precusor

March 3, 2019

The exploratory data analysis indicated that the tests commonly used at admission would not in themselves provide a basis for diagnosing Pulmonary embolism. The Classification and regression analysis indicated that factors such as age, income, gender and ethnicity are not indicative of Pulmonary Embolism. This analysis looks at other diseases which may be associated with Pulmonary Embolism and which therefore might be a useful sign that a Pulmonary Embolism might also be considered.

0.0.1 A 10,000-patient database that contains in total 10,000 patients, 36,143 admissions, and 10,726,505 lab observations.

0.0.2 PatientCorePopulatedTable

1

2 [PatientID] - a unique ID representing a patient.

3 [PatientGender] - Male/Female.

4 [PatientDateOfBirth] - Date Of Birth.

5 [PatientRace] - African American, Asian, White.

6 [PatientMaritalStatus] - Single, Married, Divorced, Separated, Widowed.

7 [PatientLanguage] - English, Icelandic, Spanish.

8 [PatientPopulationPercentageBelowPoverty] - given in %.

9

9.0.1 AdmissionsCorePopulatedTable

10

11 [PatientID] - a unique ID representing a patient.

12 [AdmissionID] - an admission ID for the patient.

13 [AdmissionStartDate] - start date.

14 [AdmissionEndDate] - end date.

15

15.0.1 AdmissionsDiagnosesCorePopulatedTable

16

17 [PatientID] - a unique ID representing a patient.

18 [AdmissionID] - an admission ID for the patient.

19 [PrimaryDiagnosisCode] - ICD10 code for admission's primary diagnosis.

20 [PrimaryDiagnosisDescription] - admission's primary diagnosis de-

```python
print(__doc__)
from copy import deepcopy
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import decomposition
from sklearn import datasets
from sklearn import preprocessing
from sklearn.impute import SimpleImputer
from scipy.ndimage import convolve
from sklearn import linear_model, datasets, metrics
from sklearn.model_selection import train_test_split
from sklearn.neural_network import BernoulliRBM
from sklearn.pipeline import Pipeline
from sklearn.base import clone
from sklearn.datasets import make_multilabel_classification
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import CCA
from matplotlib.mlab import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
from scipy.stats import ttest_ind
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.utils import shuffle
from sklearn import tree
#from sklearn.cross_validation import cross_val_score
from pydotplus import graph_from_dot_data
from sklearn.linear_model import LogisticRegression
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn import  linear_model
from sklearn.model_selection import cross_val_score
from IPython.display import display
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.calibration import calibration_curve
from sklearn import neighbors
from sklearn.calibration import calibration_curve
from sklearn.calibration import CalibratedClassifierCV
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.metrics import classification_report,confusion_matrix
#from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeClassifier
import scipy.stats as stats
#import researchpy as rp
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
        from sklearn.decomposition import PCA
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
        import matplotlib.pyplot as plt

        import scipy.stats as ss
        import itertools as it

Automatically created module for IPython interactive environment


In [225]: ############################################
        ### AdmissionsDiagnosesCorePopulatedTable ###
        ############################################

        file_to_open2 =  "AdmissionsDiagnosesCorePopulatedTable.csv"
        columnsdiagnoses = ['PatientID', 'PatientGender', 'PrimaryDiagnosisCode', 'PrimaryDia
        f2 = open(file_to_open2)
        diagnosesdf = pd.read_csv(f2, index_col=False, names=columnsdiagnoses)
        del diagnosesdf['PatientMaritalStatus']
        diagnosesdf.head()

In [234]: # Since section II showed that the Lab Results are not explainary in this case: Look

        I26df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I26')])
        I26df['PrimaryDiagnosisCode']= 'I26'
        I27df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I27')])
        I27df['PrimaryDiagnosisCode']= 'I27'
        I28df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I28')])
        I28df['PrimaryDiagnosisCode']= 'I28'
        I82df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I82')])
        I82df['PrimaryDiagnosisCode']= 'I82'
        I83df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I83')])
        I83df['PrimaryDiagnosisCode']= 'I83'
        I21df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I21')])
        I21df['PrimaryDiagnosisCode']= 'I21'
        I22df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I22')])
        I22df['PrimaryDiagnosisCode']= 'I22'
        I23df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I23')])
        I23df['PrimaryDiagnosisCode']= 'I23'
        I24df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I24')])
        I24df['PrimaryDiagnosisCode']= 'I24'
        I25df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I25')])
        I25df['PrimaryDiagnosisCode']= 'I25'
        I50df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I50')])
        I50df['PrimaryDiagnosisCode']= 'I50'
        I163df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I163')]
        I163df['PrimaryDiagnosisCode']= 'I163'
        I97df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I97')])
```

```
I97df['PrimaryDiagnosisCode']= 'I97'
I97df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('I97')])
I97df['PrimaryDiagnosisCode']= 'I97'
T67df=deepcopy(diagnosesdf[diagnosesdf.PrimaryDiagnosisCode.str.startswith('T67')])
T67df['PrimaryDiagnosisCode']= 'T67'
```

```
In [235]: precusor = [I26df, I27df,I28df, I82df,I83df, I21df, I22df, I23df,I24df, I25df,I50df,
          precusordf = pd.concat(precusor)
          precusordf.head()
```

```
In [236]: #dropping  all factors but ICD10 code
          precusordf=precusordf.drop(['PatientID','PatientGender','PrimaryDiagnosisDescription
```

```
In [237]: precusordf=pd.get_dummies(precusordf)
          precusordf.head()
```

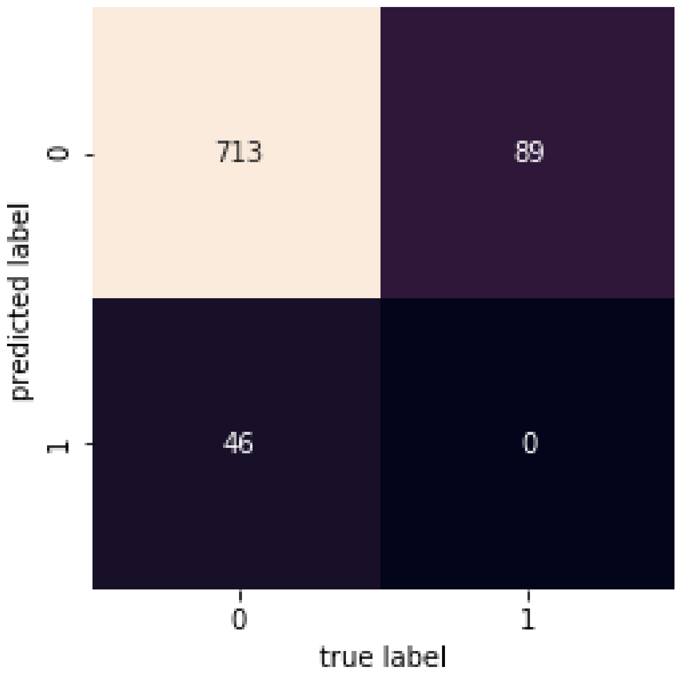
```
In [238]: def cramers_corrected_stat(confusion_matrix):
              """ calculate Cramers V statistic for categorial-categorial association.
                  uses correction from Bergsma and Wicher,
                  Journal of the Korean Statistical Society 42 (2013): 323-328
              """
              chi2 = ss.chi2_contingency(confusion_matrix)[0]
              n = confusion_matrix.sum()
              phi2 = chi2/n
              r,k = confusion_matrix.shape
              phi2corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))
              rcorr = r - ((r-1)**2)/(n-1)
              kcorr = k - ((k-1)**2)/(n-1)
              return np.sqrt(phi2corr / min( (kcorr-1), (rcorr-1)))
```

```
In [239]: list(precusordf)
```

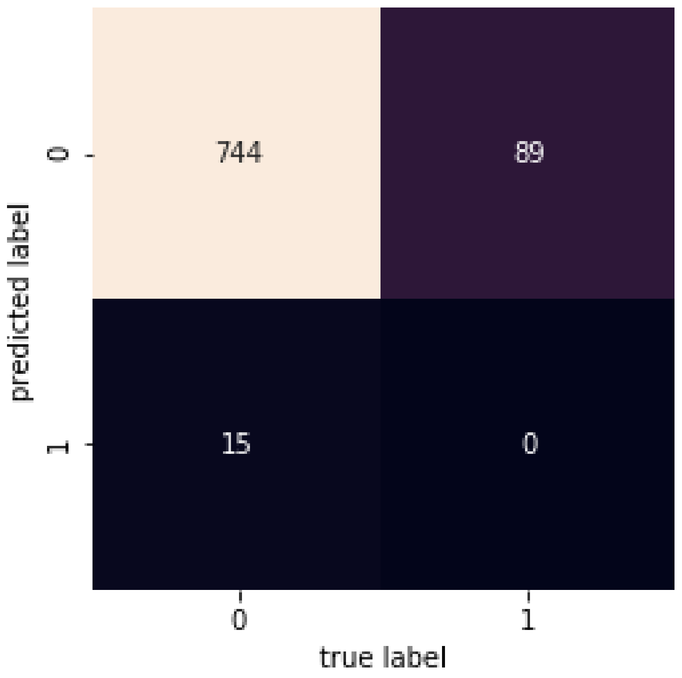
```
Out[239]: ['PrimaryDiagnosisCode_I21',
           'PrimaryDiagnosisCode_I23',
           'PrimaryDiagnosisCode_I24',
           'PrimaryDiagnosisCode_I25',
           'PrimaryDiagnosisCode_I26',
           'PrimaryDiagnosisCode_I27',
           'PrimaryDiagnosisCode_I28',
           'PrimaryDiagnosisCode_I82',
           'PrimaryDiagnosisCode_I97']
```

```
In [240]: x1=precusordf['PrimaryDiagnosisCode_I21']
          x2=precusordf['PrimaryDiagnosisCode_I23']
          x3=precusordf['PrimaryDiagnosisCode_I24']
          x4=precusordf['PrimaryDiagnosisCode_I25']
          x5=precusordf['PrimaryDiagnosisCode_I27']
          x6=precusordf['PrimaryDiagnosisCode_I28']
          x7=precusordf['PrimaryDiagnosisCode_I82']
          x8=precusordf['PrimaryDiagnosisCode_I97']
          y=precusordf['PrimaryDiagnosisCode_I26']
```

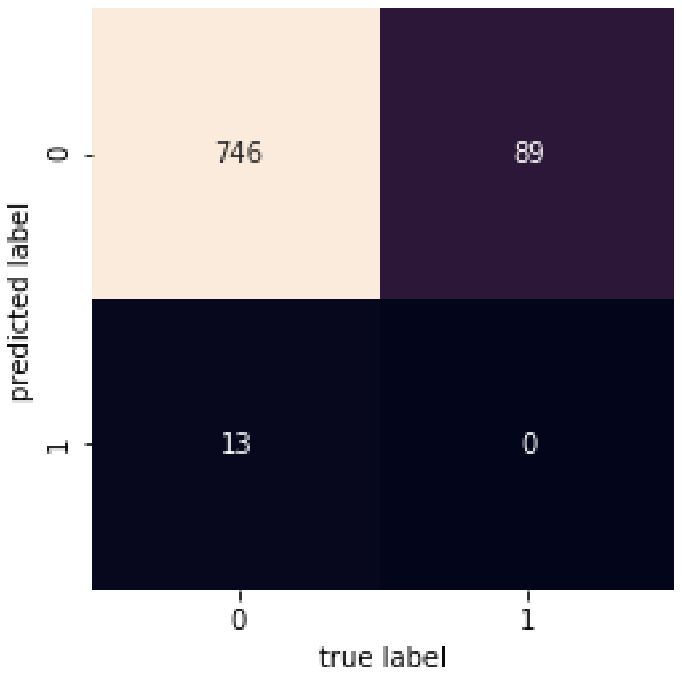
```
In [241]: confusion_matrix1=pd.crosstab(y,x1)
          confusion_matrix1

In [0]:

In [242]: mat1=confusion_matrix1
          sns.heatmap(mat1.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[242]: Text(91.68,0.5,'predicted label')

Out[242]:
```



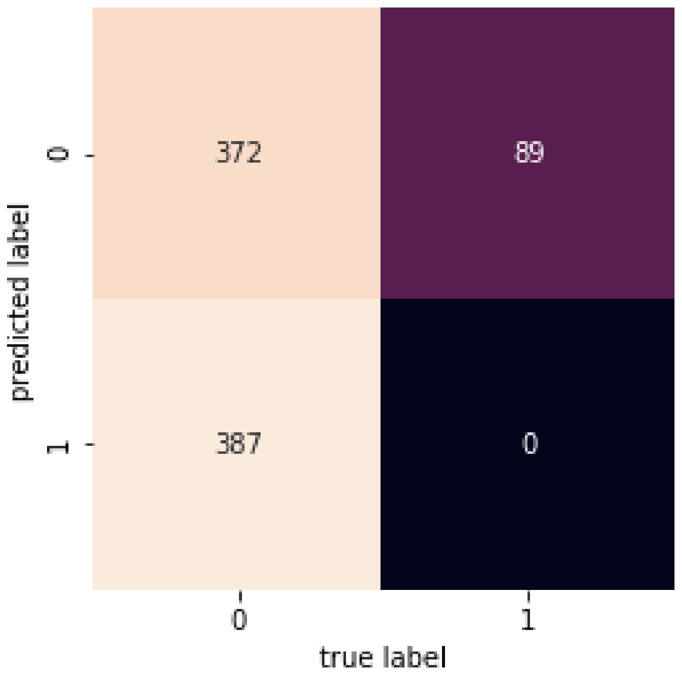
```
In [244]: confusion_matrix2=pd.crosstab(y,x2)
          confusion_matrix2

In [245]: mat2=confusion_matrix2
          sns.heatmap(mat2.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[245]: Text(91.68,0.5,'predicted label')

Out[245]:
```



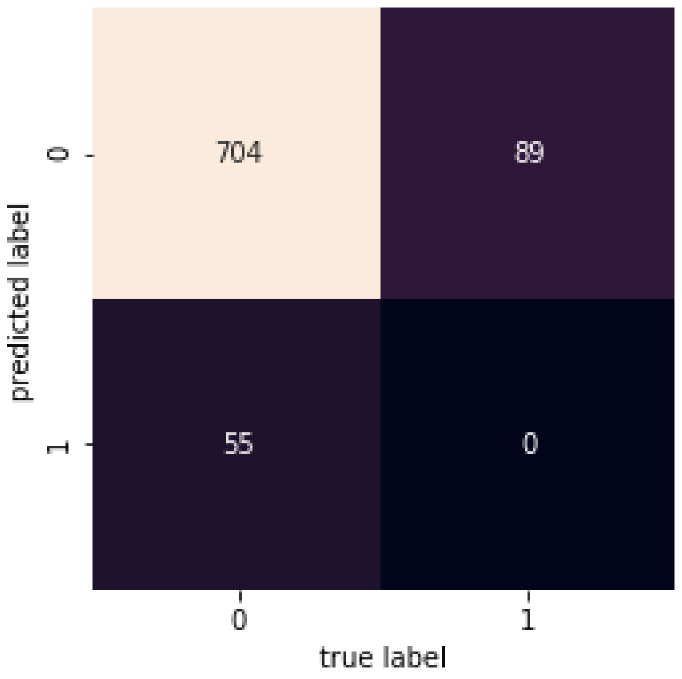
```
In [246]: confusion_matrix3=pd.crosstab(y,x3)
          confusion_matrix3

In [247]: mat3=confusion_matrix3
          sns.heatmap(mat3.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[247]: Text(91.68,0.5,'predicted label')

Out[247]:
```



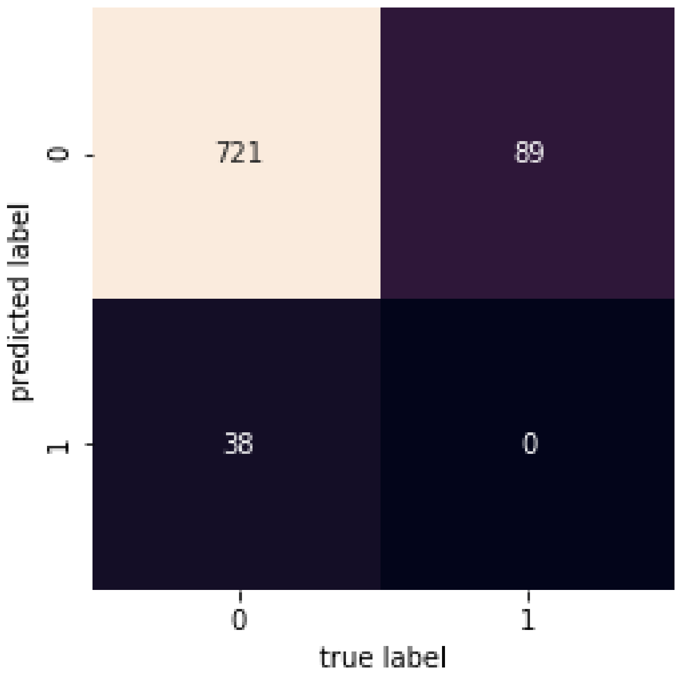
```
In [248]: confusion_matrix4=pd.crosstab(y,x4)
          confusion_matrix4

In [249]: mat4=confusion_matrix4
          sns.heatmap(mat4.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[249]: Text(91.68,0.5,'predicted label')

Out[249]:
```



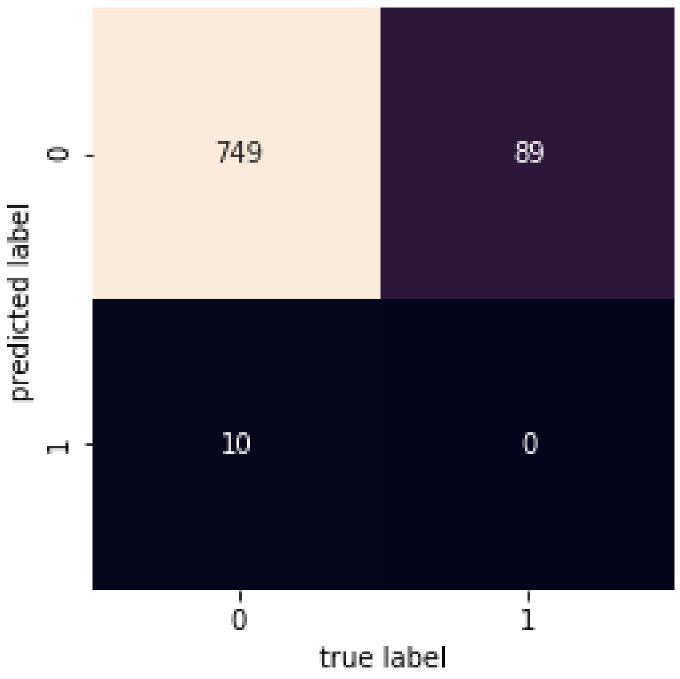
```
In [250]: confusion_matrix5=pd.crosstab(y,x5)
          confusion_matrix5

In [251]: mat5=confusion_matrix5
          sns.heatmap(mat5.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[251]: Text(91.68,0.5,'predicted label')

Out[251]:
```



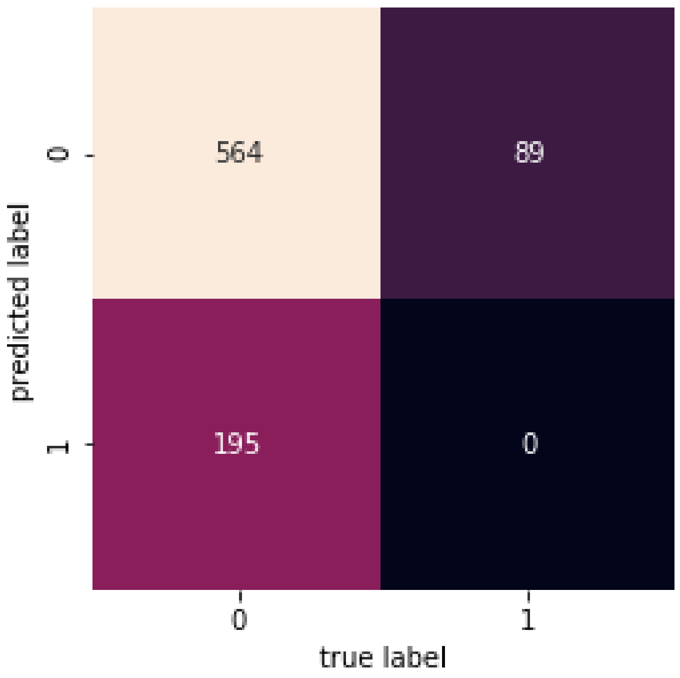
```
In [252]: confusion_matrix6=pd.crosstab(y,x6)
          confusion_matrix6

In [253]: mat6=confusion_matrix6
          sns.heatmap(mat6.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[253]: Text(91.68,0.5,'predicted label')

Out[253]:
```



```
In [254]: confusion_matrix7=pd.crosstab(y,x7)
          confusion_matrix7

In [255]: mat7=confusion_matrix7
          sns.heatmap(mat7.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[255]: Text(91.68,0.5,'predicted label')

Out[255]:
```



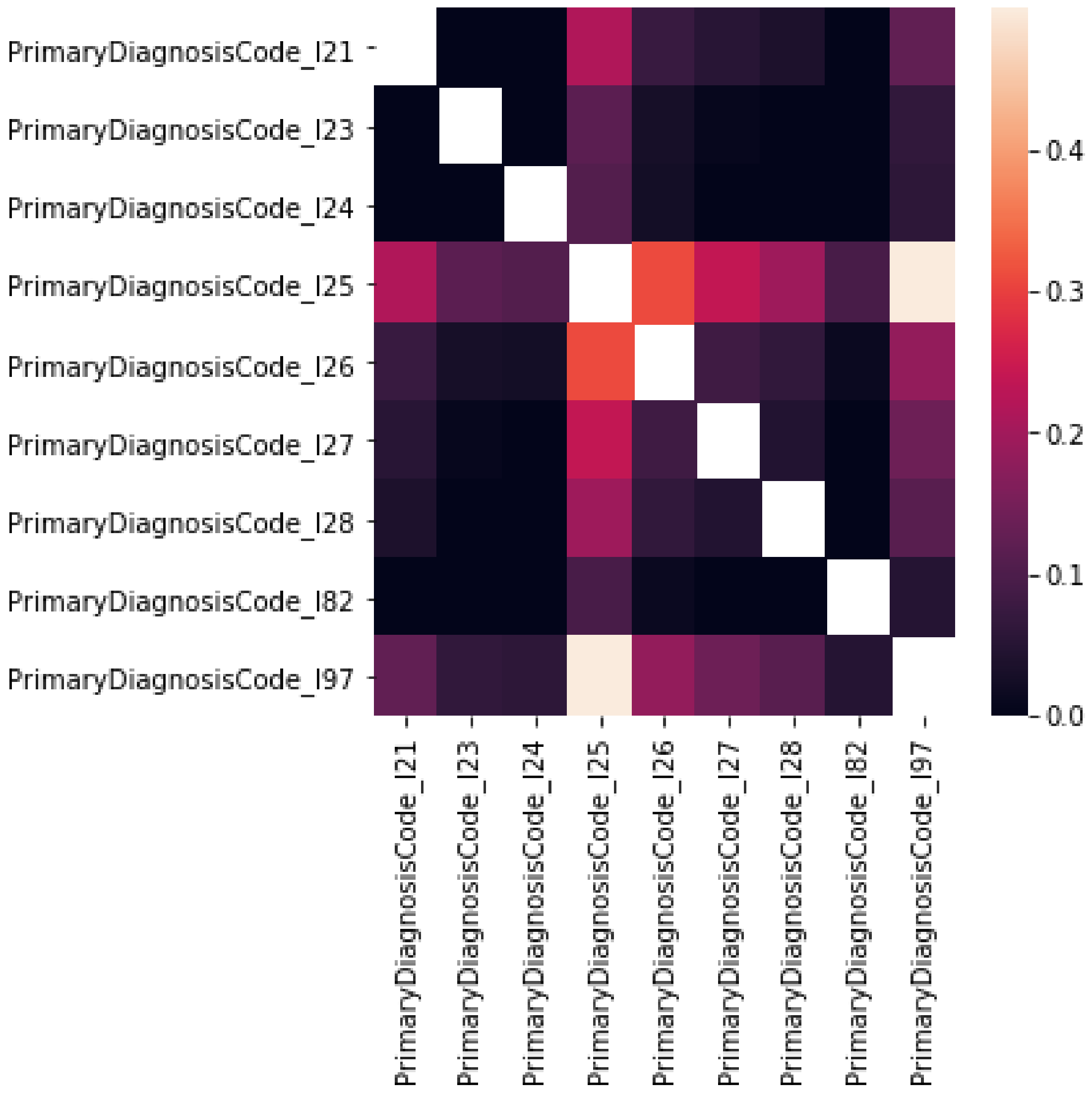
```
In [256]: confusion_matrix8=pd.crosstab(y,x8)
          confusion_matrix8

In [257]: mat8=confusion_matrix8
          sns.heatmap(mat8.T, square=True, annot=True, fmt='d', cbar=False)
          plt.xlabel('true label')
          plt.ylabel('predicted label')

Out[257]: Text(91.68,0.5,'predicted label')

Out[257]:
```



```
In [260]: def cramers_corrected_stat(x,y):

              """ calculate Cramers V statistic for categorial-categorial association.
                  uses correction from Bergsma and Wicher,
                  Journal of the Korean Statistical Society 42 (2013): 323-328
              """
              result=-1
              if len(x.value_counts())==1 :
                  print("First variable is constant")
              elif len(y.value_counts())==1:
                  print("Second variable is constant")
              else:
                  conf_matrix=pd.crosstab(x, y)

                  if conf_matrix.shape[0]==2:
                      correct=False
                  else:
                      correct=True

                  chi2 = ss.chi2_contingency(conf_matrix, correction=correct)[0]

                  n = sum(conf_matrix.sum())
                  phi2 = chi2/n
                  r,k = conf_matrix.shape
```

```
            phi2corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))
            rcorr = r - ((r-1)**2)/(n-1)
            kcorr = k - ((k-1)**2)/(n-1)
            result=np.sqrt(phi2corr / min( (kcorr-1), (rcorr-1)))
        return round(result,6)
```

```
In [261]: cramers_corrected_stat(x1,y)

Out[261]: 0.074509

In [262]: cramers_corrected_stat(x2,y)

Out[262]: 0.030528

In [263]: cramers_corrected_stat(x3,y)

Out[263]: 0.025411

In [264]: cramers_corrected_stat(x4,y)

Out[264]: 0.312044

In [265]: cramers_corrected_stat(x5,y)

Out[265]: 0.083429

In [266]: cramers_corrected_stat(x6,y)

Out[266]: 0.065769

In [267]: cramers_corrected_stat(x7,y)

Out[267]: 0.014795

In [268]: cramers_corrected_stat(x8,y)

Out[268]: 0.184053

In [269]: results = pd.DataFrame(columns=precusordf.columns, index=precusordf.columns)
          for (label1, column1), (label2, column2) in it.combinations(precusordf.items(), 2):
              results.loc[label1, label2] = results.loc[label2, label1] = cramers_corrected_sta
          results.fillna(0)
          results = results[results.columns].astype(float)
          results

In [305]: plt.rcParams["figure.figsize"]=5,5
          sns.heatmap(results)

Out[305]: <matplotlib.axes._subplots.AxesSubplot object at 0x7f3615b0f5d0>

Out[305]:
```



```
In [307]: from colorama import init, Fore, Back, Style


          for i in list(results):
              for j in list(results):
                  if ((results.at[i,j] > 0.1) & (results.at[i,j]<0.2)):
                      print(Fore.RED +'There is a weak association between ',[i,j])
                  elif ((results.at[i,j] > 0.2) & (results.at[i,j]<0.3)):
                      print(Fore.YELLOW +'There is a moderate association ',[i,j])
                  elif (results.at[i,j] > 0.3):
                        print(Fore.GREEN + 'There is a strong association between',[i,j])
```

```
There is a moderate association  ['PrimaryDiagnosisCode_I21', 'PrimaryDiagnosisCode_I25']There
```

So looking at associated illnesses has been quite successful and we can say that: Pulmonary embolism is strongly associated with Chronic ischemic heart disease. That Chronic ischemic heart disease is moderately associated with myocardial infarction That there is a moderate association between Chronic ischemic heart disease and Primary pulmonary hypertension That there is a weak association between Pulmonary Embolism and Intraoperative and postprocedural complications and disorders of circulatory system That Chronic ischemic heart disease is weakly associated with Hemopericardium as current complication following acute myocardial infarction That Chronic ischemic heart disease is weakly associated with Acute coronary thrombosis

An actionable insight from this is that a patient presenting with Chronic ischemic heart disease, primary pulmonary hypertension, Acute coronary thrombosis or a myocardial infarction should be considered at risk from Pulmonary Embolism as well and that further investigation perhaps using a D-dimer test woild be benefical.

```
In [0]:
```