DATA-X

FINAL WRITTEN REPORT

PREDICTING MOOD DISORDERS (II)

Yin Zhang, Yuyang Pan, Simarjeev Singh, Jillian Chan

Executive Summary

Mentored by Professor Dott. Ing. Roberto V. Zicari from the Frankfurt Big Data Lab, our team worked on a clinical dataset provided by the Geisinger Health System based in Pennsylvania, US. The dataset contains 7 data files with 171,543 patients' data, spanning the years of 1995-2017. After investigation in the data, our team decided to focus on finding the correlation among depression, insomnia/sleeping disorders and anxiety as well as demographic factors using logistics regression and random forest classification. The outcome of our project gives a general sense of which population are more likely to be diagnosed as depressed and will eventually help doctors identify a patient's likelihood of getting depression and thus mitigate the effects of mood disorders.

Table of Contents

1. Problems Solved & Details

1.1 Set the goal of the project

After receiving the data, the first problem we encountered was the size of the dataset. This clinical dataset is over 50 GB and is composed of a lot of clinical vocabularies that our team were having trouble understanding with. However, after meeting and investigating into the "Data Dictionary" file, which was a summary of the whole database, we chose the direction of analyzing the correlation of depression and insomnia as well as other demographic factors including age, gender, ethnicity and more.

1.2 Choose the data will be used

Once we decided on the direction, we narrowed down the scope to only two data files: one called "problem_list" -- a list of all the medical problems each patient had and the dates the disease was noted, "demographics", which contains the demographic information and 7 encounter diagnosis of each patient, as well as "social history", which records the smoking status, alcohol usage, illicit drug usage, sexual activity and use of birth control of each study ID at every encounter.

In our dataset, "study ID" refers to an identifier unique to each patient and "encounter" refers to a logged hospital visit.

1.3 Data cleaning and profiling

We chose to work on 3 main files, for each, our team started on cleaning the data (deleting NAN values and duplicate rows) and dropping the columns unnecessary for our project.

1.3.1 Problem List & Demographics

Since both "Problem List" and "Demographics" file contains the historical medical problem a patient suffered, we kind of cleaned them together and merged the result into one dataframe, which include the basic information and the history of our target diseases: insomnia, depression and anxiety.

First of all, for each Study ID, we checked both "Problem List" and "Demographics" data to find out whether this person has/had the target diseases using keywords such as "depression", "insomnia", "anxiety" and "panic disorder". Then for each target disease, we found the earliest date of the patient suffering from this disease. By comparing these dates, we created new columns to code for the chronological precedence of a medical condition over another, and named them "anx_before_ins", "ins_before_anx", etc.

In the meanwhile, we also created new columns named "insomnia", "depression" and "anxiety" to just mark whether a certain patient has/had one of the target diseases.

For the basic information of each patients, we splitted columns with more than 2 bands to do one-hot encoding.

1.3.2 Social History

The primary key for “Social History” is encounter ID due to the fact that each patient can go to the doctor more than once with different encounter ID. In this file, his/her social history factors were logged at every encounter.

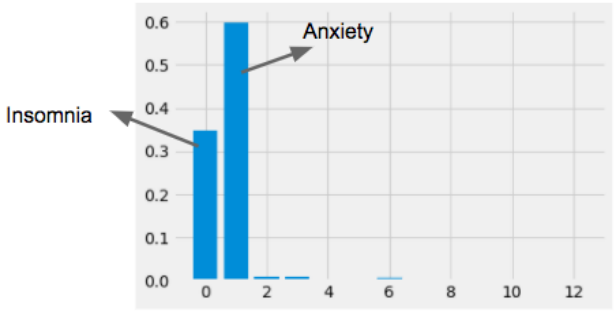
We extracted the year during which the patient exhibited the certain social history factor. Initially we tried to record each patient’s historical social history factors, i.e. whether he/she has ever declared to have used illicit drugs, alcohol usage, etc. However, many patients exhibited different social history factors throughout the span of small timeframes (i.e. changes within a year). As a result, we did not use social history with the concerns that using social history variables based on whether a patient has ever exhibited a factor would not be accurate enough.

1.4 Classification

After data profiling and data blending, our team started on using random forest to rank the importance of each feature (age, gender, ethnicity and more) in relation to depression.

1.4.1 Random Forest

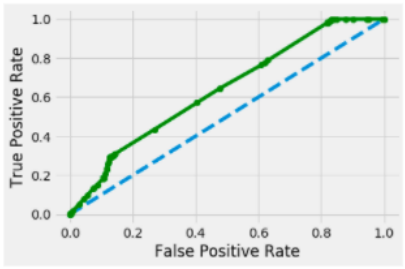
We found that insomnia and anxiety, as expected, were vastly more important than other factors in predicting depression with the use of random forest. We had a random forest accuracy of 70.4%.



1.4.2 Logistic Regression

In the meantime, we also worked on using logistic regression to find the correlation between each factor and depression based on a feature set. We obtained an accuracy of logistic regression of 67.4%, similarly with anxiety and insomnia displaying the highest significance out of all the factors in use in predicting depression.

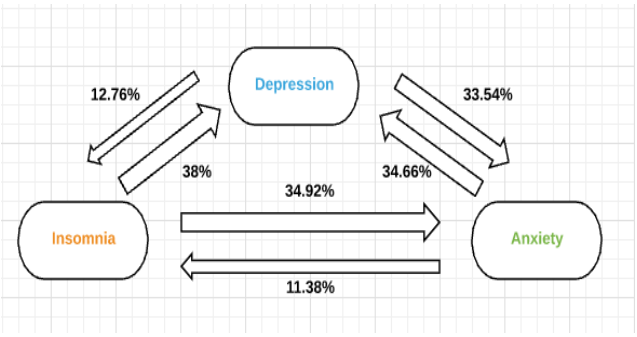
anxiety	0.92
insomnia	1.17
pt_sex	-0.0018
pt_ethnicity	0.034
age0	-0.022
age1	-0.0038
age2	-0.026
age3	-0.0113
race0	0.49
race1	0.315
race2	0.22
race3	0.24
race4	0.26
intercept	-1.72

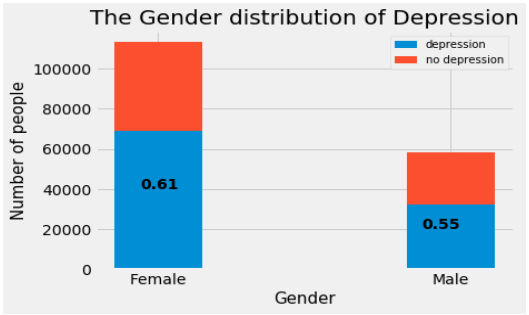


2. Full System of Solution

Our existing model is valuable because we have successfully extracted patient features from a very large dataset and have chronologically coded their medical history status. We obtained positive results for the prediction of depression based on these factors, especially “insomnia” and “anxiety”, which are the factors highlighted in medical literature.

In addition, we verify that the relationships among depression, insomnia and anxiety are multi-directional by using chronological precedence of medical condition of all the patients. Furthermore, we confirmed researches onto the factors predisposed towards depression -- females, anxiety patients, or insomnia patients, for instance.





Hence, our research could form a basis for further analysis of the data by the Geisinger team, providing them with more insights from the large amounts of datasets they receive regularly.

3. Future Improvements

Given that the current accuracy of both our models is around 70%, we are trying to figure out the ways to improve our models and increase the accuracy.

3.1 Model Accuracy: Classifier Parameters

We could tune our random forest parameters, such as testing out multiple minimum sample leaf sizes (to vary the capturing of noise in train data) or increasing the number of

RF trees. We also could look as ensemble classification techniques such as bagging (bootstrap aggregating) and boosting to enhance our model.

We could also increase our model's accuracy by a more targeted train-test-split. This could be done by sorting the patients' data by year and using the first few years' data as training set and the last few years as testing set. In this way, we can compare our testing results with the actual ones and could have a better look at the performance of our models.

3.2 Model Accuracy: Adding Features

Since we experienced technical difficulty in adding patients' social history to our models during this semester, our team is considering keeping working on it and adding this important feature (including drug use, alcohol consumption, smoking habit) to our model.

4. Experience of working with Mentor

During this semester, our team had a great experience with working with our mentors.

At the beginning of the semester, given several directions proposed by Prof. Zicari, our team chose to investigate the correlation between insomnia and depression.

While working on the data, we had several skype meetings with our mentors. The first meeting was in the middle of the semester with Prof. Ohayon and Prof. Cristina from Stanford University. By that time we just finished cleaning, blending the data and building our first logistic regression model. After looking into our results in details, Prof. Ohayon and Prof. Cristina suggested us to also include the factor of anxiety to our data and further investigate the causal relationships among three: anxiety, insomnia, and depression. And later we added this significant feature to our models.

The second meeting with Prof. Zicari occurred at the end of the semester. We proposed our results to him and got some feedback. He was happy with our outcomes and suggested us to take the factor social history into account. Although we had some trouble adding this feature in, this should be done in the future to further improve the classification models.

5. Contribution of Each Member

Jillian Chan	Data cleaning and data profiling of "social history", logistic regression and random forest,communication with mentors, document management.
Yuyang Pan	Literature review, document management, communication with mentors, help with logistic regression and random forest models.
Simarjeev Singh	Data cleaning, building logistic regression and random forest models.
Yin Zhang	Data cleaning and data profiling of "problem list" and "demographics", plot charts (flow charts, bar charts, ROC curves), logistic model analysis, document management, literature review, communication with mentors.

Appendix

3ai. Data cleaning

```
import numpy as np
import pandas as pd
demog=pd.read_pickle('C:/Users/anel/mooddisorder project/demog.pkl')
problemlist=pd.read_pickle('C:/Users/anel/mooddisorder project/problemlist.pkl')

insomnia=[]
anxiety=[]
depression=[]
ins_before_anx=[]
anx_before_ins=[]
ins_before_dep=[]
dep_before_ins=[]
anx_before_dep=[]
dep_before_anx=[]

studyid=list(set(demog['studyid']))

#for i in range(3566,len(studyid)):
for i in range(3566):
    tempdemog=demog.loc[i]
    tempprob=problemlist[problemlist['studyid']==demog.loc[i]['studyid']]

    dt_dep0 = pd.to_datetime(tempprob[tempprob['prob_dx_nm'].str.contains('DEPRESSIVE|DYSTHYMIA|DEPRESS
    dt_ins0 = pd.to_datetime(tempprob[tempprob['prob_dx_nm'].str.contains('INSOMNIA|SLEEP')]['prob_note
    dt_anx0 = pd.to_datetime(tempprob[tempprob['prob_dx_nm'].str.contains('ANXIETY|ANXIOUS|PANIC')]['pr

    dt_dep0=list(dt_dep0)
    dt_ins0=list(dt_ins0)
    dt_anx0=list(dt_anx0)

tempdemog=tempdemog.dropna()
for j in range(len(tempdemog)):
    if ('ANXIOUS' in tempdemog[j]) or ('ANXIETY' in tempdemog[j]) or ('PANIC DISORDER' in tempdemog[j]):
        dt_anx1 = pd.to_datetime(tempdemog.values[j-2])
        dt_anx0.append(dt_anx1)
    if ('INSOMNIA' in tempdemog[j]) or ('SLEEP' in tempdemog[j]):
        dt_ins1 = pd.to_datetime(tempdemog.values[j-2])
        dt_ins0.append(dt_ins1)
    if ('DEPRESSIVE' in tempdemog[j]) or ('DYSTHYMIA' in tempdemog[j]) or ('DEPRESSION' in tempdemog[j]):
        dt_dep1 = pd.to_datetime(tempdemog.values[j-2])
        dt_dep0.append(dt_dep1)

#dt_anx=dt_anx0.append(dt_anx1)
#dt_ins=dt_ins0.append(dt_ins1)
#dt_dep=dt_dep0.append(dt_dep1)

dt_dep=dt_dep0
dt_ins=dt_ins0
dt_anx=dt_anx0

if len(dt_dep)==0:
    depression.append(0)
else:
    depression.append(1)
    #e_dep=min(dt_dep)
if len(dt_ins)==0:
    insomnia.append(0)
else:
    insomnia.append(1)
    #e_ins=min(dt_ins)
if len(dt_anx)==0:
    anxiety.append(0)
else:
    anxiety.append(1)
    #e_anx=min(dt_anx)
```

```
    if len(dt_ins)>0 and len(dt_anx)>0:
        if min(dt_ins)<min(dt_anx):
            ins_before_anx.append(1)
            anx_before_ins.append(0)
        elif min(dt_ins)>min(dt_anx):
            anx_before_ins.append(1)
            ins_before_anx.append(0)
        else:
            anx_before_ins.append(1)
            ins_before_anx.append(1)
    elif len(dt_ins)>0 and len(dt_anx)==0:
        ins_before_anx.append(1)
        anx_before_ins.append(0)
    elif len(dt_ins)==0 and len(dt_anx)>0:
        anx_before_ins.append(1)
        ins_before_anx.append(0)
    else:
        anx_before_ins.append(0)
        ins_before_anx.append(0)
#######
    if len(dt_ins)>0 and len(dt_dep)>0:
        if min(dt_ins)<min(dt_dep):
            ins_before_dep.append(1)
            dep_before_ins.append(0)
        elif min(dt_ins)>min(dt_dep):
            dep_before_ins.append(1)
            ins_before_dep.append(0)
        else:
            dep_before_ins.append(1)
            ins_before_dep.append(1)
    elif len(dt_ins)>0 and len(dt_dep)==0:
        ins_before_dep.append(1)
        dep_before_ins.append(0)
    elif len(dt_ins)==0 and len(dt_dep)>0:
        dep_before_ins.append(1)
        ins_before_dep.append(0)
    else:
        dep_before_ins.append(0)
        ins_before_dep.append(0)
########
    if len(dt_anx)>0 and len(dt_dep)>0:
        if min(dt_anx)<min(dt_dep):
            anx_before_dep.append(1)
            dep_before_anx.append(0)
        elif min(dt_anx)>min(dt_dep):
            dep_before_anx.append(1)
            anx_before_dep.append(0)
        else:
            dep_before_anx.append(1)
            anx_before_dep.append(1)
    elif len(dt_anx)>0 and len(dt_dep)==0:
        anx_before_dep.append(1)
        dep_before_anx.append(0)
    elif len(dt_anx)==0 and len(dt_dep)>0:
        dep_before_anx.append(1)
        anx_before_dep.append(0)
    else:
        dep_before_anx.append(0)
        anx_before_dep.append(0)

    if i%10==0:
        print(i)
```

3ci. Social History: Smoking

```
##############change columns with more than 2 categories into 0,1##############
smoking=np.zeros((len(df_sh),10))
temp=list(df_sh['soc_hx_smoke_stts'])
for i in range(len(df_sh)):
    if temp[i]==0:
        smoking[i,0]=1
    elif temp[i]==5:
        smoking[i,0]=1
    elif temp[i]==1:
        smoking[i,1]=1
    elif temp[i]==9:
        smoking[i,1]=1
    elif temp[i]==2:
        smoking[i,2]=1
    elif temp[i]==7:
        smoking[i,2]=1
    elif temp[i]==3:
        smoking[i,3]=1
    elif temp[i]==6:
        smoking[i,3]=1
    else:
        smoking[i,3]=0

df_sh['never_smoked']=smoking[:,0]
df_sh['prev_smoked']=smoking[:,1]
df_sh['heavy_smoker']=smoking[:,2]
df_sh['light_smoker']=smoking[:,3]
```

3civ. Social History: Historical Factors Aggregated

```
def get_final_encounter(studyid):
    df_temp = df_temp = df_sh_c_renewl.loc[df_sh_c_renewl['studyid'] == studyid]
    values = []
    for index, row in df_temp.iterrows():
        values.append(row['soc_hx_ill_drug_yn'] + row['smoked_before'])
    try:
        final_encounter = values.index(2)
    except ValueError:
        try:
            final_encounter = values.index(1)
        except ValueError:
            try:
                final_encounter = values.index(10)
            except ValueError:
                try:
                    final_encounter = values.index(9)
                except ValueError:
                    final_encounter = 0

    return list(df_temp['encid'])[final_encounter]
```

```
studyids_sh = set(df_sh_c_renewl['studyid'])
final_encounters = []
for studyid in studyids_sh:
    final_encounters.append(get_final_encounter(studyid))

final_encounters
```