MACHINE LEARNING MODEL BY AURANGZEB ALAM

In [1]:
```
#Import libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()
```

In [2]:
```
#Set the working directory
os.chdir("C:/Users/auran/Desktop/Ayushi")
```

In [3]:
```
#check the working directory
os.getcwd()
```

Out[3]:
```
'C:\\Users\\auran\\Desktop\\Ayushi'
```

Exploratory Data Analysis

In [4]:
```
#Load the Data
Dataset = pd.read_csv("churnModelBase_Assignment2.csv")
```

In [5]:
```
#Dataset head
Dataset.head()
```

Out[5]:

	userID	Contact_ind	Customer_status	ActiveSinceDays	num_AcD_m0	num_AcD_m1	r_AcD_m0_m1
0	1	1	4	221	1	2	50.00000
1	2	1	4	1225	3	1	150.00000
2	3	1	3	838	18	19	120.00000
3	4	1	4	486	2	0	200.00000
4	5	1	3	928	1	1	33.33333

5 rows × 112 columns

In [6]:
```
# Separate the target variable from the Dataset
y = Dataset.iloc[:,-1:]
y["targetid"].unique()
```

Out[6]:
```
array([0, 1], dtype=int64)
```

In [7]:
```
#Shape of the Data
Dataset.shape
```

Out[7]:
```
(50000, 112)
```

In [8]:
```
#Columns of Dataset
Dataset.columns
```

Out[8]:
```
Index(['userID', 'Contact_ind', 'Customer_status', 'ActiveSinceDays',
       'num_AcD_m0', 'num_AcD_m1', 'r_AcD_m0_m12', 'r_AcD_m0_m123',
       'daystofirstload', 'dayssincelastload',
       ...
       'InAllRev_m1', 'InAllRev_m2', 'r_InAllRev_m0_m1', 'r_InAllRev_m0_m2',
       'tot_InAllRev_3m', 'OgAllRev_m0', 'OgAllRev_m1', 'OgAllRev_m2',
       'tot_OgAllRev_3m', 'targetid'],
      dtype='object', length=112)
```

In [9]:
```
# UserID has no use in the Dataset for any prediction so we will drop it
Dataset.drop(["userID"],axis=1, inplace=True)
```

In [10]:
```
#Numerical columns of the Dataset
numerical_column = Dataset.select_dtypes(include=[np.number]).columns.tolist()
numerical_column
```

Out[10]:
```
['Contact_ind',
 'Customer_status',
 'ActiveSinceDays',
 'num_AcD_m0',
 'num_AcD_m1',
 'r_AcD_m0_m12',
 'r_AcD_m0_m123',
 'daystofirstload',
 'dayssincelastload',
 'tot_RevUo_m0',
 'tot_RevUo_m1',
 'tot_RevUo_m2',
 'r_tot_RevUo_m0_m1',
 'r_tot_RevUo_m1_m2',
 'r_tot_RevUo_m0_m12',
 'avg_RevUo_m0',
 'avg_RevUo_m1',
 'avg_RevUo_m2',
 'chg_RevUo_l0_1m',
 'chg_RevUo_l1_2m',
 'chg_RevUo_l2_3m',
 'tot_LdCnt_m0',
 'tot_LdCnt_m1',
 'tot_LdCnt_m2',
 'max_RevUo_m0',
 'max_RevUo_m1',
 'max_RevUo_m2',
 'r_avg_RevUo_m0_m1',
 'r_avg_RevUo_m1_m2',
 'avgdays_btw_loads_l6',
 'tot_failed_LdTxnCnt_3m',
 'r_failed_LdTxnCnt_m0_m12',
 'tot_rtlX_m0',
 'tot_rtlX_m1',
 'tot_rtlX_m2',
 'r_tot_rtlX_m0_m1',
 'r_tot_rtlX_m1_m2',
 'r_tot_rtlX_m0_m12',
 'avg_rtlX_m0',
 'avg_rtlX_m1',
 'avg_rtlX_m2',
 'max_rtlX_m0',
 'max_rtlX_m1',
 'max_rtlX_m2',
 'r_avg_max_rtlX_m0',
 'r_avg_max_rtlX_m1',
 'r_avg_max_rtlX_m2',
 'RtlTxns_cnt_m0',
 'RtlTxns_cnt_m1',
 'RtlTxns_cnt_m2',
 'r_RtlTxns_cnt_m0_m1',
 'r_RtlTxns_cnt_m1_m2',
 'sum_r_RtlTxns_cnt',
 'fail_RtlTxncnt_m0',
 'RtlTxn_dayscnt_m0',
 'RtlTxn_dayscnt_m1',
 'RtlTxn_dayscnt_m2',
 'tot_rechX_m0',
 'tot_rechX_m1',
 'tot_rechX_m2',
```

```
 'chg_rechX_l0_1m',
 'chg_rechX_l1_2m',
 'avg_rechX_m0',
 'avg_rechX_m1',
 'avg_rechX_m2',
 'max_rechX_m0',
 'max_rechX_m1',
 'max_rechX_m2',
 'r_avg_max_rechX_m0',
 'r_avg_max_rechX_m1',
 'r_avg_max_rechX_m2',
 'rechtxns_cnt_m0',
 'rechtxns_cnt_m1',
 'rechtxns_cnt_m2',
 'rechtxn_dayscnt_m0',
 'rechtxn_dayscnt_m1',
 'rechtxn_dayscnt_m2',
 'billX_amt_m0',
 'billX_amt_m1',
 'billX_amt_m2',
 'tot_billX_amt_m012',
 'r_billX_amt_m0_m1',
 'r_billX_amt_m0_m12',
 'r_billX_amt_m1_m2',
 'billX_cnt_m0',
 'billX_cnt_m1',
 'billX_cnt_m2',
 'rech_amt_m0',
 'rech_amt_m1',
 'rech_amt_m2',
 'r_rech_amt_m0_m1',
 'r_rech_amt_m1_m2',
 'sum_r_rech_amt_m1m2_m0m1',
 'r_rech_amt_m0_m12',
 'avgbal_m0',
 'avgbal_m1',
 'avgbal_m2',
 'resolved_cnt_m2',
 'tot_resolved_cnt_m012',
 'percent_resolved_3m',
 'InAllRev_m0',
 'InAllRev_m1',
 'InAllRev_m2',
 'r_InAllRev_m0_m1',
 'r_InAllRev_m0_m2',
 'tot_InAllRev_3m',
 'OgAllRev_m0',
 'OgAllRev_m1',
 'OgAllRev_m2',
 'tot_OgAllRev_3m',
 'targetid']
```

In [11]:
```
#Categorical columns of the Dataset
categorical_column = Dataset.select_dtypes(exclude=[np.number]).columns.tolist()
categorical_column
```

Out[11]:
```
[]
```

In [12]:
```
#Dataset info of the Dataset
Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Columns: 111 entries, Contact_ind to targetid
dtypes: float64(67), int64(44)
memory usage: 42.3 MB
```

In [13]:
```
#Statistical Description of the Dataset
Dataset.describe()
```

Out[13]:

	Contact_ind	Customer_status	ActiveSinceDays	num_AcD_m0	num_AcD_m1	r_AcD_m0_m12
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.00000	50000.000000
mean	0.888900	2.822120	713.390560	4.386800	4.59760	112.890128
std	0.314259	15.547017	527.984288	3.855557	4.18118	107.234056
min	0.000000	-999.000000	-999.000000	1.000000	0.00000	4.444444
25%	1.000000	3.000000	261.000000	2.000000	2.00000	55.555556
50%	1.000000	3.000000	551.000000	3.000000	3.00000	88.888889
75%	1.000000	4.000000	1081.000000	6.000000	6.00000	133.333333
max	1.000000	4.000000	2999.000000	30.000000	31.00000	4600.000000

8 rows × 111 columns

Missing Value

In [14]:
```
#Total number of missing value in the Dataset
z = 0
for i in Dataset.columns:
    z += Dataset[i].isnull().sum()
print(z)
```

```
68650
```

In [15]:
```
Dataset.dtypes
```

Out[15]:

```
Contact_ind                  int64
Customer_status              int64
ActiveSinceDays              int64
num_AcD_m0                   int64
num_AcD_m1                   int64
r_AcD_m0_m12               float64
r_AcD_m0_m123              float64
daystofirstload              int64
dayssincelastload            int64
tot_RevUo_m0               float64
tot_RevUo_m1               float64
tot_RevUo_m2               float64
r_tot_RevUo_m0_m1          float64
r_tot_RevUo_m1_m2          float64
r_tot_RevUo_m0_m12         float64
avg_RevUo_m0               float64
avg_RevUo_m1               float64
avg_RevUo_m2               float64
chg_RevUo_l0_1m            float64
chg_RevUo_l1_2m            float64
chg_RevUo_l2_3m            float64
tot_LdCnt_m0                 int64
tot_LdCnt_m1                 int64
tot_LdCnt_m2                 int64
max_RevUo_m0               float64
max_RevUo_m1               float64
max_RevUo_m2               float64
r_avg_RevUo_m0_m1          float64
r_avg_RevUo_m1_m2          float64
avgdays_btw_loads_l6         int64
                             ...
r_billX_amt_m0_m1          float64
r_billX_amt_m0_m12         float64
r_billX_amt_m1_m2          float64
billX_cnt_m0                 int64
billX_cnt_m1                 int64
billX_cnt_m2                 int64
rech_amt_m0                  int64
rech_amt_m1                  int64
rech_amt_m2                  int64
r_rech_amt_m0_m1           float64
r_rech_amt_m1_m2           float64
sum_r_rech_amt_m1m2_m0m1   float64
r_rech_amt_m0_m12          float64
avgbal_m0                  float64
avgbal_m1                  float64
avgbal_m2                  float64
resolved_cnt_m2              int64
tot_resolved_cnt_m012        int64
percent_resolved_3m        float64
InAllRev_m0                float64
InAllRev_m1                float64
InAllRev_m2                float64
r_InAllRev_m0_m1           float64
r_InAllRev_m0_m2           float64
tot_InAllRev_3m            float64
OgAllRev_m0                float64
OgAllRev_m1                float64
OgAllRev_m2                float64
tot_OgAllRev_3m            float64
```

```
targetid                        int64
Length: 111, dtype: object
```

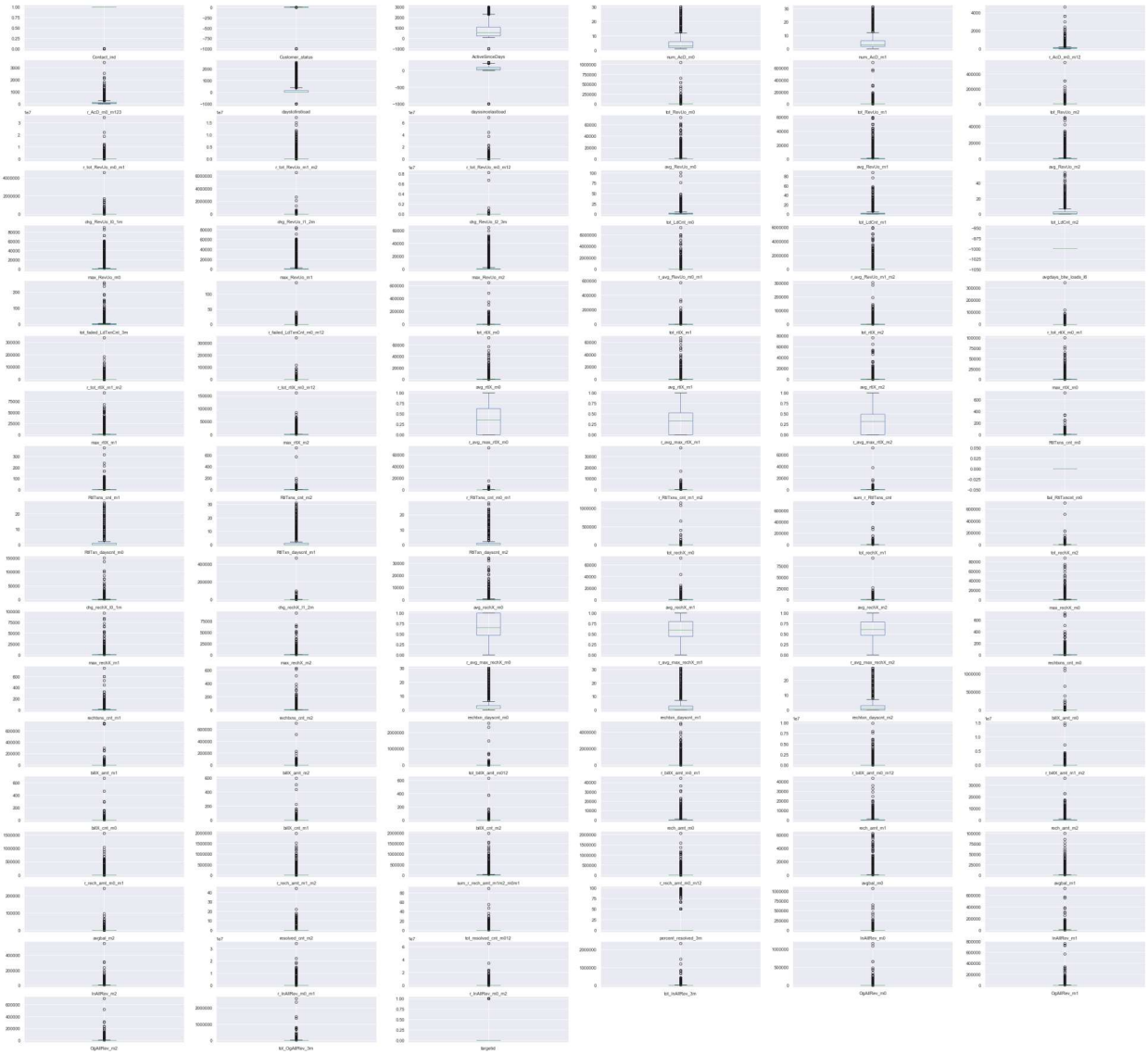
In [16]:
```
# Now lets impute the missing values with mean and most_frequent

from sklearn.preprocessing import Imputer

for i in Dataset.columns:
    if(Dataset[i].dtypes=='float64'):
        imputer = Imputer(missing_values="NaN",strategy='mean',axis=0)
        Dataset[[i]] = imputer.fit_transform(Dataset[[i]])
    else:
        imputer = Imputer(missing_values="NaN",strategy='most_frequent',axis=0)
        Dataset[[i]] = imputer.fit_transform(Dataset[[i]])
```

In [17]:
```
#Check the missing values after imputation
Dataset.isnull().sum().sum()
```

Out[17]:
```
0
```

Outliers

In [18]:
```
#Plot to see the outliers
Dataset.plot(kind="box",subplots=True,layout=(20,6),figsize=(50,50))
plt.show()
```



In [19]:
```
#Detect the outliers and make those outliers as NaN value.
for i in numerical_column:
    print(i)
    q75,q25=np.percentile(Dataset[i],[75,25])
    iqr = q75-q25
    min = q25-(iqr*1.5)
    print(min)
    max = q75+(iqr*1.5)
    print(max)
    Dataset.loc[Dataset[i]<min,i] = np.nan
    Dataset.loc[Dataset[i]>max,i] = np.nan
```

```
Contact_ind
1.0
1.0
Customer_status
1.5
5.5
ActiveSinceDays
-969.0
2311.0
num_AcD_m0
-4.0
12.0
num_AcD_m1
-4.0
12.0
r_AcD_m0_m12
-61.11111104999999
249.99999990999999
r_AcD_m0_m123
-75.0
285.0
daystofirstload
-234.0
390.0
dayssincelastload
-118.5
221.5
tot_RevUo_m0
-1965.0
3275.0
tot_RevUo_m1
-2250.0
3750.0
tot_RevUo_m2
-2325.0
3875.0
r_tot_RevUo_m0_m1
-299.850075
499.750125
r_tot_RevUo_m1_m2
-364.10234204999995
606.8372367499999
r_tot_RevUo_m0_m12
-221.2203960375
368.7006600625
avg_RevUo_m0
-903.9740625
1506.6234375
avg_RevUo_m1
-937.5
1562.5
avg_RevUo_m2
-945.23625
1575.3937500000002
chg_RevUo_l0_1m
-2457.72464067
1432.20773445
chg_RevUo_l1_2m
-2467.5
1448.5
```

```
chg_RevUo_l2_3m
-2468.1657056024997
1449.6095093375
tot_LdCnt_m0
-3.0
5.0
tot_LdCnt_m1
-3.0
5.0
tot_LdCnt_m2
-4.5
7.5
max_RevUo_m0
-1500.0
2500.0
max_RevUo_m1
-1500.0
2500.0
max_RevUo_m2
-1500.0
2500.0
r_avg_RevUo_m0_m1
-268.45280921250003
447.42134868750003
r_avg_RevUo_m1_m2
-299.40119760000005
499.0019960000001
avgdays_btw_loads_l6
-999.0
-999.0
tot_failed_LdTxnCnt_3m
-4.5
7.5
r_failed_LdTxnCnt_m0_m12
0.0
0.0
tot_rtlX_m0
-783.0
1305.0
tot_rtlX_m1
-794.14125
1323.56875
tot_rtlX_m2
-863.2574999999999
1438.7624999999998
r_tot_rtlX_m0_m1
-1.13755256625
1.8959209437499998
r_tot_rtlX_m1_m2
-0.8862699273750001
1.4771165456250002
r_tot_rtlX_m0_m12
-0.46967682300000013
0.7827947050000001
avg_rtlX_m0
-393.0
655.0
avg_rtlX_m1
-366.0
610.0
```

```
avg_rtlX_m2
-376.7587500375
627.9312500625
max_rtlX_m0
-541.5
902.5
max_rtlX_m1
-522.51
870.85
max_rtlX_m2
-551.3062500000001
918.8437500000001
r_avg_max_rtlX_m0
-0.9361941727499999
1.5603236212499998
r_avg_max_rtlX_m1
-0.7740150135000001
1.2900250225
r_avg_max_rtlX_m2
-0.734404121625
1.2240068693750001
RtlTxns_cnt_m0
-1.5
2.5
RtlTxns_cnt_m1
-1.5
2.5
RtlTxns_cnt_m2
-1.5
2.5
r_RtlTxns_cnt_m0_m1
-75.0
125.0
r_RtlTxns_cnt_m1_m2
-75.0
125.0
sum_r_RtlTxns_cnt
-199.99999995
333.33333325
fail_RtlTxncnt_m0
0.0
0.0
RtlTxn_dayscnt_m0
-1.5
2.5
RtlTxn_dayscnt_m1
-1.5
2.5
RtlTxn_dayscnt_m2
-1.5
2.5
tot_rechX_m0
-1186.0
2030.0
tot_rechX_m1
-1257.0
2095.0
tot_rechX_m2
-1062.0
1770.0
```

```
chg_rechX_l0_1m
-2514.0021040125002
1526.0035066875
chg_rechX_l1_2m
-2512.5478082625
1523.5796804375
avg_rechX_m0
-450.0
790.0
avg_rechX_m1
-412.5
687.5
avg_rechX_m2
-363.0
605.0
max_rechX_m0
-626.0
1094.0
max_rechX_m1
-643.5
1072.5
max_rechX_m2
-598.5
997.5
r_avg_max_rechX_m0
-0.3160003912500001
1.78960023475
r_avg_max_rechX_m1
-0.09548354562500017
1.3372901273750002
r_avg_max_rechX_m2
-0.01200582950000001
1.2678095585
rechtxns_cnt_m0
-2.0
6.0
rechtxns_cnt_m1
-6.0
10.0
rechtxns_cnt_m2
-4.5
7.5
rechtxn_dayscnt_m0
-2.0
6.0
rechtxn_dayscnt_m1
-4.5
7.5
rechtxn_dayscnt_m2
-4.5
7.5
billX_amt_m0
-435.0
725.0
billX_amt_m1
-441.0
735.0
billX_amt_m2
-361.5
602.5
```

```
tot_billX_amt_m012
-1635.0
2725.0
r_billX_amt_m0_m1
-132.4042127625
220.6736879375
r_billX_amt_m0_m12
-130.0627848375
216.7713080625
r_billX_amt_m1_m2
-149.0625
248.4375
billX_cnt_m0
-1.5
2.5
billX_cnt_m1
-1.5
2.5
billX_cnt_m2
-1.5
2.5
rech_amt_m0
-690.0
1150.0
rech_amt_m1
-714.0
1190.0
rech_amt_m2
-598.5
997.5
r_rech_amt_m0_m1
-439.0243902
731.7073170000001
r_rech_amt_m1_m2
-501.9658929
836.6098215
sum_r_rech_amt_m1m2_m0m1
-18736.655470924998
31350.152014555
r_rech_amt_m0_m12
-299.25
498.75
avgbal_m0
-265.54327582
442.756034412
avgbal_m1
-336.49693544999997
561.5482257499999
avgbal_m2
-303.12901785249994
506.0131249914999
resolved_cnt_m2
0.0
0.0
tot_resolved_cnt_m012
0.0
0.0
percent_resolved_3m
0.0
0.0
```

```
InAllRev_m0
-2250.0
3750.0
InAllRev_m1
-2400.0
4000.0
InAllRev_m2
-2593.21875
4322.03125
r_InAllRev_m0_m1
-306.0019605
510.0032675
r_InAllRev_m0_m2
-341.7401693625
569.5669489375
tot_InAllRev_3m
-8240.30875
15067.18125
OgAllRev_m0
-2904.5
5587.5
OgAllRev_m1
-3084.5
5671.5
OgAllRev_m2
-3200.0
5600.0
tot_OgAllRev_3m
-9990.5
19765.5
targetid
0.0
0.0
```

In [20]:
```
#Now check the total number of NaN which has been created after outliers detection.
Dataset.isnull().sum().sum()
```

Out[20]:
```
596519
```

In [21]:
```
# Again impute the NaN value which was created after we made outliers as NaN by mean a

for i in Dataset.columns:
    if(Dataset[i].dtypes=='float64'):
        imputer = Imputer(missing_values="NaN",strategy='mean',axis=0)
        Dataset[[i]] = imputer.fit_transform(Dataset[[i]])
    else:
        imputer = Imputer(missing_values="NaN",strategy='most_frequent',axis=0)
        Dataset[[i]] = imputer.fit_transform(Dataset[[i]])
```

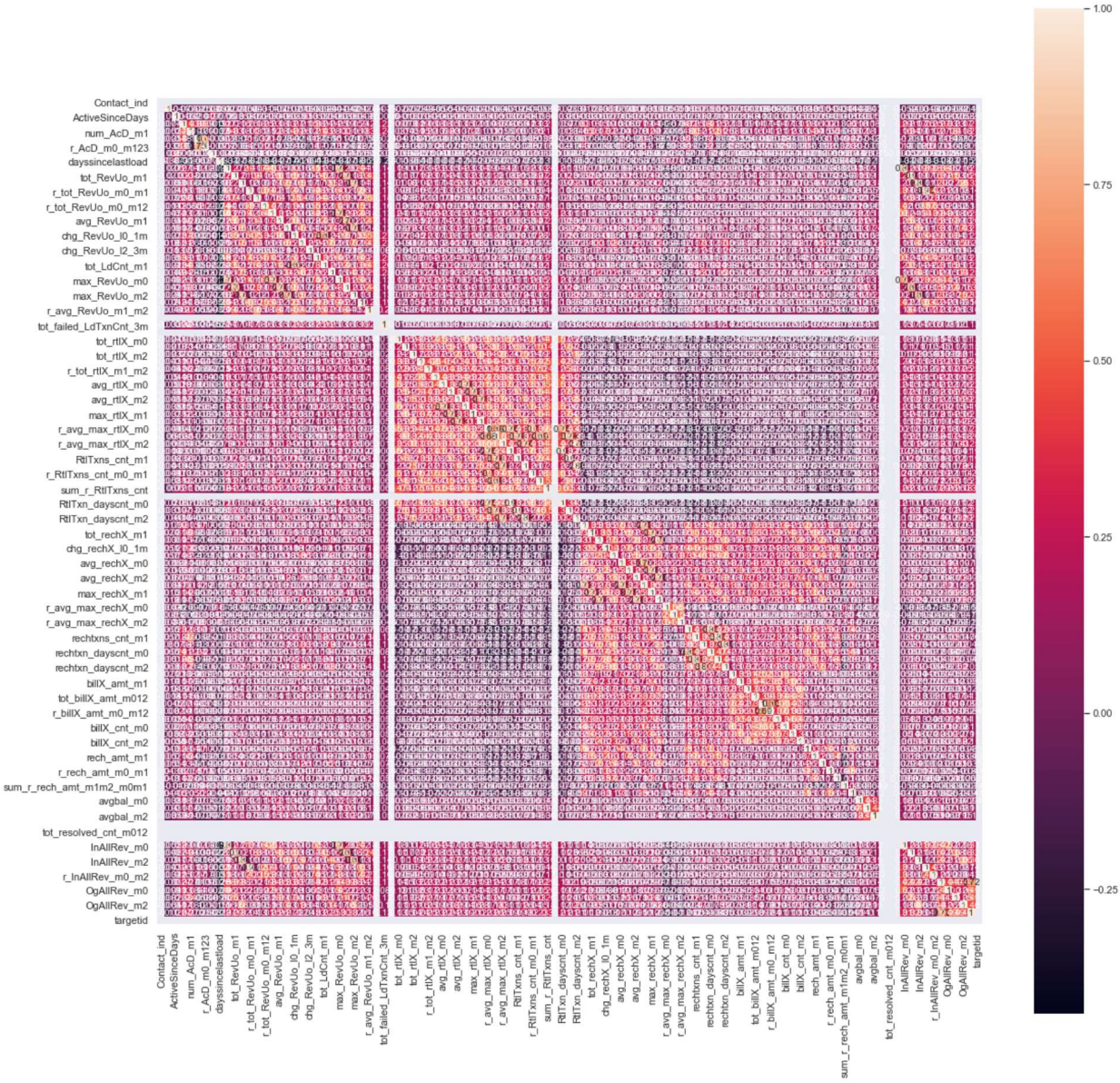
In [22]:
```
#Now check the null after we did the imputation
Dataset.isnull().sum().sum()
```

Out[22]:
```
0
```

Feature Selection

In [23]:
```
#Firstly create the correlation and then plot the correlation matrix
corr = Dataset.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr, annot=True, square=True)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22e242b9e48>
```



In [24]:
```
# Create correlation matrix and remove the correlated columns
corr_matrix = Dataset.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Find features with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.75)]

# Drop features
Dataset.drop(Dataset[to_drop], axis=1,inplace=True)
```

In [25]:
```
#After finding and removing the correlated columns we will define the predictor variab
X = Dataset
```

In [26]:
```
#Now split data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=1
```

Build the model

Logistic Regression

In [27]:
```
#Import Logistic Regression and initiate the Logistic Regression model
from sklearn.linear_model import LogisticRegression
Logistic_Regression = LogisticRegression()
Logistic_Regression.fit(X_train,y_train)
prediction1 = Logistic_Regression.predict(X_test)
```

```
C:\Users\auran\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [28]:
```
#Accuracy metrics
from sklearn import metrics
print("accuracy_score",metrics.accuracy_score(prediction1,y_test))
```

```
accuracy_score 0.8220666666666666
```

Random Forest

In [29]:
```
#Import Random Forest and initiate the Random Forest model
from sklearn.ensemble import RandomForestClassifier
Random_Forest = RandomForestClassifier(n_estimators=200)
Random_Forest.fit(X_train,y_train)
```

```
C:\Users\auran\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: De
precationWarning: numpy.core.umath_tests is an internal NumPy module and should not b
e imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
C:\Users\auran\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DataConversionWar
ning: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples,), for example using ravel().
  after removing the cwd from sys.path.
```

Out[29]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [30]:
```
prediction2 = Random_Forest.predict(X_test)
prediction2
```

Out[30]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [31]:
```
from sklearn import metrics
metrics.accuracy_score(prediction2,y_test)
```

Out[31]:
```
0.8517333333333333
```

In [ ]:
```

```