Wrangling Data Report

Table of Contents

In [ ]:

Introduction

Kowope Mart, a Nigerian-based retail company with a vision to provide quality goods, education and automobile services to its customers at affordable price and reduce if not eradicate charges on card payments and increase customer satisfaction with credit rewards that can be used within the Mall. Kowope Mart will like to have a system that profiles customers who are worthy of the card with minimum if not zero risk of defaulting.

The Goal is to predict customers who are likely to default or not.

Reference: https://zindi.africa/hackathons/dsn-ai-bootcamp-qualification-hackathon (https://zindi.africa/hackathons/dsn-ai-bootcamp-qualification-hackathon)

In [ ]:
```
pip install catboost
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.6/dist-packages (0.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from catboost) (1.15.0)
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (from catboost) (4.4.
1)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (from catboost) (0.1
0.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.6/dist-packages (from catboost)
(1.18.5)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.6/dist-packages (from catboos
t) (1.1.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from catboost)
(3.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from catboost) (1.4.1)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly-
>catboost) (1.3.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dist-packages (from
pandas>=0.24.0->catboost) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.
24.0->catboost) (2018.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matpl
otlib->catboost) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/di
st-packages (from matplotlib->catboost) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib
->catboost) (0.10.0)
```

In [ ]:
```

```

In [ ]:
```
#importing the relevant libraries and packages needed for the analysis

import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score
from sklearn.metrics import roc_auc_score
from catboost import CatBoostClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC

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

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/dr
ive", force_remount=True).
```

In [227]:
```
import os,sys
```

In [228]:
```
os.chdir('/content/drive/My Drive/DSN_2020')
```

In [228]:
```

```

In [228]:
```

```

Type *Markdown* and LaTeX: α^2

In [472]:
```
#Using pandas to get the train and test datasets and inspecting the dataframes. Next I'll carry out data
# to gain more insights in the data

df = pd.read_csv('Train.csv')
df1 = pd.read_csv('Test.csv')
df.head()
```

Out[472]:

	Applicant_ID	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	form_fie
0	Apcnt_1000000	3436.0	0.28505	1.6560	0.0	0.000	0.0	10689720.0	252072.0	4272776.0	113331
1	Apcnt_1000004	3456.0	0.67400	0.2342	0.0	0.000	0.0	898979.0	497531.0	9073814.0	25331
2	Apcnt_1000008	3276.0	0.53845	3.1510	0.0	6.282	NaN	956940.0	NaN	192944.0	10798
3	Apcnt_1000012	3372.0	0.17005	0.5050	0.0	0.000	192166.0	3044703.0	385499.0	3986472.0	36219
4	Apcnt_1000016	3370.0	0.77270	1.1010	0.0	0.000	1556.0	214728.0	214728.0	1284089.0	3617

In [473]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56000 entries, 0 to 55999
Data columns (total 52 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Applicant_ID    56000 non-null  object 
 1   form_field1     53471 non-null  float64
 2   form_field2     52156 non-null  float64
 3   form_field3     55645 non-null  float64
 4   form_field4     55645 non-null  float64
 5   form_field5     55645 non-null  float64
 6   form_field6     42640 non-null  float64
 7   form_field7     50837 non-null  float64
 8   form_field8     42640 non-null  float64
 9   form_field9     47992 non-null  float64
 10  form_field10    55645 non-null  float64
 11  form_field11    24579 non-null  float64
 12  form_field12    46105 non-null  float64
 13  form_field13    50111 non-null  float64
 14  form_field14    56000 non-null  int64  
 15  form_field15    33525 non-null  float64
 16  form_field16    42964 non-null  float64
 17  form_field17    44849 non-null  float64
 18  form_field18    45598 non-null  float64
 19  form_field19    55996 non-null  float64
 20  form_field20    55645 non-null  float64
 21  form_field21    40146 non-null  float64
 22  form_field22    35600 non-null  float64
 23  form_field23    27877 non-null  float64
 24  form_field24    42703 non-null  float64
 25  form_field25    50550 non-null  float64
 26  form_field26    48562 non-null  float64
 27  form_field27    46701 non-null  float64
 28  form_field28    55645 non-null  float64
 29  form_field29    55645 non-null  float64
 30  form_field30    30491 non-null  float64
 31  form_field31    16592 non-null  float64
 32  form_field32    50550 non-null  float64
 33  form_field33    54744 non-null  float64
 34  form_field34    55645 non-null  float64
 35  form_field35    32852 non-null  float64
```

```
 36  form_field36    54005 non-null  float64
 37  form_field37    50550 non-null  float64
 38  form_field38    55645 non-null  float64
 39  form_field39    51789 non-null  float64
 40  form_field40    12271 non-null  float64
 41  form_field41    17771 non-null  float64
 42  form_field42    54677 non-null  float64
 43  form_field43    55432 non-null  float64
 44  form_field44    50617 non-null  float64
 45  form_field45    24683 non-null  float64
 46  form_field46    40096 non-null  float64
 47  form_field47    56000 non-null  object
 48  form_field48    35111 non-null  float64
 49  form_field49    55645 non-null  float64
 50  form_field50    44944 non-null  float64
 51  default_status  56000 non-null  object
dtypes: float64(48), int64(1), object(3)
memory usage: 22.2+ MB
```

In [474]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24000 entries, 0 to 23999
Data columns (total 51 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Applicant_ID   24000 non-null  object 
 1   form_field1    22890 non-null  float64
 2   form_field2    22291 non-null  float64
 3   form_field3    23854 non-null  float64
 4   form_field4    23854 non-null  float64
 5   form_field5    23854 non-null  float64
 6   form_field6    18396 non-null  float64
 7   form_field7    21769 non-null  float64
 8   form_field8    18396 non-null  float64
 9   form_field9    20600 non-null  float64
 10  form_field10   23853 non-null  float64
 11  form_field11   10602 non-null  float64
 12  form_field12   19817 non-null  float64
 13  form_field13   21537 non-null  float64
 14  form_field14   24000 non-null  int64  
 15  form_field15   14408 non-null  float64
 16  form_field16   18526 non-null  float64
 17  form_field17   19305 non-null  float64
 18  form_field18   19631 non-null  float64
 19  form_field19   24000 non-null  float64
 20  form_field20   23853 non-null  float64
 21  form_field21   17293 non-null  float64
 22  form_field22   15276 non-null  float64
 23  form_field23   11875 non-null  float64
 24  form_field24   18395 non-null  float64
 25  form_field25   21744 non-null  float64
 26  form_field26   20828 non-null  float64
 27  form_field27   20090 non-null  float64
 28  form_field28   23853 non-null  float64
 29  form_field29   23853 non-null  float64
 30  form_field30   13092 non-null  float64
 31  form_field31   7190 non-null   float64
 32  form_field32   21744 non-null  float64
 33  form_field33   23505 non-null  float64
 34  form_field34   23853 non-null  float64
 35  form_field35   14134 non-null  float64
```

```
 36  form_field36  23097 non-null  float64
 37  form_field37  21744 non-null  float64
 38  form_field38  23853 non-null  float64
 39  form_field39  22171 non-null  float64
 40  form_field40  5172 non-null   float64
 41  form_field41  7651 non-null   float64
 42  form_field42  23422 non-null  float64
 43  form_field43  23750 non-null  float64
 44  form_field44  21638 non-null  float64
 45  form_field45  10462 non-null  float64
 46  form_field46  17115 non-null  float64
 47  form_field47  24000 non-null  object
 48  form_field48  15078 non-null  float64
 49  form_field49  23854 non-null  float64
 50  form_field50  19203 non-null  float64
dtypes: float64(48), int64(1), object(2)
memory usage: 9.3+ MB
```

In [475]:
```
#Creating a mapping of the column names to their description for easier analysis.


dict_ = {'Applicant_ID':'id', 'form_field1':'credit_score','form_field2':'credit_risk',
         'form_field3':'other_loan_default','form_field4':'auto_loan_default',
         'form_field5':'education_loan_default','form_field6':'min_credit',
         'form_field7':'max_credit_lines','form_field8':'max_credit','form_field9':'credit_sum',
         'form_field10':'total_amtcredit_line','form_field11':'amt_dues_postdefault',
         'form_field12':'amount_due','form_field13':'amount_paid_last_year',
         'form_field14':'income','form_field15':'property_val',
         'form_field16':'active_cards', 'form_field17':'no_active_cards','form_field18':'no_active_lines
         'form_field19':'active_cards_75_percent','form_field20':'active_lines_75_percent',
         'form_field21':'active_card_loan_use','form_field22':'avg_use_lines2yrs',
         'form_field23':'active_cards_last_year','form_field24':'card_use_defaulted1',
         'form_field25':'average_tenure_active_cards','form_field26':'oldest_card_tenure',
         'form_field27':'oldest_revolving_card_tenure','form_field28':'no_days_defaulted',
         'form_field29':'oldest_line_tenure','form_field30':'max_autoloan_tenure',
         'form_field31':'max_edu_loan_tenure','form_field32':'sum_of_active_cards_tenure',
         'form_field33':'sum_of_active_cards_tenure','form_field34':'active_cards_missed1',
         'form_field35':'no_revolvncards_2yrs_missed1','form_field36':'no_active_lines',
         'form_field37':'no_cards_last_two_years','form_field38':'no_line_last_two_years',
         'form_field39':'delinquent_lines','form_field40':'use_line_edu_loans',
         'form_field41':'use_line_auto_loans','form_field42':'bankrupt_index',
         'form_field43':'high_risk_loan','form_field44':'ratio_max_to_sum_amount_due',
         'form_field45':'no_mortagage_loan_missed2','form_field46':'auto_defaulted2',
         'form_field47':'product_type','form_field48':'unknown_var1',
         'form_field49':'unknown_var2','form_field50':'ratio_min_to_sum_amount_due',
         'default_status':'defaulted_or_not'
}
```

Columns Description

- id: Unique Customer Application Identification number
- credit_score: Customer Creditworthiness score based on historical data
- credit_risk: A score that measures the number and riskiness of credit enquiries made by a borrower.
- other_loan_default: Severity of default by the borrower on any loan(s).
- auto_loan_default: Severity of default by the borrower on auto loan(s).
- education_loan_default: Severity of default by the borrower on education loan(s).

- min_credit: Minimum of credit available on all credit cards that is automatically renewed as debts are paid off on the customer's cards (in NGN)
- max_credit_lines:Maximum of credit available on customer's active credit lines (in NGN)
- max_credit: Maximum of credit available on all active credit cards that is automatically renewed as debts are paid off on the customer's cards (in NGN)
- credit_sum: Sum of available credit on credit cards that the borrower has missed 1 payment (in NGN)
- total_amtcredit_line: Total amount of credit available on accepted credit lines (in NGN)
- amt_dues_postdefault: The amount of dues collected post-default where the due amount was more than 500 (in NGN)
- amount_due: Sum of the amount due on active credit cards (in NGN)
- amount_paid_last_year: Annual amount paid towards all credit cards during the previous year (in NGN)
- income: Annual income (in NGN)
- property_val: The estimated market value of a property owned/used by the borrower (in NGN)
- active_cards: Number of active credit card that is automatically renewed as debts are paid off on which full credit limit is utilized by the borrower
- no_active_cards:Number of active credit cards on which full credit limit is utilized by the borrower
- no_active_lines: Number of active credit lines on which full credit limit is utilized by the borrower
- active_cards_75_percent:Number of active credit cards on which at least 75% credit limit is utilized by the borrower
- active_lines_75_percent: Number of active credit lines on which at least 75% credit limit is utilized by the borrower
- active_card_loan_use:Average utilization of active revolving credit card loans (%)
- avg_use_lines2yrs: Average utilization of line on all active credit lines activated in last 2 years (%)
- active_cards_last_year:Average utilization of line on all active credit cards activated in last 1 year (%)
- card_use_defaulted1: Average utilization of line on credit cards on which the borrower has missed 1 payment during the last 6 months (%)
- average_tenure_active_cards: Average tenure of active revolving credit cards (in days)
- oldest_card_tenure: Tenure of oldest credit card among all active credit cards (in days)
- oldest_revolving_card_tenure: Tenure of oldest revolving credit card among all active revolving credit cards (in days)
- no_days_defaulted: Number of days since last missed payment on any credit line
- no_days_defaulted: Tenure of the oldest credit line (in days)
- max_autoloan_tenure: Maximum tenure on all auto loans (in days)
- max_edu_loan_tenure: Maximum tenure on all education loans (in days)
- sum_of_active_cards_tenure:Sum of tenures (in months) of active credit cards
- sum_of_active_cards_tenure: Sum of tenures (in months) of active credit cards
- active_cards_missed1: Number of active credit lines over the last 6 months on which the borrower has missed 1 payment
- no_revolvncards_2yrs_missed1: Number of revolving credit cards over the last 2 years on which the borrower has missed 1 payment
- no_active_lines: Number of active credit lines
- no_cards_last_two_years: Number of credit cards with an active tenure of at least 2 years
- no_line_last_two_years: Number of credit lines activated in the last 2 years

- delinquent_lines: Number of credit lines on which the borrower has current delinquency
- use_line_edu_loans: Utilization of line on active education loans (%)
- use_line_auto_loans: Utilization of line on active auto loans (%)
- bankrupt_index: Financial stress index of the borrower. This index is a function of collection trades, bankruptcies files, tax liens invoked, etc.
- high_risk_loan: Number of credit lines on which the borrower has never missed a payment in the last 2 years, yet considered as high-risk loans based on the market prediction of the economic scenario
- ratio_max_to_sum_amount_due: Ratio of the maximum amount due on all active credit lines and the sum of amounts due on all active credit lines
- no_mortagage_loan_missed2: Number of mortgage loans on which the borrower has missed 2 payments
- auto_defaulted2: Number of auto loans on which the borrower has missed 2 payments
- product_type: Type of product that the applicant applied for. (C = Charge; L = Lending)
- unknown_var1: Undefined Variable
- unknown_var2: Undefined Variable
- ratio_min_to_sum_amount_due: Ratio of the minimum amount due on all active credit lines and the sum of amounts due on all active credit lines
- default_status: defaulted or not. (yes:1, no: 0)

Wrangling

- In this section, I'm going to examine the datasets checking the table for data quality and tidyness issues and highlighting them for cleaning.

In [476]:
```
#null values in the train dataset

df.isnull().sum()
```

Out[476]:
```
Applicant_ID          0
form_field1        2529
form_field2        3844
form_field3         355
form_field4         355
form_field5         355
form_field6       13360
form_field7        5163
form_field8       13360
form_field9        8008
form_field10        355
form_field11      31421
form_field12       9895
form_field13       5889
form_field14          0
form_field15      22475
form_field16      13036
form_field17      11151
form_field18      10402
form_field19          4
form_field20        355
form_field21      15854
form_field22      20400
form_field23      28123
form_field24      13297
form_field25       5450
form_field26       7438
form_field27       9299
form_field28        355
form_field29        355
form_field30      25509
form_field31      39408
form_field32       5450
form_field33       1256
form_field34        355
form_field35      23148
form_field36       1995
form_field37       5450
form_field38        355
```

```
form_field39       4211
form_field40      43729
form_field41      38229
form_field42       1323
form_field43        568
form_field44       5383
form_field45      31317
form_field46      15904
form_field47          0
form_field48      20889
form_field49        355
form_field50      11056
default_status        0
dtype: int64
```

In [477]:
```
#null values in the test dataset
df1.isnull().sum()
```

Out[477]:
```
Applicant_ID        0
form_field1      1110
form_field2      1709
form_field3       146
form_field4       146
form_field5       146
form_field6      5604
form_field7      2231
form_field8      5604
form_field9      3400
form_field10      147
form_field11    13398
form_field12     4183
form_field13     2463
form_field14        0
form_field15     9592
form_field16     5474
form_field17     4695
form_field18     4369
form_field19        0
form_field20      147
form_field21     6707
form_field22     8724
form_field23    12125
form_field24     5605
form_field25     2256
form_field26     3172
form_field27     3910
form_field28      147
form_field29      147
form_field30    10908
form_field31    16810
form_field32     2256
form_field33      495
form_field34      147
form_field35     9866
form_field36      903
form_field37     2256
form_field38      147
form_field39     1829
```

```
form_field40    18828
form_field41    16349
form_field42      578
form_field43      250
form_field44     2362
form_field45    13538
form_field46     6885
form_field47        0
form_field48     8922
form_field49      146
form_field50     4797
dtype: int64
```

In [477]:

In [478]:
```
#using the describe method to view some basic
#statistical details like percentile, mean, std etc. of the numeric values of the data frames

df.describe()
```

Out[478]:

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	
count	53471.000000	52156.000000	55645.000000	55645.000000	55645.000000	4.264000e+04	5.083700e+04	4.264000e+04	4.799200e+04	5
mean	3491.795665	0.550737	1.052225	0.851979	1.956317	6.244479e+05	6.865210e+06	2.626690e+06	1.316002e+07	1
std	188.462426	0.820979	2.147768	3.157692	10.512396	1.433422e+06	1.912729e+07	3.927355e+06	1.977963e+07	2
min	2990.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
25%	3358.000000	0.070788	0.000000	0.000000	0.000000	1.400400e+04	6.869740e+05	1.929440e+05	1.368502e+06	4
50%	3484.000000	0.267575	0.062000	0.000000	0.000000	1.155330e+05	2.704328e+06	9.639420e+05	5.506295e+06	3
75%	3620.000000	0.719512	1.282000	0.000000	0.000000	5.259280e+05	6.993831e+06	3.751516e+06	1.694552e+07	1
max	3900.000000	18.015050	57.371600	91.672200	407.748600	5.313546e+07	2.158794e+09	1.037397e+08	3.200533e+08	2

In [479]:
```
df1.describe()
```

Out[479]:

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	
count	22890.000000	22291.000000	23854.000000	23854.000000	23854.000000	1.839600e+04	2.176900e+04	1.839600e+04	2.060000e+04	2
mean	3492.284404	0.557676	1.065443	0.859146	2.183538	6.263036e+05	6.797033e+06	2.654142e+06	1.350593e+07	1
std	190.502764	0.826543	2.198444	3.403115	11.415706	1.457540e+06	1.626022e+07	3.968185e+06	2.289125e+07	2
min	2986.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
25%	3356.000000	0.068675	0.000000	0.000000	0.000000	1.400400e+04	6.725810e+05	1.816630e+05	1.349441e+06	4
50%	3484.000000	0.273250	0.058200	0.000000	0.000000	1.155330e+05	2.719888e+06	9.594685e+05	5.529830e+06	3
75%	3624.000000	0.728850	1.304250	0.000000	0.000000	5.159112e+05	7.073576e+06	3.799849e+06	1.728658e+07	1
max	3900.000000	22.315050	34.541400	206.452800	297.885600	4.818738e+07	7.709887e+08	1.135141e+08	1.443921e+09	7

In [480]:
```
#checking for duplicated values

sum(df.Applicant_ID.duplicated())
```

Out[480]: 0

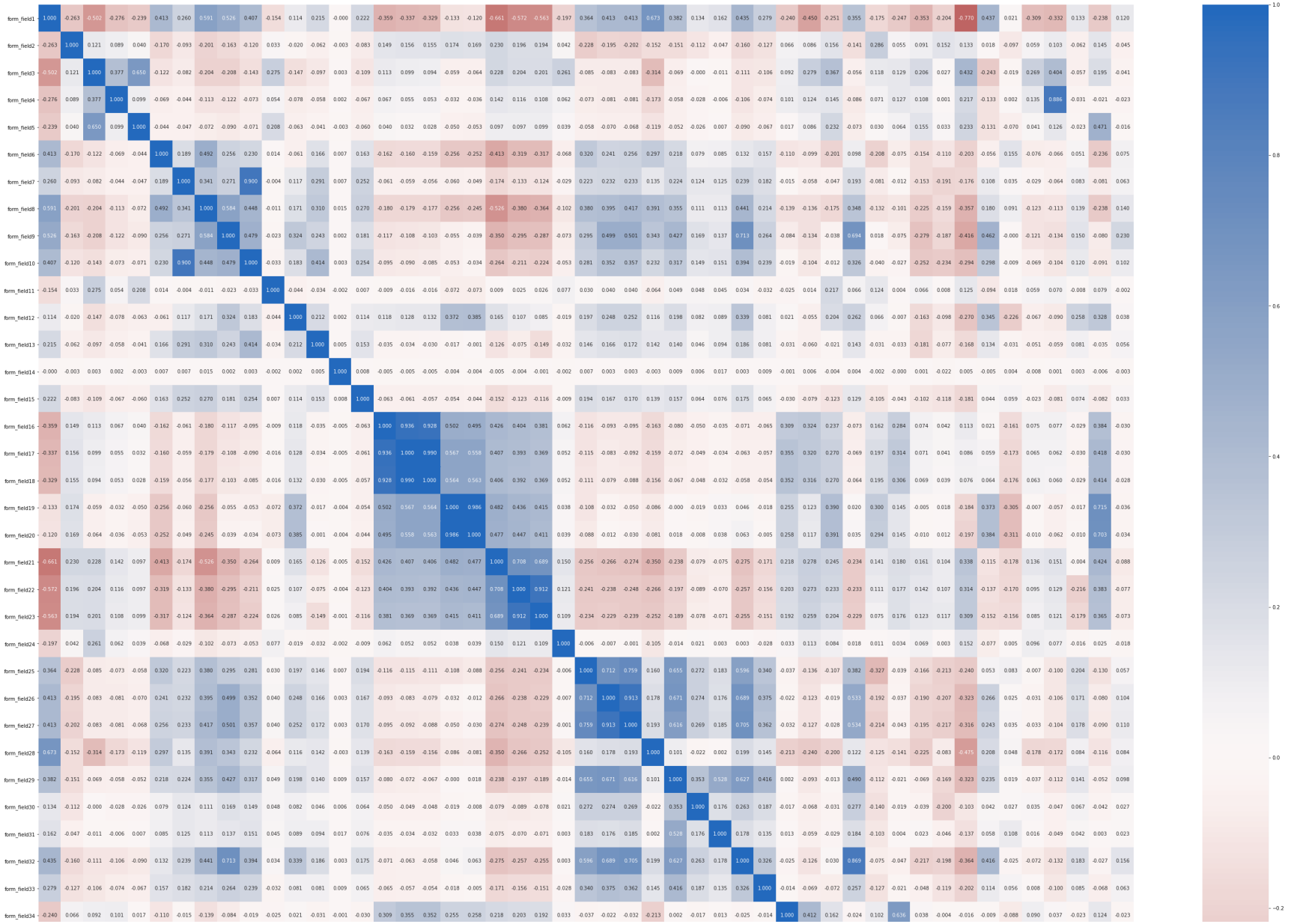
In [481]:
```

sum(df1.Applicant_ID.duplicated())
```

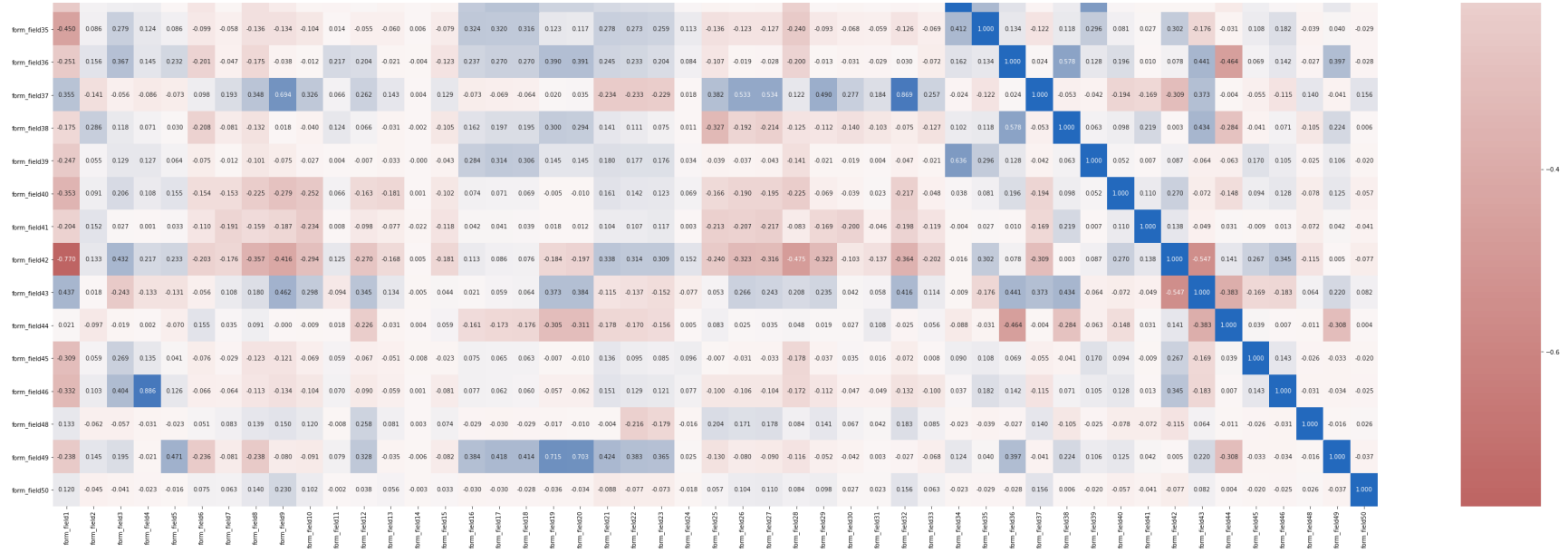
Out[481]: 0

In [482]:
```
df.form_field48.describe()
```

Out[482]:
```
count    3.511100e+04
mean     3.052449e+05
std      1.647757e+06
min      0.000000e+00
25%      7.710079e+01
50%      3.181243e+02
75%      1.153022e+03
max      5.321014e+07
Name: form_field48, dtype: float64
```

In [483]:
```
# correlation plot
numeric_vars = df.select_dtypes(exclude = object).columns
plt.figure(figsize = [50, 50])
sb.heatmap(df[numeric_vars].corr(), annot = True, fmt = '.3f',
           cmap = 'vlag_r', center = 0)
plt.show()
```





In [484]:

Quality Issues

I noticed the following issues with the dataset

- Column names are non-descriptive
- Some colums have null values more than half the dataset
- Inconsistent datatypes for columns
- form_field4, form_field5 has more than 75% as zero
- form_field48 and form_field49 are unknown variables
- form_field32 and form_field33 have same column description

In [484]:

Cleaning

Define

- drop Applicant_ID column
- drop unknown variables form_field48' and 'form_field49' because model perfomed better without them after trial and error

Code

In [485]:
```
# dropping Applicant_ID, 'form_field48','form_field49'

train_df = df.drop(['Applicant_ID','form_field48','form_field49'], axis = 1)
test_df = df1.drop(['Applicant_ID','form_field48','form_field49'],axis = 1)
```

In [486]:
```
#testing to ensure 'Applicant_ID','form_field48','form_field49' has been successfully dropped

train_df.columns
```

Out[486]:
```
Index(['form_field1', 'form_field2', 'form_field3', 'form_field4',
       'form_field5', 'form_field6', 'form_field7', 'form_field8',
       'form_field9', 'form_field10', 'form_field11', 'form_field12',
       'form_field13', 'form_field14', 'form_field15', 'form_field16',
       'form_field17', 'form_field18', 'form_field19', 'form_field20',
       'form_field21', 'form_field22', 'form_field23', 'form_field24',
       'form_field25', 'form_field26', 'form_field27', 'form_field28',
       'form_field29', 'form_field30', 'form_field31', 'form_field32',
       'form_field33', 'form_field34', 'form_field35', 'form_field36',
       'form_field37', 'form_field38', 'form_field39', 'form_field40',
       'form_field41', 'form_field42', 'form_field43', 'form_field44',
       'form_field45', 'form_field46', 'form_field47', 'form_field50',
       'default_status'],
      dtype='object')
```

In [487]:
```
#testing to ensure 'Applicant_ID','form_field48','form_field49' has been successfully dropped

test_df.columns
```

Out[487]:
```
Index(['form_field1', 'form_field2', 'form_field3', 'form_field4',
       'form_field5', 'form_field6', 'form_field7', 'form_field8',
       'form_field9', 'form_field10', 'form_field11', 'form_field12',
       'form_field13', 'form_field14', 'form_field15', 'form_field16',
       'form_field17', 'form_field18', 'form_field19', 'form_field20',
       'form_field21', 'form_field22', 'form_field23', 'form_field24',
       'form_field25', 'form_field26', 'form_field27', 'form_field28',
       'form_field29', 'form_field30', 'form_field31', 'form_field32',
       'form_field33', 'form_field34', 'form_field35', 'form_field36',
       'form_field37', 'form_field38', 'form_field39', 'form_field40',
       'form_field41', 'form_field42', 'form_field43', 'form_field44',
       'form_field45', 'form_field46', 'form_field47', 'form_field50'],
      dtype='object')
```

In [ ]:
```

```

Define: Taking Care of Null Values in the dataset

- I tried to fill the missing values of some columns with forward fill, mean, median or mode but I noticed they made the model perform worse. I found a suggestion on the discussion forum to replace all missing numeric values with -999. That did not make much sense to me but it made my model better, so I used it.

https://zindi.africa/hackathons/dsn-ai-bootcamp-qualification-hackathon/discussions/3171 (https://zindi.africa/hackathons/dsn-ai-bootcamp-qualification-hackathon/discussions/3171)

Code

In [488]:
```
train_df.isnull().sum()
```

Out[488]:
```
form_field1        2529
form_field2        3844
form_field3         355
form_field4         355
form_field5         355
form_field6       13360
form_field7        5163
form_field8       13360
form_field9        8008
form_field10        355
form_field11      31421
form_field12       9895
form_field13       5889
form_field14          0
form_field15      22475
form_field16      13036
form_field17      11151
form_field18      10402
form_field19          4
form_field20        355
form_field21      15854
form_field22      20400
form_field23      28123
form_field24      13297
form_field25       5450
form_field26       7438
form_field27       9299
form_field28        355
form_field29        355
form_field30      25509
form_field31      39408
form_field32       5450
form_field33       1256
form_field34        355
form_field35      23148
form_field36       1995
form_field37       5450
form_field38        355
form_field39       4211
form_field40      43729
form_field41      38229
```

```
form_field42       1323
form_field43        568
form_field44       5383
form_field45      31317
form_field46      15904
form_field47          0
form_field50      11056
default_status        0
dtype: int64
```

In [489]:
```
test_df.isnull().sum()
```

Out[489]:
```
form_field1      1110
form_field2      1709
form_field3       146
form_field4       146
form_field5       146
form_field6      5604
form_field7      2231
form_field8      5604
form_field9      3400
form_field10      147
form_field11    13398
form_field12     4183
form_field13     2463
form_field14        0
form_field15     9592
form_field16     5474
form_field17     4695
form_field18     4369
form_field19        0
form_field20      147
form_field21     6707
form_field22     8724
form_field23    12125
form_field24     5605
form_field25     2256
form_field26     3172
form_field27     3910
form_field28      147
form_field29      147
form_field30    10908
form_field31    16810
form_field32     2256
form_field33      495
form_field34      147
form_field35     9866
form_field36      903
form_field37     2256
form_field38      147
form_field39     1829
form_field40    18828
form_field41    16349
```

```
form_field42      578
form_field43      250
form_field44     2362
form_field45    13538
form_field46     6885
form_field47        0
form_field50     4797
dtype: int64
```

In [ ]:
```
for i in train_df.columns:
    if i in ['form_field47', 'default_status']:
        train_df[column_name].fillna(train_df[column_name].value_counts().index[0],inplace = True)
        continue
    column_name = i
    train_df[column_name].fillna(-999,inplace = True)
```

In [ ]:
```
for i in test_df.columns:
    if i in ['form_field47', 'default_status']:
        test_df[column_name].fillna(test_df[column_name].value_counts().index[0],inplace = True)
        continue
    column_name = i
    test_df[column_name].fillna(-999,inplace = True)
```

In [490]:
```
train_df.describe()
```

Out[490]:

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	
count	53471.000000	52156.000000	55645.000000	55645.000000	55645.000000	4.264000e+04	5.083700e+04	4.264000e+04	4.799200e+04	5
mean	3491.795665	0.550737	1.052225	0.851979	1.956317	6.244479e+05	6.865210e+06	2.626690e+06	1.316002e+07	1
std	188.462426	0.820979	2.147768	3.157692	10.512396	1.433422e+06	1.912729e+07	3.927355e+06	1.977963e+07	2
min	2990.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
25%	3358.000000	0.070788	0.000000	0.000000	0.000000	1.400400e+04	6.869740e+05	1.929440e+05	1.368502e+06	4
50%	3484.000000	0.267575	0.062000	0.000000	0.000000	1.155330e+05	2.704328e+06	9.639420e+05	5.506295e+06	3
75%	3620.000000	0.719512	1.282000	0.000000	0.000000	5.259280e+05	6.993831e+06	3.751516e+06	1.694552e+07	1
max	3900.000000	18.015050	57.371600	91.672200	407.748600	5.313546e+07	2.158794e+09	1.037397e+08	3.200533e+08	2

In [491]:
```
test_df.describe()
```

Out[491]:

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	
count	22890.000000	22291.000000	23854.000000	23854.000000	23854.000000	1.839600e+04	2.176900e+04	1.839600e+04	2.060000e+04	2
mean	3492.284404	0.557676	1.065443	0.859146	2.183538	6.263036e+05	6.797033e+06	2.654142e+06	1.350593e+07	1
std	190.502764	0.826543	2.198444	3.403115	11.415706	1.457540e+06	1.626022e+07	3.968185e+06	2.289125e+07	2
min	2986.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
25%	3356.000000	0.068675	0.000000	0.000000	0.000000	1.400400e+04	6.725810e+05	1.816630e+05	1.349441e+06	4
50%	3484.000000	0.273250	0.058200	0.000000	0.000000	1.155330e+05	2.719888e+06	9.594685e+05	5.529830e+06	3
75%	3624.000000	0.728850	1.304250	0.000000	0.000000	5.159112e+05	7.073576e+06	3.799849e+06	1.728658e+07	1
max	3900.000000	22.315050	34.541400	206.452800	297.885600	4.818738e+07	7.709887e+08	1.135141e+08	1.443921e+09	7

Test

In [549]:
```
train_df.isnull().sum()
```

Out[549]:
```
form_field1             0
form_field2             0
form_field3             0
form_field4             0
form_field5             0
form_field6             0
form_field7             0
form_field8             0
form_field9             0
form_field10            0
form_field11            0
form_field12            0
form_field13            0
form_field14            0
form_field15            0
form_field16            0
form_field17            0
form_field18            0
form_field19            0
form_field20            0
form_field21            0
form_field22            0
form_field23            0
form_field24            0
form_field25            0
form_field26            0
form_field27            0
form_field28            0
form_field29            0
form_field30            0
form_field31            0
form_field32            0
form_field33            0
form_field34            0
form_field35            0
form_field36            0
form_field37            0
form_field38            0
form_field39            0
form_field40            0
form_field41            0
```

```
form_field42           0
form_field43           0
form_field44           0
form_field45           0
form_field46           0
form_field50           0
form_field47_charge    0
form_field47_lending   0
dtype: int64
```

In [550]:
```
test_df.isnull().sum()
```

Out[550]:
```
form_field1     0
form_field2     0
form_field3     0
form_field4     0
form_field5     0
form_field6     0
form_field7     0
form_field8     0
form_field9     0
form_field10    0
form_field11    0
form_field12    0
form_field13    0
form_field14    0
form_field15    0
form_field16    0
form_field17    0
form_field18    0
form_field19    0
form_field20    0
form_field21    0
form_field22    0
form_field23    0
form_field24    0
form_field25    0
form_field26    0
form_field27    0
form_field28    0
form_field29    0
form_field30    0
form_field31    0
form_field32    0
form_field33    0
form_field34    0
form_field35    0
form_field36    0
form_field37    0
form_field38    0
form_field39    0
form_field40    0
form_field41    0
```

```
form_field42            0
form_field43            0
form_field44            0
form_field45            0
form_field46            0
form_field50            0
form_field47_charge     0
form_field47_lending    0
dtype: int64
```

In [551]:
```
train_df.describe()
```

Out[551]:

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9
count	56000.000000	56000.000000	56000.000000	56000.000000	56000.000000	5.600000e+04	5.600000e+04	5.600000e+04	5.600000e+04
mean	3288.988125	-68.061282	-5.287392	-5.486368	-4.389031	4.752342e+05	6.232170e+06	1.999799e+06	1.127799e+07
std	950.560828	252.735737	79.400587	79.418227	80.131613	1.278889e+06	1.833216e+07	3.605357e+06	1.888158e+07
min	-999.000000	-999.000000	-999.000000	-999.000000	-999.000000	-9.990000e+02	-9.990000e+02	-9.990000e+02	-9.990000e+02
25%	3336.000000	0.047700	0.000000	0.000000	0.000000	0.000000e+00	3.248150e+05	0.000000e+00	4.217732e+05
50%	3474.000000	0.226700	0.060000	0.000000	0.000000	4.006700e+04	2.174704e+06	3.765520e+05	3.712227e+06
75%	3610.000000	0.675700	1.270800	0.000000	0.000000	3.023502e+05	6.335740e+06	2.361230e+06	1.403308e+07
max	3900.000000	18.015050	57.371600	91.672200	407.748600	5.313546e+07	2.158794e+09	1.037397e+08	3.200533e+08

In [552]:
```
test_df.describe()
```

Out[552]:

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9
count	24000.000000	24000.000000	24000.000000	24000.000000	24000.000000	2.400000e+04	2.400000e+04	2.400000e+04	2.400000e+04
mean	3284.562500	-70.619160	-5.018289	-5.223330	-3.906996	4.798285e+05	6.165099e+06	2.034167e+06	1.159245e+07
std	961.477817	257.065359	77.795717	77.822785	78.679254	1.303373e+06	1.561130e+07	3.651210e+06	2.172455e+07
min	-999.000000	-999.000000	-999.000000	-999.000000	-999.000000	-9.990000e+02	-9.990000e+02	-9.990000e+02	-9.990000e+02
25%	3332.000000	0.046350	0.000000	0.000000	0.000000	0.000000e+00	3.084770e+05	0.000000e+00	4.240100e+05
50%	3472.000000	0.228800	0.055900	0.000000	0.000000	4.240100e+04	2.178400e+06	3.619645e+05	3.665936e+06
75%	3614.000000	0.680050	1.292650	0.000000	0.000000	3.084770e+05	6.392729e+06	2.499811e+06	1.434048e+07
max	3900.000000	22.315050	34.541400	206.452800	297.885600	4.818738e+07	7.709887e+08	1.135141e+08	1.443921e+09

In [ ]:
```

```

Define form_field32 and form_field33 have same column description

- I checked the correlation of columns form_field32 and form_field33, they were not correlated so I didnt drop them.

In [ ]:
```

```

In [ ]:
```

```

Removing outliers for important features gotten from model

I noticed the following columns were the top 6 important features gotten from the model using

```
from catboost import CatBoostClassifier, Pool

f_imp = model.get_feature_importance(Pool(xval, yval), prettified=True)
```

- form_field1
- form_field2
- form_field47_lending
- form_field47_charge
- form_field42
- form_field6
- form_field15

I tried removing rows (using quantile range) I considered to be outliers for these features. However, doing so did not improve the model (performed worse). Therefore, I did not remove thier outliers on any columns.

In [492]:
```
# x1 = train_df.form_field2
# x2 = test_df.form_field2

# train_df.form_field2 = x1[x1.between(x1.quantile(0), x1.quantile(.99))] # without outliers
# test_df.form_field2 = x2[x2.between(x2.quantile(0), x2.quantile(.99))] # without outliers
```

In [493]:
```
# x3 = train_df.form_field6
# x4 = test_df.form_field6

# train_df.form_field6 = x1[x1.between(x1.quantile(0), x1.quantile(.99))] # without outliers
# test_df.form_field6 = x2[x2.between(x2.quantile(0), x2.quantile(.99))] # without outliers
```

In [494]:
```
# x5 = train_df.form_field42
# x6 = test_df.form_field42

# train_df.form_field42 = x1[x1.between(x1.quantile(0), x1.quantile(.99))] # without outliers
# test_df.form_field42 = x2[x2.between(x2.quantile(0), x2.quantile(.99))] # without outliers
```

In [494]:
```

```

In [496]:
```

```

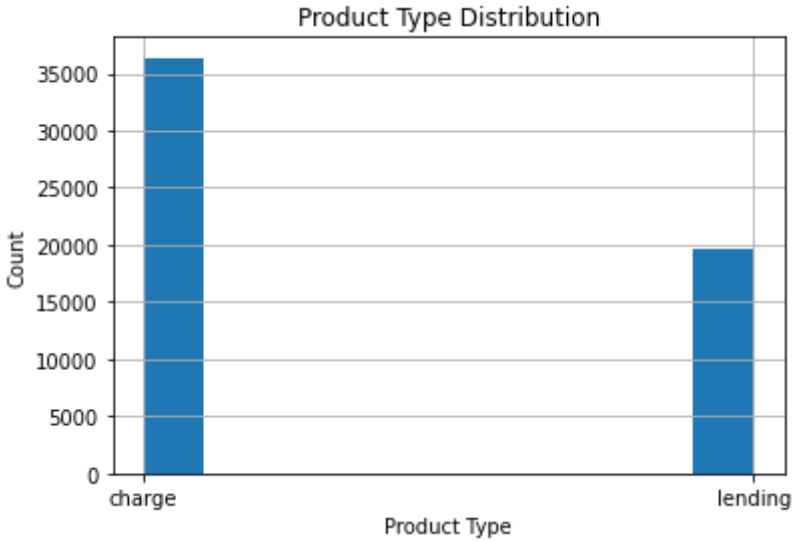
In [ ]:
```

```

Exploration

In this section, the dataset will be explored and possible relationships between the columns will be identified.

In [501]:
```
train_df.form_field47.hist()
plt.xlabel('Product Type')
plt.ylabel('Count')
plt.title('Product Type Distribution')
```

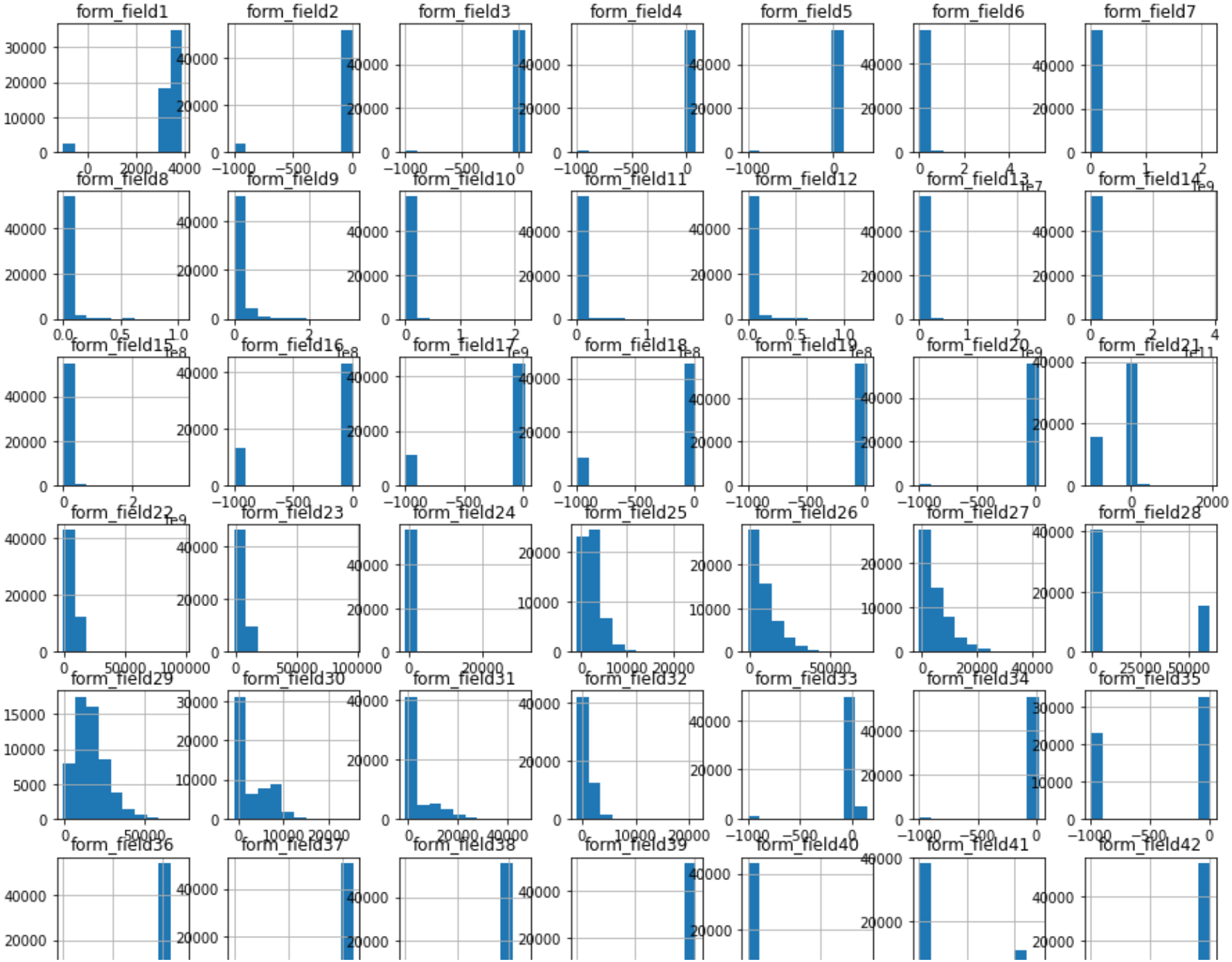
Out[501]:
```
Text(0.5, 1.0, 'Product Type Distribution')
```

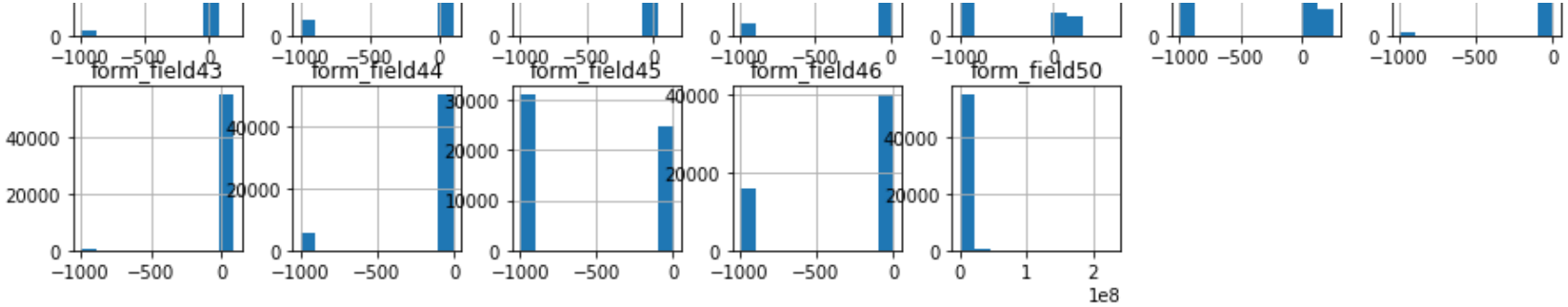


From the plot of product types, it can be observed that most applicants applied for credit card charges than lending.

In [501]:
```

```

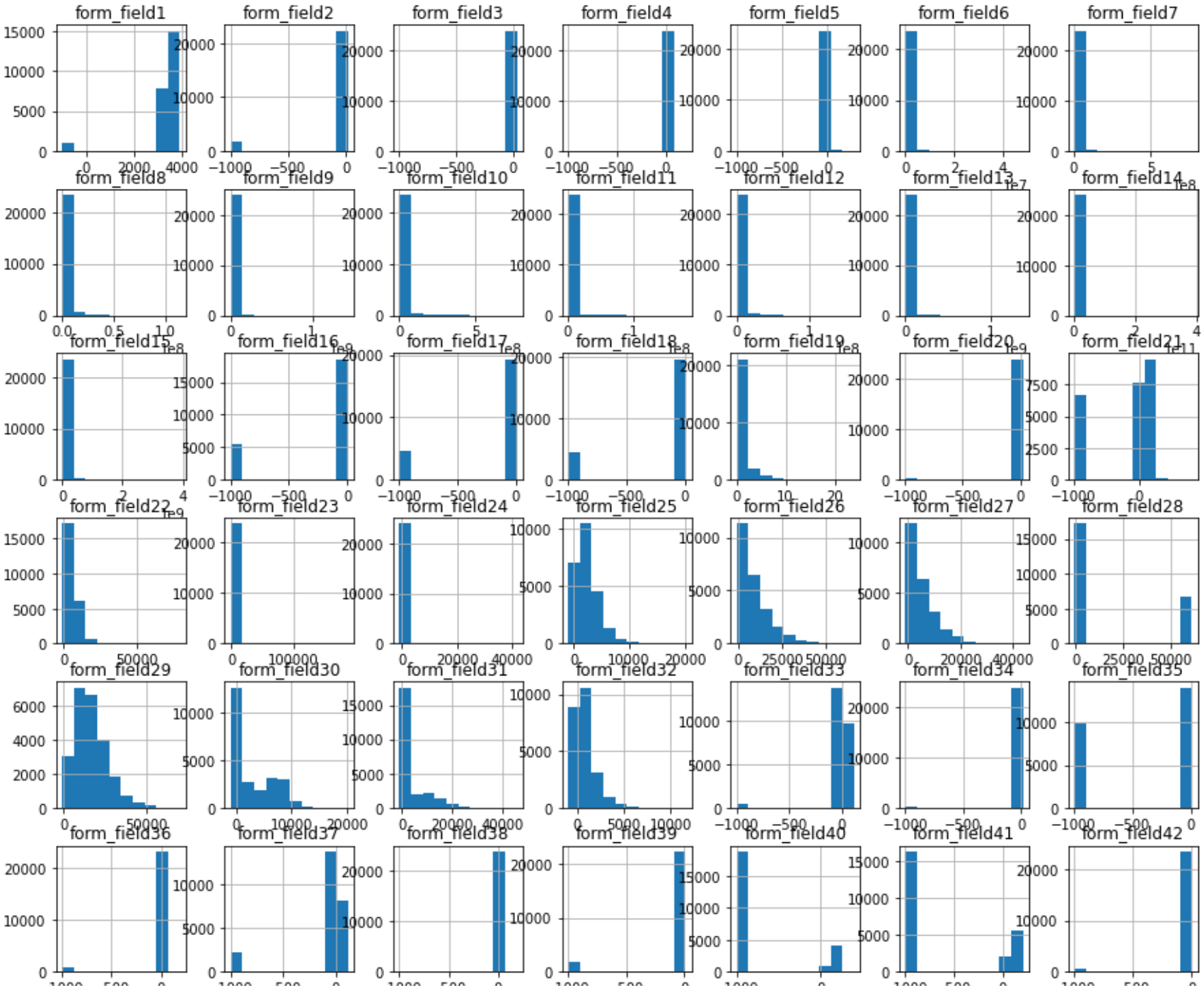
In [502]:
```
#exploring the data set to see the distribution

train_df.hist(figsize=(15,15));
```

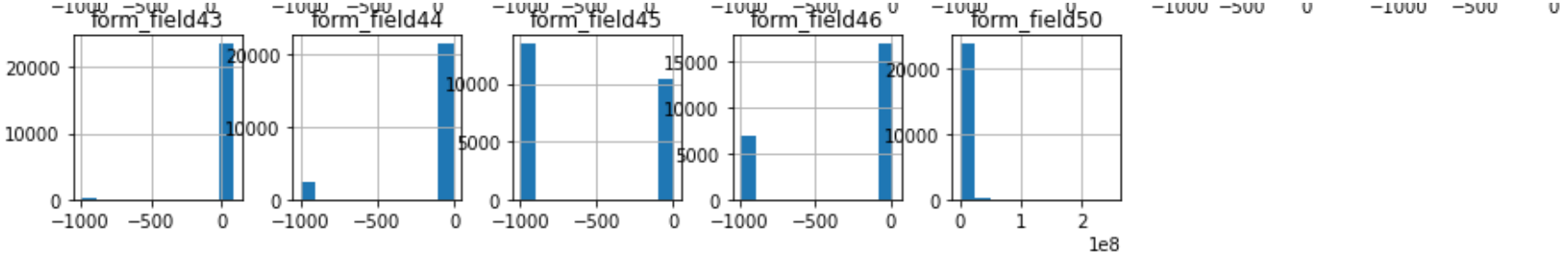



form_field43
form_field44
form_field45
form_field46
form_field50
1e8

In [502]:

In [503]:
```
test_df.hist(figsize=(15,15));
```





Bivariate Exploration

Plotting two variables against each other

In [505]:
```
'''
Function to plot the scatter distribution between two varibles
where data is the dataframe, x is the x-axis variable and
y is the y-axis variable

'''

def scatterplots(data,x,y):
    #Function to plot scatter plots, taking three parameters
    plt.figure(figsize =[6, 6])
    plt.scatter(data = data, x = x, y = y, alpha = 1/3)
    plt.xlabel('{}(mins)'.format(x))
    plt.ylabel('{}(mins)'.format(y))
```

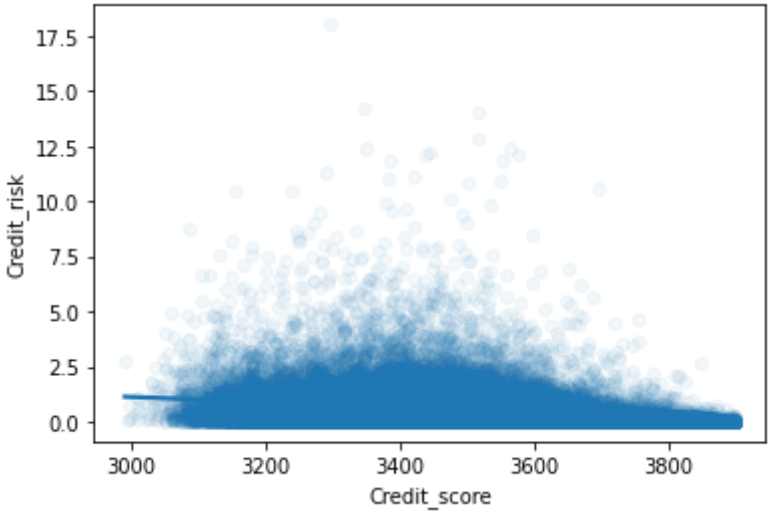
In [506]:
```
#scatter plot between Credit_score Vs Credit_risk

scatterplots(df,'form_field1','form_field2')
plt.title('Credit_score Vs Credit_risk');
```

From the analysis, it can be seen that credit score and credit risk low correlation

By applying jitters and a regression line, we see the plots more clearly below

In [507]:
```
sb.regplot(data=df, x = 'form_field1', y = 'form_field2', x_jitter = 0.3,
           scatter_kws = {'alpha': 1/20})
plt.xlabel ('Credit_score')
plt.ylabel('Credit_risk')
```

Out[507]:
```
Text(0, 0.5, 'Credit_risk')
```

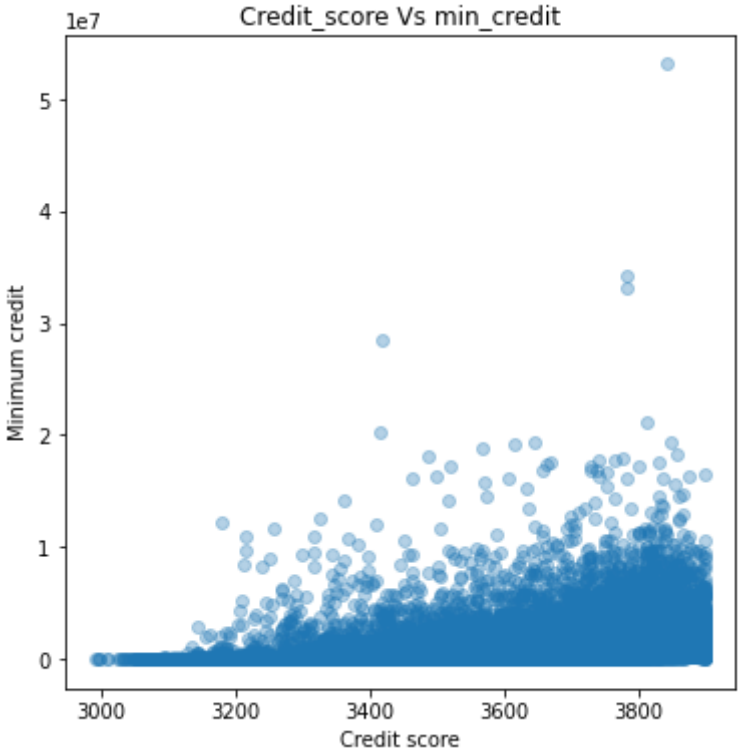


From the analysis, it can be seen that credit score and credit risk low correlation

In [507]:

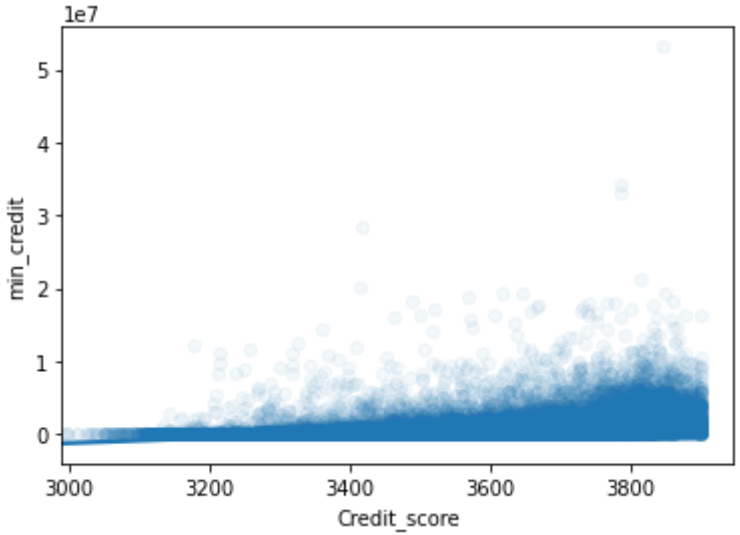
In [508]:
```
#plotting Credit score against Minimum credit

scatterplots(df,'form_field1','form_field6')
plt.title('Credit_score Vs min_credit');
plt.xlabel('Credit score')
plt.ylabel('Minimum credit')
```

Out[508]:
```
Text(0, 0.5, 'Minimum credit')
```



In [509]:
```
sb.regplot(data=df, x = 'form_field1', y = 'form_field6', x_jitter = 0.3,
           scatter_kws = {'alpha': 1/20})
plt.xlabel ('Credit_score')
plt.ylabel('min_credit')
```

Out[509]:
```
Text(0, 0.5, 'min_credit')
```



From the analysis, it can be seen that credit score and minimum credit have a somewhat positive correlation

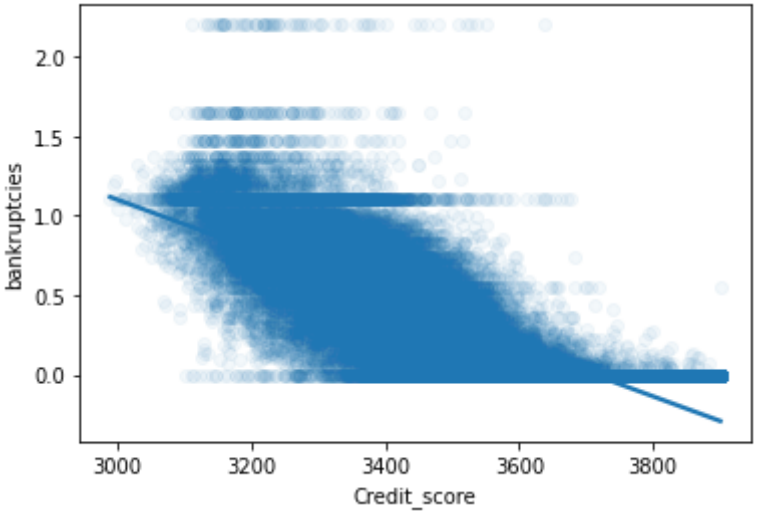
In [509]:
```

```

Next, I will check the relationship between Credit Score and Financial Stress Index of applicants

In [510]:
```
scatterplots(df,'form_field1','form_field42')
plt.title('Credit_score Vs bankruptcies');
```

In [511]:
```
sb.regplot(data=df, x = 'form_field1', y = 'form_field42', x_jitter = 0.3,
           scatter_kws = {'alpha': 1/20})
plt.xlabel ('Credit_score')
plt.ylabel('bankruptcies')
```

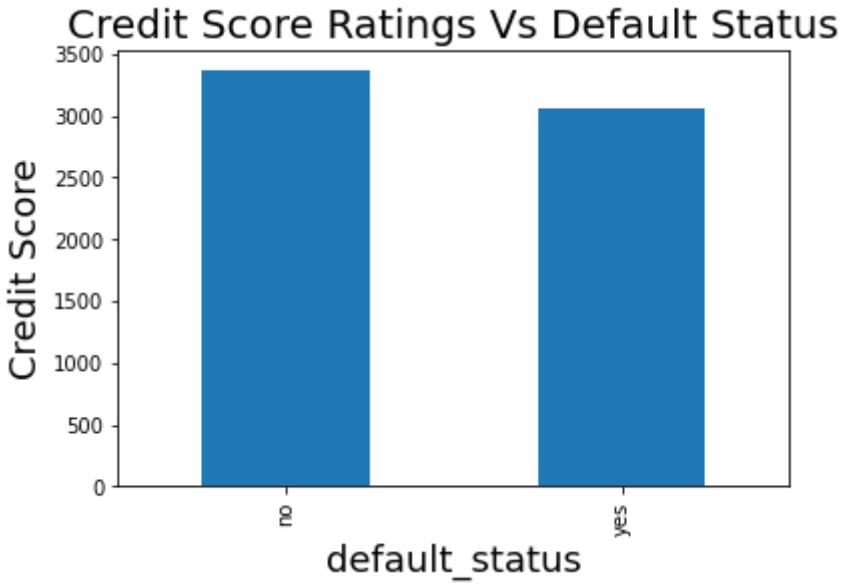
Out[511]:
```
Text(0, 0.5, 'bankruptcies')
```



From the analysis, it can be seen that credit score and Financial stress of applicants have a negative correlation. This is somewhat expected as the higher an applicant's credit score,the lower the Financial stress index of the borrower

In [511]:
```

```

In [512]:
```
train_df.groupby('default_status')['form_field1'].mean().plot(kind='bar')
plt.title('Credit Score Ratings Vs Default Status', fontsize = 20)
plt.xlabel('default_status', fontsize = 18)
plt.ylabel('Credit Score', fontsize = 18);
```

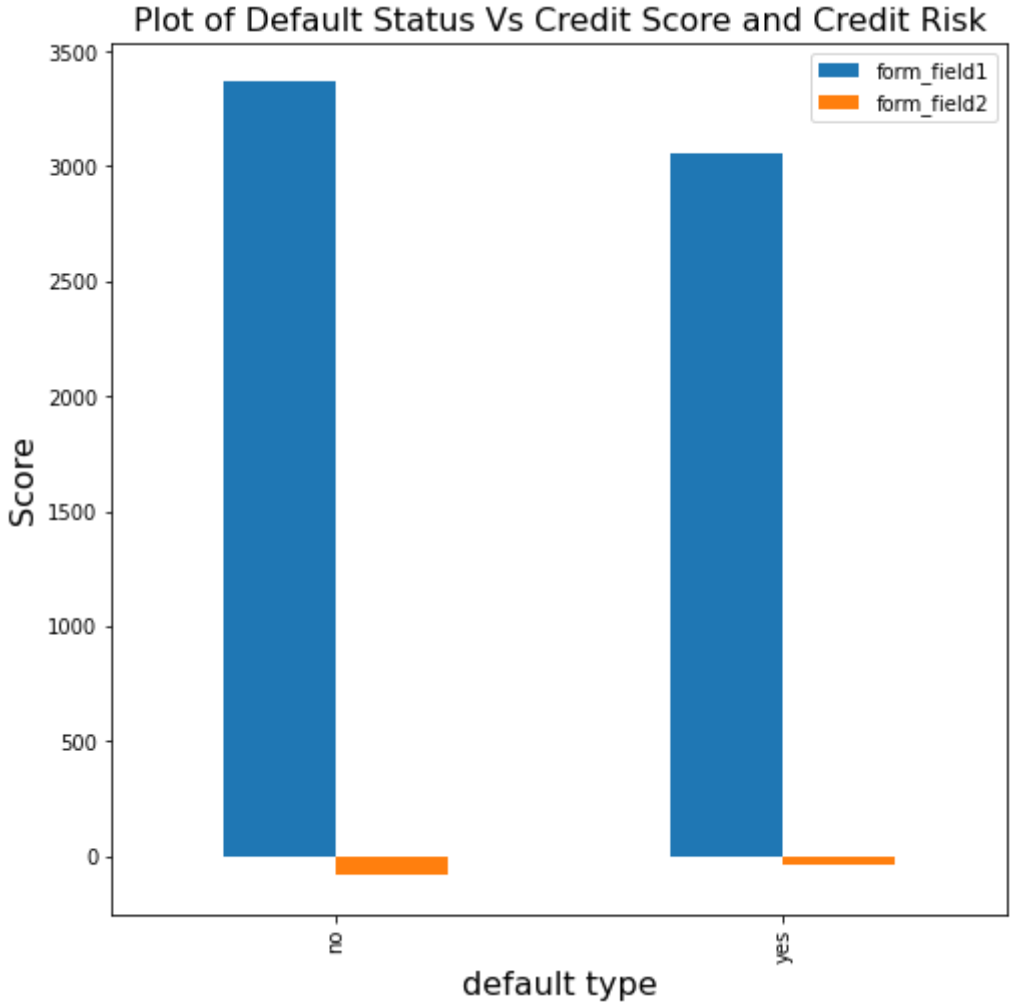


From the distribution above, we can clearly see that non-defaulted applicants tend to have higher credit scores

In [512]:
```

```

Multi-variate Distribution

In [513]:
```
train_df.groupby(['default_status'])['form_field1','form_field2'].mean().plot(kind='bar',figsize=(8,8))
plt.title('Plot of Default Status Vs Credit Score and Credit Risk', fontsize = 16)
plt.ylabel('Score', fontsize = 16)
plt.xlabel('default type', fontsize = 16);
```



In [513]:

Conclusion

The analysis done above may be subject to errors. This is due to the amount of inaccurate data and missing values (null, nan) contained in the dataframe. The data set had a lot of invalid entries and as such, the accuracy of the analysis may be compromised. However, the following conclusions can be made.

- Credit score seem to have a positve relationship with where a borrower status was defaulted or not
- Financial stress of applicants seem to have a negative correlation with credit scores
- credit score and minimum credit seem to have a somewhat positive correlation

In [513]:

In [513]:

Model

Next, I'll be moving on the the modeling, where I will be making use of Catboost. But first, I will use onehot encoding on product_type.

Categorical Features

In [514]:
```
train_df.form_field47.value_counts()
#using onehot encoding on product_type

train_df=pd.get_dummies(train_df,columns=['form_field47'])
```

In [515]:
```
test_df.form_field47.value_counts()
#using onehot encoding on product_type

test_df=pd.get_dummies(test_df,columns=['form_field47'])
```

output label y (default_status)

In [516]:
```
train_df['default_status'] = train_df['default_status'].map({'yes': 1, 'no': 0})
y = train_df['default_status']
y
```

Out[516]:
```
0         0
1         0
2         1
3         0
4         0
         ..
55995     0
55996     1
55997     0
55998     0
55999     0
Name: default_status, Length: 56000, dtype: int64
```

In [517]:
```
#dropping default status from the training data

train_df.drop('default_status',axis=1,inplace = True)
```

In [518]:
```
train_df
```

Out[518]:

	form_field1	form_field2	form_field3	form_field4	form_field5	form_field6	form_field7	form_field8	form_field9	form_field10	form_1
0	3436.0	0.28505	1.6560	0.0000	0.000	0.0	10689720.0	252072.0	4272776.0	11333126.0	439
1	3456.0	0.67400	0.2342	0.0000	0.000	0.0	898979.0	497531.0	9073814.0	2533168.0	24
2	3276.0	0.53845	3.1510	0.0000	6.282	-999.0	956940.0	-999.0	192944.0	1079864.0	
3	3372.0	0.17005	0.5050	0.0000	0.000	192166.0	3044703.0	385499.0	3986472.0	3621979.0	
4	3370.0	0.77270	1.1010	0.0000	0.000	1556.0	214728.0	214728.0	1284089.0	361770.0	39
...	...	...	...	...	...	...	...	...	...	...	
55995	3740.0	0.01730	0.0000	0.0000	0.000	770998.0	9637475.0	4047934.0	11641992.0	19910965.0	
55996	3360.0	2.01145	0.6252	0.0000	0.000	-999.0	927765.0	-999.0	-999.0	1849306.0	57
55997	3500.0	0.76640	0.0000	0.0000	0.000	118645.0	3662435.0	3662435.0	3585024.0	704090.0	
55998	3280.0	0.05235	2.0916	2.2212	0.000	-999.0	3458599.0	-999.0	115533.0	3458599.0	50
55999	3522.0	0.46930	0.0000	0.0000	0.000	98806.0	2053920.0	523983.0	14903368.0	5430440.0	

56000 rows × 49 columns

In [518]:
```
```

Machine Learning

In [525]:
```
columns = train_df.select_dtypes(exclude = object).columns
```

In [529]:
```
X = train_df[columns]
y
```

Out[529]:
```
0         0
1         0
2         1
3         0
4         0
         ..
55995     0
55996     1
55997     0
55998     0
55999     0
Name: default_status, Length: 56000, dtype: int64
```

In [530]:
```
def metric_roc_auc_score(y, pred):
    return roc_auc_score(y, pred, labels=[0, 1])
```

In [531]:
```
#setting the number of seeds
seed = 42
```

In [532]:
```
#setting the number of folds
num_skfold = 15
kf = StratifiedKFold(num_skfold)
```

In [534]:
```
scores = []
score = 0
test_oofs = []
```

In [536]:
```
'''
using catboost classifier and 15 kfolds

Catboost classifier was used for this problem because of its ease and automatic handling of categorical
The links below were used during this research

https://medium.com/devcareers/loan-prediction-using-selected-machine-learning-algorithms-1bdc00717631
https://www.kaggle.com/c/home-credit-default-risk/discussion/59347

'''


#setting the hyperparameters
params = {
    # 'iterations':5000,
    'n_estimators': 3000,
    'learning_rate': 0.01,
    'max_depth': 9,
    'objective': 'CrossEntropy',
      'eval_metric':'AUC',
    'random_seed': seed,
    # 'early_stopping_rounds': 200,
    'use_best_model': True,
     'od_type': 'Iter',
     'od_wait':500,
     'bagging_temperature':0.2
}
for i, (train_idex, valid_idex) in enumerate(kf.split(X, y)):

    xtrain, ytrain = X.loc[train_idex, columns], y.loc[train_idex]
    xval, yval = X.loc[valid_idex, columns], y.loc[valid_idex]

    model = CatBoostClassifier(**params)
    model.fit(xtrain, ytrain, eval_set=[(xval,yval)], verbose=100)

    p = model.predict_proba(xval)[:, 1]
    score_ = metric_roc_auc_score(yval, p)
    scores.append(score_)
    score += score_/num_skfold

    pred = model.predict_proba(test_df[columns])[:, 1]
    test_oofs.append(pred)
```

```
    print('Fold {} : {}'.format(i, score_))

print()
print()
print('Average log : ', score)
```

```
Shrink model to first 2192 iterations.
Fold 13 : 0.8371402091000192
0:      test: 0.8171308 best: 0.8171308 (0)     total: 106ms    remaining: 5m 16s
100:    test: 0.8345862 best: 0.8345862 (100)   total: 10.8s    remaining: 5m 9s
200:    test: 0.8390876 best: 0.8390876 (200)   total: 21.3s    remaining: 4m 56s
300:    test: 0.8414440 best: 0.8414440 (300)   total: 31.6s    remaining: 4m 42s
400:    test: 0.8433427 best: 0.8433427 (400)   total: 41.9s    remaining: 4m 31s
500:    test: 0.8445066 best: 0.8445066 (500)   total: 52.3s    remaining: 4m 20s
600:    test: 0.8454400 best: 0.8454400 (600)   total: 1m 2s    remaining: 4m 9s
700:    test: 0.8463447 best: 0.8463447 (700)   total: 1m 12s   remaining: 3m 59s
800:    test: 0.8469785 best: 0.8469785 (800)   total: 1m 23s   remaining: 3m 48s
900:    test: 0.8475370 best: 0.8475417 (899)   total: 1m 33s   remaining: 3m 38s
1000:   test: 0.8477590 best: 0.8477609 (999)   total: 1m 43s   remaining: 3m 27s
1100:   test: 0.8482748 best: 0.8482930 (1094)  total: 1m 54s   remaining: 3m 17s
1200:   test: 0.8485919 best: 0.8486295 (1184)  total: 2m 7s    remaining: 3m 10s
1300:   test: 0.8488341 best: 0.8488562 (1290)  total: 2m 21s   remaining: 3m 4s
1400:   test: 0.8490165 best: 0.8490522 (1384)  total: 2m 31s   remaining: 2m 52s
1500:   test: 0.8491345 best: 0.8492152 (1460)  total: 2m 42s   remaining: 2m 41s
1600:   test: 0.8491539 best: 0.8492152 (1460)  total: 2m 55s   remaining: 2m 33s
1700:   test: 0.8491057 best: 0.8492350 (1659)  total: 3m 5s    remaining: 2m 21s
```

In [536]:

In [537]:
```
f"{num_skfold} fold CV, score: {score}"
```

Out[537]: '15 fold CV, score: 0.8414551057040357'

In [538]:
```
oof_prediction = pd.DataFrame(test_oofs).T
```

In [539]:
```
oof_prediction.columns = ['fold_'+ str(i) for i in range(1, num_skfold + 1)]
```

In [540]:
```
oof_prediction.head()
```

Out[540]:

	fold_1	fold_2	fold_3	fold_4	fold_5	fold_6	fold_7	fold_8	fold_9	fold_10	fold_11	fold_12	fold_13	fc
0	0.291674	0.296713	0.306309	0.289369	0.298745	0.283045	0.272852	0.319247	0.293048	0.286980	0.295281	0.285792	0.271415	0.2
1	0.414129	0.367098	0.399131	0.394489	0.420300	0.358673	0.419413	0.375734	0.376223	0.387857	0.315365	0.381680	0.416843	0.3
2	0.383402	0.396645	0.382177	0.388441	0.412786	0.390261	0.411076	0.398738	0.391841	0.377870	0.434896	0.389955	0.358820	0.3
3	0.753500	0.734978	0.774917	0.741115	0.743234	0.752518	0.733484	0.771642	0.762443	0.762864	0.787798	0.786938	0.710153	0.7
4	0.132696	0.126377	0.177693	0.171935	0.162253	0.153498	0.146131	0.158045	0.149779	0.168673	0.111491	0.148417	0.183843	0.1

Extracting the important features from the catboost model

In [548]:
```
from catboost import CatBoostClassifier, Pool

f_imp = model.get_feature_importance(Pool(xtrain, ytrain), prettified=True)
f_imp
```

Out[548]:

	Feature Id	Importances
0	form_field2	6.829987
1	form_field1	5.922405
2	form_field47_lending	3.730274
3	form_field47_charge	3.677936
4	form_field42	2.940870
5	form_field15	2.655562
6	form_field3	2.552360
7	form_field6	2.540605
8	form_field33	2.535348
9	form_field14	2.513518
10	form_field24	2.405521
11	form_field43	2.394873
12	form_field11	2.370554
13	form_field9	2.313072
14	form_field29	2.279732
15	form_field7	2.211839
16	form_field38	2.129438
17	form_field10	2.107282
18	form_field30	2.076423
19	form_field25	2.051827
20	form_field27	2.042251
21	form_field8	2.034704
22	form_field26	1.957339

	Feature Id	Importances
23	form_field44	1.949719
24	form_field13	1.946544
25	form_field36	1.941832
26	form_field37	1.914349
27	form_field40	1.887076
28	form_field34	1.861697
29	form_field22	1.764951
30	form_field32	1.750571
31	form_field21	1.702493
32	form_field41	1.598054
33	form_field12	1.575543
34	form_field28	1.492918
35	form_field19	1.448778
36	form_field20	1.387880
37	form_field45	1.339535
38	form_field23	1.303371
39	form_field31	1.283691
40	form_field4	1.268889
41	form_field50	1.236108
42	form_field35	1.074368
43	form_field5	1.050157
44	form_field18	0.836688
45	form_field46	0.657765
46	form_field17	0.615686
47	form_field16	0.442581
48	form_field39	0.395039

In [542]:
```
sub = pd.read_csv('SampleSubmission.csv')
```

In [543]:
```
sub['default_status'] =  np.mean(test_oofs, axis = 0)
#mean of all folds
```

In [546]:
```
sub.to_csv('chichi12_final_submission.csv', index = False)
```

In [ ]:

In [ ]:

In [ ]:

Notes on Results

- 13 0.8415139755203778
- 12 0.84177582339947 // add cat_features to catbooth model tuning. remove onehot encoding better score locally but worse on zindi
- 11 0.8416445807261457 // add cat_features to catbooth model tuning. better score locally but worse on zindi.

0.8414551057040357 // best score on zindi so far, Got 0.844733426980167 on Zindi. Dropped formfield 48 & 49 and fill na. zindi score

- 9. 0.8414551057040357
- 8. 0.8415711816094578
- 7. 0.8412555255076545
- 6. 0.8406273993127695
- 5. 0.8415711816094578
- 4. 0.8414979672319591
- 3. Median with plenty columns gave 0.839808864006457
- 2. Fillforward with small columns gave 0.828415288044175
- 1. Median with small columns gave 0.834775522965092

Type *Markdown* and LaTeX: α^2