Appendix 1 - Python

Week 5

Name : Ayachit Madhukar

Course : DSC630

Instructor : Fadi Alsaleem

Date : 25 Sep 2021

Import

In [185…

```
# Importing required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

import pandas_profiling as pp
```

In [73]:

```
import sys
# installing pandas-profiing
#!{sys.executable} -m pip install pandas-profiling
```

Data

In [74]:

```
# Load Source Data
datafile='Data/er_data.txt'
df = pd.read_csv(datafile,sep="|")
df.head()
```

Out[74]:

	AGE	SEX	RACE_ETHNICITY	PLAN_TYPE	STATE_CODE	PLAN_REGION	COMPLEXCARE_IND	MMP_
0	38.0	F	White	MARKETPLACE	FL	SOUTHEAST	0	
1	81.0	M	White	MEDICAID	NY	NORTHEAST	1	
2	30.0	F	White	MARKETPLACE	TX	SOUTHWEST	0	
3	88.0	F	White	MEDICARE	TX	SOUTHWEST	0	
4	1.0	F	Hispanic	MEDICAID	NE	MIDDLESTATES	0	

5 rows × 46 columns

In [75]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69000 entries, 0 to 68999
Data columns (total 46 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   AGE                        69000 non-null  float64
 1   SEX                        69000 non-null  object
 2   RACE_ETHNICITY             69000 non-null  object
 3   PLAN_TYPE                  69000 non-null  object
 4   STATE_CODE                 69000 non-null  object
 5   PLAN_REGION                69000 non-null  object
 6   COMPLEXCARE_IND            69000 non-null  int64
 7   MMP_DUAL_IND               69000 non-null  int64
 8   DUAL_PRODUCT_IND           69000 non-null  int64
 9   LTC_IND                    69000 non-null  int64
 10  MEDICAID_ELIGIBLE          69000 non-null  int64
 11  MEDICARE_ELIGIBLE          69000 non-null  int64
 12  BEHAVIORAL_ELIGIBLE        69000 non-null  int64
 13  COMMERCIAL_ELIGIBLE        69000 non-null  int64
 14  OTHER_ELIGIBLE             69000 non-null  int64
 15  RISK_TYPE_DESC             6891 non-null   object
 16  MEMBER_MONTHS_PRE          68998 non-null  float64
 17  ADD_STATE                  68113 non-null  object
 18  COUNTY_CLEAN               50817 non-null  object
 19  REG_REGION_DESC            69000 non-null  object
 20  RISK_SCORE                 68935 non-null  float64
 21  PRIOR_TOTAL_COSTS_ANNUAL   68935 non-null  float64
 22  PRIOR_RX_COSTS_ANNUAL      68935 non-null  float64
 23  ANNUAL_IP_COSTS            68935 non-null  float64
 24  ANNUAL_ER_COSTS            68935 non-null  float64
 25  ANNUAL_OTHER_COSTS         68935 non-null  float64
 26  FUTURE_RISK_INPATIENT      68935 non-null  float64
 27  BH_RISK_SCORE              68935 non-null  float64
 28  RX_RISK_SCORE              68935 non-null  float64
 29  ER_RISK_SCORE              68935 non-null  float64
 30  ORCA_SCORE                 65600 non-null  float64
 31  ORCA_RISK_GROUP            65600 non-null  object
 32  SUD_SEG_VALUE              68935 non-null  float64
 33  SUD_SEG_DEF                68935 non-null  object
 34  ENG_SCORE                  68935 non-null  float64
 35  POPHEALTHCAT_GROUPED       69000 non-null  object
 36  INTERVENABLE_IND           69000 non-null  int64
 37  SHORT_DESC                 68935 non-null  object
 38  SHORT_DESC_2               69000 non-null  object
 39  RISK_CAT_RECODE            68935 non-null  object
 40  MEDICAID_CLAIMS            69000 non-null  int64
 41  MEDICARE_CLAIMS            69000 non-null  int64
 42  BEHAVIORAL_CLAIMS          69000 non-null  int64
 43  COMMERCIAL_CLAIMS          69000 non-null  int64
 44  OTHER_CLAIMS               69000 non-null  int64
 45  MORE_THAN_4_ER_VISITS      69000 non-null  int64
dtypes: float64(15), int64(16), object(15)
memory usage: 24.2+ MB
```

VARIABLE DEFINITION

AGE
The age of the patient at the time the data was gathered

SEX
The Gender of the patient (Male or Female)

RACE_ETHNICITY
The race or ethnicity of the patient

PLAN_TYPE
The type of plan or benefit the patient is on such as medicaid, medicare, marketplace (ObamaCare) or Commerical Insurance

STATE_CODE
The State in which the patient gets benefits from

PLAN_REGION
The region of the U.S the patient lives in: Midwest, Southwest....

COMPLEXCARE_IND
Specify whether the patient is deemed to require complex care services

MMP_DUAL_IND
Specify whether the patient has both medicare and medicaid coverage

DUAL_PRODUCT_IND
Specify whether the patient has more than one public benefit, such social security, TANF, food stamps...

LTC_IND
Specify whether the patient has long term care needs

MEDICAID_ELIGIBLE Specify whether the patient is eligible for medicaid

MEDICARE_ELIGIBLE
specify whether the patient is eligible for medicare

BEHAVIORAL_ELIGIBLE
specify whether the patient is eligibile for behavioral health services

COMMERCIAL_ELIGIBLE
specify whether the patient is eligible for health coverage through an employer

OTHER_ELIGIBLE
Specify whether the patient has some other type of medical coverages

RISK_TYPE_DESC
he type of risk that the patient represent to their health plan, specify whether the insurer takes on the full risk, or share the risk

MEMBER_MONTHS_PRE
The total number of months the member has coverage during the previous 12 months

ADD_STATE
The state in which the patient lives

COUNTY_CLEAN
The county in which the patient lives if available

REG_REGION_DESC
The regio in which the patient lives

RISK_SCORE
The overall health risk score attributed to the patient. The higher the score the worse the patient

PRIOR_TOTAL_COSTS_ANNUAL
The total medical or healthcare cost incurred by the patients during the prior year

PRIOR_RX_COSTS_ANNUAL
The total Pharmacy or drugs cost incurred by the patients during the prior year

ANNUAL_IP_COSTS
The total inpatient or hospitalization cost incurred by the patients during the prior year

ANNUAL_ER_COSTS
The total emergency room (ER) cost incurred by the patients during the prior year

ANNUAL_OTHER_COSTS
All other medical services cost incurred by the patients during the prior year

FUTURE_RISK_INPATIENT
A score that's designed to be predictive of the future risk of hospitalization of the patient

BH_RISK_SCORE A score that's designed to be predictive of the future risk of behavioral health needs of the patient

RX_RISK_SCORE A score that's designed to be predictive of the future medication needs of the patient

ER_RISK_SCORE A score that's designed to be predictive of the future emergency care needs of the patient

ORCA_SCORE Opioid risk classification algorithm/ The likelihood of the patient abusing opioid

ORCA_RISK_GROUP A grouping of the patient based on the ORCA score

SUD_SEG_VALUE The substance use disorder segment that the member belongs to

SUD_SEG_DEF A definition of the SUD_SEG_VALUE

ENG_SCORE The likelihood of the member successfully completing a care management program

POPHEALTHCAT_GROUPED

The population health category that the patient belongs to based on their medical history

INTERVENABLE_IND Specify whether the patient is likely to benefit from an intervention

SHORT_DESC Description of the condition(s) that the patient might be suffering from

SHORT_DESC_2 Description of the condition(s) that the patient might be suffering from

RISK_CAT_RECODE A grouping of the type of healthcare needs the patient requires

MEDICAID_CLAIMS The total number of healthcare or medical claims that the patients incurred using medicaid

MEDICARE_CLAIMS The total number of healthcare or medical claims that the patients incurred using medicare

BEHAVIORAL_CLAIMS The total number of healthcare or medical claims that the patients incurred using behavioral health coverage

COMMERCIAL_CLAIMS The total number of healthcare or medical claims that the patients incurred using commercial or employer coverage

OTHER_CLAIMS The total number of all other healthcare or medical claims that the patients incurred

***MORE_THAN_4_ER_VISITS** Specify whether or not the patient has had 4 or more ER visits previously **(This is the target to predict)**.

In [77]:
```
df.shape
```

Out[77]:
```
(69000, 46)
```

Identifying and Handling Non Numerical data

In [76]:
```
df.describe(include="O").columns
```

Out[76]:
```
Index(['SEX', 'RACE_ETHNICITY', 'PLAN_TYPE', 'STATE_CODE', 'PLAN_REGION',
       'RISK_TYPE_DESC', 'ADD_STATE', 'COUNTY_CLEAN', 'REG_REGION_DESC',
       'ORCA_RISK_GROUP', 'SUD_SEG_DEF', 'POPHEALTHCAT_GROUPED', 'SHORT_DESC',
       'SHORT_DESC_2', 'RISK_CAT_RECODE'],
      dtype='object')
```

In [179…
```
object_columns=['SEX', 'RACE_ETHNICITY', 'PLAN_TYPE', 'STATE_CODE', 'PLAN_REGION',
        'RISK_TYPE_DESC', 'ADD_STATE', 'COUNTY_CLEAN', 'REG_REGION_DESC',
        'ORCA_RISK_GROUP', 'SUD_SEG_DEF', 'POPHEALTHCAT_GROUPED', 'SHORT_DESC',
        'SHORT_DESC_2', 'RISK_CAT_RECODE']
```

In [79]:

```
###  Handling Non Numerical data using Label Encoder

from sklearn import preprocessing
labelencoder = preprocessing.LabelEncoder()
cleaned_df=df
for c in object_columns:
    cleaned_df[c]=labelencoder.fit_transform(cleaned_df[c])

cleaned_df
```

Out[79]:

	AGE	SEX	RACE_ETHNICITY	PLAN_TYPE	STATE_CODE	PLAN_REGION	COMPLEXCARE_IND	MM
0	38.0	0	6	4	6	3	0	
1	81.0	1	6	5	26	1	1	
2	30.0	0	6	4	33	4	0	
3	88.0	0	6	6	33	4	0	
4	1.0	0	3	5	21	0	0	
...	...	...	...	...	...	...	...	
68995	0.0	1	6	5	13	3	1	
68996	0.0	1	5	5	9	0	0	
68997	0.0	1	5	5	10	0	1	
68998	0.0	1	6	5	27	0	0	
68999	0.0	1	5	5	10	0	0	

69000 rows × 46 columns

In [80]:

```
cleaned_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69000 entries, 0 to 68999
Data columns (total 46 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   AGE                       69000 non-null  float64
 1   SEX                       69000 non-null  int64  
 2   RACE_ETHNICITY            69000 non-null  int64  
 3   PLAN_TYPE                 69000 non-null  int64  
 4   STATE_CODE                69000 non-null  int64  
 5   PLAN_REGION               69000 non-null  int64  
 6   COMPLEXCARE_IND           69000 non-null  int64  
 7   MMP_DUAL_IND              69000 non-null  int64  
 8   DUAL_PRODUCT_IND          69000 non-null  int64  
 9   LTC_IND                   69000 non-null  int64  
 10  MEDICAID_ELIGIBLE         69000 non-null  int64  
 11  MEDICARE_ELIGIBLE         69000 non-null  int64  
 12  BEHAVIORAL_ELIGIBLE       69000 non-null  int64  
 13  COMMERCIAL_ELIGIBLE       69000 non-null  int64  
 14  OTHER_ELIGIBLE            69000 non-null  int64  
 15  RISK_TYPE_DESC            69000 non-null  int64  
 16  MEMBER_MONTHS_PRE         68998 non-null  float64
 17  ADD_STATE                 69000 non-null  int64  
 18  COUNTY_CLEAN            69000 non-null  int64
 19  REG_REGION_DESC         69000 non-null  int64
 20  RISK_SCORE              68935 non-null  float64
 21  PRIOR_TOTAL_COSTS_ANNUAL  68935 non-null  float64
 22  PRIOR_RX_COSTS_ANNUAL   68935 non-null  float64
 23  ANNUAL_IP_COSTS         68935 non-null  float64
 24  ANNUAL_ER_COSTS         68935 non-null  float64
 25  ANNUAL_OTHER_COSTS      68935 non-null  float64
 26  FUTURE_RISK_INPATIENT   68935 non-null  float64
 27  BH_RISK_SCORE           68935 non-null  float64
 28  RX_RISK_SCORE           68935 non-null  float64
 29  ER_RISK_SCORE           68935 non-null  float64
 30  ORCA_SCORE              65600 non-null  float64
 31  ORCA_RISK_GROUP         69000 non-null  int64
 32  SUD_SEG_VALUE           68935 non-null  float64
 33  SUD_SEG_DEF             69000 non-null  int64
 34  ENG_SCORE               68935 non-null  float64
 35  POPHEALTHCAT_GROUPED    69000 non-null  int64
 36  INTERVENABLE_IND        69000 non-null  int64
 37  SHORT_DESC              69000 non-null  int64
 38  SHORT_DESC_2            69000 non-null  int64
 39  RISK_CAT_RECODE         69000 non-null  int64
 40  MEDICAID_CLAIMS         69000 non-null  int64
 41  MEDICARE_CLAIMS         69000 non-null  int64
 42  BEHAVIORAL_CLAIMS       69000 non-null  int64
 43  COMMERCIAL_CLAIMS       69000 non-null  int64
 44  OTHER_CLAIMS            69000 non-null  int64
 45  MORE_THAN_4_ER_VISITS   69000 non-null  int64
dtypes: float64(15), int64(31)
memory usage: 24.2 MB
```

Identifying Null values and replacing it with median

In [92]:

```python
#looking for null values
s=cleaned_df.isnull().sum()
s=s[s!=0]
s
```

Out[92]:

```
MEMBER_MONTHS_PRE            2
RISK_SCORE                  65
PRIOR_TOTAL_COSTS_ANNUAL    65
PRIOR_RX_COSTS_ANNUAL       65
ANNUAL_IP_COSTS             65
ANNUAL_ER_COSTS             65
ANNUAL_OTHER_COSTS          65
FUTURE_RISK_INPATIENT       65
BH_RISK_SCORE               65
RX_RISK_SCORE               65
ER_RISK_SCORE               65
ORCA_SCORE                3400
SUD_SEG_VALUE               65
ENG_SCORE                   65
dtype: int64
```

In [180…

```python
# replacing null with median value
Null_columns=['MEMBER_MONTHS_PRE','RISK_SCORE','PRIOR_TOTAL_COSTS_ANNUAL','PRIOR_RX_COS
for c in Null_columns:
    median = cleaned_df[c].median()
    cleaned_df[c].fillna(median, inplace=True)

cleaned_df.isnull().sum()
```

Out[180…

```
AGE                         0
SEX                         0
RACE_ETHNICITY              0
PLAN_TYPE                   0
STATE_CODE                  0
PLAN_REGION                 0
COMPLEXCARE_IND             0
MMP_DUAL_IND                0
DUAL_PRODUCT_IND            0
LTC_IND                     0
MEDICAID_ELIGIBLE           0
MEDICARE_ELIGIBLE           0
BEHAVIORAL_ELIGIBLE         0
COMMERCIAL_ELIGIBLE         0
OTHER_ELIGIBLE              0
RISK_TYPE_DESC              0
MEMBER_MONTHS_PRE           0
ADD_STATE                   0
COUNTY_CLEAN                0
REG_REGION_DESC             0
RISK_SCORE                  0
PRIOR_TOTAL_COSTS_ANNUAL    0
PRIOR_RX_COSTS_ANNUAL       0
ANNUAL_IP_COSTS             0
ANNUAL_ER_COSTS             0
ANNUAL_OTHER_COSTS          0
FUTURE_RISK_INPATIENT       0
BH_RISK_SCORE               0
RX_RISK_SCORE               0
ER_RISK_SCORE               0
ORCA_SCORE                  0
ORCA_RISK_GROUP             0
SUD_SEG_VALUE               0
SUD_SEG_DEF                 0
ENG_SCORE                   0
POPHEALTHCAT_GROUPED        0
INTERVENABLE_IND            0
SHORT_DESC                  0
SHORT_DESC_2                0
RISK_CAT_RECODE             0
MEDICAID_CLAIMS             0
MEDICARE_CLAIMS             0
BEHAVIORAL_CLAIMS           0
COMMERCIAL_CLAIMS           0
OTHER_CLAIMS                0
MORE_THAN_4_ER_VISITS       0
dtype: int64
```

Exploration

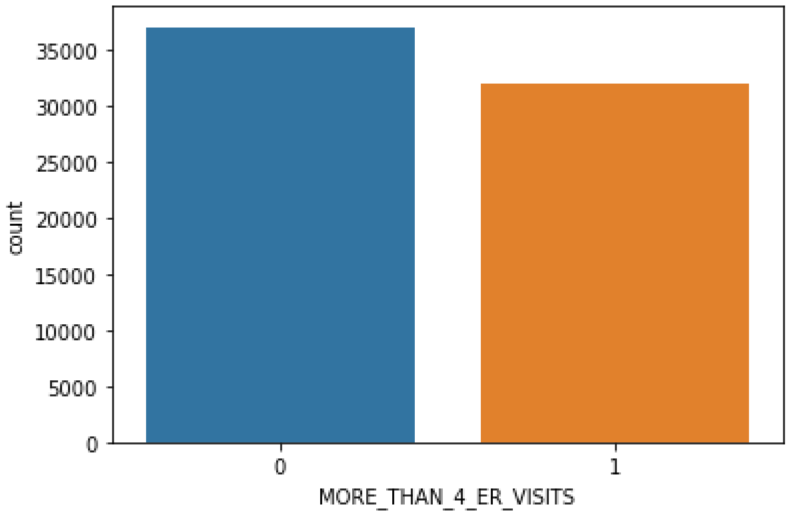
In [246…

```python
# exiting count breakup  of 4+ ER Visits

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

ax = sns.countplot(cleaned_df.MORE_THAN_4_ER_VISITS,label="Count")
print(y.value_counts())
```

```
0    37000
```

```
1    32000
Name: MORE_THAN_4_ER_VISITS, dtype: int64
```

```
/Users/madhukarayachit/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:
36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Outlier Detection and cleaning

In [247…

```python
# Outlier detection
import scipy.stats as stats
#Internally studentized method (z-score)
def z_score_method(df, variable_name):
    #Takes two parameters: dataframe & variable of interest as string
    columns = df.columns
    z = np.abs(stats.zscore(df))
    threshold = 3
    outlier = []
    index=0
    for item in range(len(columns)):
        if columns[item] == variable_name:
            index = item
    for i, v in enumerate(z[:, index]):
        if v > threshold:
            outlier.append(i)
        else:
            continue
    return outlier

outlier_z = z_score_method(cleaned_df, 'AGE')
for c in cleaned_df.columns:
    outlier_z = z_score_method(cleaned_df, c)

    if (len(outlier_z)>0):
        print (len(outlier_z) , ' outliers in ' , c)
        print(cleaned_df[c].iloc[outlier_z])

        # replacing outlier with median value
        median =cleaned_df[c].median()
        cleaned_df[c].iloc[outlier_z] = np.nan
        cleaned_df.fillna(median,inplace=True)
```

```
/Users/madhukarayachit/opt/anaconda3/lib/python3.8/site-packages/scipy/stats/stats.py:25
00: RuntimeWarning: invalid value encountered in true_divide
  return (a - mns) / sstd
31  outliers in  PLAN_TYPE
37       3.0
110      3.0
457      3.0
488      3.0
1222     3.0
1691     3.0
2056     3.0
2621     3.0
3366     3.0
5790     3.0
6354     3.0
6857     3.0
7351     3.0
7799     3.0
7822     3.0
9005     3.0
9278     3.0
10319    3.0
11494    3.0
12533    3.0
16542    3.0
17146    3.0
22581    3.0
25822    3.0
26634    3.0
26786    3.0
26927    3.0
31305    3.0
50998    3.0
51010    3.0
51063    3.0
Name: PLAN_TYPE, dtype: float64
/Users/madhukarayachit/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.p
y:670: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value)
3514  outliers in  RISK_TYPE_DESC
17       2.0
25       3.0
30       2.0
55       3.0
61       3.0
        ... 
68836    3.0
68864    3.0
68879    2.0
68884    3.0
68922    2.0
Name: RISK_TYPE_DESC, Length: 3514, dtype: float64
1753  outliers in  RISK_SCORE
24       15.4592
195      15.4639
326      18.4911
356      16.5013
458      15.5998
          ...   
68788    20.9538
```

```
68877    21.2681
68951    15.8066
68975    14.6363
68985    15.0308
Name: RISK_SCORE, Length: 1753, dtype: float64
2077  outliers in  PRIOR_TOTAL_COSTS_ANNUAL
13        84033.83
24        73343.52
183      137381.44
200      100995.35
276       83579.71
           ...
68771     73661.84
68788     71369.40
68902    117157.42
68926     70520.05
68998    121874.55
Name: PRIOR_TOTAL_COSTS_ANNUAL, Length: 2077, dtype: float64
1655  outliers in  PRIOR_RX_COSTS_ANNUAL
127      56771.11
162      23609.76
195      54173.52
200      19538.45
260      26276.60
           ...
68109    29542.89
68113    19338.72
68238    34305.35
68711    42646.47
68877    19945.49
Name: PRIOR_RX_COSTS_ANNUAL, Length: 1655, dtype: float64
1843  outliers in  ANNUAL_IP_COSTS
25       22955.83
28       61275.36
207      36366.40
244      59605.56
329      51881.23
           ...
68926    23370.55
68975    57147.82
68981    22823.47
68982    26982.38
68998    36512.64
Name: ANNUAL_IP_COSTS, Length: 1843, dtype: float64
1777  outliers in  ANNUAL_ER_COSTS
819      6715.20
1123     6397.73
1300     4627.05
1547     4837.70
2146     5797.05
          ...
68695    4958.63
68788    6101.40
68822    4430.98
68823    4596.71
68914    4273.62
Name: ANNUAL_ER_COSTS, Length: 1777, dtype: float64
1995  outliers in  ANNUAL_OTHER_COSTS
323      40842.62
413      62465.00
519      50696.49
584      40204.66
620      43321.34
           ...
68743    58468.65
```

```
68788    51178.95
68877    45421.84
68926    46607.97
68934    48342.40
Name: ANNUAL_OTHER_COSTS, Length: 1995, dtype: float64
2569  outliers in  FUTURE_RISK_INPATIENT
50       17.6432
200      23.8625
224      19.7088
286      16.0005
374      20.9119
          ...
67929    18.7353
67957    16.4742
68113    20.7151
68474    22.6004
68794    22.8755
Name: FUTURE_RISK_INPATIENT, Length: 2569, dtype: float64
2280  outliers in  BH_RISK_SCORE
93       28.568
131      34.478
407      31.145
553      25.013
570      34.642
          ...
67840    31.442
67957    35.285
67973    27.400
68065    32.369
68872    33.550
Name: BH_RISK_SCORE, Length: 2280, dtype: float64
1749  outliers in  RX_RISK_SCORE
195      12.9797
234      11.0840
268      17.2066
286      13.4253
318      11.2428
          ...
68531    11.7058
68542    11.1915
68649    12.7739
68757    11.8065
68951    15.1222
Name: RX_RISK_SCORE, Length: 1749, dtype: float64
1679  outliers in  ER_RISK_SCORE
891      22.8315
1052     23.1255
1234     22.8801
1689     25.2189
2082     23.2170
          ...
67973    23.7476
68045    26.1962
68335    22.8101
68572    23.0872
68984    24.6212
Name: ER_RISK_SCORE, Length: 1679, dtype: float64
1921  outliers in  SUD_SEG_VALUE
13       2.0
78       2.0
2481     2.0
2546     2.0
2565     2.0
        ...
67581    2.0
```

```
67599    2.0
67700    2.0
67957    2.0
68579    2.0
Name: SUD_SEG_VALUE, Length: 1921, dtype: float64
4232  outliers in  SUD_SEG_DEF
66       2.0
135      2.0
161      2.0
217      2.0
247      2.0
        ...
68093    2.0
68105    2.0
68794    2.0
68872    2.0
68917    2.0
Name: SUD_SEG_DEF, Length: 4232, dtype: float64
```

In [102…

```python
columns = np.full((cleaned_df.corr().shape[0],), True, dtype=bool)
for i in range(cleaned_df.corr().shape[0]):
    for j in range(i+1, cleaned_df.corr().shape[0]):
        if cleaned_df.corr().iloc[i,j] >= 0.9:
            if columns[j]:
                columns[j] = False
selected_columns = cleaned_df.columns[columns]
data = cleaned_df[selected_columns]
data
```

Out[102…

	AGE	SEX	RACE_ETHNICITY	PLAN_TYPE	STATE_CODE	PLAN_REGION	COMPLEXCARE_IND	MM
0	38.0	0	6	4.0	6	3	0	
1	81.0	1	6	5.0	26	1	1	
2	30.0	0	6	4.0	33	4	0	
3	88.0	0	6	6.0	33	4	0	
4	1.0	0	3	5.0	21	0	0	
...	...	...	...	...	...	...	...	
68995	0.0	1	6	5.0	13	3	1	
68996	0.0	1	5	5.0	9	0	0	
68997	0.0	1	5	5.0	10	0	1	
68998	0.0	1	6	5.0	27	0	0	
68999	0.0	1	5	5.0	10	0	0	

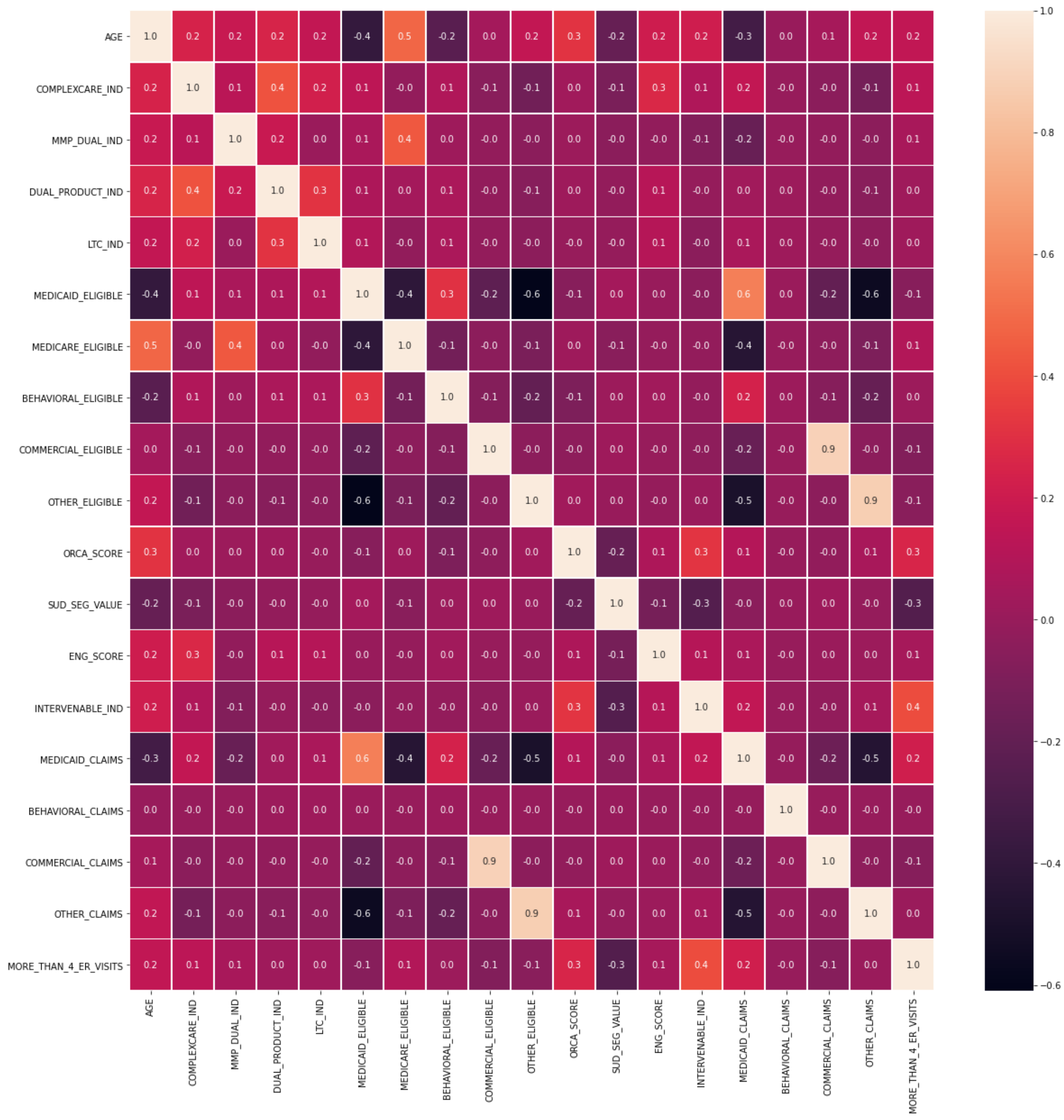
69000 rows × 46 columns

Feature selection using corelation

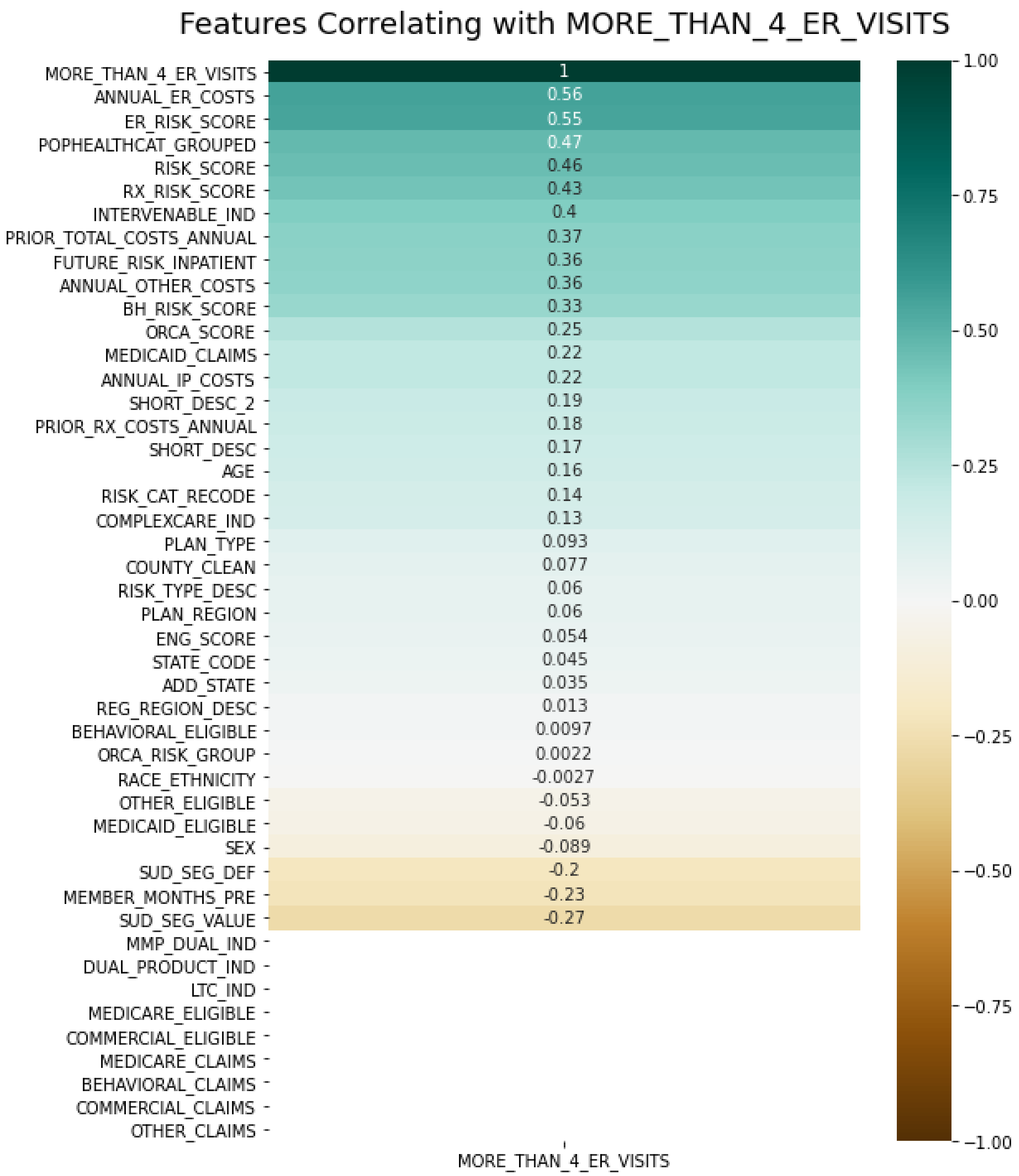
In [32]:

```python
# Corelation map
f,ax = plt.subplots(figsize=(20, 20))
sns.heatmap(data.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

```
Out[32]: <AxesSubplot:>
```



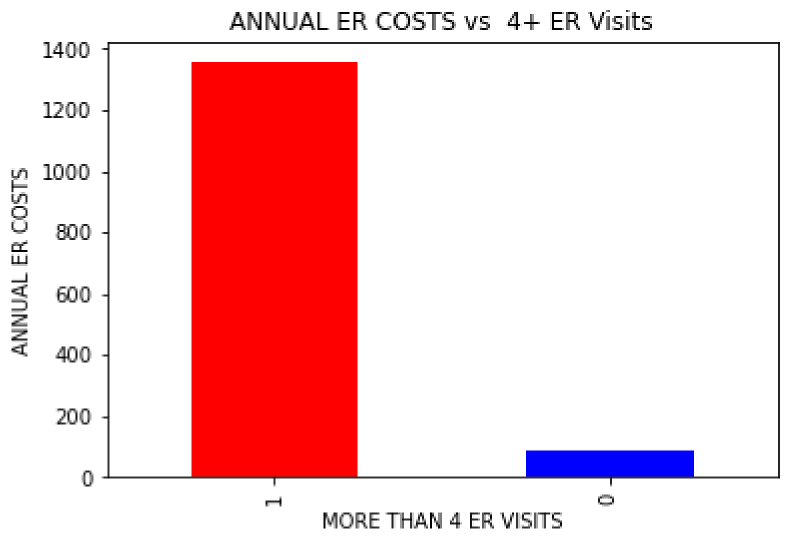
```
In [103…
# corelation with target variable
plt.figure(figsize=(8, 12))
heatmap = sns.heatmap(data.corr()[['MORE_THAN_4_ER_VISITS']].sort_values(by='MORE_THAN_
heatmap.set_title('Features Correlating with MORE_THAN_4_ER_VISITS', fontdict={'fontsiz
```



Bar graph for top 3 corelations

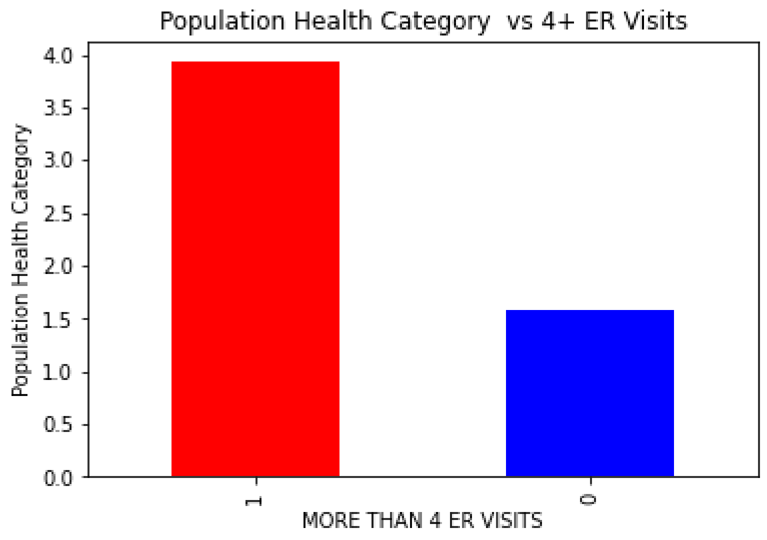
```
In [249…
erdata=data.groupby("MORE_THAN_4_ER_VISITS")['ANNUAL_ER_COSTS'].describe().sort_values(
erdata["mean"].plot(kind='bar',color=['red',  'blue', ])
plt.xlabel('MORE THAN 4 ER VISITS')
plt.ylabel("ANNUAL ER COSTS")
plt.title("ANNUAL ER COSTS vs  4+ ER Visits")
```

```
Out[249… Text(0.5, 1.0, 'ANNUAL ER COSTS vs  4+ ER Visits')
```



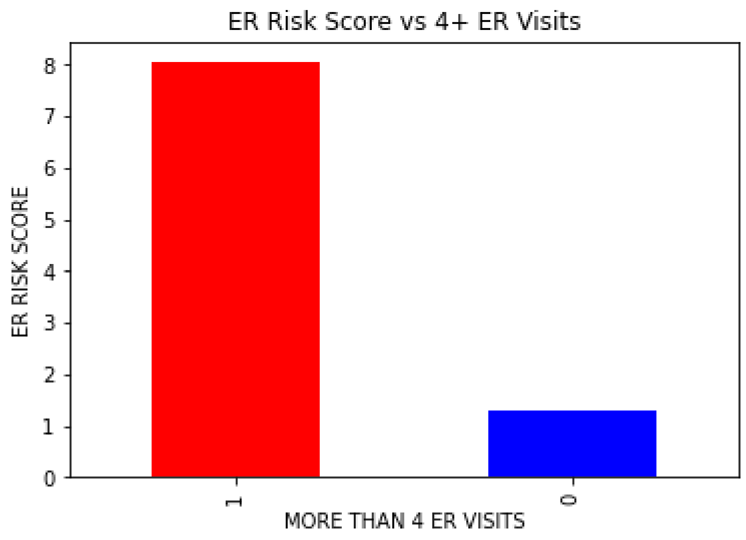
```
In [250…
erdata=data.groupby("MORE_THAN_4_ER_VISITS")['POPHEALTHCAT_GROUPED'].describe().sort_va
erdata["mean"].plot(kind='bar',color=['red',  'blue', ])
plt.xlabel('MORE THAN 4 ER VISITS')
plt.ylabel("Population Health Category")
plt.title("Population Health Category  vs 4+ ER Visits")
```

```
Out[250… Text(0.5, 1.0, 'Population Health Category  vs 4+ ER Visits')
```



```
In [251…
erdata=data.groupby("MORE_THAN_4_ER_VISITS")['ER_RISK_SCORE'].describe().sort_values('m
erdata["mean"].plot(kind='bar',color=['red',  'blue', ])
plt.xlabel('MORE THAN 4 ER VISITS')
plt.ylabel("ER RISK SCORE")
plt.title("ER Risk Score vs 4+ ER Visits")
```

```
Out[251… Text(0.5, 1.0, 'ER Risk Score vs 4+ ER Visits')
```



Preparing data for model

In [252…

```
# Preparing model data
selected_columns=['MORE_THAN_4_ER_VISITS',
                  'ANNUAL_ER_COSTS',
                  'ER_RISK_SCORE',
                  'POPHEALTHCAT_GROUPED',
                  'RISK_SCORE',
                  'RX_RISK_SCORE',
                  'INTERVENABLE_IND',
                  'PRIOR_TOTAL_COSTS_ANNUAL',
                  'FUTURE_RISK_INPATIENT',
                  'ANNUAL_OTHER_COSTS',
                  'BH_RISK_SCORE',
                  'ORCA_SCORE',
                  'MEDICAID_CLAIMS',
                  'ANNUAL_IP_COSTS',
                  'SHORT_DESC_2',
                  'PRIOR_RX_COSTS_ANNUAL',
                  'SHORT_DESC',
                  'AGE',
                  'RISK_CAT_RECODE',
                  'COMPLEXCARE_IND',
                  'PLAN_TYPE',
                  'COUNTY_CLEAN',
                  'RISK_TYPE_DESC',
                  'PLAN_REGION',
                  'ENG_SCORE',
                  'STATE_CODE',
                  'ADD_STATE',
                  'REG_REGION_DESC',
                  'BEHAVIORAL_ELIGIBLE',
                  'ORCA_RISK_GROUP',
                  'RACE_ETHNICITY',
                  'OTHER_ELIGIBLE',
                  'MEDICAID_ELIGIBLE',
                  'SEX',
                  'SUD_SEG_DEF',
                  'MEMBER_MONTHS_PRE',
                  'SUD_SEG_VALUE']
```

```
model_data=data[selected_columns]
model_data
```

Out[252…

	MORE_THAN_4_ER_VISITS	ANNUAL_ER_COSTS	ER_RISK_SCORE	POPHEALTHCAT_GROUPED	RISK_
0	0	0.00	1.9154	5	
1	0	0.00	8.3131	4	
2	0	0.00	0.6467	0	
3	0	0.00	2.0956	2	
4	0	0.00	0.9482	0	
...	...	...	...	...	
68995	1	777.07	7.5670	4	1
68996	1	2763.28	16.0033	1	
68997	1	941.15	2.6039	1	
68998	1	1079.95	4.9284	4	
68999	1	104.98	1.5763	1	

69000 rows × 37 columns

In [168…

```
import statsmodels.api as sm

y=model_data.MORE_THAN_4_ER_VISITS
X=model_data.drop("MORE_THAN_4_ER_VISITS",axis=1)


X.describe()
```

Out[168…

	ANNUAL_ER_COSTS	ER_RISK_SCORE	POPHEALTHCAT_GROUPED	RISK_SCORE	RX_RISK_SCORE	I
count	69000.000000	69000.000000	69000.000000	69000.000000	69000.000000	
mean	673.549401	4.417997	2.672145	2.507399	2.268842	
std	1133.082080	6.084783	2.472848	3.567680	2.916019	
min	0.000000	0.289600	0.000000	0.100000	0.134700	
25%	0.000000	0.667100	0.000000	0.354300	0.471700	
50%	121.940000	1.525200	2.000000	1.064300	0.974900	
75%	940.637500	5.366350	4.000000	3.206975	2.905525	
max	7676.760000	26.544900	10.000000	24.902200	17.221500	

8 rows × 36 columns

Modeling

In [169…

```
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.199834
         Iterations 9
                              Results: Logit
=========================================================================
Model:                Logit               Pseudo R-squared:  0.711
Dependent Variable:   MORE_THAN_4_ER_VISITS AIC:               27649.0715
Date:                 2021-09-25 21:54     BIC:               27978.1785
No. Observations:     69000               Log-Likelihood:    -13789.
Df Model:             35                  LL-Null:           -47646.
Df Residuals:         68964               LLR p-value:       0.0000
Converged:            1.0000              Scale:             1.0000
No. Iterations:       9.0000
-------------------------------------------------------------------------
                          Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
-------------------------------------------------------------------------
ANNUAL_ER_COSTS          0.0031   0.0000  73.2131 0.0000  0.0030  0.0032
ER_RISK_SCORE            0.2968   0.0063  46.9788 0.0000  0.2844  0.3092
POPHEALTHCAT_GROUPED     0.1720   0.0094  18.3388 0.0000  0.1536  0.1904
RISK_SCORE               0.1687   0.0085  19.7526 0.0000  0.1520  0.1854
RX_RISK_SCORE            0.0607   0.0092   6.6293 0.0000  0.0428  0.0787
INTERVENABLE_IND        -0.2281   0.0419  -5.4463 0.0000 -0.3103 -0.1460
PRIOR_TOTAL_COSTS_ANNUAL 0.0000   0.0000   2.1624 0.0306  0.0000  0.0000
FUTURE_RISK_INPATIENT    0.0232   0.0061   3.8301 0.0001  0.0113  0.0351
ANNUAL_OTHER_COSTS       0.0000   0.0000   6.6473 0.0000  0.0000  0.0000
BH_RISK_SCORE            0.0110   0.0033   3.3061 0.0009  0.0045  0.0175
ORCA_SCORE               0.0060   0.0005  12.9559 0.0000  0.0051  0.0070
MEDICAID_CLAIMS          3.4167   0.0758  45.0908 0.0000  3.2682  3.5652
ANNUAL_IP_COSTS          0.0000   0.0000   4.9325 0.0000  0.0000  0.0000
SHORT_DESC_2            -0.0085   0.0011  -7.6187 0.0000 -0.0107 -0.0063
PRIOR_RX_COSTS_ANNUAL   -0.0000   0.0000 -10.9507 0.0000 -0.0000 -0.0000
SHORT_DESC               0.0118   0.0010  11.6581 0.0000  0.0098  0.0138
AGE                     -0.0083   0.0011  -7.7053 0.0000 -0.0104 -0.0062
RISK_CAT_RECODE          0.0177   0.0019   9.2522 0.0000  0.0139  0.0214
COMPLEXCARE_IND          0.1299   0.0522   2.4884 0.0128  0.0276  0.2321
PLAN_TYPE                0.5660   0.0442  12.8033 0.0000  0.4794  0.6526
COUNTY_CLEAN            -0.0000   0.0000  -0.5012 0.6163 -0.0001  0.0001
RISK_TYPE_DESC          -0.7328   0.0504 -14.5268 0.0000 -0.8317 -0.6339
PLAN_REGION              0.0552   0.0123   4.4880 0.0000  0.0311  0.0794
ENG_SCORE               -0.0042   0.0006  -7.4727 0.0000 -0.0053 -0.0031
STATE_CODE              -0.0098   0.0028  -3.4307 0.0006 -0.0154 -0.0042
ADD_STATE               -0.0034   0.0019  -1.7580 0.0787 -0.0072  0.0004
REG_REGION_DESC          0.0001   0.0003   0.2288 0.8190 -0.0006  0.0007
BEHAVIORAL_ELIGIBLE      0.1212   0.0437   2.7712 0.0056  0.0355  0.2069
ORCA_RISK_GROUP         -0.0299   0.0229  -1.3032 0.1925 -0.0749  0.0151
RACE_ETHNICITY          -0.0593   0.0091  -6.5488 0.0000 -0.0771 -0.0416
OTHER_ELIGIBLE           1.2677   0.0944  13.4316 0.0000  1.0827  1.4527
MEDICAID_ELIGIBLE       -1.4501   0.0694 -20.9078 0.0000 -1.5860 -1.3141
SEX                      0.0484   0.0324   1.4951 0.1349 -0.0150  0.1119
SUD_SEG_DEF             -0.0491   0.0369  -1.3314 0.1831 -0.1214  0.0232
MEMBER_MONTHS_PRE       -0.4464   0.0050 -89.6315 0.0000 -0.4562 -0.4367
SUD_SEG_VALUE           -0.0619   0.0334  -1.8533 0.0638 -0.1273  0.0036
=========================================================================
```

Modeling

Removing variables with higher p-values

In [170…

```
model_data=model_data.drop("COUNTY_CLEAN",axis=1)
```

In [171…

```
model_data=model_data.drop("REG_REGION_DESC",axis=1)
```

In [172…

```
y=model_data.MORE_THAN_4_ER_VISITS
X=model_data.drop("MORE_THAN_4_ER_VISITS",axis=1)
X.describe()
```

Out[172…

	ANNUAL_ER_COSTS	ER_RISK_SCORE	POPHEALTHCAT_GROUPED	RISK_SCORE	RX_RISK_SCORE	I
count	69000.000000	69000.000000	69000.000000	69000.000000	69000.000000	
mean	673.549401	4.417997	2.672145	2.507399	2.268842	
std	1133.082080	6.084783	2.472848	3.567680	2.916019	
min	0.000000	0.289600	0.000000	0.100000	0.134700	
25%	0.000000	0.667100	0.000000	0.354300	0.471700	
50%	121.940000	1.525200	2.000000	1.064300	0.974900	
75%	940.637500	5.366350	4.000000	3.206975	2.905525	
max	7676.760000	26.544900	10.000000	24.902200	17.221500	

8 rows × 34 columns

In [173…

```
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.199837
         Iterations 9
                             Results: Logit
=========================================================================
Model:                 Logit                  Pseudo R-squared:  0.711
Dependent Variable:    MORE_THAN_4_ER_VISITS  AIC:               27645.4561
Date:                  2021-09-25 21:54       BIC:               27956.2794
No. Observations:      69000                  Log-Likelihood:    -13789.
Df Model:              33                     LL-Null:           -47646.
Df Residuals:          68966                  LLR p-value:       0.0000
Converged:             1.0000                 Scale:             1.0000
No. Iterations:        9.0000
-------------------------------------------------------------------------
                          Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
-------------------------------------------------------------------------
ANNUAL_ER_COSTS           0.0031   0.0000  73.2627 0.0000  0.0030  0.0032
ER_RISK_SCORE             0.2969   0.0063  47.0237 0.0000  0.2846  0.3093
POPHEALTHCAT_GROUPED      0.1716   0.0094  18.3424 0.0000  0.1533  0.1899
RISK_SCORE                0.1687   0.0085  19.7535 0.0000  0.1520  0.1855
RX_RISK_SCORE             0.0607   0.0092   6.6242 0.0000  0.0427  0.0786
INTERVENABLE_IND         -0.2272   0.0419  -5.4267 0.0000 -0.3092 -0.1451
PRIOR_TOTAL_COSTS_ANNUAL  0.0000   0.0000   2.1592 0.0308  0.0000  0.0000
FUTURE_RISK_INPATIENT     0.0233   0.0061   3.8368 0.0001  0.0114  0.0351
ANNUAL_OTHER_COSTS        0.0000   0.0000   6.6355 0.0000  0.0000  0.0000
```

```
BH_RISK_SCORE             0.0110   0.0033   3.3034 0.0010  0.0045  0.0175
ORCA_SCORE                0.0060   0.0005  12.9573 0.0000  0.0051  0.0070
MEDICAID_CLAIMS           3.4163   0.0755  45.2450 0.0000  3.2683  3.5643
ANNUAL_IP_COSTS           0.0000   0.0000   4.9778 0.0000  0.0000  0.0000
SHORT_DESC_2             -0.0085   0.0011  -7.6168 0.0000 -0.0107 -0.0063
PRIOR_RX_COSTS_ANNUAL    -0.0000   0.0000 -10.9592 0.0000 -0.0000 -0.0000
SHORT_DESC                0.0118   0.0010  11.6571 0.0000  0.0098  0.0138
AGE                      -0.0082   0.0011  -7.6957 0.0000 -0.0103 -0.0061
RISK_CAT_RECODE           0.0177   0.0019   9.2560 0.0000  0.0139  0.0214
COMPLEXCARE_IND           0.1317   0.0520   2.5306 0.0114  0.0297  0.2337
PLAN_TYPE                 0.5644   0.0441  12.7943 0.0000  0.4779  0.6508
RISK_TYPE_DESC           -0.7354   0.0493 -14.9034 0.0000 -0.8321 -0.6387
PLAN_REGION               0.0547   0.0122   4.4631 0.0000  0.0307  0.0787
ENG_SCORE                -0.0042   0.0006  -7.4787 0.0000 -0.0053 -0.0031
STATE_CODE               -0.0096   0.0028  -3.4119 0.0006 -0.0151 -0.0041
ADD_STATE                -0.0035   0.0019  -1.8209 0.0686 -0.0073  0.0003
BEHAVIORAL_ELIGIBLE       0.1302   0.0406   3.2089 0.0013  0.0507  0.2097
ORCA_RISK_GROUP          -0.0297   0.0229  -1.2958 0.1951 -0.0747  0.0152
RACE_ETHNICITY           -0.0596   0.0090  -6.5879 0.0000 -0.0773 -0.0418
OTHER_ELIGIBLE            1.2701   0.0940  13.5124 0.0000  1.0859  1.4544
MEDICAID_ELIGIBLE        -1.4446   0.0686 -21.0638 0.0000 -1.5790 -1.3102
SEX                       0.0487   0.0324   1.5034 0.1327 -0.0148  0.1121
SUD_SEG_DEF              -0.0488   0.0369  -1.3230 0.1858 -0.1211  0.0235
MEMBER_MONTHS_PRE        -0.4464   0.0050 -89.9396 0.0000 -0.4561 -0.4367
SUD_SEG_VALUE            -0.0621   0.0334  -1.8610 0.0627 -0.1276  0.0033
==========================================================================
```

Accuracy

In [175…

```python
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.sc
```

```
Accuracy of logistic regression classifier on test set: 0.87
/Users/madhukarayachit/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_l
ogistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Confusion Matrix

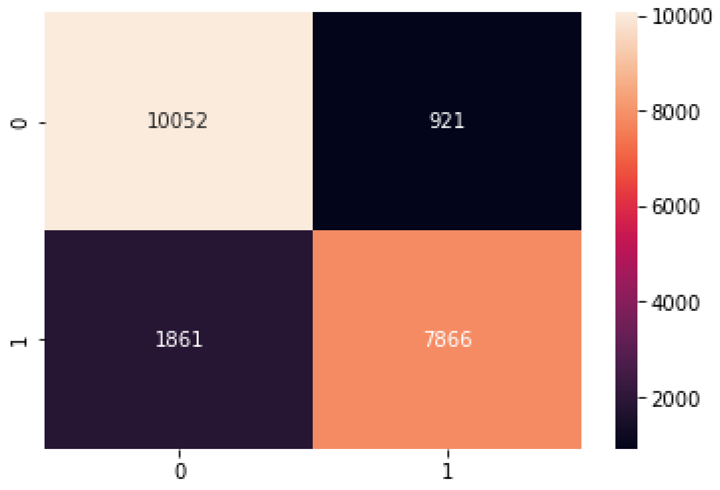
In [176…

```python
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
sns.heatmap(confusion_matrix,annot=True,fmt="d")
```

```
[[10052   921]
 [ 1861  7866]]
```

Out[176…

```
<AxesSubplot:>
```



Clasification Report

In [177…

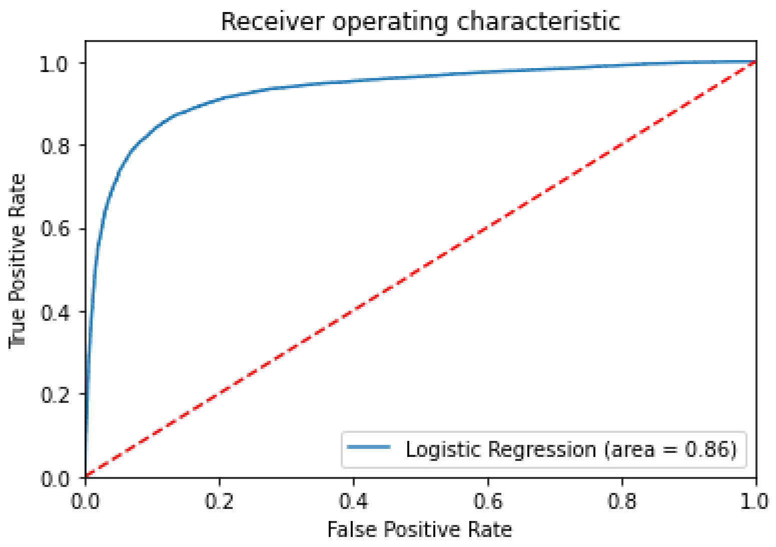
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.92      0.88     10973
           1       0.90      0.81      0.85      9727

    accuracy                           0.87     20700
   macro avg       0.87      0.86      0.86     20700
weighted avg       0.87      0.87      0.86     20700
```

ROC Curve

In [178…

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```


Receiver operating characteristic
True Positive Rate
False Positive Rate
Logistic Regression (area = 0.86)
0.0
0.2
0.4
0.6
0.8
1.0