1.pickle","rb")
fs_1=pickle.load(pickle_in)
pickle_in.close()

#2. fs_2
pickle_in=open(pickle_path+"fs_2.pickle","rb")
fs_2=pickle.load(pickle_in)
pickle_in.close()

#3. fs_3
pickle_in=open(pickle_path+"fs_3.pickle","rb")
fs_3=pickle.load(pickle_in)
pickle_in.close()

#4. fs_4
pickle_in=open(pickle_path+"fs_4.pickle","rb")
fs_4=pickle.load(pickle_in)
pickle_in.close()

#5. fs_5
pickle_in=open(pickle_path+"fs_5.pickle","rb")
fs_5=pickle.load(pickle_in)
pickle_in.close()

#6. fs_6
pickle_in=open(pickle_path+"fs_6.pickle","rb")
fs_6=pickle.load(pickle_in)
pickle_in.close()

#7. fs_7
pickle_in=open(pickle_path+"fs_7.pickle","rb")
fs_7=pickle.load(pickle_in)
pickle_in.close()

#8. fs_8
pickle_in=open(pickle_path+"fs_8.pickle","rb")
fs_8=pickle.load(pickle_in)
pickle_in.close()

#9. fs_9
pickle_in=open(pickle_path+"fs_9.pickle","rb")
fs_9=pickle.load(pickle_in)
pickle_in.close()
```

In [ ]:
```
dfs=[fs_1,fs_2,fs_3,fs_4,fs_5,fs_6,fs_7,fs_8,fs_9]
Final_feature_selection=pd.concat([x.set_index('feature') for x in dfs],axis=1).reset_index()
Final_feature_selection.head(5)
```

Out[ ]:

	feature	ranking	RFE_LR_decisions	correlation	Correlation_decision	Coeff_Lasso	Lasso_decision	ranking	RFE_DT_decisions	ranking
0	EXT_SOURCE_2	1	True	0.276816	True	-1.94	True	1	True	1
1	DAYS_DETAILS_CHANGE_MUL	1	True	0.128967	True	0.59	True	1	True	1
2	AMT_CREDIT	1	True	0.083425	True	10.47	True	2	False	1
3	NAME_CONTRACT_STATUS_MEAN_ALL	1	True	0.113167	True	0.59	True	4	False	4
4	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True	0.038589	False	-2.67	True	1	True	1

In [ ]:
```
Final_feature_selection['votes']=np.sum([Final_feature_selection['RFE_LR_decisions'],
                                        Final_feature_selection['Correlation_decision'],
                                        Final_feature_selection['Lasso_decision'],
                                        Final_feature_selection['RFE_DT_decisions'],
                                        Final_feature_selection['RFE_RF_decisions'],
                                        Final_feature_selection['RFE_GB_decisions'],
                                        Final_feature_selection['RFE_Pe_decisions'],
                                        Final_feature_selection['ExtraTreeClf_decision'],
                                        Final_feature_selection['Permutation_decision']],axis=0)
```

In [ ]:
```
Final_feature_selection.head(5)
```

Out[ ]:

	feature	ranking	RFE_LR_decisions	correlation	Correlation_decision	Coeff_Lasso	Lasso_decision	ranking	RFE_DT_decisions	ranking
0	EXT_SOURCE_2	1	True	0.276816	True	-1.94	True	1	True	1
1	DAYS_DETAILS_CHANGE_MUL	1	True	0.128967	True	0.59	True	1	True	1
2	AMT_CREDIT	1	True	0.083425	True	10.47	True	2	False	1
3	NAME_CONTRACT_STATUS_MEAN_ALL	1	True	0.113167	True	0.59	True	4	False	4
4	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True	0.038589	False	-2.67	True	1	True	1

In [ ]:

```
selected_feature_df=Final_feature_selection.loc[Final_feature_selection['votes']>5]
selected_feature_df=selected_feature_df.sort_values(by=['votes'],ascending=False)
print('shape of selected feature dataframe is ',selected_feature_df.shape)
selected_feature_df.head(15)
```

```
shape of selected feature dataframe is  (15, 20)
```

Out[ ]:

	feature	ranking	RFE_LR_decisions	correlation	Correlation_decision	Coeff_Lasso	Lasso_decision	ranking	RFE_DT_decisions
0	EXT_SOURCE_2	1	True	0.276816	True	-1.94	True	1	True
17	EXT_SOURCE_3	1	True	0.234374	True	-1.24	True	1	True
4	DAYS_PAYMENT_RATIO_MAX_Latest_year	1	True	0.038589	False	-2.67	True	1	True
10	AMT_GOODS_PRICE	1	True	0.102654	True	-10.45	True	1	True
12	CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year	1	True	0.099438	True	3.45	True	1	True
13	EXT_SOURCE_1	1	True	0.142267	True	-1.77	True	1	True
1	DAYS_DETAILS_CHANGE_MUL	1	True	0.128967	True	0.59	True	1	True
2	AMT_CREDIT	1	True	0.083425	True	10.47	True	2	False
7	DAYS_BIRTH	1	True	0.154611	True	-0.60	True	1	True
19	DEF_60_CREDIT_RATIO	1	True	0.087042	True	-0.97	True	1	True
8	AMT_PAYMENT_DIFF_MAX_Latest_year	1	True	0.058890	False	8.53	True	1	True
9	INTEREST_SHARE_MEAN_LAST_5	1	True	0.041503	False	1.17	True	1	True
20	REGION_POPULATION_RELATIVE	2	False	0.093559	True	-0.20	False	1	True
24	DAYS_LAST_PHONE_CHANGE	3	False	0.123175	True	-0.17	False	1	True
26	EMPLOYED_TO_AGE_RATIO	4	False	0.080600	True	-1.12	True	1	True

In [ ]:

```
selected_variable=list(selected_feature_df['feature'])
print('Number of finally selected variables are',len(selected_variable))
selected_variable
```

```
Number of finally selected variables are 15
```

Out[ ]:

```
['EXT_SOURCE_2',
 'EXT_SOURCE_3',
 'DAYS_PAYMENT_RATIO_MAX_Latest_year',
 'AMT_GOODS_PRICE',
 'CNT_DRAWINGS_ATM_CURRENT_MAX_Latest_year',
 'EXT_SOURCE_1',
 'DAYS_DETAILS_CHANGE_MUL',
 'AMT_CREDIT',
 'DAYS_BIRTH',
 'DEF_60_CREDIT_RATIO',
 'AMT_PAYMENT_DIFF_MAX_Latest_year',
 'INTEREST_SHARE_MEAN_LAST_5',
 'REGION_POPULATION_RELATIVE',
 'DAYS_LAST_PHONE_CHANGE',
 'EMPLOYED_TO_AGE_RATIO']
```

Creating Final dataframes ready for modeling

In [ ]:

```
# creating final dataframes
X_train_final=X_train[selected_variable]
X_train_smote_final=X_train_smote[selected_variable]
X_test_final=X_test[selected_variable]
X_oot_final=X_oot[selected_variable]
X_test_oot_final=X_test_oot[selected_variable]

y_train_final=y_train.copy()
y_train_smote_final=y_train_smote.copy()
y_test_final=y_test.copy()
y_oot_final=y_oot.copy()

print('Shape of X_train_final is {} and of y_train_final is {} '.format(X_train_final.shape,y_train_final.shape))
print('Shape of X_train_smote_final is {} and of y_train_smote_final is {} '.format(X_train_smote_final.shape,y_train_smote_final.shape))
print('Shape of X_test_final is {} and of y_test_final is {} '.format(X_test_final.shape,y_test_final.shape))
print('Shape of X_oot_final is {} and of y_oot_final is {} '.format(X_oot_final.shape,y_oot_final.shape))
print('Shape of X_test_oot_final is {} '.format(X_test_oot_final.shape))
```

```
Shape of X_train_final is (246008, 15) and of y_train_final is (246008,)
Shape of X_train_smote_final is (452272, 15) and of y_train_smote_final is (452272,)
Shape of X_test_final is (30751, 15) and of y_test_final is (30751,)
Shape of X_oot_final is (30752, 15) and of y_oot_final is (30752,)
Shape of X_test_oot_final is (48744, 15)
```

In [ ]:

```
# Lets store the dataframes to reuse
pickle_path = '/content/drive/MyDrive/Career/DS/Case Studies/Home Credit Default Risk/'
pickle_out=open(pickle_path+'X_train_final.pickle', 'wb')
pickle.dump(X_train_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'X_train_smote_final.pickle', 'wb')
pickle.dump(X_train_smote_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'X_test_final.pickle', 'wb')
pickle.dump(X_test_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'X_oot_final.pickle', 'wb')
pickle.dump(X_oot_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'X_test_oot_final.pickle', 'wb')
pickle.dump(X_test_oot_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_train_final.pickle', 'wb')
pickle.dump(y_train_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_train_smote_final.pickle', 'wb')
pickle.dump(y_train_smote_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_test_final.pickle', 'wb')
pickle.dump(y_test_final,pickle_out)
pickle_out.close()

pickle_out=open(pickle_path+'y_oot_final.pickle', 'wb')
pickle.dump(y_oot_final,pickle_out)
pickle_out.close()
```

Summary:

Following steps are performed to select the top 15 features from the set of 1771 features:

1. Information value of all the features is calculated and 40 different features with high information value are selected
2. Different 9 algorithms are further developed on the SMOTE train data and important 20 features of each algorithm are selected.
3. For each feature, count of votes based on its importance in vaious 9 algorithms is counted. Amongst this, top 15 features were selected.