imd-pipeline-genetic-16march2024

March 16, 2024

The IMD Machine Learning Model: Credit to: `Isaac Okello Opio` The following were perfromed in this step: 1. Take a peek at the raw data. 2. Review the dimensions of my dataset. 3. Review the data types of attributes in the data.

[14]:
```
#Import libraries
from pycaret import*
import pandas as pd #this is for dataframe manipulation
import numpy as np #this is for numerical / mathematical computing
import matplotlib.pyplot as plt #this is for visualisation
from IPython.display import display
```

[19]:
```
#import data
IMD_data = pd.read_stata ("DATASET16MARCH2024.dta")
```

[20]:
```
#EDA
IMD_data.head(5)
```

[20]:

	PatientID	age	agecatak	AgeCategory	sex1	religion1	\
0	CHAKA/01/01/0001	11.668720	9-12	Adolescent	Female	Born Again	
1	CHAKA/01/01/0002	15.000000	13-17	Adolescent	Male	Protestant	
2	CHAKA/01/01/0003	6.020534	9-12	Adolescent	Female	Muslim	
3	CHAKA/01/02/0152	5.242984	5-8	Children	Female	Catholic	
4	CHAKA/01/02/0153	11.696098	9-12	Adolescent	Female	Muslim	

	childeduc1	heightst1	weightst1	height_m	…	stin2vntr_	httlpr1	\
0	Pre-primary	134.0	31.0	1.340	…	NaN	SS	
1	Pre-primary	137.0	35.0	1.370	…	12/12	SS	
2	Pre-primary	120.1	19.0	1.201	…	10/10	LL	
3	Pre-primary	99.0	15.0	0.990	…	12/12	LS	
4	Pre-primary	131.6	30.0	1.316	…	12/12	LL	

	HTTLPRrs35531	rs35531	Rs10482605	Rs1360780	rs1386494	rs1843809	rs34517220	\
0	NaN	NaN	A:A	C:T	A:A	G:G	A:A	
1	SA/SA	A:A	A:A	C:T	A:A	G:G	A:A	
2	LA/LG	A:G	G:A	C:T	G:A	T:G	A:A	
3	SA/LG	A:G	A:A	C:C	G:A	T:G	A:A	
4	LA/LA	A:A	A:A	C:C	G:G	T:G	G:A	

```
        _merge
0  matched (3)
1  matched (3)
2  matched (3)
3  matched (3)
4  matched (3)

[5 rows x 51 columns]
```

[21]:
```
# List variable names
variable_names = IMD_data.columns.tolist()

# Print the variable names
print(variable_names)
```

```
['PatientID', 'age', 'agecatak', 'AgeCategory', 'sex1', 'religion1',
'childeduc1', 'heightst1', 'weightst1', 'height_m', 'BMI', 'rounded_bmi',
'BMI_category', 'childtrib1', 'motherali1', 'fatherali1', 'orphanhood', 'ses',
'ses_cat', 'livelihood1', 'sexualever1', 'childstay1', 'childartk1',
'childworst1', 'childpremt1', 'chilborhiv', 'tobacco_status', 'alcohol1',
'Stress', 'GroupCategory', 'cd4takeoff1', 'CD4_category', 'Viralload',
'Viralload_Category', 'tlbase', 'ptsd', 'gad', 'mdd', 'pc', 'intdis', 'IMDs',
'stin2vntr_', 'httlpr1', 'HTTLPRrs35531', 'rs35531', 'Rs10482605', 'Rs1360780',
'rs1386494', 'rs1843809', 'rs34517220', '_merge']
```

0.0.1 FOR JOINT ANALYSIS (ALL THREE CATEGORIES)

[22]:
```
#Dropping Non-Important and Redundant Variables
# List of variables to drop

columns_to_drop = ['PatientID', 'age', 'height_m', 'BMI', 'rounded_bmi',␣
 ↪'motherali1', 'fatherali1', 'ses', 'cd4takeoff1', 'Viralload', 'ptsd',␣
 ↪'gad', 'mdd', 'pc', 'intdis', '_merge' ]

# Dropping the specified columns
IMD_data = IMD_data.drop(columns=columns_to_drop)
```

Note: Removing MDD, PTSD, and GAD as they are used to get IMDs

Dimensions of the Data I had to check how much data I have, both in terms of rows and columns. - Too many rows and algorithms may take too long to train. Too few and perhaps I do not have enough data to train the algorithms. - Too many features and some algorithms can be distracted or super poor performance due to the curse of dimensionality.

[23]:
```
print("Number of rows:", IMD_data.shape[0])
print("Number of columns:", IMD_data.shape[1])
#OR
```

```
print("The dimension is:", IMD_data.shape)
```

```
Number of rows: 736
Number of columns: 35
The dimension is: (736, 35)
```

Data Type For Each Attribute The type of each attribute is important. Strings may need to be converted to foating point values or integers to represent categorical or ordinal values.

[24]:
```
IMD_data.dtypes
```

[24]:
```
agecatak               object
AgeCategory            object
sex1                 category
religion1            category
childeduc1           category
heightst1             float64
weightst1             float64
BMI_category         category
childtrib1           category
orphanhood           category
ses_cat              category
livelihood1          category
sexualever1          category
childstay1           category
childartk1           category
childworst1          category
childpremt1          category
chilborhiv           category
tobacco_status         object
alcohol1             category
Stress               category
GroupCategory        category
CD4_category         category
Viralload_Category   category
tlbase                float64
IMDs                 category
stin2vntr_           category
httlpr1              category
HTTLPRrs35531        category
rs35531              category
Rs10482605             object
Rs1360780              object
rs1386494              object
rs1843809              object
rs34517220             object
dtype: object
```

Descriptive Statistics Descriptive statistics can give a great insight into the shape of each attribute.

[25]:
```
IMD_data.describe().T
```

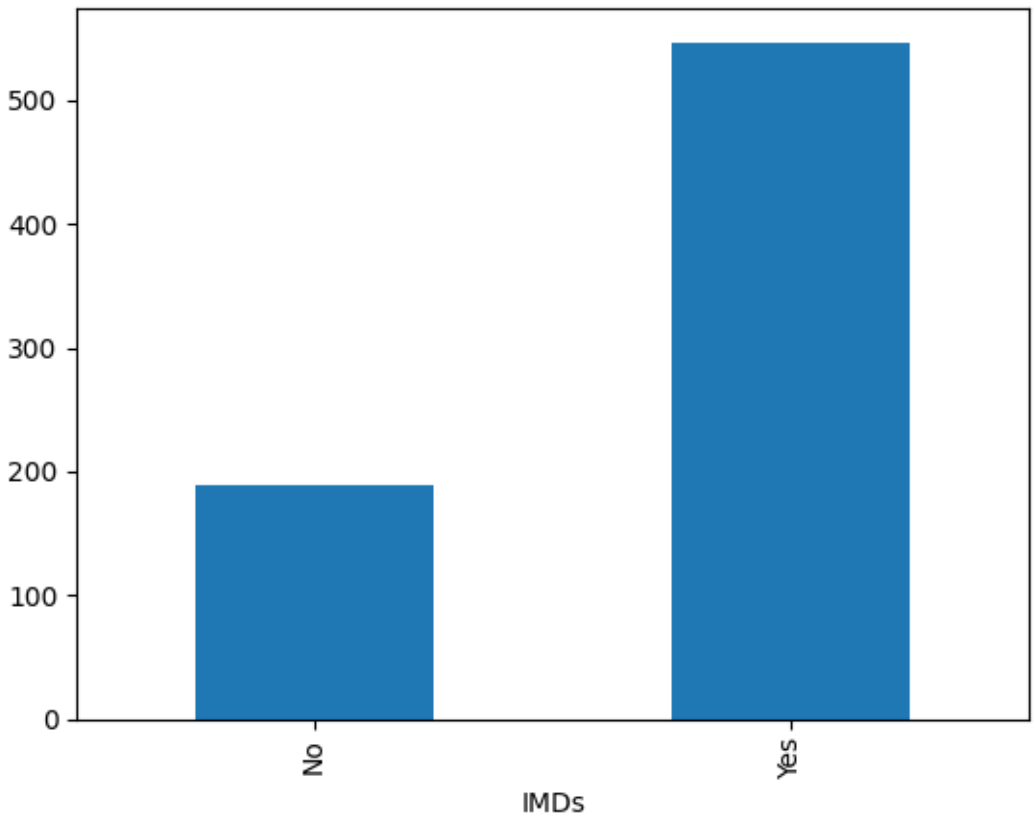
[25]:
```
           count        mean        std        min        25%         50%  \
heightst1  733.0  131.050887  22.690718  12.500000  119.00000  130.600000
weightst1  736.0   29.878111  11.915348  10.000000   21.00000   27.000000
tlbase     613.0    1.147457   0.362307   0.019943    0.91413    1.133582

                75%         max
heightst1  144.000000  427.200000
weightst1   36.000000  153.000000
tlbase       1.358633    2.179359
```

Dependent Variable (DV) Note that: The DV in this cases was being diagnosed with Internalizing Mental Disorder, IMD (Yes) or Not (No), This was calculated by ensuring all participants without any from of GAD, MDD, or PTSD, was assigned as "Not having IMD", while all those with any form of GAD, MDD, or PTSD, was amrked as having IMD.

[26]:
```
##Plotting the distribution of the dependent variable (DV).
IMD_data.groupby('IMDs').size().plot(kind='bar')
```

[26]:
```
<Axes: xlabel='IMDs'>
```



[28]:
```
#Checking at the distribution of the data
#IMD_data.hist()
```

NOTE: The above was not useful as the package (pycaret) cound perfrom transformations in all datasets. Unlike in scikit-learn that I would have to decide the data to transform based on the above plot.

MODEL BUILDING USING PYCARET Since this was a classification problem (Supervised machine learning),the classification package from pycaret was used. Thisn package conatins several algorithms:

- Logistic Regression
- Support Vector Machines
- Decision Trees
- Naive Bayes
- K-Nearest Neighbors
- Random Forests

[29]:
```
#Import classification model from pycaret
from pycaret.classification import *
```

[30]:
```
# Setting up the data for machine learning modeling using the pycaret setup function
# IMD_data: Your dataset or data structure
# target='IMDs': Specifying 'IMDs' as the target variable to predict
# session_id=123: Setting a session ID (seed) for reproducibility

s = setup(IMD_data, target='IMDs', session_id=123)
```

```
<pandas.io.formats.style.Styler at 0x27627fab650>
```

[31]:
```
#Model training
# Comparing and selecting the best-performing model based on default evaluation metric
best = s.compare_models()

# Retrieve the metrics dataframe for all compared models
metrics_df = pull()
print(metrics_df)
```

```
Initiated  . . . . . .                13:20:34
Status     . . . . . .  Selecting Estimator
Estimator  . . . . . .  Logistic Regression
```

```
<pandas.io.formats.style.Styler at 0x276289ddfd0>
```

```
<IPython.core.display.HTML object>
```

	Model	Accuracy	AUC	Recall	Prec.	F1	\
dummy	Dummy Classifier	0.7437	0.0	0.7437	0.5531	0.6344	
rf	Random Forest Classifier	0.7360	0.0	0.7360	0.6513	0.6599	
gbc	Gradient Boosting Classifier	0.7320	0.0	0.7320	0.6887	0.6975	
et	Extra Trees Classifier	0.7242	0.0	0.7242	0.6292	0.6571	
lr	Logistic Regression	0.7183	0.0	0.7183	0.6717	0.6818	
ada	Ada Boost Classifier	0.7165	0.0	0.7165	0.6771	0.6856	
knn	K Neighbors Classifier	0.7164	0.0	0.7164	0.6616	0.6710	
ridge	Ridge Classifier	0.7067	0.0	0.7067	0.6510	0.6616	
lda	Linear Discriminant Analysis	0.6912	0.0	0.6912	0.6562	0.6644	
dt	Decision Tree Classifier	0.6701	0.0	0.6701	0.6851	0.6748	
svm	SVM - Linear Kernel	0.6582	0.0	0.6582	0.5812	0.5524	
nb	Naive Bayes	0.3202	0.0	0.3202	0.6416	0.2615	
qda	Quadratic Discriminant Analysis	0.3031	0.0	0.3031	0.6427	0.2132	

	Kappa	MCC	TT (Sec)
dummy	0.0000	0.0000	0.246
rf	0.0520	0.0678	0.382
gbc	0.1621	0.1727	0.380
et	0.0470	0.0489	0.398

```
lr      0.1213  0.1296     1.155
ada     0.1332  0.1424     0.332
knn     0.0860  0.0970     0.410
ridge   0.0606  0.0710     0.363
lda     0.0826  0.0893     0.284
dt      0.1683  0.1708     0.367
svm     0.0047  0.0130     0.462
nb      0.0036  0.0154     0.331
qda     0.0147  0.0547     0.285
```

[32]:
```
#Print the best model
print(best)
```

```
DummyClassifier(constant=None, random_state=123, strategy='prior')
```

[33]:
```
s.evaluate_model(best)
```

```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',),␣
 ↪options=(('Pipeline Plot', 'pipelin…
```

[36]:
```
# Save specific plots using plot_model with save=True

# List of plot types to include
#plot_types = ['auc', 'confusion_matrix', 'feature', 'learning']

# Iterate over plot types and save each one
#for plot_type in plot_types:
    #plot_model(best, plot=plot_type, save=True, verbose=False)
```

[38]:
```
#DISPLAY THE PLOTS
#Import the package
from IPython.display import Image, display

# Display the PNG images
# display(Image('Calibration Curve.png'))
# display(Image("AUC.png"))
# display(Image("Confusion Matrix.png"))
# display(Image("Feature Importance.png"))
# display(Image("Learning Curve.png"))
```

The Predict Hold DATA (Testing/assess perfromace) The predict_holdout is a portion of the dataset that is intentionally set aside and not used during the training of the model. This set is reserved for evaluating the model's performance on unseen data.

[39]:
```
#predict/testing
predict_holdout =s.predict_model(best) #Making predictions on the holdout set␣
 ↪using the best model (predict_model)
```

```
#Retrieve the metrics dataframe
metrics_df = pull() #Calculating various classification metrics based on the␣
 ↪predictions made
print(metrics_df) #Printing the metrics dataframe to the console
```

```
<pandas.io.formats.style.Styler at 0x2762aced690>

              Model  Accuracy  AUC  Recall   Prec.      F1  Kappa  MCC
0  Dummy Classifier    0.7421  0.5  0.7421  0.5507  0.6322    0.0  0.0
```

[40]:
```
# Predicting on new data

# Creating a copy of the original data without the 'IMDs' column
New_IMD_data = IMD_data.copy().drop('IMDs', axis=1)

# Making predictions on the new data using the best-performing model
New_Predictions = s.predict_model(best, New_IMD_data)

# Displaying the first 5 rows of the predictions
New_Predictions.head(5)
```

```
<IPython.core.display.HTML object>
```

[40]:

	agecatak	AgeCategory	sex1	religion1	childeduc1	heightst1	\
0	9-12	Adolescent	Female	Born Again	Pre-primary	134.000000	
1	13-17	Adolescent	Male	Protestant	Pre-primary	137.000000	
2	9-12	Adolescent	Female	Muslim	Pre-primary	120.099998	
3	5-8	Children	Female	Catholic	Pre-primary	99.000000	
4	9-12	Adolescent	Female	Muslim	Pre-primary	131.600006	

	weightst1	BMI_category	childtrib1	orphanhood	\
0	31.0	Underweight	Non-Munganda but Ugandan	Single parent alive	
1	35.0	Normal weight	Non-Munganda but Ugandan	Single parent alive	
2	19.0	Underweight	Muganda	Both parents alive	
3	15.0	Underweight	Non-Munganda but Ugandan	NaN	
4	30.0	Underweight	Muganda	Both parents alive	

	…	httlpr1	HTTLPRrs35531	rs35531	Rs10482605	Rs1360780	rs1386494	rs1843809	\
0	…	SS	NaN	NaN	A:A	C:T	A:A	G:G	
1	…	SS	SA/SA	A:A	A:A	C:T	A:A	G:G	
2	…	LL	LA/LG	A:G	G:A	C:T	G:A	T:G	
3	…	LS	SA/LG	A:G	A:A	C:C	G:A	T:G	
4	…	LL	LA/LA	A:A	A:A	C:C	G:G	T:G	

	rs34517220	prediction_label	prediction_score
0	A:A	Yes	0.7437
1	A:A	Yes	0.7437

```
2        A:A             Yes          0.7437
3        A:A             Yes          0.7437
4        G:A             Yes          0.7437

[5 rows x 36 columns]
```