Predicting the stress level:

By Youssef ARRASSEN



Abstract

The aim of this project is to predict the stress of the user, we get the data from Kaggle, and we did some cleaning and optimization of data. And trained a Model to predict the stress level.

Keywords: DASS, Stress, Predicting Stress.

1. Introduction

When we look at stress it plays a factor in performance, and we need to minimize it, for this a set of question in place to determine the level of a person stress level, there is a test called DASS (Depression Anxiety Stress Scales), we created a simple ML that can predict if the user is stressed or not.

The dataset is from Kaggle, and updated last 2 months (version 13), and I took a copy and hosted it on my GitHub account just updating the model and improving it in the future, more details in “Dataset”

1. Dataset Description

The Dataset is from Kaggle ([https://www.kaggle.com/)](https://www.kaggle.com/), and it include 47 features we kept only what is related to stress:

Q1 I found myself getting upset by quite trivial things.

Q6 I tended to over-react to situations.

Q8 I found it difficult to relax.

Q11 I found myself getting upset rather easily.

Q12 I felt that I was using a lot of nervous energy.

Q14 I found myself getting impatient when I was delayed in any way (eg, elevators, traffic lights, being kept waiting).

Q18 I felt that I was rather touchy.

Q22 I found it hard to wind down.

Q27 I found that I was very irritable.

Q29 I found it hard to calm down after something upset me.

Q32 I found it difficult to tolerate interruptions to what I was doing.

Q33 I was in a state of nervous tension.

Q35 I was intolerant of anything that kept me from getting on with what I was doing.

Q39 I found myself getting agitated.

Each item was presented one at a time in a random order for each new participant along with a 4 point rating scale asking the user to indicate how often that had been true of them in the past week where

1 = Normal

2 = Mild

3 = Moderate

4 = Severe

1+2 = Not stressed

3+4 = Stressed

On the next page was a generic demographics survey with many different questions.

TIPI1 Extraverted, enthusiastic.

TIPI2 Critical, quarrelsome.

TIPI3 Dependable, self-disciplined.

TIPI4 Anxious, easily upset.

TIPI5 Open to new experiences, complex.

TIPI6 Reserved, quiet.

TIPI7 Sympathetic, warm.

TIPI8 Disorganized, careless.

TIPI9 Calm, emotionally stable.

TIPI10 Conventional, uncreative.

The TIPI items were rated "I see myself as:" \_\_\_\_\_ such that

1 = Disagree strongly

2 = Disagree moderately

3 = Disagree a little

4 = Neither agree nor disagree

5 = Agree a little

6 = Agree moderately

7 = Agree strongly

|  |  |
| --- | --- |
| Education:  "How much education have you completed?" | 1=Less than high school  2=High school  3=University degree  4=Graduate degree |
| Urban:  "What type of area did you live when you were a child?" | 1=Rural (country side)  2=Suburban  3=Urban (town, city) |
| English native:  "Is English your native language?" | 1=Yes,  2=No |
| Age:  "How many years old are you?" | (Num) |
| Religion:  "What is your religion?" | 1=Agnostic,  2=Atheist,  3=Buddhist,  4=Christian (Catholic),  5=Christian (Mormon),  6=Christian (Protestant),  7=Christian (Other),  8=Hindu,  9=Jewish,  10=Muslim,  11=Sikh,  12=Other |
| Married:  "What is your marital status?" | 1=Never married,  2=Currently married,  3=Previously married |
| Family size:  "Including you, how many children did your mother have?" | (Num) |
| Screen size | 1=device with small screen (phone, etc)  2=device with big screen (laptop, desktop, etc) |

\*We achieved higher accuracy when working with QA alone

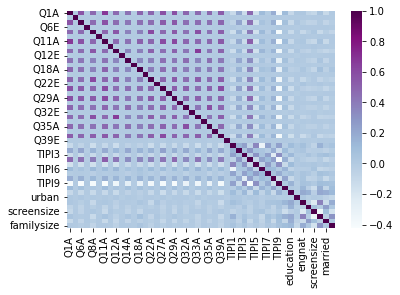
1. Methods

Since we want to classify this dataset, We picked 3 type of models to train:

* Decision Tree
* K-NN
* Random Forest

We removed the not needed features such as QA related to depression and anxiety, the goal here is to find the best QA that affect the Stress level.

From the correlation graph:

Figure 3.1 : dataset correlation

And also after removing the not related features:

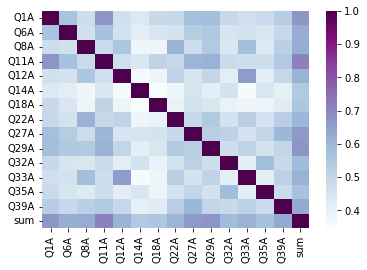


Figure 3.2: dataset of QA correlation.

1. Experimental Results

The Methods I used are Decision Tree, K-NN, Random Forest, and then I compared the 3 models with different parameters,

**With** **Decision Tree**, we achieved an accuracy of 0.8, then I got the feature importance plotted

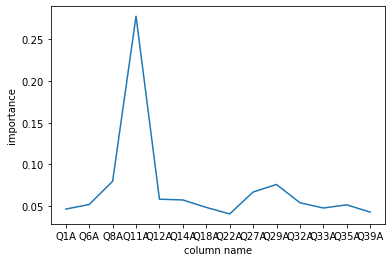


Figure 4.1: Feature Importance for the Stress Questions

These 3 are the top and very important:

* Q8 I found it difficult to relax.
* Q11 I found myself getting upset rather easily.
* Q29 I found it hard to calm down after something upset me.

**With K-NN**, we tested the dataset with multiple to check for the best one, accuracy is lower from 0.89 to 0.94.

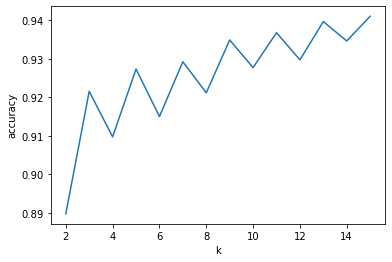


Figure 4.2: K-NN with different K

With Random Forest

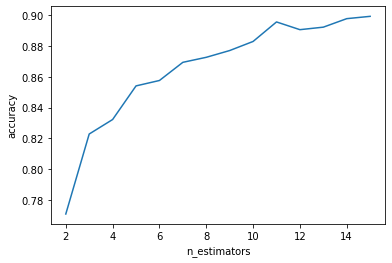


Figure 4.3: Random Forest tested on QA\_dataset

We got the following confusion matrix (4x4):

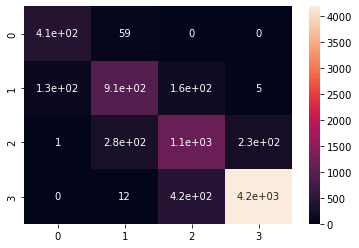


Figure 4.4: Confusion Matrix generated from the Random forest model

1. Conclusions

The One that perform best is Random Forest, I am still planning to improve the model, so I put it a github repo for future improvement.

the model is able to predict the stress category the tester is in. We can have an accuracy up to 0.95 but still can improved while reducing the amount of questions, as the original Goal was.

References

**Depression Anxiety Stress Scales (DASS)**

<https://www.psytoolkit.org/survey-library/depression-anxiety-stress-dass.html>

**Predicting Depression, Anxiety and Stress**

https://www.kaggle.com/yamqwe/depression-anxiety-stress-scales

Source code is hosted on github:

<https://github.com/youssef02/ML-and-PR->

Sourcecode :

**Predicting the stress level:**

1. the Data are from Kaggle.
2. [<https://www.kaggle.com/yamqwe/depression-anxiety-stress-scales>]

*#ignore if have files in the directory*

**!**mkdir DASS\_data\_21.02.19

**!**wget https://raw.githubusercontent.com/youssef02/ML-and-PR-/main/DASS\_data\_21.02.19/data.csv -O ./DASS\_data\_21.02.19/data.csv

**import** pandas **as** pd

**import** numpy **as** np

*#loading data*

pd**.**options**.**display**.**max\_columns **=** 172

df **=** pd**.**read\_csv('./DASS\_data\_21.02.19/data.csv', sep**=**r'\t', engine**=**'python')

df**.**head()

*# ne pas oublier de changer*

*# we don't need I because the order does not matter*

**for** i **in** df:

**if** i**.**endswith('I'):

df**.**drop(i, axis**=**1, inplace**=True**)

df**.**head()

df**.**drop(['introelapse'] , axis**=**1, inplace**=True**)

df**.**drop(['testelapse'] , axis**=**1, inplace**=True**)

df**.**drop(['surveyelapse'] , axis**=**1, inplace**=True**)

df**.**drop(['gender'] , axis**=**1, inplace**=True**)

df**.**drop(['voted'] , axis**=**1, inplace**=True**)

df**.**drop(['orientation'] , axis**=**1, inplace**=True**)

df**.**drop(['race'] , axis**=**1, inplace**=True**)

df**.**drop(['hand'] , axis**=**1, inplace**=True**)

df**.**drop(['uniquenetworklocation'] , axis**=**1, inplace**=True**)

*#not needed*

df**.**drop(['source'] , axis**=**1, inplace**=True**)

**for** i **in** df:

**if** i**.**startswith('VCL'):

df**.**drop(i, axis**=**1, inplace**=True**)

df['age']**.**sort\_values(ascending**=False**)**.**head(n**=**10)

StressQ **=** [1, 6, 8, 11, 12, 14, 18, 22, 27, 29, 32, 33, 35, 39]

**for** i **in** range(1,43):

a **=** ["Q"**+**str(i)**+**"E"]

b **=** ["Q"**+**str(i)**+**"A"]

**if** i **not** **in** StressQ:

df**.**drop(a, axis**=**1, inplace**=True**)

df**.**drop(b, axis**=**1, inplace**=True**)

print ("deleted not related questions to stress")

columns\_drop\_list **=** []

df**.**dtypes

*#dara cleaning*

*#df.drop('major', axis=1, inplace=True)*

*#find strings in columns*

**for** i **in** df**.**columns:

**if** df[i]**.**dtype **==** 'object':

columns\_drop\_list**.**append(i)

**for** i **in** columns\_drop\_list:

df**.**drop(i, axis**=**1, inplace**=True**)

*#get 'Q1A' column rows*

*#implelement the crossvalidation using skilearn*

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(df**.**iloc[:,:**-**1], df**.**iloc[:,**-**1], test\_size**=**0.2, random\_state**=**0)

print(X\_train**.**shape)

print(X\_test**.**shape)

print(y\_train**.**shape)

print(y\_test**.**shape)

*#plotting the data*

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

*#data visualization*

sns**.**heatmap(df\_QA**.**corr(), cmap**=**'BuPu')

*#get list of columns that start with Q and end with A*

columns\_QA **=** []

**for** i **in** df**.**columns:

**if** i**.**startswith('Q') **and** i**.**endswith('A'):

columns\_QA**.**append(i)

df["sum"] **=** df[columns\_QA]**.**sum(axis**=**1)

columns\_QA**.**append("sum")

print(columns\_QA)

df**.**head()

*#get 3 max sum*

df**.**sort\_values(by**=**['sum'], ascending**=False**)**.**head(n**=**3)

df\_QA **=** df[columns\_QA]

df\_QA**.**head()

*#from https://www.psytoolkit.org/survey-library/depression-anxiety-stress-dass.html*

*#if sum is less than 14 then it is 0*

*#if sum is between 15 and 18 then it is 1*

*#if sum is between 19 and 25 then it is 2*

*#if sum is between 26 and 33 then it is 3*

*#if it more than 33 then it is 4*

**for** i **in** range(len(df\_QA)):

**if** df\_QA**.**iloc[i, **-**1] **<** 14:

df\_QA**.**iloc[i, **-**1] **=** 0

**elif** df\_QA**.**iloc[i, **-**1] **<** 19:

df\_QA**.**iloc[i, **-**1] **=** 1

**elif** df\_QA**.**iloc[i, **-**1] **<** 26:

df\_QA**.**iloc[i, **-**1] **=** 2

**elif** df\_QA**.**iloc[i, **-**1] **<** 33:

df\_QA**.**iloc[i, **-**1] **=** 3

**else**:

df\_QA**.**iloc[i, **-**1] **=** 4

df\_QA**.**head()

*#plotthe count of each value of sum*

df\_QA["sum"]**.**value\_counts()

plt**.**plot(df\_QA["sum"]**.**value\_counts())

plt**.**pie(df\_QA["sum"]**.**value\_counts(), labels**=**["1", "2", "3", "4"], autopct**=**'%1.1f%%')

plt**.**show()

*# chnging the*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(df\_QA**.**iloc[:,:**-**1], df\_QA**.**iloc[:,**-**1], test\_size**=**0.2, random\_state**=**0)

*#show head of all the train and test data*

*#new dataframe where has only the columns that start with Q and end with A*

*#train test split*

*#shuffle the data*

df\_QA **=** df\_QA**.**sample(frac**=**1)**.**reset\_index(drop**=True**)

*#train test split*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(df\_QA**.**iloc[:,:**-**1], df\_QA**.**iloc[:,**-**1], test\_size**=**0.2, random\_state**=**0)

*#knn on df\_QA*

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.metrics **import** accuracy\_score

acclist\_QA **=** []

**for** i **in** range(2,16):

knn **=** KNeighborsClassifier(n\_neighbors**=**i)

knn**.**fit(X\_train, y\_train)

y\_pred **=** knn**.**predict(X\_test)

acclist\_QA**.**append(accuracy\_score(y\_test, y\_pred))

*#plotting the accuracy*

plt**.**plot(range(2,16), acclist\_QA)

plt**.**xlabel('k')

plt**.**ylabel('accuracy')

plt**.**show()

**for** i **in** range(len(df)):

**if** df**.**iloc[i, **-**1] **<** 14:

df**.**iloc[i, **-**1] **=** 0

**elif** df**.**iloc[i, **-**1] **<** 19:

df**.**iloc[i, **-**1] **=** 1

**elif** df**.**iloc[i, **-**1] **<** 26:

df**.**iloc[i, **-**1] **=** 2

**elif** df**.**iloc[i, **-**1] **<** 33:

df**.**iloc[i, **-**1] **=** 3

**else**:

df**.**iloc[i, **-**1] **=** 4

df**.**head()

*#shuffle the data*

df **=** df**.**sample(frac**=**1)**.**reset\_index(drop**=True**)

*#train test split*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(df**.**iloc[:,:**-**1], df**.**iloc[:,**-**1], test\_size**=**0.2, random\_state**=**0)

*#random forest on df*

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.metrics **import** accuracy\_score

acclist\_df **=** []

**for** i **in** range(2,16):

rf **=** RandomForestClassifier(n\_estimators**=**i)

rf**.**fit(X\_train, y\_train)

y\_pred **=** rf**.**predict(X\_test)

acclist\_df**.**append(accuracy\_score(y\_test, y\_pred))

*#plotting the accuracy*

plt**.**plot(range(2,16), acclist\_df)

plt**.**xlabel('n\_estimators')

plt**.**ylabel('accuracy')

plt**.**show()

*#implement random forest on df\_QA*

column\_TIPI **=** []

**for** i **in** df**.**columns:

**if** i**.**startswith('TIPI'):

column\_TIPI**.**append(i)

print(column\_TIPI)

df\_TIPI **=** df[column\_TIPI]

*#select colomns that start with TIPI*

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.metrics **import** accuracy\_score

*#train test split*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(df\_QA**.**iloc[:,:**-**1], df\_QA**.**iloc[:,**-**1], test\_size**=**0.2, random\_state**=**0)

acclist\_QA **=** []

**for** i **in** range(2,50):

rf **=** RandomForestClassifier(n\_estimators**=**i)

rf**.**fit(X\_train, y\_train)

y\_pred **=** rf**.**predict(X\_test)

acclist\_QA**.**append(accuracy\_score(y\_test, y\_pred))

*#plotting the accuracy*

plt**.**plot(range(2,50), acclist\_QA)

*#decision tree regressor*

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.metrics **import** accuracy\_score

*#train test split*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(df\_QA**.**iloc[:,:**-**1], df\_QA**.**iloc[:,**-**1], test\_size**=**0.2, random\_state**=**0)

dtc **=** DecisionTreeClassifier()

dtc**.**fit(X\_train, y\_train)

y\_pred **=** dtc**.**predict(X\_test)

print(accuracy\_score(y\_test, y\_pred))

imp **=** dtc**.**feature\_importances\_

plt**.**plot(imp)

*#plot with column name*

*#plot x axis with column name*

plt**.**xlabel('column name')

plt**.**ylabel('importance')

plt**.**xticks(range(len(imp)), df\_QA**.**columns[:**-**1])

plt**.**show()

*#print top 3 features*

list\_imp **=** []

**for** i **in** range(len(imp)):

**if** imp[i] **>** 0.05:

print(df\_QA**.**columns[i], imp[i])

list\_imp**.**append([df\_QA**.**columns[i], imp[i]])

*#sort list\_imp*

list\_imp**.**sort(key**=lambda** x: x[1], reverse**=True**)

*#print top 3*

**for** i **in** range(3):

print(list\_imp[i])

*#plot age vs. Q1A*

**import** seaborn **as** sns

sns**.**lmplot(x**=**'age', y**=**'sum', data**=**df, fit\_reg**=False**)

*#random forest on df with k=4*

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.metrics **import** accuracy\_score

rf **=** RandomForestClassifier(n\_estimators**=**4)

rf**.**fit(X\_train, y\_train)

print(accuracy\_score(y\_test, y\_pred))

*#generate confusion matrix*

**from** sklearn.metrics **import** confusion\_matrix

y\_pred **=** rf**.**predict(X\_test)

cm **=** confusion\_matrix(y\_test, y\_pred)

*#heatmap the cm*

sns**.**heatmap(cm, annot**=True**)

df**.**shape