Investigation of Bank Fraud

```
Problem Statement: PredCatch Analytics' Australian banking client's
profitability and reputation are being hit by fraudulent ATM transactions.
They want PredCatch to help them in reducing and if possible completely
eliminating such fraudulent transactions. PredCatch believes it can do the
same by building a predictive model to catch such fraudulent transactions in
real time and decline them. Your job as PredCatch's Data Scientist is to build
this fraud detection & prevention predictive model in the first step. If
successful, in the 2nd step you will have to present your solutions and
explain how it works to the client. The data has been made available to you.
The challenging part of the problem is that the data contains very few fraud
instances in comparison to the overall population. To give more edge to the
solution they have also collected data regarding location [geo_scores] of the
transactions, their own proprietary index [Lambda_wts], on network turn around
times [Qset_tats] and vulnerability qualification score [instance_scores]. As
of now you don't need to understand what they mean.
Training data contains masked variables pertaining to each transaction id .
Your prediction target here is 'Target' .


1: Fraudulent transactions


0: Clean transactions
```

Importing the Basic Libraries

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

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

Display all the columns of the dataframe

In [2]:
```
pd.pandas.set_option('display.max_columns',None)
```

Import the Datasets

In [3]:
```
data1 = pd.read_csv('Geo_scores.csv')
data2 = pd.read_csv('instance_scores.csv')
data3 = pd.read_csv('Lambda_wts.csv')
data4 = pd.read_csv('Qset_tats.csv')
data5 = pd.read_csv('test_share.csv')
data6 = pd.read_csv('train (5).csv')
```

save data in diff variables

In [4]:
```
geo = data1
score_inst = data2
Lambda = data3
Qset = data4
test = data5
train = data6
```

save all data in one list

In [5]:
```
dataset= [('Geo_scores',geo),('instance_scores',score_inst),('Lambda_wts', Lam
        ('Qset_tats', Qset),('Test', test),('Train',train)]
dataset
```

```
 [1424035 rows x 2 columns]),
('Lambda_wts',
         Group  lambda_wt
 0      Grp936       3.41
 1      Grp347      -2.88
 2      Grp188       0.39
 3     Grp1053      -2.75
 4       Grp56      -0.83
 ...       ...        ...
 1395   Grp892       4.24
 1396  Grp1072      -7.28
 1397   Grp785      -2.63
 1398    Grp50       0.79
 1399    Grp37      -0.16

 [1400 rows x 2 columns]),
('Qset_tats',
             id  qsets_normalized_tat
```

Read all dataset

Head

In [6]:
```
data1.head()
```

Out[6]:

	id	geo_score
0	26674	4.48
1	204314	4.48
2	176521	5.17
3	48812	-2.41
4	126870	6.55

In [7]:
```
data2.head()
```

Out[7]:

	id	instance_scores
0	173444	-0.88
1	259378	1.50
2	161170	0.44
3	191161	0.76
4	34521	-0.84

In [8]:
```
data3.head()
```

Out[8]:

	Group	lambda_wt
0	Grp936	3.41
1	Grp347	-2.88
2	Grp188	0.39
3	Grp1053	-2.75
4	Grp56	-0.83

In [9]:
```
data4.head()
```

Out[9]:

	id	qsets_normalized_tat
0	9983	2.41
1	266000	3.10
2	77525	1.03
3	160765	-11.63
4	138220	-4.48

In [10]:
```
data5.head()
```

Out[10]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	P
0	146574	Grp229	-0.300000	1.540000	0.220000	-0.280000	0.570000	0.260000	0.700000	1.076
1	268759	Grp141	0.633333	0.953333	0.810000	0.466667	0.910000	0.253333	1.040000	0.550
2	59727	Grp188	1.043333	0.740000	0.860000	1.006667	0.583333	0.616667	0.630000	0.686
3	151544	Grp426	1.283333	0.300000	0.576667	0.636667	0.256667	0.543333	0.356667	0.663
4	155008	Grp443	1.186667	0.326667	0.476667	0.866667	0.436667	0.680000	0.476667	0.686

In [11]:
```
data6.head()
```

Out[11]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

shape

In [12]:
```
data1.shape
```

Out[12]: (1424035, 2)

In [13]:
```
data2.shape
```

Out[13]: (1424035, 2)

In [14]:
```
data3.shape
```

Out[14]: (1400, 2)

In [15]:
```
data4.shape
```

Out[15]: (1424035, 2)

In [16]:
```
data5.shape
```

Out[16]: (56962, 27)

In [17]:
```
data6.shape
```

Out[17]: (227845, 28)

Shape Using For Loop Function

In [18]:
```
for name, data in dataset:
    print(30*'++')
    print(name, ' :' , data.shape)
    print(30*'++')
    print(data.head())
    print()
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Geo_scores  : (1424035, 2)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  geo_score
0   26674       4.48
1  204314       4.48
2  176521       5.17
3   48812      -2.41
4  126870       6.55

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
instance_scores  : (1424035, 2)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  instance_scores
0  173444            -0.88
1  259378             1.50
2  161170             0.44
3  191161             0.76
4   34521            -0.84

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Lambda_wts  : (1400, 2)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
     Group  lambda_wt
0   Grp936       3.41
1   Grp347      -2.88
2   Grp188       0.39
3  Grp1053      -2.75
4    Grp56      -0.83

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Qset_tats  : (1424035, 2)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  qsets_normalized_tat
0    9983                  2.41
1  266000                  3.10
2   77525                  1.03
3  160765                -11.63
4  138220                 -4.48

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Test  : (56962, 27)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id   Group      Per1      Per2      Per3      Per4      Per5      Per6
\
0  146574  Grp229 -0.300000  1.540000  0.220000 -0.280000  0.570000  0.260000
1  268759  Grp141  0.633333  0.953333  0.810000  0.466667  0.910000  0.253333
2   59727  Grp188  1.043333  0.740000  0.860000  1.006667  0.583333  0.616667
3  151544  Grp426  1.283333  0.300000  0.576667  0.636667  0.256667  0.543333
4  155008  Grp443  1.186667  0.326667  0.476667  0.866667  0.436667  0.680000

       Per7      Per8      Per9      Dem1      Dem2      Dem3      Dem4  \
0  0.700000  1.076667  0.930000  0.156667  0.546667  0.530000  0.876667
1  1.040000  0.550000  0.543333  0.433333  0.966667  0.760000  0.576667
2  0.630000  0.686667  0.593333  1.250000  0.826667  0.826667  0.653333
3  0.356667  0.663333  1.156667  1.186667  0.900000  0.433333  0.230000
4  0.476667  0.686667  1.476667  1.213333  0.853333  0.583333  0.850000
```

```
       Dem5      Dem6      Dem7      Dem8      Dem9     Cred1     Cred2  \
0  0.450000  0.370000  0.786667  0.546667  0.313333  0.703333  0.813333
1  0.653333  0.553333  0.636667  0.770000  0.993333  0.536667  0.703333
2  0.663333  0.453333  0.626667  0.756667  0.953333  0.623333  0.753333
3  1.323333  0.403333  0.480000  0.460000  0.260000  0.800000  0.606667
4  1.090000  0.550000  0.706667  0.740000  0.823333  0.670000  0.896667

      Cred3     Cred4     Cred5     Cred6  Normalised_FNT
0  0.776667  0.796667  0.823333  0.783333       -249.7500
1  0.806667  0.630000  0.673333  0.673333       -249.8125
2  0.870000  0.596667  0.680000  0.670000       -248.1200
3  0.456667  0.320000  0.676667  0.660000       -222.9875
4  0.566667  0.546667  0.650000  0.663333       -196.2200

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Train  : (227845, 28)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id   Group      Per1      Per2      Per3      Per4      Per5      Per6
\
0  112751  Grp169  1.070000  0.580000  0.480000  0.766667  1.233333  1.993333
1   18495  Grp161  0.473333  1.206667  0.883333  1.430000  0.726667  0.626667
2   23915  Grp261  1.130000  0.143333  0.946667  0.123333  0.080000  0.836667
3   50806  Grp198  0.636667  1.090000  0.750000  0.940000  0.743333  0.346667
4  184244  Grp228  0.560000  1.013333  0.593333  0.416667  0.773333  0.460000

       Per7      Per8      Per9      Dem1      Dem2      Dem3      Dem4  \
0  0.340000  1.010000  0.863333  0.460000  0.643333  0.736667  0.756667
1  0.810000  0.783333  0.190000  0.470000  0.613333  0.883333  0.653333
2  0.056667  0.756667  0.226667  0.660000  0.730000  0.873333  0.923333
3  0.956667  0.633333  0.486667  1.096667  0.466667  0.670000  0.526667
4  0.853333  0.796667  0.516667  0.756667  0.683333  0.296667  0.780000

       Dem5      Dem6      Dem7      Dem8      Dem9     Cred1     Cred2  \
0  0.813333  0.693333  0.666667  0.680000  0.726667  0.606667  1.010000
1  0.463333  0.483333  0.583333  0.716667  0.743333  0.680000  0.690000
2  1.223333  0.686667  0.606667  0.690000  0.820000  0.600000  0.383333
3  0.783333  0.856667  0.716667  0.720000  0.900000  0.680000  0.846667
4  0.636667  0.783333  0.630000  0.603333  0.486667  0.693333  0.526667

      Cred3     Cred4     Cred5     Cred6  Normalised_FNT  Target
0  0.933333  0.603333  0.686667  0.673333       -245.7500       0
1  0.560000  0.670000  0.553333  0.653333       -248.0000       0
2  0.763333  0.670000  0.686667  0.673333       -233.1250       0
3  0.423333  0.520000  0.846667  0.760000       -249.7775       0
4  0.520000  0.716667  0.706667  0.673333       -247.5775       0
```

Information about data

In [19]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1424035 entries, 0 to 1424034
Data columns (total 2 columns):
 #   Column     Non-Null Count    Dtype  
---  ------     --------------    -----  
 0   id         1424035 non-null  int64  
 1   geo_score  1352492 non-null  float64
dtypes: float64(1), int64(1)
memory usage: 21.7 MB
```

In [20]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1424035 entries, 0 to 1424034
Data columns (total 2 columns):
 #   Column           Non-Null Count    Dtype  
---  ------           --------------    -----  
 0   id               1424035 non-null  int64  
 1   instance_scores  1424035 non-null  float64
dtypes: float64(1), int64(1)
memory usage: 21.7 MB
```

In [21]:
```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1400 entries, 0 to 1399
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Group      1400 non-null   object 
 1   lambda_wt  1400 non-null   float64
dtypes: float64(1), object(1)
memory usage: 22.0+ KB
```

In [22]:
```
data4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1424035 entries, 0 to 1424034
Data columns (total 2 columns):
 #   Column                Non-Null Count    Dtype  
---  ------                --------------    -----  
 0   id                    1424035 non-null  int64  
 1   qsets_normalized_tat  1320834 non-null  float64
dtypes: float64(1), int64(1)
memory usage: 21.7 MB
```

In [23]:
```
data5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56962 entries, 0 to 56961
Data columns (total 27 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              56962 non-null  int64  
 1   Group           56962 non-null  object 
 2   Per1            56962 non-null  float64
 3   Per2            56962 non-null  float64
 4   Per3            56962 non-null  float64
 5   Per4            56962 non-null  float64
 6   Per5            56962 non-null  float64
 7   Per6            56962 non-null  float64
 8   Per7            56962 non-null  float64
 9   Per8            56962 non-null  float64
 10  Per9            56962 non-null  float64
 11  Dem1            56962 non-null  float64
 12  Dem2            56962 non-null  float64
 13  Dem3            56962 non-null  float64
 14  Dem4            56962 non-null  float64
 15  Dem5            56962 non-null  float64
 16  Dem6            56962 non-null  float64
 17  Dem7            56962 non-null  float64
 18  Dem8            56962 non-null  float64
 19  Dem9            56962 non-null  float64
 20  Cred1           56962 non-null  float64
 21  Cred2           56962 non-null  float64
 22  Cred3           56962 non-null  float64
 23  Cred4           56962 non-null  float64
 24  Cred5           56962 non-null  float64
 25  Cred6           56962 non-null  float64
 26  Normalised_FNT  56962 non-null  float64
dtypes: float64(25), int64(1), object(1)
memory usage: 11.7+ MB
```

In [24]:
```
data6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227845 entries, 0 to 227844
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   id              227845 non-null  int64  
 1   Group           227845 non-null  object 
 2   Per1            227845 non-null  float64
 3   Per2            227845 non-null  float64
 4   Per3            227845 non-null  float64
 5   Per4            227845 non-null  float64
 6   Per5            227845 non-null  float64
 7   Per6            227845 non-null  float64
 8   Per7            227845 non-null  float64
 9   Per8            227845 non-null  float64
 10  Per9            227845 non-null  float64
 11  Dem1            227845 non-null  float64
 12  Dem2            227845 non-null  float64
 13  Dem3            227845 non-null  float64
 14  Dem4            227845 non-null  float64
 15  Dem5            227845 non-null  float64
 16  Dem6            227845 non-null  float64
 17  Dem7            227845 non-null  float64
 18  Dem8            227845 non-null  float64
 19  Dem9            227845 non-null  float64
 20  Cred1           227845 non-null  float64
 21  Cred2           227845 non-null  float64
 22  Cred3           227845 non-null  float64
 23  Cred4           227845 non-null  float64
 24  Cred5           227845 non-null  float64
 25  Cred6           227845 non-null  float64
 26  Normalised_FNT  227845 non-null  float64
 27  Target          227845 non-null  int64  
dtypes: float64(25), int64(2), object(1)
memory usage: 48.7+ MB
```

In [25]:
```
for name, data in dataset:
    print(30*'++')
    print(name, ' :' , data.info())
    print(30*'++')
    print(data.head())
    print()
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1424035 entries, 0 to 1424034
Data columns (total 2 columns):
 #   Column     Non-Null Count    Dtype  
---  ------     --------------    -----  
 0   id         1424035 non-null  int64  
 1   geo_score  1352492 non-null  float64
dtypes: float64(1), int64(1)
memory usage: 21.7 MB
Geo_scores  : None
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  geo_score
0   26674       4.48
1  204314       4.48
2  176521       5.17
3   48812      -2.41
4  126870       6.55
```

.nunique()

In [26]:
```
data1.nunique()
```

Out[26]:
```
id           284807
geo_score     25523
dtype: int64
```

In [27]:
```
data2.nunique()
```

Out[27]:
```
id                284807
instance_scores    11158
dtype: int64
```

In [28]:
```
data3.nunique()
```

Out[28]:
```
Group        1400
lambda_wt    1400
dtype: int64
```

In [29]:
```
data4.nunique()
```

Out[29]:
```
id                      284807
qsets_normalized_tat     24832
dtype: int64
```

In [30]:
```
data5.nunique()
```

Out[30]:
```
id                56962
Group               915
Per1               1275
Per2               1586
Per3               1095
Per4               1069
Per5               1253
Per6               1092
Per7               1210
Per8               1265
Per9                916
Dem1                699
Dem2                696
Dem3                645
Dem4                716
Dem5                658
Dem6                715
Dem7                956
Dem8                910
Dem9                605
Cred1               809
Cred2               399
Cred3               497
Cred4               388
Cred5               635
Cred6               489
Normalised_FNT    13855
dtype: int64
```

In [31]:
```
for name, data in dataset:
    print(30*'++')
    print(name, ' :' , data.nunique())
    print(30*'++')
    print(data.head())
    print()
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Geo_scores  : id           284807
geo_score     25523
dtype: int64
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  geo_score
0   26674       4.48
1  204314       4.48
2  176521       5.17
3   48812      -2.41
4  126870       6.55

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
instance_scores  : id                 284807
instance_scores     11158
dtype: int64
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  instance_scores
0  173444            -0.88
```

Checking missing values data

In [32]:
```
data1.isnull().sum()
```

Out[32]:
```
id              0
geo_score    71543
dtype: int64
```

In [33]:
```
data2.isnull().sum()
```

Out[33]:
```
id                0
instance_scores    0
dtype: int64
```

In [34]:
```
data3.isnull().sum()
```

Out[34]:
```
Group        0
lambda_wt    0
dtype: int64
```

In [35]:
```
data4.isnull().sum()
```

Out[35]:
```
id                           0
qsets_normalized_tat    103201
dtype: int64
```

In [36]:
```
data5.isnull().sum()
```

Out[36]:
```
id              0
Group           0
Per1            0
Per2            0
Per3            0
Per4            0
Per5            0
Per6            0
Per7            0
Per8            0
Per9            0
Dem1            0
Dem2            0
Dem3            0
Dem4            0
Dem5            0
Dem6            0
Dem7            0
Dem8            0
Dem9            0
Cred1           0
Cred2           0
Cred3           0
Cred4           0
Cred5           0
Cred6           0
Normalised_FNT  0
dtype: int64
```

In [37]:
```
data6.isnull().sum()
```

Out[37]:
```
id               0
Group            0
Per1             0
Per2             0
Per3             0
Per4             0
Per5             0
Per6             0
Per7             0
Per8             0
Per9             0
Dem1             0
Dem2             0
Dem3             0
Dem4             0
Dem5             0
Dem6             0
Dem7             0
Dem8             0
Dem9             0
Cred1            0
Cred2            0
Cred3            0
Cred4            0
Cred5            0
Cred6            0
Normalised_FNT   0
Target           0
dtype: int64
```

In [38]:
```
for name, data in dataset:
    print(30*'++')
    print(name, ' :' , data.isnull().sum())
    print(30*'++')
    print(data.head())
    print()
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Geo_scores  : id              0
geo_score    71543
dtype: int64
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  geo_score
0   26674       4.48
1  204314       4.48
2  176521       5.17
3   48812      -2.41
4  126870       6.55

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
instance_scores  : id                0
instance_scores    0
dtype: int64
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       id  instance_scores
0  173444            -0.88
```

In [39]:
```
# In geo_score    71543 missing values
# In gsets normalized tat    103201 missing values
```

In [ ]:
```

```

Treat the missing values

In [40]:
```
# geo
```

In [41]:
```
geo.isnull().sum()/len(geo)*100
```

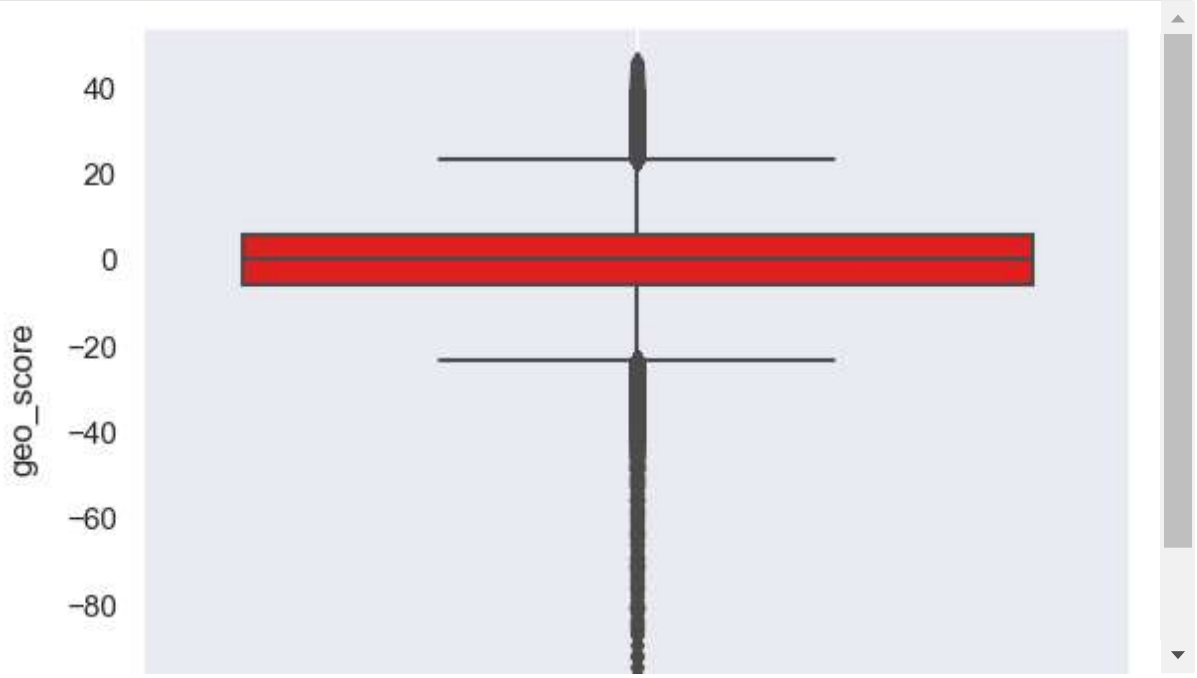
Out[41]:
```
id           0.000000
geo_score    5.023964
dtype: float64
```

In [42]:
```
geo.describe().style.background_gradient(cmap='Reds',low=0.5,high=1.0,axis=1)
```

Out[42]:

	id	geo_score
count	1424035.000000	1352492.000000
mean	142403.000000	-0.000009
std	82216.727925	7.827199
min	0.000000	-109.390000
25%	71201.000000	-5.860000
50%	142403.000000	0.180000
75%	213605.000000	5.860000
max	284806.000000	45.810000

In [43]:
```
sns.boxplot(y='geo_score', data=geo,color='red')
plt.grid();
```




```
# qsets
```

In [44]:
```
data4.isnull().sum()/len(data4)*100
```

Out[44]:
```
id                      0.000000
qsets_normalized_tat    7.247083
dtype: float64
```

In [ ]:

In [45]:
```
data.describe().style.background_gradient(cmap='Reds',low=0.5,high=1.0,axis=1)
```

Out[45]:

	id	Per1	Per2	Per3	Per4	Per
count	227845.000000	227845.000000	227845.000000	227845.000000	227845.000000	227845.00000
mean	142404.076201	0.666006	0.667701	0.666315	0.666687	0.66672
std	82170.248170	0.654133	0.548305	0.506357	0.471956	0.46139
min	0.000000	-18.136667	-23.573333	-15.443333	-1.226667	-37.24666
25%	71325.000000	0.360000	0.470000	0.370000	0.383333	0.43666
50%	142374.000000	0.670000	0.690000	0.726667	0.660000	0.65000
75%	213492.000000	1.103333	0.933333	1.010000	0.913333	0.87000
max	284805.000000	1.483333	8.020000	3.793333	6.163333	12.26666

In [46]:
```
#sns.boxplot(y='qsets', data=qsets,color='red')
#plt.grid();
```

Fill the missing values

In [47]:
```
geo['geo_score'] = geo['geo_score'].fillna(geo['geo_score'].median())
```

In [48]:
```
geo['geo_score'].isnull().sum()
```

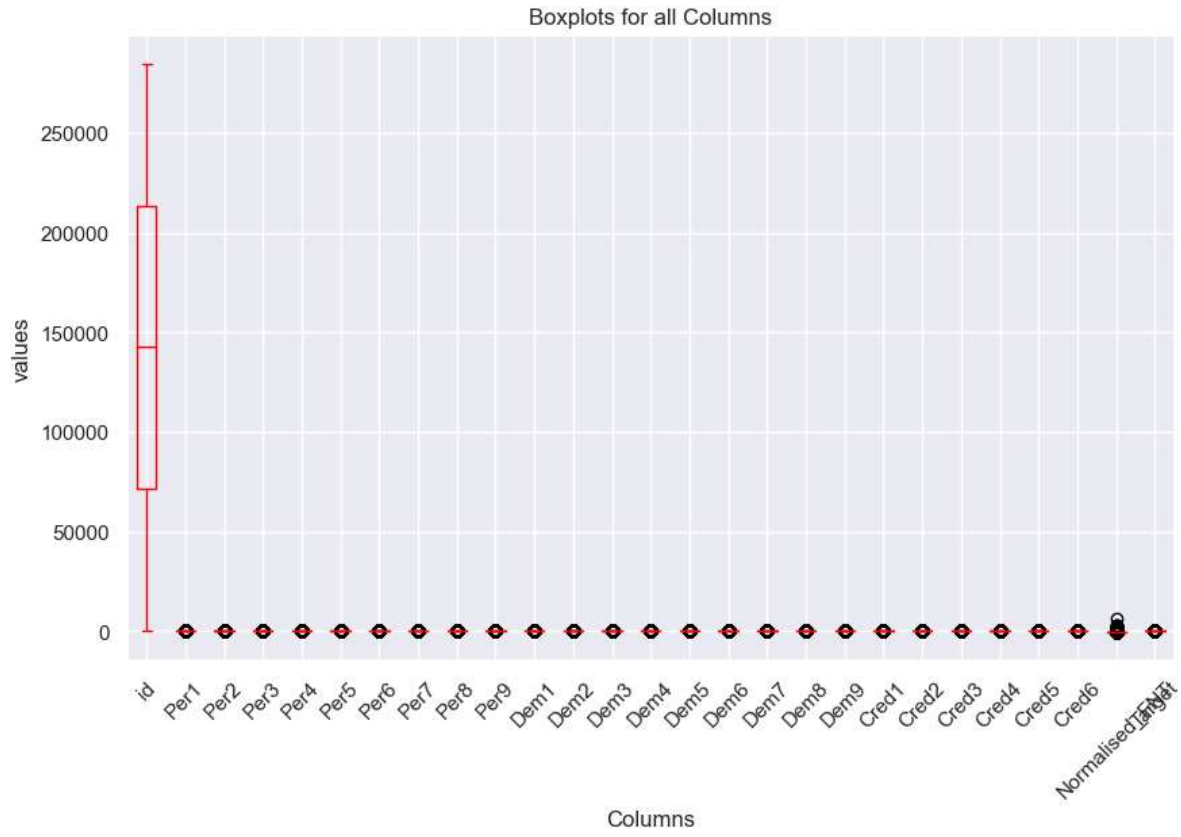
Out[48]: 0

In [49]:
```
data4['qsets normalized tat'] = data4['qsets normalized tat'].fillna(data4['qs
```

In [50]:
```
data4['qsets normalized tat'].isnull().sum()
```

Out[50]: 0

In [51]:
```
num_col = data.select_dtypes(exclude='object').columns.tolist()
plt.figure(figsize = (10,6))
data.boxplot(column = num_col,color='red')
plt.title('Boxplots for all Columns')
plt.xticks(rotation = 45)
plt.xlabel('Columns')
plt.ylabel('values')
plt.grid(True)
plt.show()
```



Check Duplicate values

In [52]:
```
data1.duplicated().sum()
```

Out[52]: 55349

In [53]:
```
data2.duplicated().sum()
```

Out[53]: 33600

In [54]:
```
data3.duplicated().sum()
```

Out[54]: 0

In [55]:
```
data4.duplicated().sum()
```

Out[55]: 59314

In [56]:
```
data5.duplicated().sum()
```

Out[56]: 0

In [57]:
```
data6.duplicated().sum()
```

Out[57]: 0

In [58]:
```
for name, data in dataset:
    print(' '*15,'  Duplicate entries ', name, ' : ', data.duplicated().sum())
    print()
```

```
                Duplicate entries  Geo_scores  :  55349

                Duplicate entries  instance_scores  :  33600

                Duplicate entries  Lambda_wts  :  0

                Duplicate entries  Qset_tats  :  59314

                Duplicate entries  Test  :  0

                Duplicate entries  Train  :  0
```

drop duplicates from train and test

In [59]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227845 entries, 0 to 227844
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   id              227845 non-null  int64  
 1   Group           227845 non-null  object 
 2   Per1            227845 non-null  float64
 3   Per2            227845 non-null  float64
 4   Per3            227845 non-null  float64
 5   Per4            227845 non-null  float64
 6   Per5            227845 non-null  float64
 7   Per6            227845 non-null  float64
 8   Per7            227845 non-null  float64
 9   Per8            227845 non-null  float64
 10  Per9            227845 non-null  float64
 11  Dem1            227845 non-null  float64
 12  Dem2            227845 non-null  float64
 13  Dem3            227845 non-null  float64
 14  Dem4            227845 non-null  float64
 15  Dem5            227845 non-null  float64
 16  Dem6            227845 non-null  float64
 17  Dem7            227845 non-null  float64
 18  Dem8            227845 non-null  float64
 19  Dem9            227845 non-null  float64
 20  Cred1           227845 non-null  float64
 21  Cred2           227845 non-null  float64
 22  Cred3           227845 non-null  float64
 23  Cred4           227845 non-null  float64
 24  Cred5           227845 non-null  float64
 25  Cred6           227845 non-null  float64
 26  Normalised_FNT  227845 non-null  float64
 27  Target          227845 non-null  int64  
dtypes: float64(25), int64(2), object(1)
memory usage: 48.7+ MB
```

In [60]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56962 entries, 0 to 56961
Data columns (total 27 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              56962 non-null  int64  
 1   Group           56962 non-null  object 
 2   Per1            56962 non-null  float64
 3   Per2            56962 non-null  float64
 4   Per3            56962 non-null  float64
 5   Per4            56962 non-null  float64
 6   Per5            56962 non-null  float64
 7   Per6            56962 non-null  float64
 8   Per7            56962 non-null  float64
 9   Per8            56962 non-null  float64
 10  Per9            56962 non-null  float64
 11  Dem1            56962 non-null  float64
 12  Dem2            56962 non-null  float64
 13  Dem3            56962 non-null  float64
 14  Dem4            56962 non-null  float64
 15  Dem5            56962 non-null  float64
 16  Dem6            56962 non-null  float64
 17  Dem7            56962 non-null  float64
 18  Dem8            56962 non-null  float64
 19  Dem9            56962 non-null  float64
 20  Cred1           56962 non-null  float64
 21  Cred2           56962 non-null  float64
 22  Cred3           56962 non-null  float64
 23  Cred4           56962 non-null  float64
 24  Cred5           56962 non-null  float64
 25  Cred6           56962 non-null  float64
 26  Normalised_FNT  56962 non-null  float64
dtypes: float64(25), int64(1), object(1)
memory usage: 11.7+ MB
```

In [61]:
```
print("geo id :", data1['id'].nunique())
print("**********"*10)
print("instance id:", data2['id'].nunique())
print("**********"*10)
print("lambdawts Group :", data3['Group'].nunique())
print("**********"*10)
print("qset id :", data4['id'].nunique())
print("**********"*10)
print("test - id :", data5['id'].nunique())
print("**********"*10)
print("test - Group :", data6['Group'].nunique())
print("**********"*10)
print("train - id :", train['id'].nunique())
print("**********"*10)
print("train - Group :", train['Group'].nunique())
```

```
geo id : 284807
**********************************************************************
**********
instance id: 284807
**********************************************************************
**********
lambdawts Group : 1400
**********************************************************************
**********
qset id : 284807
**********************************************************************
**********
test - id : 56962
**********************************************************************
**********
test - Group : 1301
**********************************************************************
**********
train - id : 227845
**********************************************************************
**********
train - Group : 1301
```

In [62]:
```
train['data'] ='train'
test['data'] = 'test'
```

In [63]:
```
train.head()
```

Out[63]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

In [64]:
```
train.head()
```

Out[64]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100(
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

Check Columns

In [65]:
```
train.columns
```

Out[65]:
```
Index(['id', 'Group', 'Per1', 'Per2', 'Per3', 'Per4', 'Per5', 'Per6', 'Per7',
       'Per8', 'Per9', 'Dem1', 'Dem2', 'Dem3', 'Dem4', 'Dem5', 'Dem6', 'Dem
7',
       'Dem8', 'Dem9', 'Cred1', 'Cred2', 'Cred3', 'Cred4', 'Cred5', 'Cred6',
       'Normalised_FNT', 'Target', 'data'],
      dtype='object')
```

In [66]:
```
test.columns
```

Out[66]:
```
Index(['id', 'Group', 'Per1', 'Per2', 'Per3', 'Per4', 'Per5', 'Per6', 'Per7',
       'Per8', 'Per9', 'Dem1', 'Dem2', 'Dem3', 'Dem4', 'Dem5', 'Dem6', 'Dem
7',
       'Dem8', 'Dem9', 'Cred1', 'Cred2', 'Cred3', 'Cred4', 'Cred5', 'Cred6',
       'Normalised_FNT', 'data'],
      dtype='object')
```

Check shape

In [67]:
```
print(train.shape)
print(test.shape)
```

```
(227845, 29)
(56962, 28)
```

Concat train and test data

In [68]:
```
all_data = pd.concat([train, test], axis=0)
```

In [69]:
```
all_data.head()
```

Out[69]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100(
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

In [70]:
```
all_data.tail()
```

Out[70]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	
56957	18333	Grp102	0.553333	1.043333	1.096667	0.686667	0.673333	0.340000	0.900000	0
56958	244207	Grp504	1.353333	0.616667	0.276667	0.783333	0.690000	0.650000	0.473333	0
56959	103277	Grp78	1.083333	0.433333	0.806667	0.490000	0.243333	0.316667	0.533333	0
56960	273294	Grp134	0.566667	1.153333	0.370000	0.616667	0.793333	0.226667	0.910000	0
56961	223337	Grp18	1.426667	0.110000	-0.006667	-0.200000	0.983333	1.870000	0.033333	0

Check null values

In [71]:
```
all_data.isnull().sum()
```

Out[71]:
```
id                   0
Group                0
Per1                 0
Per2                 0
Per3                 0
Per4                 0
Per5                 0
Per6                 0
Per7                 0
Per8                 0
Per9                 0
Dem1                 0
Dem2                 0
Dem3                 0
Dem4                 0
Dem5                 0
Dem6                 0
Dem7                 0
Dem8                 0
Dem9                 0
Cred1                0
Cred2                0
Cred3                0
Cred4                0
Cred5                0
Cred6                0
Normalised_FNT       0
Target           56962
data                 0
dtype: int64
```

In [72]:
```
print("geo id :", data1['id'].nunique())
print("**********"*10)
print("instance id:", data2['id'].nunique())
print("**********"*10)
print("lambdawts Group :", data3['Group'].nunique())
print("**********"*10)
print("qset id :", data4['id'].nunique())
print("**********"*10)
print("all_data id :", all_data['id'].nunique())
print("**********"*10)
print("all_data Group", all_data['Group'].nunique())
```

```
geo id : 284807
**********************************************************************
**********
instance id: 284807
**********************************************************************
**********
lambdawts Group : 1400
**********************************************************************
**********
qset id : 284807
**********************************************************************
**********
all_data id : 284807
**********************************************************************
**********
all_data Group 1400
```

Check all_data shape

In [73]:
```
all_data.shape
```

Out[73]: (284807, 29)

check geo Column

In [74]:
```
geo = geo.groupby('id').mean()
```

In [75]:
```
geo.shape
```

Out[75]: (284807, 1)

In [76]:
```
geo.head()
```

Out[76]:

id	geo_score
0	-0.620
1	1.106
2	0.070
3	0.180
4	0.540

In [77]:
```
# Merge all_data with geo
```

In [78]:
```
all_data = pd.merge(all_data, geo, on='id', how='left')
```

In [79]:
```
all_data.head()
```

Out[79]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

In [80]:
```
all_data.shape
```

Out[80]: (284807, 30)

In [ ]:

check instance Column

In [81]:
```
data2['id'].nunique()
```

Out[81]: 284807

In [82]:
```
data2.shape
```

Out[82]: (1424035, 2)

In [83]:
```
data2 = data2.groupby('id').mean()
```

In [84]:
```
data2.shape
```

Out[84]: (284807, 1)

In [85]:
```
# Merge all_data with instance
```

In [86]:
```
all_data = pd.merge(all_data, data2, on='id', how='left')
```

In [87]:
```
all_data.shape
```

Out[87]: (284807, 31)

In [88]:
```
all_data.head()
```

Out[88]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

check lambdawts Column

In [89]:
```
data3['Group'].nunique()
```

Out[89]: 1400

In [90]:
```
data3.shape
```

Out[90]: (1400, 2)

In [91]:
```
all_data.shape
```

Out[91]: (284807, 31)

In [92]:
```
all_data['Group'].nunique()
```

Out[92]: 1400

In [93]:
```
# Merge all data with lambdawts
```

In [94]:
```
all_data = pd.merge(all_data, data3, on='Group', how='left')
```

In [95]:
```
all_data.shape
```

Out[95]: (284807, 32)

In [96]:
```
all_data.head()
```

Out[96]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100(
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

Check qset Column

In [97]:
```
data4['id'].nunique()
```

Out[97]: 284807

In [98]:
```
data4.shape
```

Out[98]: (1424035, 2)

In [99]:
```
data4 = data4.groupby('id').mean()
```

In [100]:
```
all_data.shape
```

Out[100]: (284807, 32)

In [101]:
```
all_data['id'].nunique()
```

Out[101]: 284807

In [102]:
```
# Merge all data with lambdawts
```

In [103]:
```
all_data = pd.merge(all_data, data4, on='id', how='left')
```

In [104]:
```
all_data.head()
```

Out[104]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100(
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

In [105]:
```
all_data.shape
```

Out[105]:
```
(284807, 33)
```

In [106]:
```
all_data.hist(bins=60,figsize=(25,30),color='purple')
plt.show()
```

In [ ]:
```
```

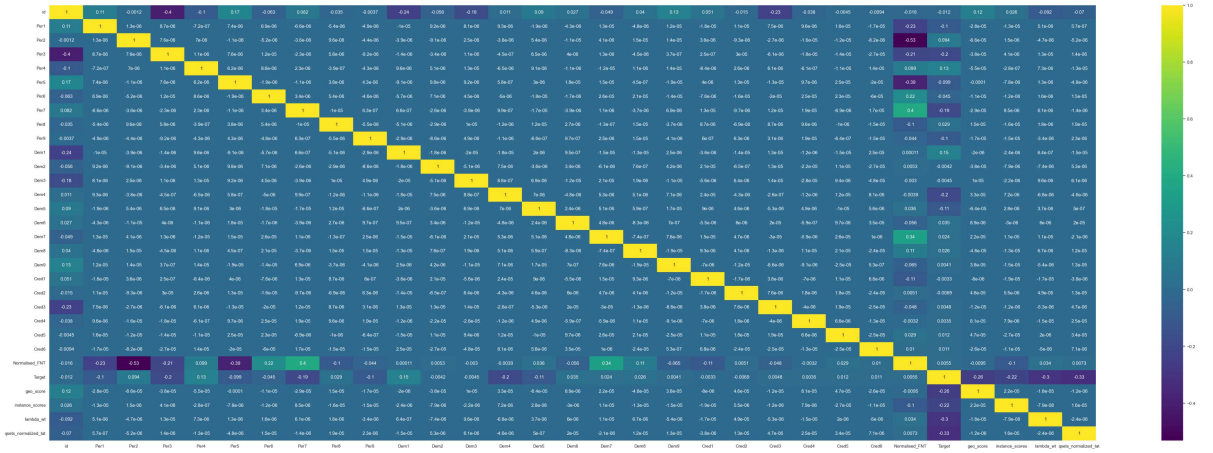
split the train and test data seperately

In [107]:
```
train = all_data[all_data['data']=='train']
test = all_data[all_data['data']=='test']
```

In [108]:
```
print(train.shape)
print(test.shape)
```

```
(227845, 33)
(56962, 33)
```

In [109]:
```
plt.figure(figsize=(60,20))
corr = all_data.corr()
sns.heatmap(corr, annot=True, cmap='viridis')
plt.show()
```



Target - train dataset

split the data into ind variable and dependent variable

In [110]:
```
train.head()
```

Out[110]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Pe
0	112751	Grp169	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.0100(
1	18495	Grp161	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.7833
2	23915	Grp261	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.7566
3	50806	Grp198	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.6333
4	184244	Grp228	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.7966

In [111]:
```
Fraud = train[train['Target']==1]
Valid = train[train['Target']==0]
outlier_fraction = (len(Fraud)/(len(train)))*100
print(outlier_fraction)
```

```
0.17292457591783889
```

In [112]:
```
print(len(Fraud))
print(len(Valid))
```

```
394
227451
```

In [113]:
```
x = train.drop(['id','Group','Target','data'], axis=1)
y = train[['Target']]
```

In [114]:
```
x.head()
```

Out[114]:

	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Per8	Per9	
0	1.070000	0.580000	0.480000	0.766667	1.233333	1.993333	0.340000	1.010000	0.863333	0.46
1	0.473333	1.206667	0.883333	1.430000	0.726667	0.626667	0.810000	0.783333	0.190000	0.47
2	1.130000	0.143333	0.946667	0.123333	0.080000	0.836667	0.056667	0.756667	0.226667	0.66
3	0.636667	1.090000	0.750000	0.940000	0.743333	0.346667	0.956667	0.633333	0.486667	1.09
4	0.560000	1.013333	0.593333	0.416667	0.773333	0.460000	0.853333	0.796667	0.516667	0.75

In [115]:
```
y.head()
```

Out[115]:

	Target
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0

Target - test dataset

split the data into ind variable and dependent variable

In [116]:
```
test.head()
```

Out[116]:

	id	Group	Per1	Per2	Per3	Per4	Per5	Per6	Per7
227845	146574	Grp229	-0.300000	1.540000	0.220000	-0.280000	0.570000	0.260000	0.700000
227846	268759	Grp141	0.633333	0.953333	0.810000	0.466667	0.910000	0.253333	1.040000
227847	59727	Grp188	1.043333	0.740000	0.860000	1.006667	0.583333	0.616667	0.630000
227848	151544	Grp426	1.283333	0.300000	0.576667	0.636667	0.256667	0.543333	0.356667
227849	155008	Grp443	1.186667	0.326667	0.476667	0.866667	0.436667	0.680000	0.476667

In [117]:
```
test = test.drop(['id', 'Group', 'Target', 'data'], axis=1)
```

In [118]:
```
test.head()
```

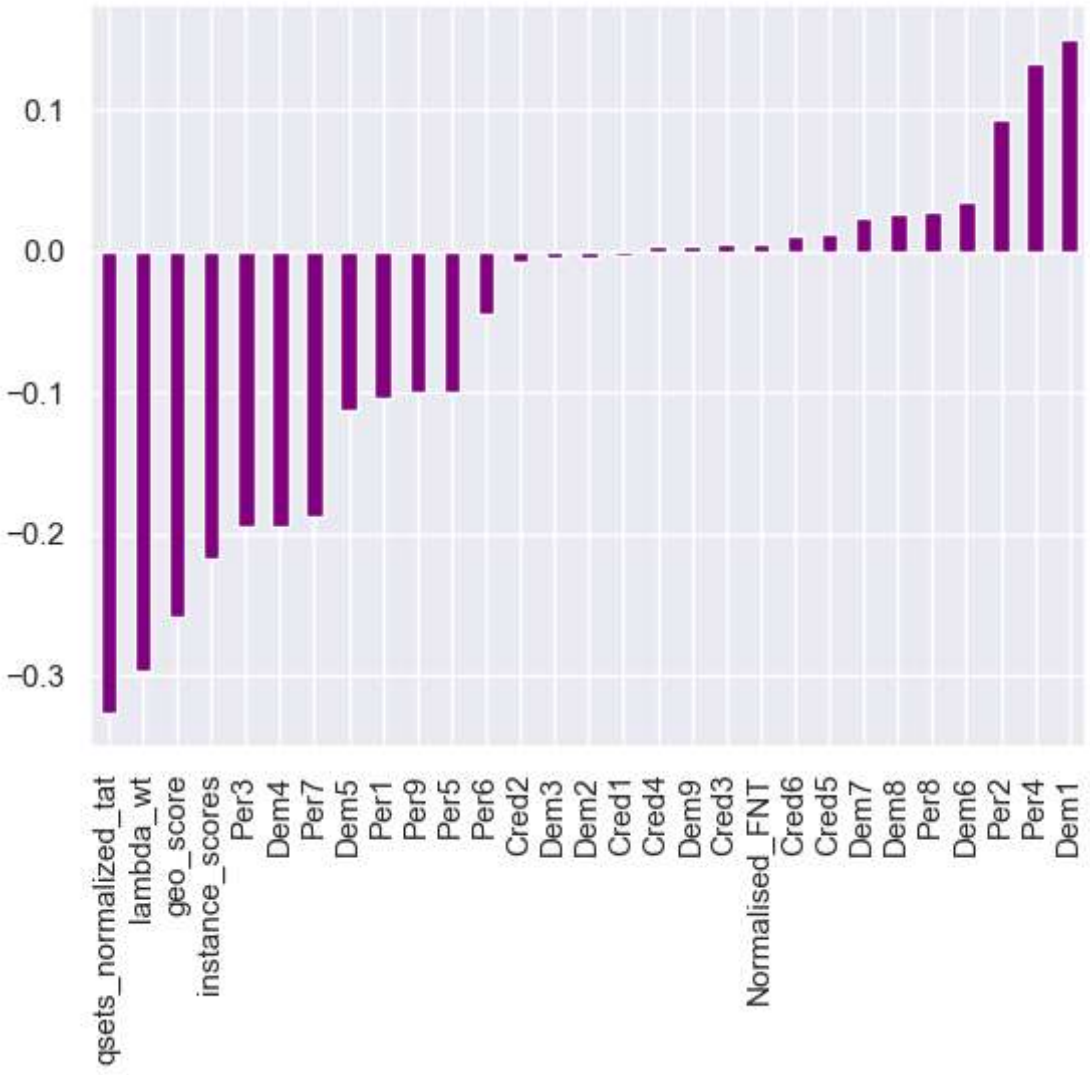
Out[118]:

	Per1	Per2	Per3	Per4	Per5	Per6	Per7	Per8	Pe
227845	-0.300000	1.540000	0.220000	-0.280000	0.570000	0.260000	0.700000	1.076667	0.9300
227846	0.633333	0.953333	0.810000	0.466667	0.910000	0.253333	1.040000	0.550000	0.5433
227847	1.043333	0.740000	0.860000	1.006667	0.583333	0.616667	0.630000	0.686667	0.5933
227848	1.283333	0.300000	0.576667	0.636667	0.256667	0.543333	0.356667	0.663333	1.1566
227849	1.186667	0.326667	0.476667	0.866667	0.436667	0.680000	0.476667	0.686667	1.4766

In [119]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 227845 entries, 0 to 227844
Data columns (total 29 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   Per1                  227845 non-null  float64
 1   Per2                  227845 non-null  float64
 2   Per3                  227845 non-null  float64
 3   Per4                  227845 non-null  float64
 4   Per5                  227845 non-null  float64
 5   Per6                  227845 non-null  float64
 6   Per7                  227845 non-null  float64
 7   Per8                  227845 non-null  float64
 8   Per9                  227845 non-null  float64
 9   Dem1                  227845 non-null  float64
 10  Dem2                  227845 non-null  float64
 11  Dem3                  227845 non-null  float64
 12  Dem4                  227845 non-null  float64
 13  Dem5                  227845 non-null  float64
 14  Dem6                  227845 non-null  float64
 15  Dem7                  227845 non-null  float64
 16  Dem8                  227845 non-null  float64
 17  Dem9                  227845 non-null  float64
 18  Cred1                 227845 non-null  float64
 19  Cred2                 227845 non-null  float64
 20  Cred3                 227845 non-null  float64
 21  Cred4                 227845 non-null  float64
 22  Cred5                 227845 non-null  float64
 23  Cred6                 227845 non-null  float64
 24  Normalised_FNT        227845 non-null  float64
 25  geo_score             227845 non-null  float64
 26  instance_scores       227845 non-null  float64
 27  lambda_wt             227845 non-null  float64
 28  qsets_normalized_tat  227845 non-null  float64
dtypes: float64(29)
memory usage: 52.1 MB
```

In [120]:
```
all_data.iloc[:,1:].corr()['Target'].sort_values().head(29).plot(kind='bar',co
plt.show()
```



Data Preprocessing

1) missing value imputation completed

2) Encoding not required as there is no char variable

3) Outlier Treatement - not required in this dataset

4) Feature Scaling - completed

In [ ]:

```
```

In [121]:

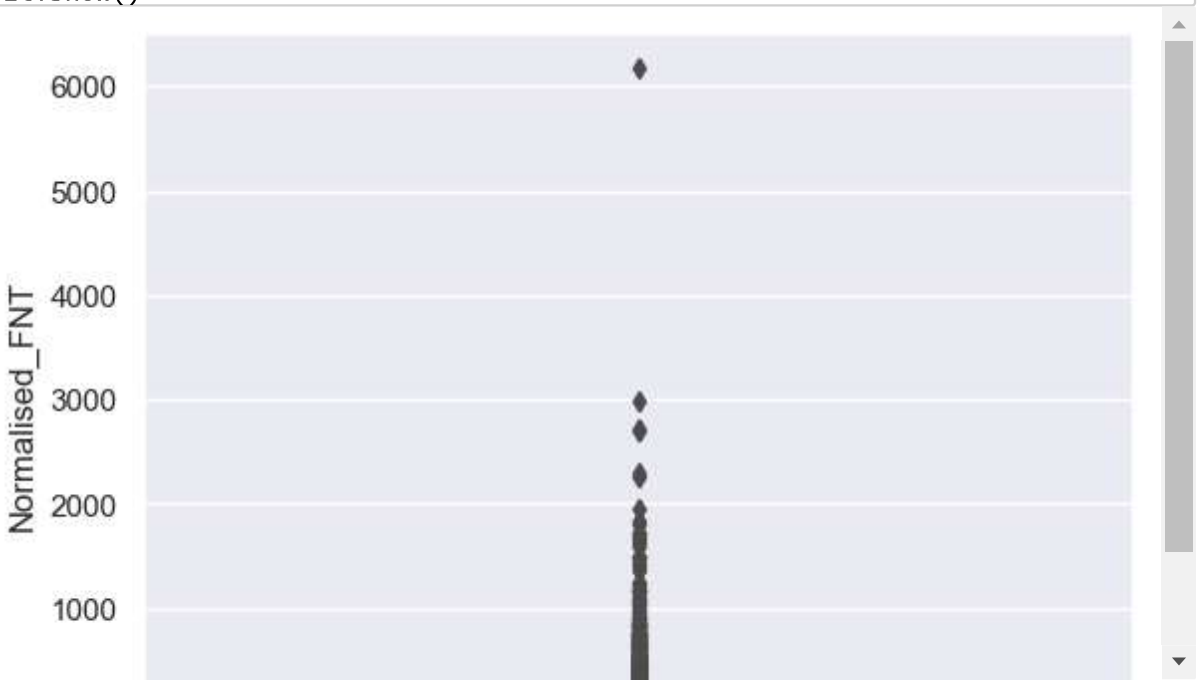
```
x.describe().style.background_gradient(cmap='Reds',low=0.5,high=1.0,axis=1)
```

Out[121]:

	Per1	Per2	Per3	Per4	Per5	Per
count	227845.000000	227845.000000	227845.000000	227845.000000	227845.000000	227845.00000
mean	0.666006	0.667701	0.666315	0.666687	0.666723	0.66737
std	0.654133	0.548305	0.506357	0.471956	0.461393	0.44457
min	-18.136667	-23.573333	-15.443333	-1.226667	-37.246667	-8.05333
25%	0.360000	0.470000	0.370000	0.383333	0.436667	0.41000
50%	0.670000	0.690000	0.726667	0.660000	0.650000	0.57666
75%	1.103333	0.933333	1.010000	0.913333	0.870000	0.80000
max	1.483333	8.020000	3.793333	6.163333	12.266667	25.10000

In [122]:

```
sns.boxplot(y='Normalised_FNT', data=x)
plt.show()
```



In [123]:

```
IQR = -230.750000 + 248.617500
IQR
```

Out[123]:

```
17.867500000000007
```

In [124]:
```
x['Normalised_FNT'].describe()
```

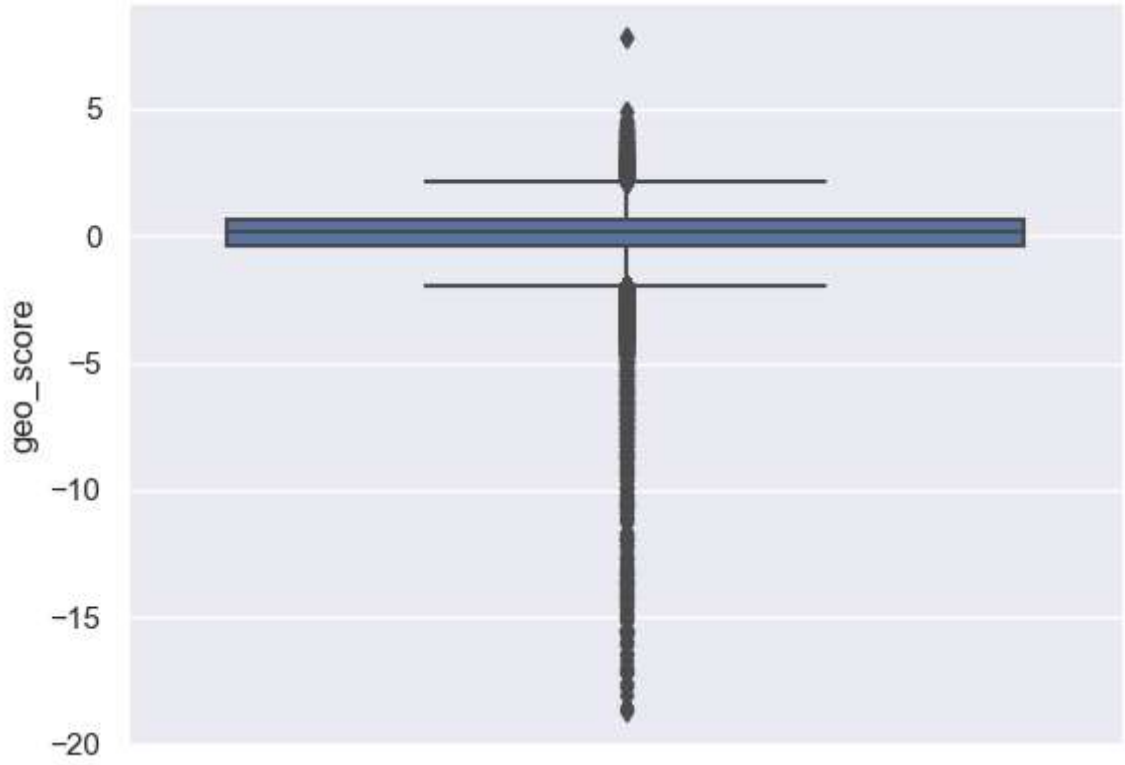
Out[124]:
```
count    227845.000000
mean       -227.954170
std          61.951661
min        -250.000000
25%        -248.617500
50%        -244.510000
75%        -230.750000
max        6172.790000
Name: Normalised_FNT, dtype: float64
```

In [125]:
```
# pos_outlier_range = Q3 + 1.5*IQR
pos_outlier_range = -230.750000 + (1.5*IQR)
pos_outlier_range
```

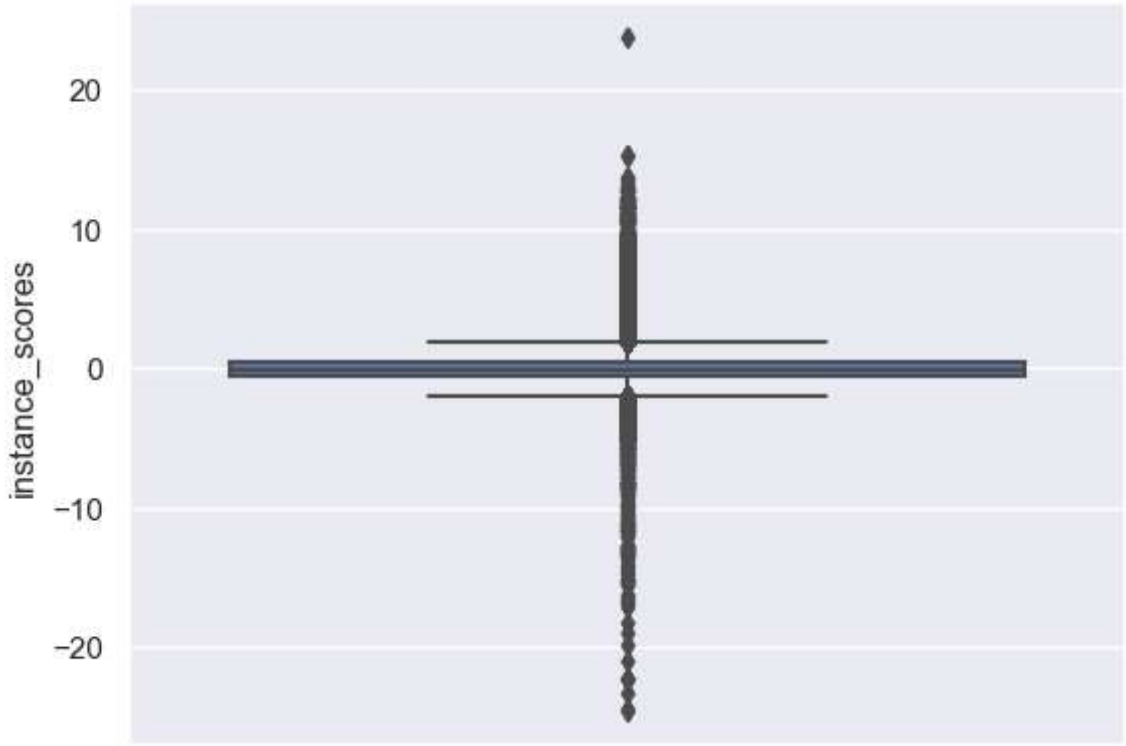
Out[125]:
```
-203.94875
```

In [126]:
```
# Holding capping method right now as positive outlier range is -203.94 which
```

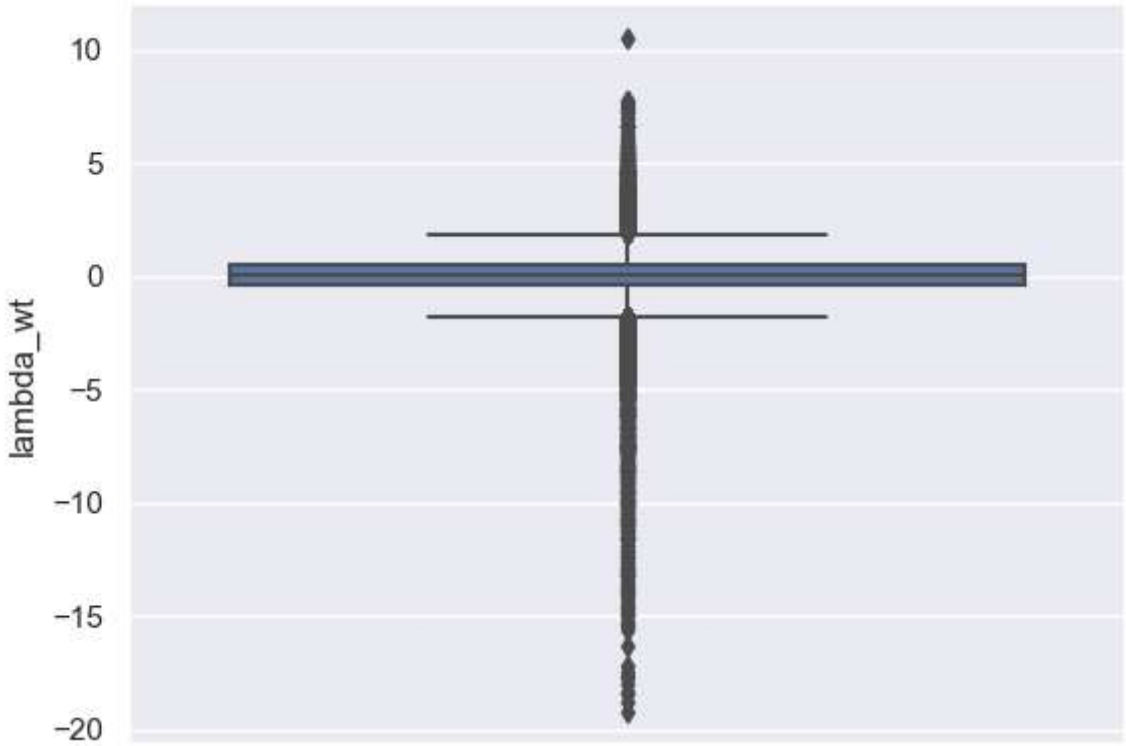
In [127]:
```
sns.boxplot(y='geo_score', data=x)
plt.show()
```



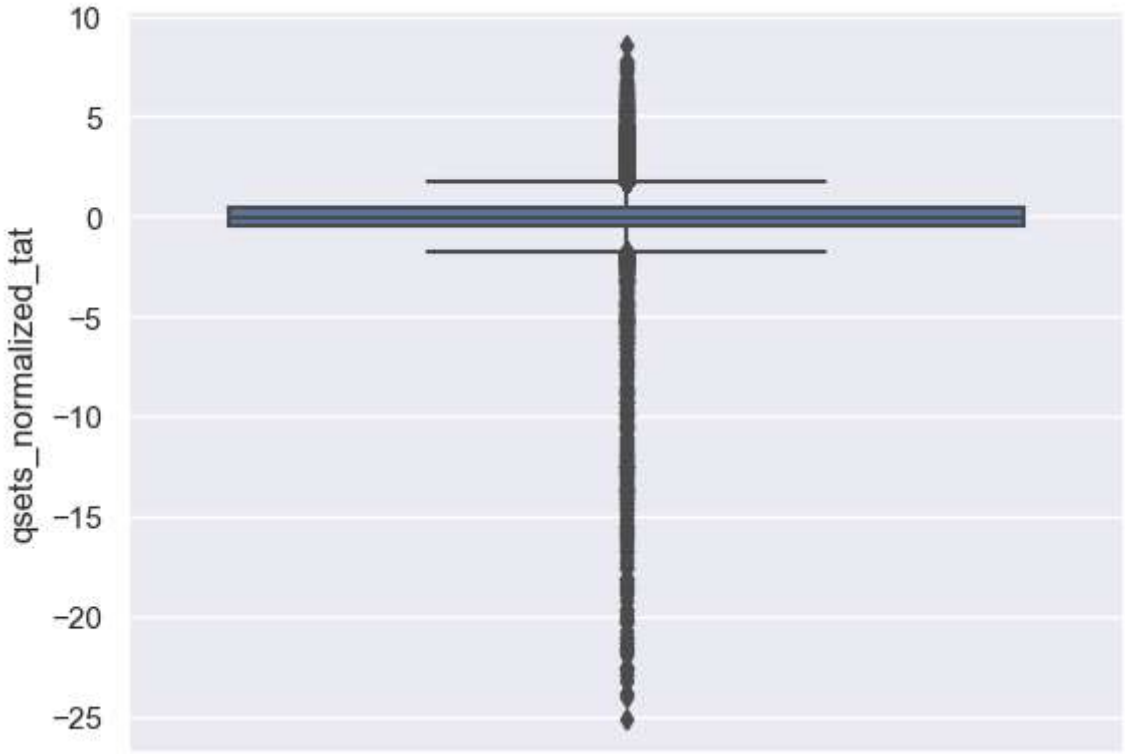
In [128]:
```
sns.boxplot(y='instance_scores', data=x)
plt.show()
```



In [129]:
```
sns.boxplot(y='lambda_wt', data=x)
plt.show()
```



In [130]:
```
sns.boxplot(y='qsets_normalized_tat', data=x)
plt.show()
```



Feature scaling

In [131]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc_x = sc.fit_transform(x)
```

In [132]:
```
pd.DataFrame(sc_x).describe()
```

Out[132]:

	0	1	2	3	4	5
count	2.278450e+05	2.278450e+05	2.278450e+05	2.278450e+05	2.278450e+05	2.278450e+05
mean	2.469880e-16	-1.035354e-16	2.169253e-16	3.414173e-16	-1.947214e-16	-2.275907e-16
std	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00
min	-2.874447e+01	-4.421098e+01	-3.181486e+01	-4.011726e+00	-8.217170e+01	-1.961595e+01
25%	-4.678047e-01	-3.605683e-01	-5.851915e-01	-6.003821e-01	-4.986139e-01	-5.789326e-01
50%	6.105872e-03	4.066939e-02	1.191877e-01	-1.416771e-02	-3.624526e-02	-2.040406e-01
75%	6.685615e-01	4.844626e-01	6.787413e-01	5.226069e-01	4.405724e-01	2.983147e-01
max	1.249484e+00	1.340918e+01	6.175532e+00	1.164655e+01	2.514117e+01	5.495757e+01

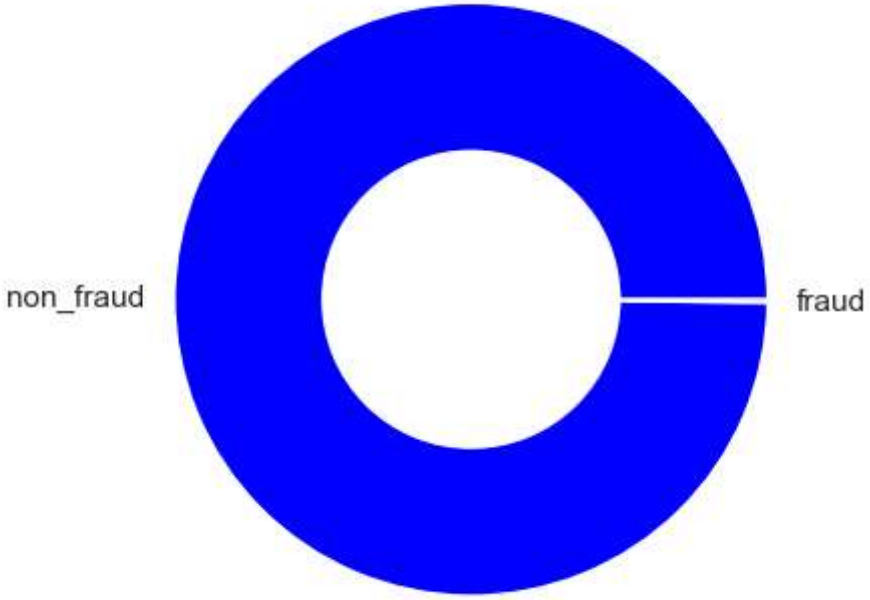
Cheak imbalance dataset

In [133]:
```
y.value_counts()
```

Out[133]:
```
Target
0.0       227451
1.0          394
dtype: int64
```

In [ ]:
```
```

In [134]:
```
target=['non_fraud','fraud']
quantity=[227451,394]
plt.pie(quantity,labels=target,radius=1,colors=['blue','red'])
plt.pie([1],colors=['w'],radius=0.5)
plt.show()
```



In [135]:
```
x.shape
```

Out[135]:
```
(227845, 29)
```

Since data is imbalance, so we can build model with both aproach

1) balance the data and perform model building

2) model building with balance the data

In [136]:
```
import imblearn
from imblearn.over_sampling import SMOTE
ros = SMOTE()
x_ros, y_ros = ros.fit_resample(sc_x, y)
print(y.value_counts())
print(y_ros.value_counts())
```

```
Target
0.0       227451
1.0          394
dtype: int64
Target
0.0       227451
1.0       227451
dtype: int64
```

Split the data into training and testing for model building

In [137]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.
```

model use

Logistic Regression

Decision Tree

Randam Forest Classification

XGBoost Classifier

Logistic Regression

In [138]:
```
from sklearn.linear_model import LogisticRegression
```

In [139]:
```
lr = LogisticRegression()
lr.fit(x_train, y_train)
```

Out[139]:
```
LogisticRegression
LogisticRegression()
```

In [140]:
```
# fitting the model
lr.fit(x_train,y_train)
```

Out[140]:
```
LogisticRegression
LogisticRegression()
```

In [141]:
```
# predicting the test set result
lr_pred = lr.predict(x_test)
```

In [142]:
```
# Training accuracy
print('Training accuracy  ',lr.score(x_train,y_train))
```

```
Training accuracy   0.941320608718081
```

In [143]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confusion_matrix(y_test,lr_pred), annot=labels, fmt='')
```

Out[143]: <Axes: >

Decision Tree

In [144]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [145]:
```
dtc=DecisionTreeClassifier(criterion='entropy')
dtc.fit(x_train, y_train)
```

Out[145]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [146]:
```
## fitting the model
dtc.fit(x_train,y_train)
```

Out[146]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

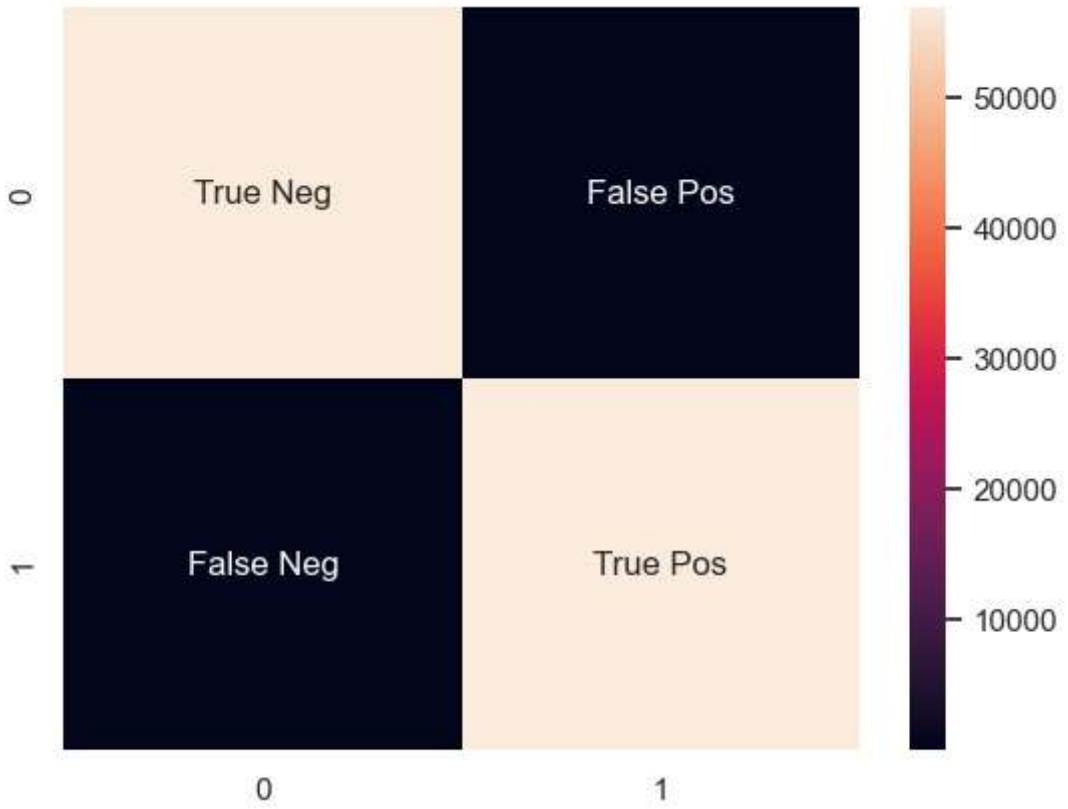
In [147]:
```
# predicting the test set result
dtc_pred = dtc.predict(x_test)
```

In [148]:
```
# Training accuracy
print('Training accuracy  ',dtc.score(x_train,y_train))
```

```
Training accuracy   1.0
```

In [149]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confusion_matrix(y_test,dtc_pred), annot=labels, fmt='')
```

Out[149]:
```
<Axes: >
```



Randam Forest Classification

In [150]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [151]:
```
rfc=DecisionTreeClassifier(criterion='entropy')
rfc.fit(x_train, y_train)
```

Out[151]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [152]:
```
## fitting the model
rfc.fit(x_train,y_train)
```

Out[152]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

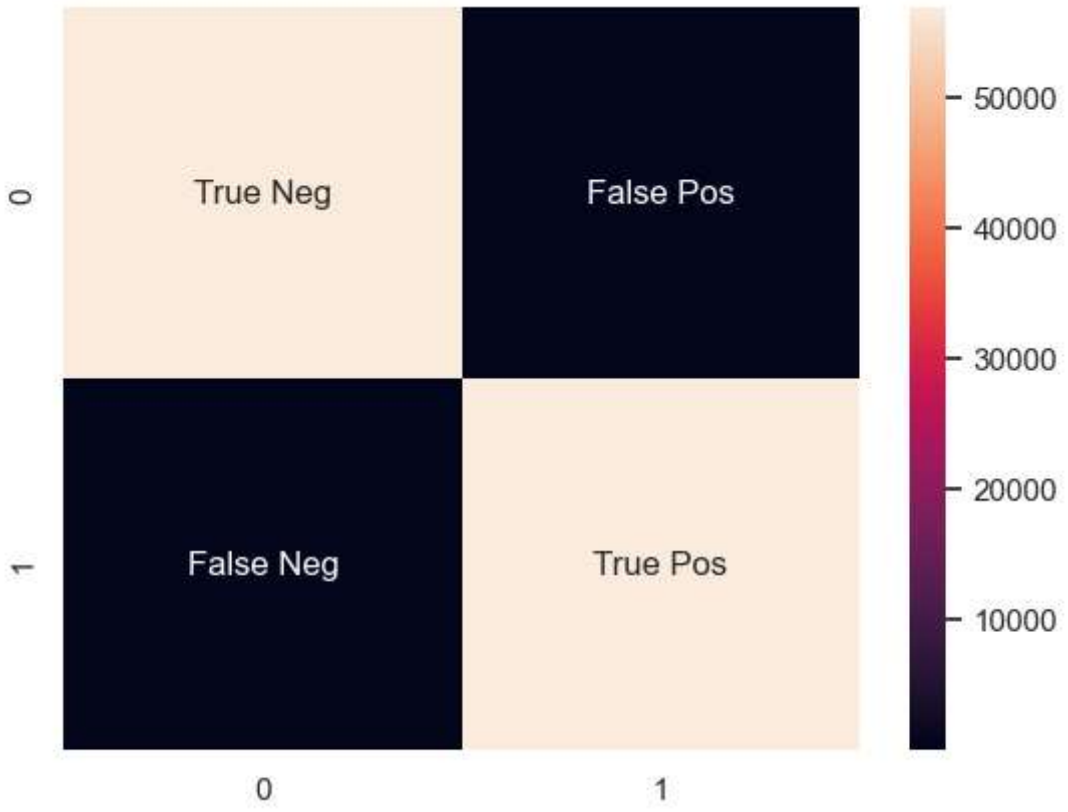
In [153]:
```
# predicting the test set result
rfc_pred = rfc.predict(x_test)
```

In [154]:
```
# Training accuracy
print('Training accuracy  ',rfc.score(x_train,y_train))
```

```
Training accuracy   1.0
```

In [155]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confusion_matrix(y_test,rfc_pred), annot=labels, fmt='')
```

Out[155]: <Axes: >



XGBoost Classifier

In [156]:
```
from xgboost import XGBClassifier
```

In [168]:
```
xgb = XGBClassifier()
#xgb.fit(x_train, y_train)
```

In [158]:
```
## fitting the model
xgb.fit(x_train,y_train)
```

Out[158]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=No
ne,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=N
one,
              interaction_constraints=None, learning_rate=None, max_bin=No
ne,
              max_cat_threshold=None, max_cat_to_onehot=None,
```

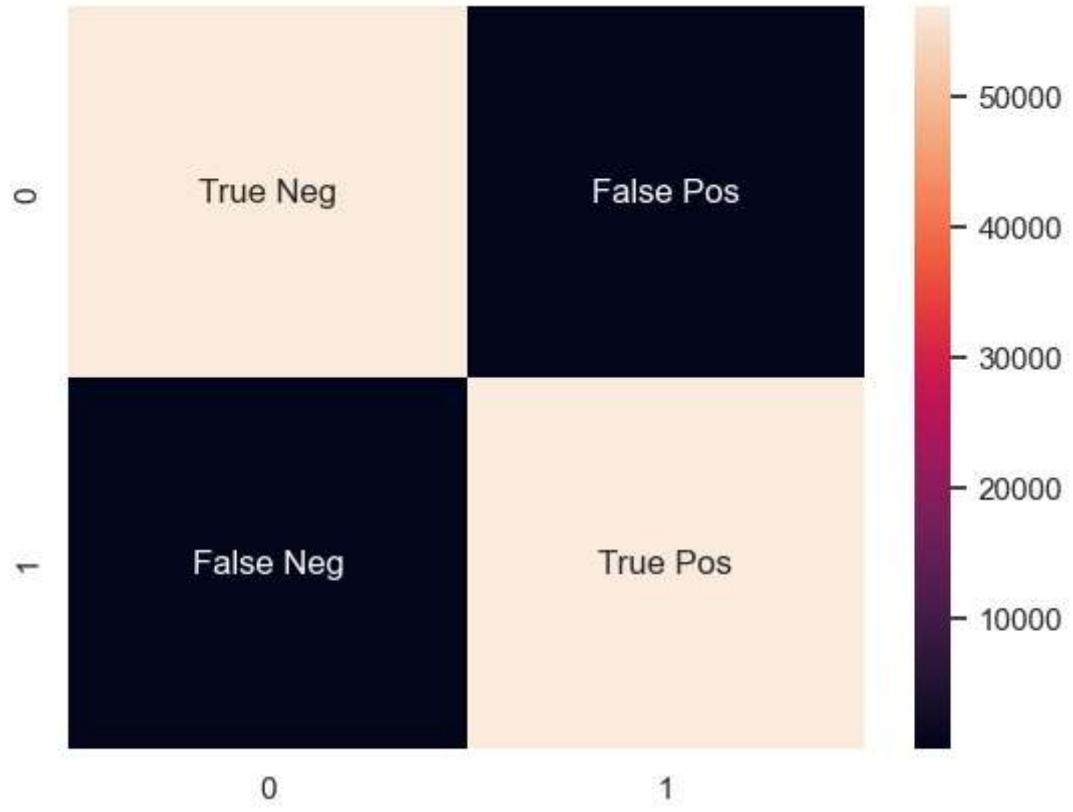
In [159]:
```
# predicting the test set result
xgb_pred = xgb.predict(x_test)
```

In [160]:
```
# Training accuracy
print('Training accuracy  ',xgb.score(x_train,y_train))
```

```
Training accuracy   0.9999970689614744
```

In [161]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confusion_matrix(y_test,xgb_pred), annot=labels, fmt='')
```

Out[161]: <Axes: >



Classification Report

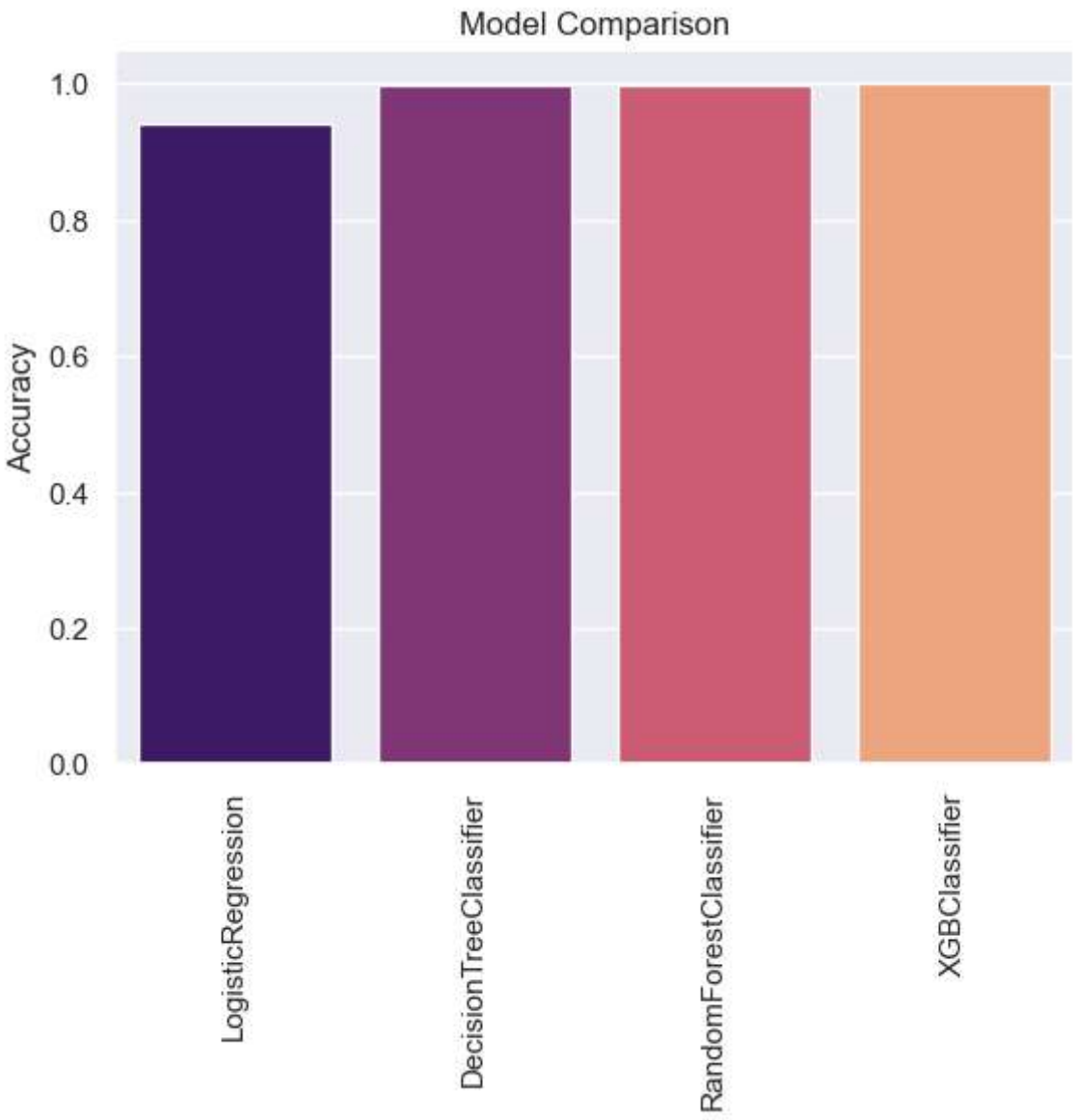
In [163]:
```
print('LogisticRegression: \n',classification_report(y_test,lr_pred))
print('DecisionTreeClassifier: \n',classification_report(y_test,dtc_pred))
print('RandomForestClassifier: \n',classification_report(y_test,rfc_pred))
print('XGBClassifier: \n',classification_report(y_test,xgb_pred))
```

```
LogisticRegression:
               precision    recall  f1-score   support

         0.0       0.91      0.97      0.94     56824
         1.0       0.97      0.91      0.94     56902

    accuracy                           0.94    113726
   macro avg       0.94      0.94      0.94    113726
weighted avg       0.94      0.94      0.94    113726

DecisionTreeClassifier:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     56824
         1.0       1.00      1.00      1.00     56902

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726

RandomForestClassifier:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     56824
         1.0       1.00      1.00      1.00     56902

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726

XGBClassifier:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     56824
         1.0       1.00      1.00      1.00     56902

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

Model Comparison

In [167]:
```
model = ['LogisticRegression','DecisionTreeClassifier','RandomForestClassifier
accuracy = [accuracy_score(y_test,lr_pred),accuracy_score(y_test,dtc_pred),acc
sns.barplot(x = model,y = accuracy,palette = 'magma').set_title('Model Compari
plt.xticks(rotation=90)
plt.ylabel('Accuracy')
```

Out[167]:
```
Text(0, 0.5, 'Accuracy')
```



Conclusion

In [ ]:

```
Fraudulent transactions : 394
Clean transactions: 227451
************************
1. LogisticRegression:0.94
2. DecisionTreeClassifier:1.0
3. RandomForestClassifier:1.0
4. XGBClassifier:0.99
```

In [ ]:

In [ ]: