CMS_Fraud_Predict

December 17, 2025

1 CMS Exclusions Dataset Modeling

1.1 ## DATA SOURCES

I used data public US government data published by Center for Medicare and Medicaid Services (CMS) and Office of Inspector General (OIG) of Human and Health Services

- Data source 1:
  - publishing US agency - CMS
  - The Medicare Physician & Other Practitioners by Provider dataset provides information on use, payments, submitted charges and beneficiary demographic and health characteristics organized by National Provider Identifier (NPI).
  - URL: https://data.cms.gov/provider-summary-by-type-of-service/medicare-physician-other-practitioners/medicare-physician-other-practitioners-by-provider
- Data Source 2:
  - publishing US agency - OIG
  - OIG has the authority to exclude individuals and entities from Federally funded health care programs
  - URL: https://oig.hhs.gov/exclusions/exclusions_list.asp

[258]:
```
import numpy as np
import pandas as pd
import os
```

[259]:
```
#exclusions_file = 'data/exclusions_052020.csv'
exclusions_file = 'cms_data/leie.csv'
exclusions = pd.read_csv(exclusions_file)
exclusions = exclusions[exclusions.NPI > 0].set_index('NPI')
```

```
/tmp/ipykernel_11146/1939962845.py:3: DtypeWarning: Columns (3) have mixed
types. Specify dtype option on import or set low_memory=False.
  exclusions = pd.read_csv(exclusions_file)
```

[260]:
```
len(exclusions)
```

[260]: 8297

1.2 ## DATA PREPARATION

The first goal for this step is to prepare data by adding labels. We do this by joining the Provider data with the Exclusions data by the National Provider Identifier (NPI). Our dataset is very unbalanced. Only a small fraction (less than tenth of a percent) of our data has a label. Classification algorithms perform better when the classes in the dataset are well balanced. We artificially increase the fraction of the labeled data by dramatically reducing the fraction of unlabeled records.

For each input data file we do the following: - load raw data to a dataframe - join with the Exclusions data by the Provider NPI - rebalance the dataset by reducing the number of non-excluded providers by a factor of 100 - save the preprocessed data

[261]:
```
files = os.listdir('cms_data')
files
```

[261]:
```
['MUP_PHY_R25_P07_V10_D19_Prov.csv',
 'MUP_PHY_R25_P07_V10_D22_Prov.csv',
 '.ipynb_checkpoints',
 'MUP_PHY_R25_P07_V10_D21_Prov.csv',
 'leie.csv',
 'y2022_prep.csv',
 'y2019_prep.csv',
 'MUP_PHY_R25_P05_V20_D23_Prov.csv',
 'MUP_PHY_R25_P07_V10_D20_Prov.csv',
 'y2023_prep.csv']
```

[262]:
```
fil = files[0]
data = pd.read_csv('cms_data/'+fil)
data.head()
```

```
/tmp/ipykernel_11146/2032392280.py:2: DtypeWarning: Columns (10,11) have mixed
types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv('cms_data/'+fil)
```

[262]:
```
   Rndrng_NPI Rndrng_Prvdr_Last_Org_Name Rndrng_Prvdr_First_Name  \
0  1003000126                  ENKESHAFI                 ARDALAN
1  1003000134                     CIBULL                  THOMAS
2  1003000142                     KHALIL                  RASHID
3  1003000423                    VELOTTA                JENNIFER
4  1003000480                  ROTHCHILD                   KEVIN

  Rndrng_Prvdr_MI Rndrng_Prvdr_Crdntls Rndrng_Prvdr_Ent_Cd  \
0             NaN                 M.D.                   I
1               L                 M.D.                   I
2             NaN                 M.D.                   I
3               A                 M.D.                   I
4               B                   MD                   I

           Rndrng_Prvdr_St1   Rndrng_Prvdr_St2 Rndrng_Prvdr_City  \
```

```
0             900 SETON DR                NaN        CUMBERLAND
1            2650 RIDGE AVE  EVANSTON HOSPITAL          EVANSTON
2  4126 N HOLLAND SYLVANIA RD          SUITE 220            TOLEDO
3          11100 EUCLID AVE                NaN         CLEVELAND
4          12605 E 16TH AVE                NaN            AURORA

  Rndrng_Prvdr_State_Abrvtn  … Bene_CC_PH_Diabetes_V2_Pct  \
0                        MD  …                       54.0
1                        IL  …                       21.0
2                        OH  …                       34.0
3                        OH  …                       19.0
4                        CO  …                       38.0

  Bene_CC_PH_HF_NonIHD_V2_Pct  Bene_CC_PH_Hyperlipidemia_V2_Pct  \
0                        46.0                              75.0
1                        11.0                              72.0
2                        21.0                              65.0
3                         NaN                              58.0
4                        18.0                              49.0

  Bene_CC_PH_Hypertension_V2_Pct Bene_CC_PH_IschemicHeart_V2_Pct  \
0                           75.0                            50.0
1                           64.0                            20.0
2                           75.0                            27.0
3                           53.0                             NaN
4                           72.0                            20.0

  Bene_CC_PH_Osteoporosis_V2_Pct Bene_CC_PH_Parkinson_V2_Pct  \
0                           13.0                         3.0
1                           15.0                         3.0
2                           10.0                         NaN
3                           15.0                         0.0
4                           17.0                         NaN

   Bene_CC_PH_Arthritis_V2_Pct  Bene_CC_PH_Stroke_TIA_V2_Pct  \
0                         59.0                          27.0
1                         42.0                           6.0
2                         75.0                          10.0
3                         43.0                           0.0
4                         55.0                           NaN

   Bene_Avg_Risk_Scre
0              2.5917
1              1.1246
2              1.6146
3              0.9065
4              1.7191
```

```
[5 rows x 81 columns]
```

[263]:
```python
len(data.columns)
```

[263]:
```
81
```

[264]:
```python
data.columns
```

[264]:
```
Index(['Rndrng_NPI', 'Rndrng_Prvdr_Last_Org_Name', 'Rndrng_Prvdr_First_Name',
       'Rndrng_Prvdr_MI', 'Rndrng_Prvdr_Crdntls', 'Rndrng_Prvdr_Ent_Cd',
       'Rndrng_Prvdr_St1', 'Rndrng_Prvdr_St2', 'Rndrng_Prvdr_City',
       'Rndrng_Prvdr_State_Abrvtn', 'Rndrng_Prvdr_State_FIPS',
       'Rndrng_Prvdr_Zip5', 'Rndrng_Prvdr_RUCA', 'Rndrng_Prvdr_RUCA_Desc',
       'Rndrng_Prvdr_Cntry', 'Rndrng_Prvdr_Type',
       'Rndrng_Prvdr_Mdcr_Prtcptg_Ind', 'Tot_HCPCS_Cds', 'Tot_Benes',
       'Tot_Srvcs', 'Tot_Sbmtd_Chrg', 'Tot_Mdcr_Alowd_Amt',
       'Tot_Mdcr_Pymt_Amt', 'Tot_Mdcr_Stdzd_Amt', 'Drug_Sprsn_Ind',
       'Drug_Tot_HCPCS_Cds', 'Drug_Tot_Benes', 'Drug_Tot_Srvcs',
       'Drug_Sbmtd_Chrg', 'Drug_Mdcr_Alowd_Amt', 'Drug_Mdcr_Pymt_Amt',
       'Drug_Mdcr_Stdzd_Amt', 'Med_Sprsn_Ind', 'Med_Tot_HCPCS_Cds',
       'Med_Tot_Benes', 'Med_Tot_Srvcs', 'Med_Sbmtd_Chrg',
       'Med_Mdcr_Alowd_Amt', 'Med_Mdcr_Pymt_Amt', 'Med_Mdcr_Stdzd_Amt',
       'Bene_Avg_Age', 'Bene_Age_LT_65_Cnt', 'Bene_Age_65_74_Cnt',
       'Bene_Age_75_84_Cnt', 'Bene_Age_GT_84_Cnt', 'Bene_Feml_Cnt',
       'Bene_Male_Cnt', 'Bene_Race_Wht_Cnt', 'Bene_Race_Black_Cnt',
       'Bene_Race_API_Cnt', 'Bene_Race_Hspnc_Cnt', 'Bene_Race_NatInd_Cnt',
       'Bene_Race_Othr_Cnt', 'Bene_Dual_Cnt', 'Bene_Ndual_Cnt',
       'Bene_CC_BH_ADHD_OthCD_V1_Pct', 'Bene_CC_BH_Alcohol_Drug_V1_Pct',
       'Bene_CC_BH_Tobacco_V1_Pct', 'Bene_CC_BH_Alz_NonAlzdem_V2_Pct',
       'Bene_CC_BH_Anxiety_V1_Pct', 'Bene_CC_BH_Bipolar_V1_Pct',
       'Bene_CC_BH_Mood_V2_Pct', 'Bene_CC_BH_Depress_V1_Pct',
       'Bene_CC_BH_PD_V1_Pct', 'Bene_CC_BH_PTSD_V1_Pct',
       'Bene_CC_BH_Schizo_OthPsy_V1_Pct', 'Bene_CC_PH_Asthma_V2_Pct',
       'Bene_CC_PH_Afib_V2_Pct', 'Bene_CC_PH_Cancer6_V2_Pct',
       'Bene_CC_PH_CKD_V2_Pct', 'Bene_CC_PH_COPD_V2_Pct',
       'Bene_CC_PH_Diabetes_V2_Pct', 'Bene_CC_PH_HF_NonIHD_V2_Pct',
       'Bene_CC_PH_Hyperlipidemia_V2_Pct', 'Bene_CC_PH_Hypertension_V2_Pct',
       'Bene_CC_PH_IschemicHeart_V2_Pct', 'Bene_CC_PH_Osteoporosis_V2_Pct',
       'Bene_CC_PH_Parkinson_V2_Pct', 'Bene_CC_PH_Arthritis_V2_Pct',
       'Bene_CC_PH_Stroke_TIA_V2_Pct', 'Bene_Avg_Risk_Scre'],
      dtype='object')
```

Join with Exclusions data:

[265]:
```python
data = data.join(
    exclusions,
    on='Rndrng_NPI',
```

```
    how='left'
)
```

2 Feature Engineering

[266]:
```
ratio_pairs = [
    ('Drug_Mdcr_Alowd_Amt','Drug_Mdcr_Pymt_Amt'),
    ('Tot_Mdcr_Alowd_Amt','Med_Sbmtd_Chrg'),
    ('Drug_Tot_Benes','Tot_Benes'),
   #  ('total_drug_medicare_payment_amt','total_med_medicare_payment_amt')
]
for p in ratio_pairs:
    data[f'rat_{p[0]}_{p[1]}'] = data[p[0]]/data[p[1]]
```

2.1 Objective

Excluded providers have non-null exclusion part of the dataframe. The model needs a numeric column as a target (or objective column) to perform supervised learning. Bolow we construct the objective column "excluded" and set it to one for the excluded providers and zero othervise. Here we assume that a provider is fraudulent if he/she is found in LEIE database regardless of the type of exclusion.

[267]:
```
data['EXCLTYPE'] = data['EXCLTYPE'].fillna('UNK')
```

[268]:
```
excluded  = np.ones(len(data),dtype=int)
excluded[np.isnan(data.EXCLDATE)] = 0
print(f'Total records in the data set is {len(data)}.')
print(f"Number of excluded providers in the set: {excluded.sum()}.")
data['excluded'] = excluded
```

```
Total records in the data set is 1155883.
Number of excluded providers in the set: 841.
```

2.2 Rebalancing

[269]:
```
data['rand'] = np.random.rand(len(data))
```

[270]:
```
df = data[(data.excluded == 1) | (data.rand > 0.99)]
```

[271]:
```
df['excluded'].sum()
```

[271]:
```
np.int64(841)
```

Save to CSV file:

[272]:
```
df.to_csv('cms_data/y2023_prep.csv')
```

2.3 Load the preprocessed data

[273]:
```
dataset = pd.concat([
    pd.read_csv('cms_data/y2019_prep.csv'),
    pd.read_csv('cms_data/y2022_prep.csv'),
    pd.read_csv('cms_data/y2023_prep.csv')
    ]).drop('Unnamed: 0',axis=1)
```

```
/tmp/ipykernel_11146/1459560532.py:3: DtypeWarning: Columns (12) have mixed
types. Specify dtype option on import or set low_memory=False.
  pd.read_csv('cms_data/y2022_prep.csv'),
```

[274]:
```
dataset = dataset.drop('rand',axis=1)
```

2.4 Feature set

Features are the variables or the columns that will be used by the model. The ML models can use numerical columns directly, however they can't deal with categorical values. We need to preprocess the categorical values to numerical features using one-hot encoding:

[275]:
```
def encode_feature(name):
    enc = LabelEncoder().fit([str(it) for it in dataset[name]])
    return enc.transform(dataset[name])
```

[276]:
```
from sklearn.ensemble import␣
  ↪RandomForestClassifier,GradientBoostingClassifier,ExtraTreesClassifier
from sklearn.model_selection import train_test_split as tts
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
```

[277]:
```
dataset['state'] = encode_feature('Rndrng_Prvdr_State_FIPS')
```

[278]:
```
dataset['type'] = encode_feature('Rndrng_Prvdr_Type')
```

[279]:
```
num_cols = dataset.describe().columns
num_cols = num_cols[np.logical_not(np.isin(num_cols,exclusions.columns))]
```

[280]:
```
dataset.head()
```

[280]:
```
   Rndrng_NPI Rndrng_Prvdr_Last_Org_Name Rndrng_Prvdr_First_Name  \
0  1003000134                     CIBULL                  THOMAS
1  1003005315                      SMITH                    ADAM
2  1003009861                      BANNA                MOUSTAFA
3  1003010570                       CHOW                    LING
4  1003017443                     BLOKAR                 MIRJANA

  Rndrng_Prvdr_MI Rndrng_Prvdr_Crdntls Rndrng_Prvdr_Ent_Cd  \
0               L                 M.D.                   I
1               B                   MD                   I
2             NaN                   MD                   I
```

```
3             S              M.D.                I
4           NaN              M.D.                I

     Rndrng_Prvdr_St1   Rndrng_Prvdr_St2 Rndrng_Prvdr_City  \
0       2650 RIDGE AVE  EVANSTON HOSPITAL          EVANSTON
1      4977 SKYVIEW CT                NaN     TRAVERSE CITY
2  5859 W. TALAVI BLVD          SUITE 100          GLENDALE
3   900 E BROADWAY AVE                NaN          BISMARCK
4       65 BLEECKER ST        12TH FLOOR          NEW YORK

  Rndrng_Prvdr_State_Abrvtn  …    EXCLDATE REINDATE  WAIVERDATE WVRSTATE  \
0                        IL  …         NaN      NaN         NaN      NaN
1                        MI  …  20221220.0      0.0         0.0      NaN
2                        AZ  …         NaN      NaN         NaN      NaN
3                        ND  …         NaN      NaN         NaN      NaN
4                        NY  …         NaN      NaN         NaN      NaN

  rat_Drug_Mdcr_Alowd_Amt_Drug_Mdcr_Pymt_Amt  \
0                                        NaN
1                                        NaN
2                                   1.253709
3                                        NaN
4                                        NaN

  rat_Tot_Mdcr_Alowd_Amt_Med_Sbmtd_Chrg rat_Drug_Tot_Benes_Tot_Benes  \
0                              0.246364                     0.000000
1                                   NaN                          NaN
2                              0.498749                     0.111872
3                              0.485200                     0.000000
4                              0.527974                     0.000000

   excluded  state  type
0         0     18    72
1         1     35    78
2         0     59    11
3         0     55    43
4         0     51    82

[5 rows x 104 columns]
```

[281]:

```python
label = 'excluded'
Y = dataset[label]
X = np.concatenate([
    dataset.loc[:,num_cols[1:]].drop(label,axis=1).fillna(0.0),
   # enc.fit_transform(data[cat_cols].fillna('UNK')).toarray()
],axis=1)
```

```
X_train,X_test,Y_train,Y_test,excltype_train,excltype_test = tts(X,Y,dataset.
 ↪EXCLTYPE,test_size=0.2)
```

[282]:
```
#feature_names = list(num_cols[2:-2]) #+list(enc.get_feature_names())
feature_names = dataset.loc[:,num_cols[1:]].drop(label,axis=1).columns
#feature_names
```

[283]:
```
len(excltype_test)
```

[283]: 7466

3 Classification

Here we care just if a provider was excluded or not regardless of a type of exclusion.

[284]:
```
from sklearn.metrics import␣
 ↪roc_auc_score,roc_curve,auc,classification_report,confusion_matrix
import matplotlib.pyplot as plt
```

[285]:
```
clf =␣
 ↪RandomForestClassifier(n_estimators=200,max_depth=8,class_weight='balanced').
 ↪fit(X_train, Y_train)
#clf_et =␣
 ↪ExtraTreesClassifier(n_estimators=200,max_depth=5,class_weight='balanced').
 ↪fit(X_train, Y_train)
#clf = GradientBoostingClassifier(verbose=1).fit(X_train, Y_train)
```

[286]:
```
Y_pred = clf.predict(X_test)
print(classification_report(Y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.92      0.95      7073
           1       0.28      0.54      0.37       393

    accuracy                           0.90      7466
   macro avg       0.62      0.73      0.66      7466
weighted avg       0.94      0.90      0.92      7466
```

[287]:
```
Y_test.sum(),Y_pred.sum()
```

[287]: (np.int64(393), np.int64(762))

4 Feature importance

[288]:
```
forest = clf
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]
```

[289]:
```
X.shape
```

[289]:
```
(37328, 68)
```

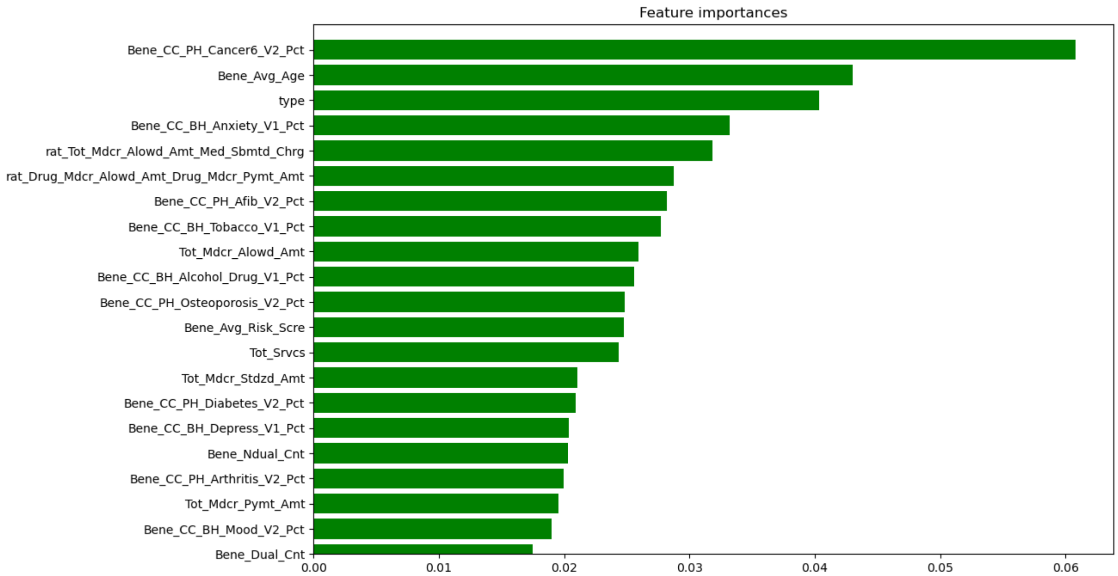
[290]:
```
# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))
```

```
Feature ranking:
1. feature 50 (0.060799)
2. feature 22 (0.043035)
3. feature 67 (0.040325)
4. feature 41 (0.033174)
5. feature 64 (0.031814)
6. feature 63 (0.028740)
7. feature 49 (0.028204)
8. feature 39 (0.027722)
9. feature 5 (0.025966)
10. feature 38 (0.025570)
11. feature 58 (0.024828)
12. feature 62 (0.024758)
13. feature 3 (0.024365)
14. feature 7 (0.021067)
15. feature 53 (0.020952)
16. feature 44 (0.020360)
17. feature 36 (0.020274)
18. feature 60 (0.019973)
19. feature 6 (0.019563)
20. feature 43 (0.019030)
21. feature 35 (0.017490)
22. feature 25 (0.016635)
23. feature 52 (0.016190)
24. feature 51 (0.015996)
25. feature 17 (0.015436)
26. feature 24 (0.014860)
27. feature 40 (0.014513)
28. feature 21 (0.014488)
29. feature 19 (0.013879)
```

```
30. feature 15 (0.013583)
31. feature 20 (0.013457)
32. feature 42 (0.013373)
33. feature 23 (0.012358)
34. feature 4 (0.012290)
35. feature 29 (0.012035)
36. feature 47 (0.011688)
37. feature 16 (0.011321)
38. feature 1 (0.010878)
39. feature 65 (0.010877)
40. feature 2 (0.010726)
41. feature 27 (0.010529)
42. feature 18 (0.010525)
43. feature 28 (0.010303)
44. feature 66 (0.010194)
45. feature 57 (0.009959)
46. feature 54 (0.009927)
47. feature 10 (0.008148)
48. feature 14 (0.007794)
49. feature 13 (0.007016)
50. feature 26 (0.006892)
51. feature 55 (0.006887)
52. feature 61 (0.006620)
53. feature 8 (0.006378)
54. feature 9 (0.006242)
55. feature 48 (0.006011)
56. feature 12 (0.005744)
57. feature 56 (0.005555)
58. feature 11 (0.004692)
59. feature 45 (0.004160)
60. feature 30 (0.003792)
61. feature 32 (0.003471)
62. feature 0 (0.003439)
63. feature 59 (0.003008)
64. feature 37 (0.002706)
65. feature 34 (0.002670)
66. feature 46 (0.002156)
67. feature 31 (0.002022)
68. feature 33 (0.000567)
```

[291]:
```
plt.figure(figsize=(12,8))
plt.title("Feature importances")
names = [feature_names[i] for i in indices]
plt.barh(range(X.shape[1]), importances[indices],
       color="g", align="center")
plt.yticks(range(X.shape[1]), names)
plt.ylim([20,-1])
```

```
plt.show()
#xerr=std[indices]
```



[292]:
```
[feature_names[i] for i in indices[:10]]
```

[292]:
```
['Bene_CC_PH_Cancer6_V2_Pct',
 'Bene_Avg_Age',
 'type',
 'Bene_CC_BH_Anxiety_V1_Pct',
 'rat_Tot_Mdcr_Alowd_Amt_Med_Sbmtd_Chrg',
 'rat_Drug_Mdcr_Alowd_Amt_Drug_Mdcr_Pymt_Amt',
 'Bene_CC_PH_Afib_V2_Pct',
 'Bene_CC_BH_Tobacco_V1_Pct',
 'Tot_Mdcr_Alowd_Amt',
 'Bene_CC_BH_Alcohol_Drug_V1_Pct']
```

[293]:
```
X_train_10 = X_train[:,indices[:10]]
X_test_10 = X_test[:,indices[:10]]
```

[294]:
```
clf_10 =␣
↪RandomForestClassifier(n_estimators=200,max_depth=8,class_weight='balanced').
↪fit(X_train_10, Y_train)
```

[295]:
```
Y_pred_10 = clf_10.predict(X_test_10)
print(classification_report(Y_test,Y_pred_10))
```

```
              precision    recall  f1-score   support
```

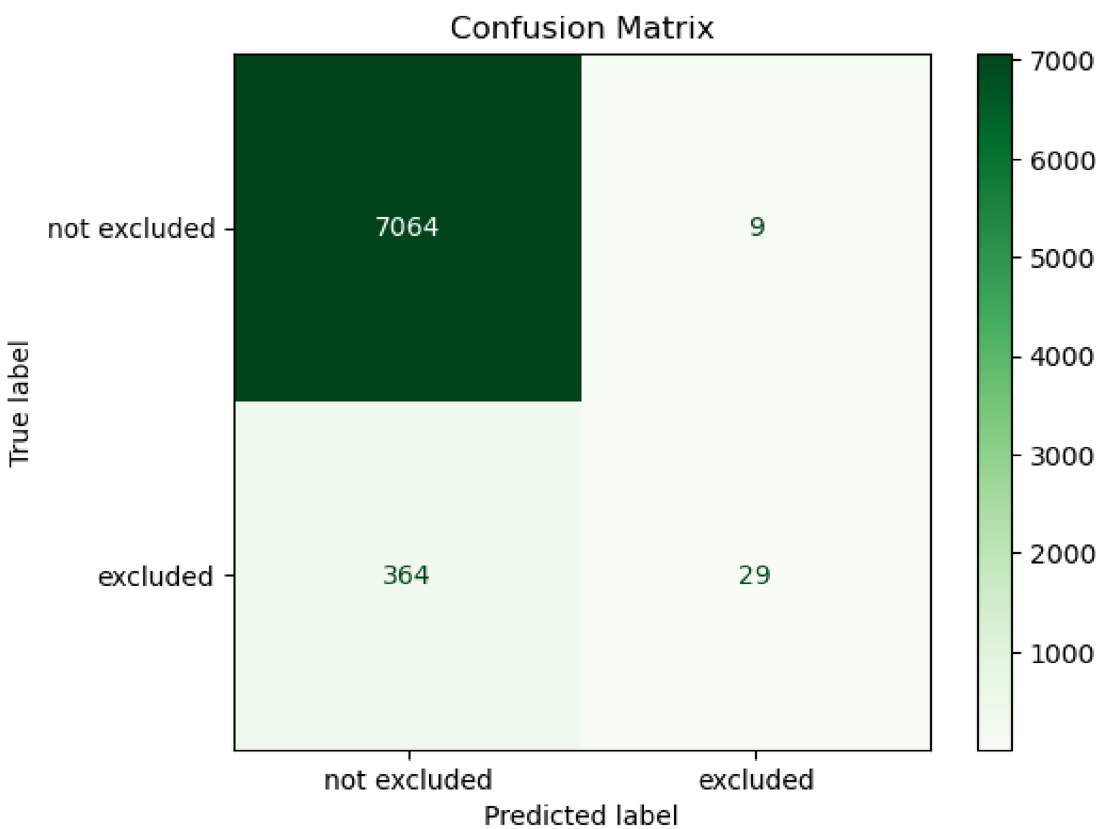
```
              0       0.98      0.86      0.91      7073
              1       0.20      0.63      0.30       393

       accuracy                           0.85      7466
      macro avg       0.59      0.75      0.61      7466
   weighted avg       0.94      0.85      0.88      7466
```

4.1 ## Confusion Matrix

[296]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

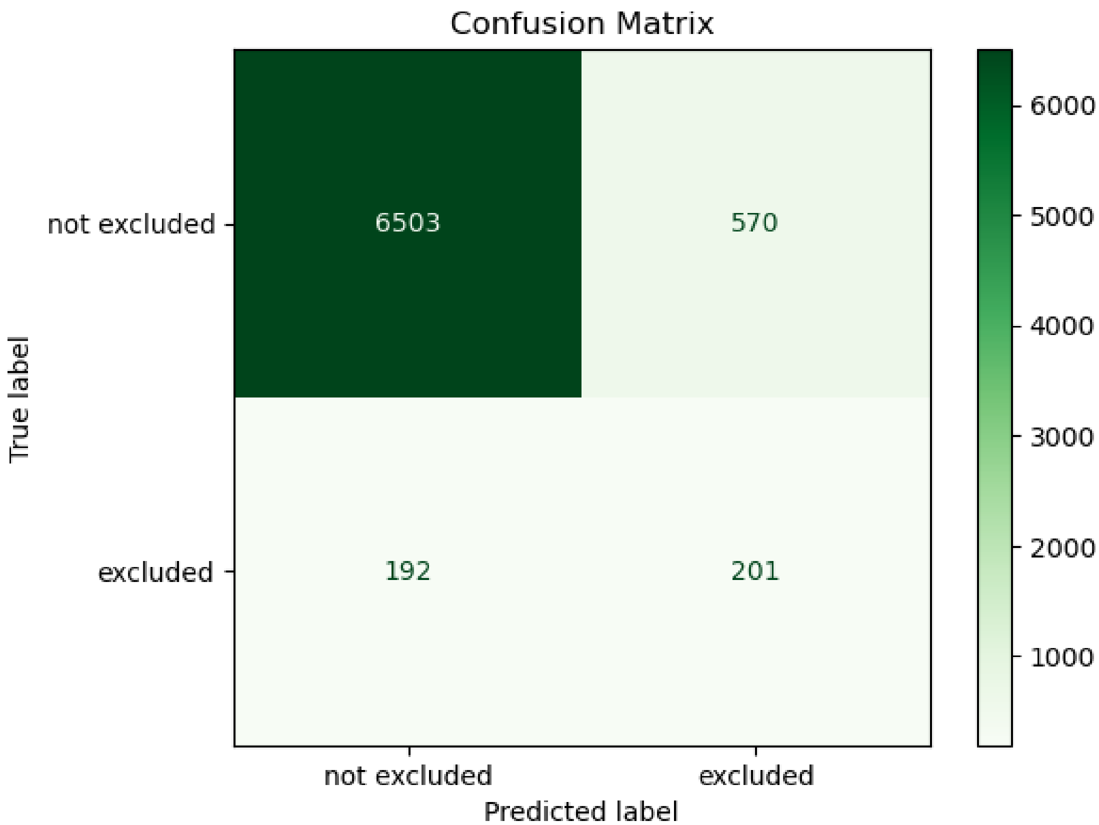
[298]:
```
Y_pred_proba = clf.predict_proba(X_test)
Y_pred_proba = Y_pred_proba[:,1]
```

[299]:
```
 cm = confusion_matrix(Y_test, Y_pred_proba > 0.8)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['not␣
 ↪excluded','excluded'])
disp.plot(cmap=plt.cm.Greens) # Customize colormap if desired
plt.title('Confusion Matrix')
plt.style.use("seaborn-v0_8-dark-palette")
plt.show()
```



[300]:
```
Y_pred_proba_10 = clf_10.predict_proba(X_test_10)
Y_pred_proba_10 = Y_pred_proba_10[:,1]
```

[301]:
```
 cm = confusion_matrix(Y_test, Y_pred_proba_10 > 0.55)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['not␣
↪excluded','excluded'])
disp.plot(cmap=plt.cm.Greens) # Customize colormap if desired
plt.title('Confusion Matrix')
plt.style.use("seaborn-v0_8-dark-palette")
plt.show()
```

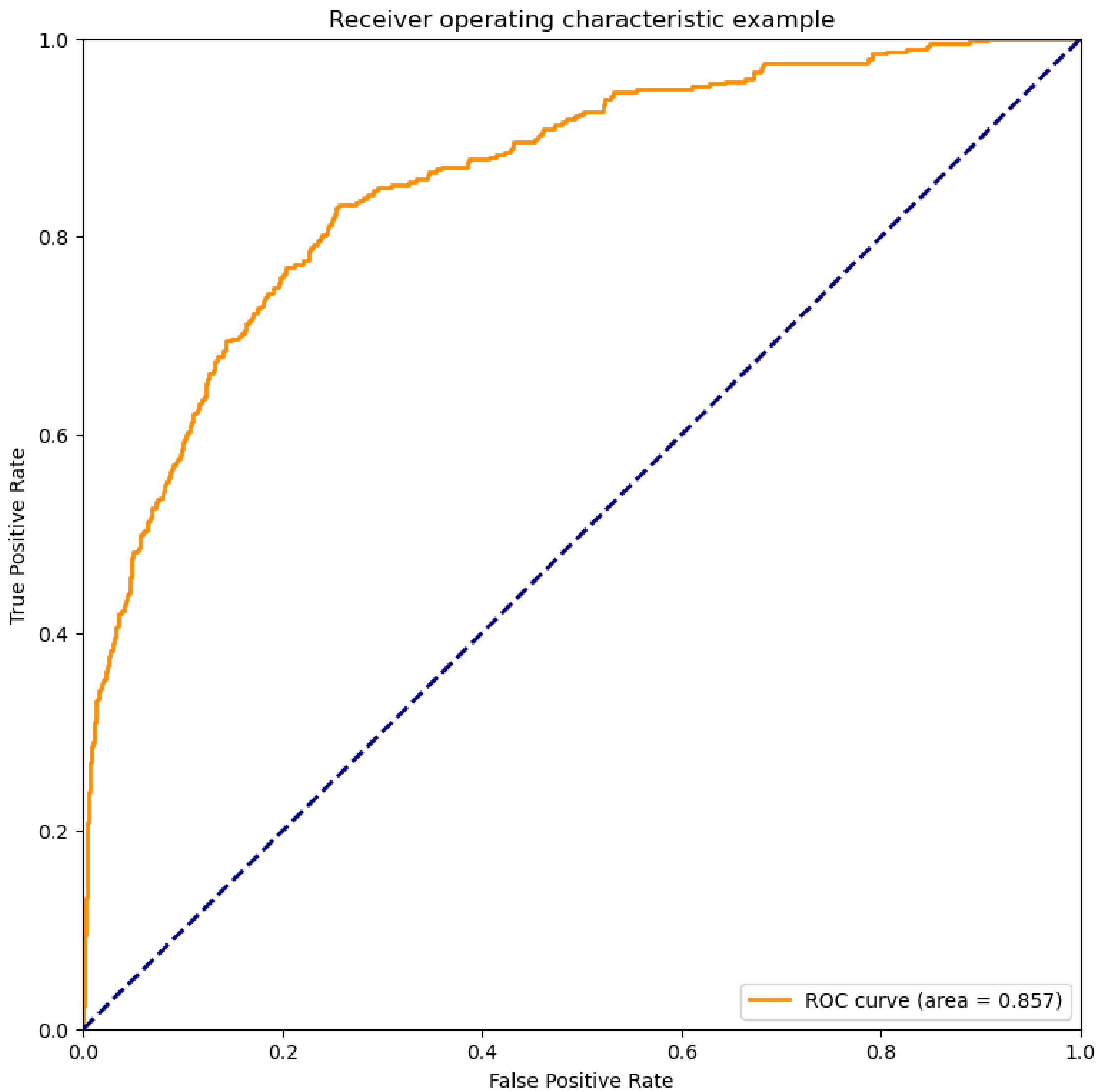


4.2 ROC Curve

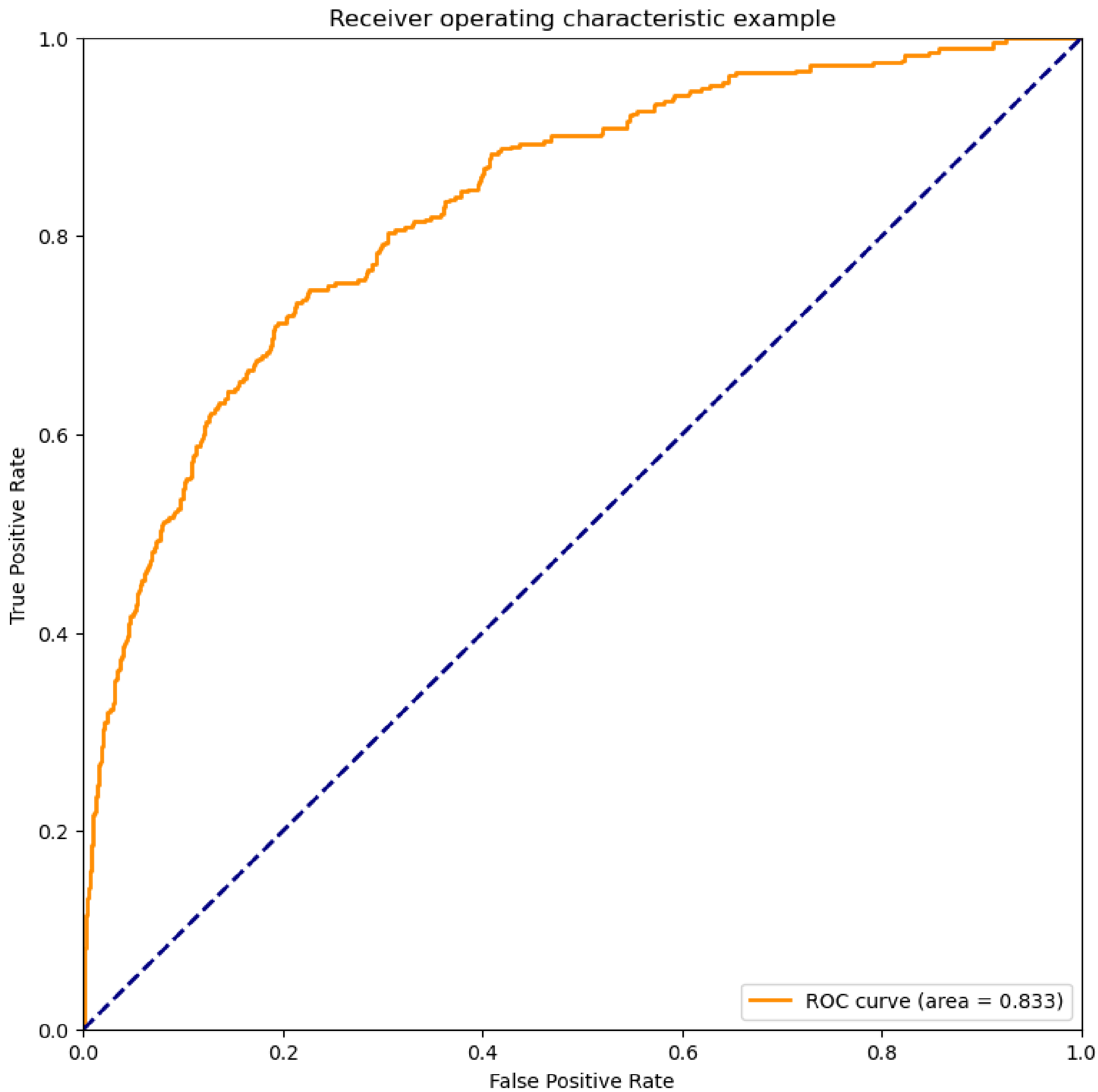
ROC curve based on total exclusion counts regardless of the exclusion type:

[302]:
```
#partb_data.dtypes
Y_pred = clf.predict_proba(X_test)[:,1]
```

```
fpr,tpr, _ = roc_curve(Y_test,Y_pred)
roc_auc = auc(fpr,tpr)
plt.figure(figsize=(9,9))
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



[303]:
```
Y_pred = clf_10.predict_proba(X_test_10)[:,1]

fpr,tpr, _ = roc_curve(Y_test,Y_pred)
roc_auc = auc(fpr,tpr)
plt.figure(figsize=(9,9))
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



5 Select 10 most important features

from xgboost import XGBClassifier as XGBC

[304]:
```
from xgboost import XGBClassifier as XGBC
```

[305]:
```
eval_set = [(X_train,Y_train),(X_test,Y_test)]
```

[306]:
```
clf = XGBC(learning_rate=0.05,eval_metric='logloss',max_depth=8).fit(X_train,
 ↪Y_train,eval_set=eval_set,verbose=True)
```

```
[0]	validation_0-logloss:0.19837	validation_1-logloss:0.20097
[1]	validation_0-logloss:0.19231	validation_1-logloss:0.19653
```

```
[2]	validation_0-logloss:0.18739	validation_1-logloss:0.19339
[3]	validation_0-logloss:0.18259	validation_1-logloss:0.19039
[4]	validation_0-logloss:0.17822	validation_1-logloss:0.18762
[5]	validation_0-logloss:0.17426	validation_1-logloss:0.18497
[6]	validation_0-logloss:0.17067	validation_1-logloss:0.18283
[7]	validation_0-logloss:0.16720	validation_1-logloss:0.18055
[8]	validation_0-logloss:0.16388	validation_1-logloss:0.17838
[9]	validation_0-logloss:0.16037	validation_1-logloss:0.17651
[10]	validation_0-logloss:0.15726	validation_1-logloss:0.17489
[11]	validation_0-logloss:0.15430	validation_1-logloss:0.17311
[12]	validation_0-logloss:0.15144	validation_1-logloss:0.17122
[13]	validation_0-logloss:0.14863	validation_1-logloss:0.16953
[14]	validation_0-logloss:0.14599	validation_1-logloss:0.16787
[15]	validation_0-logloss:0.14376	validation_1-logloss:0.16646
[16]	validation_0-logloss:0.14096	validation_1-logloss:0.16503
[17]	validation_0-logloss:0.13858	validation_1-logloss:0.16383
[18]	validation_0-logloss:0.13636	validation_1-logloss:0.16253
[19]	validation_0-logloss:0.13422	validation_1-logloss:0.16135
[20]	validation_0-logloss:0.13215	validation_1-logloss:0.15995
[21]	validation_0-logloss:0.13012	validation_1-logloss:0.15873
[22]	validation_0-logloss:0.12819	validation_1-logloss:0.15763
[23]	validation_0-logloss:0.12650	validation_1-logloss:0.15675
[24]	validation_0-logloss:0.12481	validation_1-logloss:0.15589
[25]	validation_0-logloss:0.12310	validation_1-logloss:0.15488
[26]	validation_0-logloss:0.12149	validation_1-logloss:0.15398
[27]	validation_0-logloss:0.11987	validation_1-logloss:0.15295
[28]	validation_0-logloss:0.11830	validation_1-logloss:0.15208
[29]	validation_0-logloss:0.11675	validation_1-logloss:0.15118
[30]	validation_0-logloss:0.11527	validation_1-logloss:0.15030
[31]	validation_0-logloss:0.11379	validation_1-logloss:0.14952
[32]	validation_0-logloss:0.11249	validation_1-logloss:0.14867
[33]	validation_0-logloss:0.11146	validation_1-logloss:0.14807
[34]	validation_0-logloss:0.11010	validation_1-logloss:0.14735
[35]	validation_0-logloss:0.10900	validation_1-logloss:0.14671
[36]	validation_0-logloss:0.10752	validation_1-logloss:0.14596
[37]	validation_0-logloss:0.10666	validation_1-logloss:0.14542
[38]	validation_0-logloss:0.10559	validation_1-logloss:0.14479
[39]	validation_0-logloss:0.10497	validation_1-logloss:0.14448
[40]	validation_0-logloss:0.10416	validation_1-logloss:0.14414
[41]	validation_0-logloss:0.10339	validation_1-logloss:0.14355
[42]	validation_0-logloss:0.10246	validation_1-logloss:0.14311
[43]	validation_0-logloss:0.10170	validation_1-logloss:0.14278
[44]	validation_0-logloss:0.10083	validation_1-logloss:0.14222
[45]	validation_0-logloss:0.10023	validation_1-logloss:0.14186
[46]	validation_0-logloss:0.09926	validation_1-logloss:0.14132
[47]	validation_0-logloss:0.09872	validation_1-logloss:0.14097
[48]	validation_0-logloss:0.09782	validation_1-logloss:0.14061
[49]	validation_0-logloss:0.09727	validation_1-logloss:0.14028
```

```
[50]	validation_0-logloss:0.09682	validation_1-logloss:0.14005
[51]	validation_0-logloss:0.09606	validation_1-logloss:0.13963
[52]	validation_0-logloss:0.09535	validation_1-logloss:0.13936
[53]	validation_0-logloss:0.09461	validation_1-logloss:0.13885
[54]	validation_0-logloss:0.09385	validation_1-logloss:0.13845
[55]	validation_0-logloss:0.09316	validation_1-logloss:0.13809
[56]	validation_0-logloss:0.09277	validation_1-logloss:0.13784
[57]	validation_0-logloss:0.09221	validation_1-logloss:0.13753
[58]	validation_0-logloss:0.09178	validation_1-logloss:0.13737
[59]	validation_0-logloss:0.09144	validation_1-logloss:0.13717
[60]	validation_0-logloss:0.09074	validation_1-logloss:0.13676
[61]	validation_0-logloss:0.09012	validation_1-logloss:0.13638
[62]	validation_0-logloss:0.08986	validation_1-logloss:0.13622
[63]	validation_0-logloss:0.08971	validation_1-logloss:0.13611
[64]	validation_0-logloss:0.08915	validation_1-logloss:0.13583
[65]	validation_0-logloss:0.08840	validation_1-logloss:0.13548
[66]	validation_0-logloss:0.08767	validation_1-logloss:0.13501
[67]	validation_0-logloss:0.08712	validation_1-logloss:0.13474
[68]	validation_0-logloss:0.08681	validation_1-logloss:0.13456
[69]	validation_0-logloss:0.08656	validation_1-logloss:0.13438
[70]	validation_0-logloss:0.08605	validation_1-logloss:0.13405
[71]	validation_0-logloss:0.08563	validation_1-logloss:0.13389
[72]	validation_0-logloss:0.08492	validation_1-logloss:0.13349
[73]	validation_0-logloss:0.08406	validation_1-logloss:0.13316
[74]	validation_0-logloss:0.08362	validation_1-logloss:0.13305
[75]	validation_0-logloss:0.08295	validation_1-logloss:0.13265
[76]	validation_0-logloss:0.08252	validation_1-logloss:0.13240
[77]	validation_0-logloss:0.08228	validation_1-logloss:0.13222
[78]	validation_0-logloss:0.08161	validation_1-logloss:0.13187
[79]	validation_0-logloss:0.08138	validation_1-logloss:0.13174
[80]	validation_0-logloss:0.08119	validation_1-logloss:0.13159
[81]	validation_0-logloss:0.08075	validation_1-logloss:0.13130
[82]	validation_0-logloss:0.08021	validation_1-logloss:0.13096
[83]	validation_0-logloss:0.08003	validation_1-logloss:0.13083
[84]	validation_0-logloss:0.07948	validation_1-logloss:0.13050
[85]	validation_0-logloss:0.07901	validation_1-logloss:0.13026
[86]	validation_0-logloss:0.07888	validation_1-logloss:0.13016
[87]	validation_0-logloss:0.07858	validation_1-logloss:0.12996
[88]	validation_0-logloss:0.07789	validation_1-logloss:0.12964
[89]	validation_0-logloss:0.07762	validation_1-logloss:0.12941
[90]	validation_0-logloss:0.07719	validation_1-logloss:0.12911
[91]	validation_0-logloss:0.07704	validation_1-logloss:0.12899
[92]	validation_0-logloss:0.07667	validation_1-logloss:0.12891
[93]	validation_0-logloss:0.07620	validation_1-logloss:0.12861
[94]	validation_0-logloss:0.07570	validation_1-logloss:0.12843
[95]	validation_0-logloss:0.07540	validation_1-logloss:0.12832
[96]	validation_0-logloss:0.07491	validation_1-logloss:0.12803
[97]	validation_0-logloss:0.07461	validation_1-logloss:0.12787
```

```
[98]    validation_0-logloss:0.07419    validation_1-logloss:0.12764
[99]    validation_0-logloss:0.07394    validation_1-logloss:0.12745
```

[307]:

```
excl_types = excltype_test.unique()[1:]
print(excl_types)
```

```
['1128b5' '1128b4' '1128a4' '1128a1' '1128a2' '1128a3' '1128b7' 'BRCH SA'
 '1128b6' '1128b3' '1128b1' '1128b2']
```

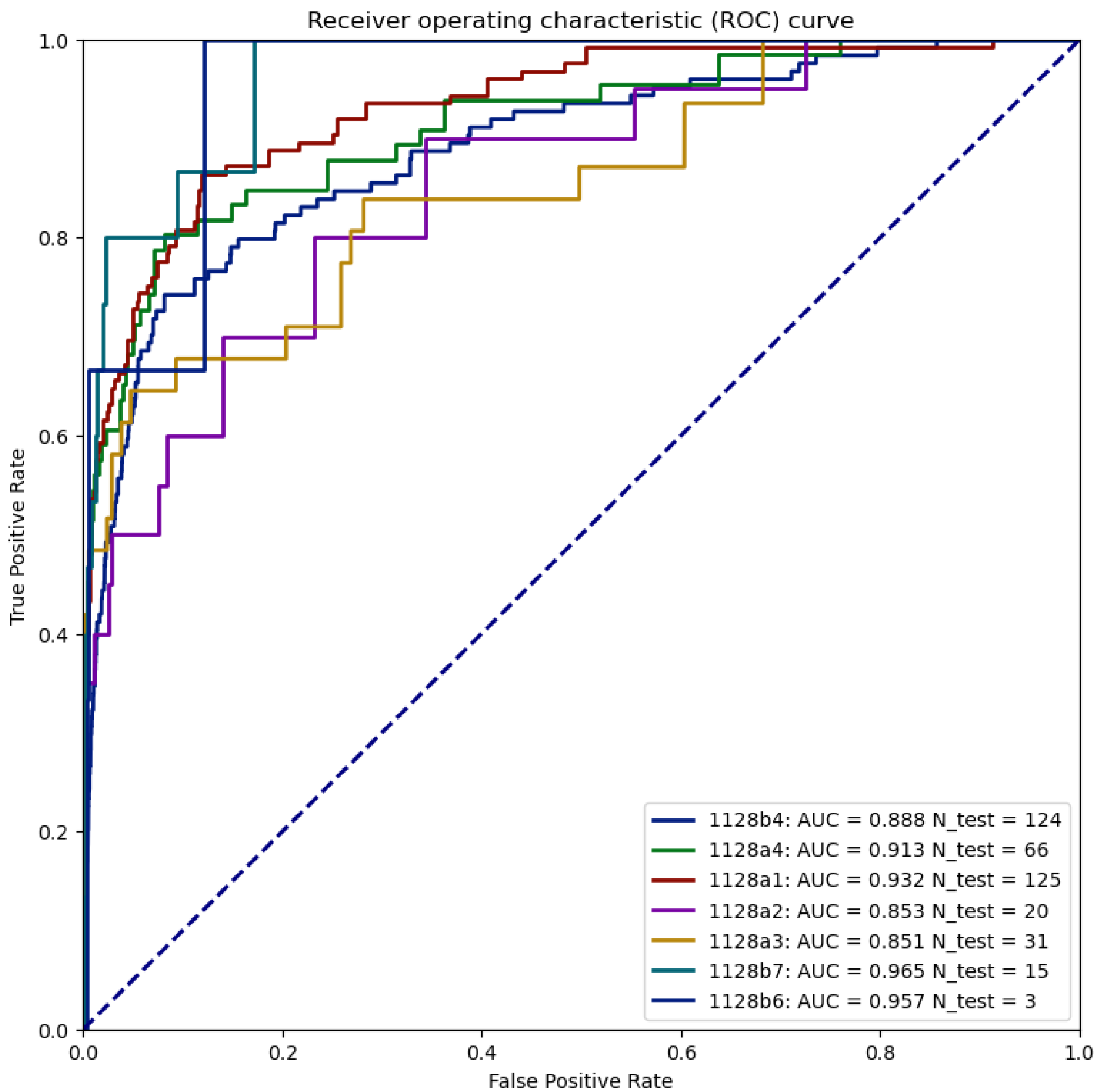
[308]:

```
excltype_map = {
    '1128a1': 'Conviction - Medicare Fraud',
    '1128a2': 'Patient Abuse/Neglect',
    '1128a3': 'Felony - Drugs',
    '1128a4': 'Felony - Healthcare Fraud',
    '1128b1': 'Misdemeanor - Fraud',
    '1128b2': 'Default on Student Loan',
    '1128b3': 'License Revocation',
    '1128b4': 'Unlawful Claims',
    '1128b5': 'Kickbacks/Bribery',
    '1128b6': 'False Claims',
    '1128b7': 'Obstruction of Audit',
    '1128b8': 'Controlled Substances Violation',
    '1128b9': 'Insurance Fraud',
    '1128b10': 'Unlawful Billing',
    '1128b11': 'Quality of Care Violation',
    '1128b12': 'Civil Monetary Penalty',
    '1128b13': 'False Statement',
    '1128b14': 'Suspension/Exclusion',
    '1128b15': 'License Suspension',
    '1128b16': 'Federal Program Violation',
}
```

[309]:

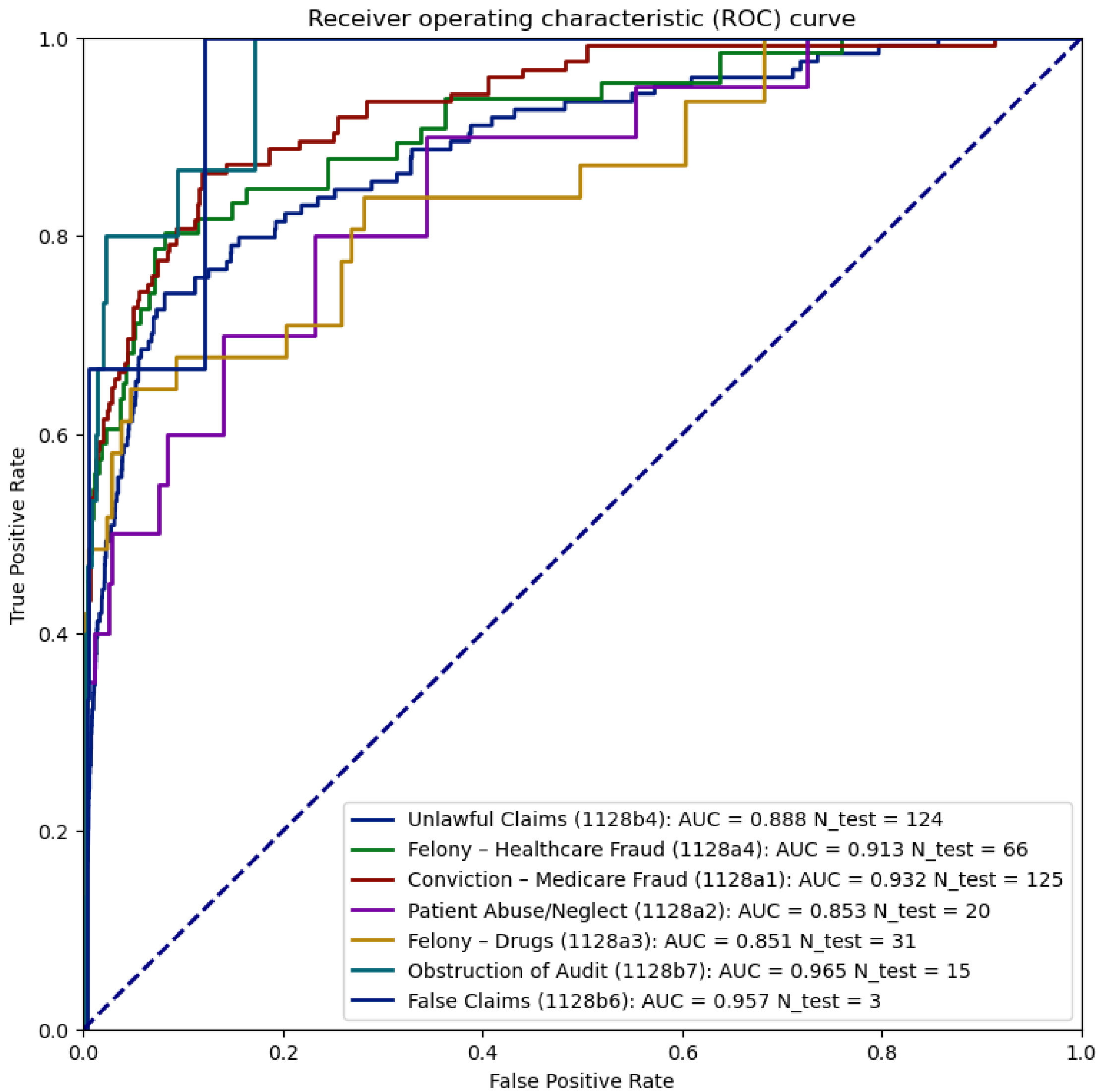
```
plt.figure(figsize=(9,9))
lw = 2

for excl_type in excl_types:


    excl_idx = np.isin(excltype_test.values,['UNK',excl_type])
    Y_pred = clf.predict_proba(X_test[excl_idx])[:,1]
    fpr,tpr, _ = roc_curve(Y_test[excl_idx],Y_pred)
    roc_auc = auc(fpr,tpr)
    n_test = Y_test[excl_idx].sum()
    if n_test < 3: continue
    plt.plot(fpr, tpr,
         lw=lw, label=f"{excl_type}: AUC = {roc_auc:0.3f} N_test = {n_test}")
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()
```



[312]:
```
plt.figure(figsize=(9, 9))
lw = 2

for excl_type in excl_types:
    excl_idx = np.isin(excltype_test.values, ['UNK', excl_type])
    Y_pred = clf.predict_proba(X_test[excl_idx])[:, 1]
```

```
    fpr, tpr, _ = roc_curve(Y_test[excl_idx], Y_pred)
    roc_auc = auc(fpr, tpr)
    n_test = Y_test[excl_idx].sum()
    if n_test < 3:
        continue

    # Map code -> description (fallback to code if not found)
    desc = excltype_map.get(excl_type, excl_type)

    plt.plot(
        fpr,
        tpr,
        lw=lw,
        label=f"{desc} ({excl_type}): AUC = {roc_auc:0.3f} N_test = {n_test}"
    )

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()
```



[311]:
```python
d1 = Counter(excltype_train)
d2 = Counter(excltype_test)
label_map = {t:i for i,t in enumerate(d1 | d2)}
label_map['UNK'] = -1
label_map
```

[311]:
```
{'1128a1': 0,
 'UNK': -1,
 '1128b4': 2,
 '1128a3': 3,
 '1128a2': 4,
 '1128a4': 5,
 '1128b7': 6,
```

```
 '1128b3': 7,
 '1128b1': 8,
 'BRCH SA': 9,
 '1128b6': 10,
 '1128b2': 11,
 '1128b5': 12}
```

[133]:
```
from collections import Counter
Counter(excltype_train)
```

[133]:
```
Counter({'UNK': 29273,
         '1128b4': 362,
         '1128a1': 348,
         '1128a4': 142,
         '1128a3': 64,
         '1128b7': 41,
         '1128a2': 37,
         '1128b1': 11,
         '1128b3': 7,
         'BRCH SA': 5,
         '1128b6': 3,
         '1128b2': 1})
```

[141]:
```
Y_train_ml = [label_map[y] for y in excltype_train]
Y_test_ml = [label_map[y] for y in excltype_test]
```

[142]:
```
#dataset['EXCLTYPE'][:10]
```

[143]:
```
clf_ml =␣
 ↪RandomForestClassifier(n_estimators=150,max_depth=5,class_weight='balanced').
 ↪fit(X_train, Y_train_ml)
```

[144]:
```
Y_pred_ml = clf_ml.predict(X_test)
```

[145]:
```
print(classification_report(Y_test_ml,Y_pred_ml))
```

```
              precision    recall  f1-score   support

          -1       0.99      0.52      0.68      7331
           0       0.02      0.10      0.03        83
           2       0.05      0.28      0.09        87
           3       0.03      0.28      0.06        36
           4       0.00      0.33      0.01        15
           5       0.01      0.30      0.02        10
           6       0.00      0.17      0.01         6
           7       0.02      0.50      0.04         2
           8       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         1
```

```
          11       0.00      0.00      0.00         1
          12       0.00      0.00      0.00         1

    accuracy                           0.51      7574
   macro avg       0.09      0.21      0.08      7574
weighted avg       0.96      0.51      0.66      7574


/opt/conda/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/conda/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/conda/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

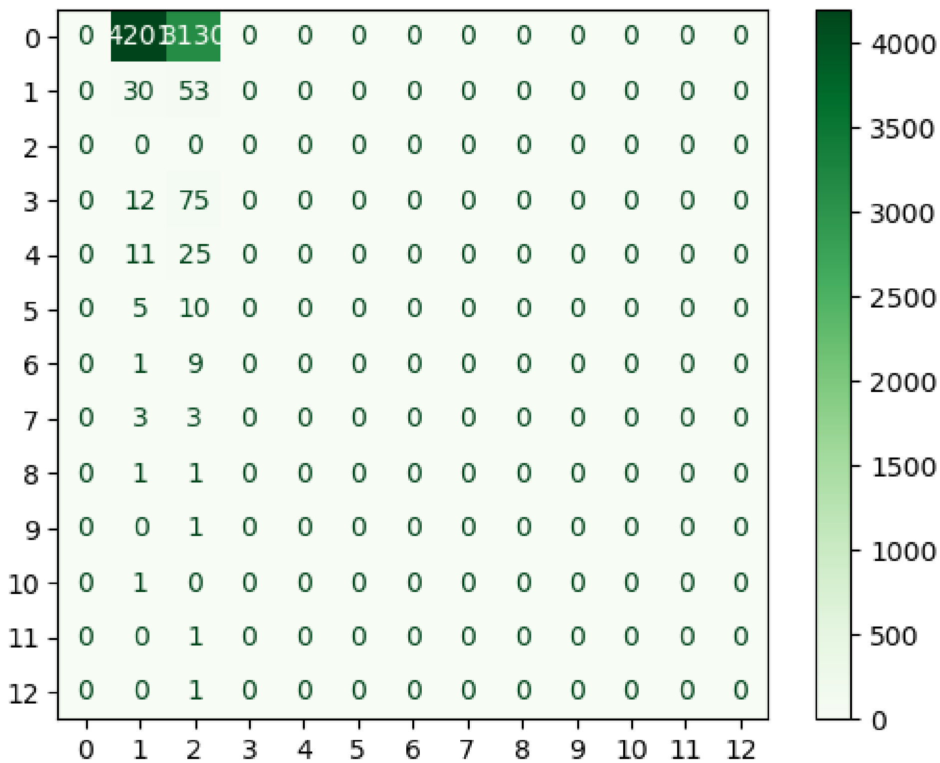
[147]:

```python
cm = confusion_matrix(Y_test_ml, Y_pred_ml > 0.4)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['not␣
↪excluded','excluded'])
disp.plot(cmap=plt.cm.Greens) # Customize colormap if desired
plt.title('Confusion Matrix')
plt.style.use("seaborn-v0_8-dark-palette")
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[147], line 4
      1 cm = confusion_matrix(Y_test_ml, Y_pred_ml > 0.4)
      3 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['not␣
↪excluded','excluded'])
----> 4 disp.plot(cmap=plt.cm.Greens) # Customize colormap if desired
      5 plt.title('Confusion Matrix')
      6 plt.style.use("seaborn-v0_8-dark-palette")

File /opt/conda/lib/python3.12/site-packages/sklearn/metrics/_plot/
↪confusion_matrix.py:181, in ConfusionMatrixDisplay.plot(self, include_values,␣
↪cmap, xticks_rotation, values_format, ax, colorbar, im_kw, text_kw)
    179 if colorbar:
    180     fig.colorbar(self.im_, ax=ax)
```

```
--> 181 ax.set(
    182     xticks=np.arange(n_classes),
    183     yticks=np.arange(n_classes),
    184     xticklabels=display_labels,
    185     yticklabels=display_labels,
    186     ylabel="True label",
    187     xlabel="Predicted label",
    188 )
    190 ax.set_ylim((n_classes - 0.5, -0.5))
    191 plt.setp(ax.get_xticklabels(), rotation=xticks_rotation)

File /opt/conda/lib/python3.12/site-packages/matplotlib/artist.py:147, in Artist.
 ↪__init_subclass__.<locals>.<lambda>(self, **kwargs)
    139 if not hasattr(cls.set, '_autogenerated_signature'):
    140     # Don't overwrite cls.set if the subclass or one of its parents
    141     # has defined a set method set itself.
    142     # If there was no explicit definition, cls.set is inherited from
    143     # the hierarchy of auto-generated set methods, which hold the
    144     # flag _autogenerated_signature.
    145     return
--> 147 cls.set = lambda self, **kwargs: Artist.set(self, **kwargs)
    148 cls.set.__name__ = "set"
    149 cls.set.__qualname__ = f"{cls.__qualname__}.set"

File /opt/conda/lib/python3.12/site-packages/matplotlib/artist.py:1224, in␣
 ↪Artist.set(self, **kwargs)
   1220 def set(self, **kwargs):
   1221     # docstring and signature are auto-generated via
   1222     # Artist._update_set_signature_and_docstring() at the end of the
   1223     # module.
-> 1224     return self._internal_update(cbook.normalize_kwargs(kwargs, self))

File /opt/conda/lib/python3.12/site-packages/matplotlib/artist.py:1216, in␣
 ↪Artist._internal_update(self, kwargs)
   1209 def _internal_update(self, kwargs):
   1210     """
   1211     Update artist properties without prenormalizing them, but generatin
   1212     errors as if calling `set`.
   1213
   1214     The lack of prenormalization is to maintain backcompatibility.
   1215     """
-> 1216     return self._update_props(
   1217␣
 ↪        kwargs, "{cls.__name__}.set() got an unexpected keyword argument "
   1218             "{prop_name!r}")

File /opt/conda/lib/python3.12/site-packages/matplotlib/artist.py:1192, in␣
 ↪Artist._update_props(self, props, errfmt)
```

```
   1189             if not callable(func):
   1190                 raise AttributeError(
   1191                     errfmt.format(cls=type(self), prop_name=k))
-> 1192             ret.append(func(v))
   1193 if ret:
   1194     self.pchanged()

File /opt/conda/lib/python3.12/site-packages/matplotlib/axes/_base.py:74, in␣
 ↪_axis_method_wrapper.__set_name__.<locals>.wrapper(self, *args, **kwargs)
     73 def wrapper(self, *args, **kwargs):
---> 74     return get_method(self)(*args, **kwargs)

File /opt/conda/lib/python3.12/site-packages/matplotlib/axis.py:2071, in Axis.
 ↪set_ticklabels(self, labels, minor, fontdict, **kwargs)
   2067 elif isinstance(locator, mticker.FixedLocator):
   2068     # Passing [] as a list of labels is often used as a way to
   2069     # remove all tick labels, so only error for > 0 labels
   2070     if len(locator.locs) != len(labels) and len(labels) != 0:
-> 2071         raise ValueError(
   2072             "The number of FixedLocator locations"
   2073             f" ({len(locator.locs)}), usually from a call to"
   2074             " set_ticks, does not match"
   2075             f" the number of labels ({len(labels)}).")
   2076     tickd = {loc: lab for loc, lab in zip(locator.locs, labels)}
   2077     func = functools.partial(self._format_with_dict, tickd)

ValueError: The number of FixedLocator locations (13), usually from a call to␣
 ↪set_ticks, does not match the number of labels (2).
```


0 4201 3130 0 0 0 0 0 0 0 0 0 0
0 30 53 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 12 75 0 0 0 0 0 0 0 0 0 0
0 11 25 0 0 0 0 0 0 0 0 0 0
0 5 10 0 0 0 0 0 0 0 0 0 0
0 1 9 0 0 0 0 0 0 0 0 0 0
0 3 3 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9 10 11 12
4000
3500
3000
2500
2000
1500
1000
500
0