Depression Detection with DM

Umme Marzia Haque[1]

[1]American International University - Bangladesh

Dec 17, 2021

1

dx.doi.org/10.17504/protocols.io.bzm8p49w

Umme Marzia Haque
American International University - Bangladesh, University o...

The study has used data from YMM. The Yes/No variables that had a low correlation with target variable have been removed. To extract the most relevant features , the high correlated variables with the target variable , the Boruta method was used in conjunction with a Random Forest(RF) Classifier. To select suitable supervised learning models, the Tree-based Pipeline Optimization Tool To select suitable supervised learning models, the Tree-based Pipeline Optimization Tool (TPOTclassifier) has been employed. RF, XGBoost (XGB), Decision Tree (DT), and Gaussian Naive Bayes (GaussianNB) have been employed in the depression identification step.has been employed. RF, XGBoost (XGB), Decision Tree (DT), and Gaussian Naive Bayes (GaussianNB) were employed in the depression identification step.

DOI

dx.doi.org/10.17504/protocols.io.bzm8p49w

https://doi.org/10.1371/journal.pone.0261131

Umme Marzia Haque 2021. Depression Detection with DM. **protocols.io** https://dx.doi.org/10.17504/protocols.io.bzm8p49w

document

Haque UM, Kabir E, Khanam R (2021) Detection of child depression using machine learning methods. PLoS ONE 16(12): e0261131. doi: 10.1371/journal.pone.0261131

document ,

Nov 01, 2021

Dec 17, 2021

Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

54688

The study has used data from YMM. The Yes/No variables that had a low correlation with target variable have been removed. To extract the most relevant features , the high correlated variables with the target variable , the Boruta method was used in conjunction with a Random Forest(RF) Classifier. To select suitable supervised learning models, the Tree-based Pipeline Optimization Tool To select suitable supervised learning models, the Tree-based Pipeline Optimization Tool (TPOTclassifier) has been employed. RF, XGBoost (XGB), Decision Tree (DT), and Gaussian Naive Bayes (GaussianNB) have been employed in the depression identification step.has been employed. RF, XGBoost (XGB), Decision Tree (DT), and Gaussian Naive Bayes (GaussianNB) were employed in the depression identification step.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)


# In[2]:


df = pd.read_csv("new_parent_file.csv")
df.head()


# In[3]:


# take a backup copy
df1 = df
len(df1.index) # number of rows


# In[7]:


# taking columns those contains Yes and No
df3 = df[['householdID'] + [col for col in df2.columns if((df2[col].all() == 'Yes') | (df2[col].all() == 'No') )]]
#df3.dropna(inplace=True)
df3.head()
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
# In[8]:


# version 2
df3.fillna('unknown',inplace=True)
df3.head()


# In[11]:


for x in df3.columns:
  col_val = df3[x].unique()
  #version 1
  #if((len(col_val) > 2) or ('Yes' not in col_val ) or ('No' not in col_val )):
  #    df3.drop(x, axis=1, inplace=True)
  #version 2
  if ((len(col_val) == 3) and ((['Yes','No','Unknown'] == col_val).any()== False)):
    df3.drop(x, axis=1, inplace=True)

  elif(len(col_val) > 3):
    df3.drop(x, axis=1, inplace=True)

df3.head(20)


# In[12]:


for x in df3.columns:
  df3[x].replace({'No': 0, 'Yes': 1, 'Unknown': 2}, inplace=True)


# In[1]:


from sklearn.preprocessing import LabelEncoder
for x in df3.columns:
  LE = LabelEncoder()
  #df3[x] = LE.fit_transform(df3[x])
  df3[x] = LE.fit_transform(df3[x].astype(str))
  #df3[x] = df3.apply(lambda x: LE.fit_transform(x), axis=0, result_type='expand')
print(df3)
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

In[21]:

```
df4=df3
```

In[22]:

```
#if distdf.loc[i, 'DISTINGUISH']==distdf.loc[i+1, 'DISTINGUISH']:
for i in range(len(df4)):
  if((df4.loc[i,'PFI19B_10']==0) and (df4.loc[i,'pmdy']==0) and (df4.loc[i,'pmdm']==0) and
(df4.loc[i,'PCH3G'] ==0)):
    df4.loc[i,'depressed']=0

  else:
    df4.loc[i,'depressed']=1
```

In[23]:

```
df4.drop(['sup1','SUP2a','SUP2b','SUP2d','sup3','sup12','sup13','sup14','SUP17a','SUP17b','SUP17d','sup18
','sup19','PFI20G_9','panxmim','pbehyim','panxyim','podmim','pcdmim','pad3mim','pad2mim','pad1mim','p
admim','poccmim','pocomim','pocmim','pgamim','psamim','psomim','podyim','pcdyim','pad3ymim','pad2y
mim','pad1ymim','padymim','pad3yim','pad2yim','pad1yim','poccyim','pocoyim','pocyim','pgayim','psayim',
'psoyim','pbehm','panxm','panxy','podysubt','podm','pody','padysubt','pad3m','pad2m','pad1m','padym','pa
d3ym','pad2ym','pad1ym','pady','pad3y','pad2y','pad1y','pocysubt','pocon','pocoz','pocm','pocy','poccm','po
ccy','pocom','pocoy','pgaysubt','pgam','pgay','psaysubt','psam','psay','psoysubt','psom','psoy','pbehmim','p
mdysubt','pcdy','pcdm','pcdysubt','panxdepy','pDISCy','panxdepm','padyim','pmdyim','pmdmim','panxdep
mim','panxdepyim','PFI20G_66','PFI19B_1','PFI19B_2','PFI19B_3','PFI19B_4','PFI19B_5','PFI19B_6','PFI19B
_7','PFI19B_8','PFI19B_9','PFI19B_66','PFI19B_77','PFI19B_99','PCH3A','PCH3B', 'PCH3C', 'PCH3D',
'PCH3E', 'PCH3F', 'PCH3H', 'PCH3I','PCH3J', 'PCH3K', 'PCH3L', 'PCH3M', 'PCH3N', 'PCH3O', 'PCH3P',
    'PCH3Z', 'PCH3_77',
'PCH3_99','PFI19B_10','pmdy','pmdm','PCH3G','PCMHDiagnosis','pDISCm','pDISCyim','health_GP',
'health_paediatrician',
    'health_psychiatrist', 'health_psychologist', 'health_nurse','health_socialworker', 'health_OT',
'health_counsfamtherapist',
    'health_otherprofessional', 'health_anyprofessional', 'loc_school','loc_private', 'loc_hospital',
'loc_CAMHS', 'loc_public',
    'loc_headspace', 'loc_community', 'loc_support', 'loc_other','health_anyservice', 'hospitalservice',
'services_any',
    'pneed_filter', 'needany', 'needanyinfo', 'needanymeds','needanycouns', 'needanyskills', 'cneed_filter',
'anyneed_info',
    'anyneed_couns', 'anyneed_relation', 'anyneed_probs','anyneed_parent', 'anyneed_respite',
'anyneed_support',
    'anyneed_help', 'anyneedb','MHDiagnosis','pDISCmim'], axis=1, inplace=True)
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

In[24]:

```
df4['depressed']=df4['depressed'].apply(lambda x: int(x))
```

In[26]:

```
df4['depressed'].value_counts()
```

In[30]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.model_selection import KFold
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_predict
from math import sqrt
cv = KFold(n_splits=10, random_state=None, shuffle=True)
classifier_pipeline = make_pipeline(StandardScaler(), KNeighborsRegressor(n_neighbors=10))
y=df4.depressed
vals = [.06,.07,.08,.09,0.1,0.2]
for val in vals:
  features = abs(df4.corr()["depressed"][abs(df4.corr()
["depressed"])>val].drop("depressed")).index.tolist()
  X = df4.drop(columns='depressed')
  X = X[features]

  print(features)

  y_pred = cross_val_predict(classifier_pipeline, X, y, cv=cv)
  print("RMSE:" + str(round(sqrt(mean_squared_error(y,y_pred)),2)))
  print("R_squared: " + str(round(r2_score(y,y_pred),2)))
```

In[34]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
Y = labelencoder.fit_transform(y)


# In[36]:


import numpy as np
feature_names = np.array(X.columns)


# In[40]:


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)


# In[41]:


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30, random_state=42)


# In[113]:


from sklearn.ensemble import RandomForestClassifier
class_weight = {0: 5839.,
                1: 471.}
model = RandomForestClassifier(n_estimators=10, class_weight=class_weight)


# In[114]:


from boruta import BorutaPy


# In[115]:


feat_selector = BorutaPy(model, n_estimators='auto', verbose=2, random_state=1)
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
# In[116]:


get_ipython().run_cell_magic('time', '', 'feat_selector.fit(X_train, y_train)')


# In[117]:


print(feat_selector.support_)


# In[118]:


X_filtered = feat_selector.transform(X_train)


# In[119]:


feature_ranks = list(zip(feature_names,
                         feat_selector.ranking_,
                         feat_selector.support_))


# In[120]:


get_ipython().run_cell_magic('time', '', "for feat in feature_ranks:\n    print('Feature: {:<30} Rank: {},  Keep:
{}'.format(feat[0], feat[1], feat[2]))")


# In[ ]:


from tpot import  TPOTClassifier
from __future__ import print_function
import sys,tempfile, urllib, os

tpot =  TPOTClassifier(generations=4, population_size=10, verbosity=3)
tpot.fit(X_train_over, y_train_over)
print(tpot.score(X_test, y_test))

%%time
import multiprocessing
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
if __name__ == '__main__':
    multiprocessing.set_start_method('forkserver', force=True)
    tpot =  TPOTClassifier(generations=2, population_size=20, verbosity=2,n_jobs = 20, random_state=50)
    tpot.fit(X_train_over, y_train_over)

print(tpot.score(X_test, y_test))

tpot =  TPOTClassifier(generations=4, population_size=10, verbosity=3)
tpot.fit(X_train_over, y_train_over)
print(tpot.score(X_test, y_test))
tpot.export('tpot_churn_pipeline.py')
get_ipython().system('cat tpot_churn_pipeline.py')
get_ipython().run_cell_magic('time', '', 'tpot.evaluated_individuals_')
tpot.fitted_pipeline_


# In[121]:


#import xgboost as xgb
#model = xgb.XGBClassifier()
#model = xgb.XGBClassifier(scale_pos_weight=4.527)
from sklearn.ensemble import RandomForestClassifier
#model = RandomForestClassifier(n_estimators=10, class_weight=class_weight)
model = RandomForestClassifier(n_estimators=10)
#from sklearn import tree
#model = tree.DecisionTreeClassifier()
#from sklearn.naive_bayes import GaussianNB
#model = GaussianNB()


# In[122]:


X_filtered = feat_selector.transform(X_train)


# In[123]:


model.fit(X_filtered, y_train)


# In[124]:


X_test_filtered = feat_selector.transform(X_test)
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
prediction = (model.predict_proba(X_test_filtered)[:,1] >= 0.79).astype(bool)


# In[125]:


len(X_test_filtered)


# In[127]:


from sklearn import metrics
print ("Accuracy = ", metrics.accuracy_score(y_test, prediction))


# In[128]:


len(prediction)


# In[129]:


from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, prediction)
print(cm)
#sns.heatmap(cm, annot=True)


# In[130]:


group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
            cm.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                cm.flatten()/np.sum(cm)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
        zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cm, annot=labels, fmt='', cmap='Blues')
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
# In[131]:


from sklearn.metrics import classification_report

print(classification_report(y_test, prediction))


# In[132]:


import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc, roc_auc_score

false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, model.predict(X_test_filtered))
print (auc(false_positive_rate, true_positive_rate))

print (roc_auc_score(y_test, model.predict_proba(X_test_filtered)[:,1]))


probs = model.predict(X_test_filtered)
preds = probs[:]
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(false_positive_rate, true_positive_rate)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()


# In[197]:


from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(model, X_train, y_train, cv=3 ) #cv=3,5,10
print("CV average score: %f" % cv_scores.mean())
```


Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Citation: Umme Marzia Haque Depression Detection with DM https://dx.doi.org/10.17504/protocols.io.bzm8p49w

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited