imd-pipeline-08march2024

March 9, 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.

[202]:
```
#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
```

[203]:
```
#import data
IMD_data = pd.read_excel ("DATASET08MARCH2024.xlsx")
```

[204]:
```
#EDA
IMD_data.head(5)
```

[204]:
```
               patid        age  agecatak AgeCategory    sex1   religion1  \
0  CHAKA/01/01/0001  11.668720   9 to 12  Adolescent  Female  Born Again
1  CHAKA/01/01/0002  15.000000  13 to 17  Adolescent    Male  Protestant
2  CHAKA/01/01/0003   6.020534   9 to 12  Adolescent  Female      Muslim
3  CHAKA/01/02/0152   5.242984    5 to 8    Children  Female    Catholic
4  CHAKA/01/02/0153  11.696100   9 to 12  Adolescent  Female      Muslim

    childeduc1  heightst1  weightst1  height_m  …  stin2vntr_  httlpr1  \
0  Pre-primary      134.0       31.0     1.340  …         NaN      3.0
1  Pre-primary      137.0       35.0     1.370  …         3.0      3.0
2  Pre-primary      120.1       19.0     1.201  …         1.0      1.0
3  Pre-primary       99.0       15.0     0.990  …         3.0      2.0
4  Pre-primary      131.6       30.0     1.316  …         3.0      1.0

  HTTLPRrs35531 rs35531 ptsd  gad  mdd  pc intdis IMDs
0           NaN     NaN  0.0  0.0  1.0   1      1  Yes
1           7.0     1.0  0.0  4.0  6.0   1     10  Yes
2           2.0     2.0  0.0  0.0  3.0   0      3  Yes
3           6.0     2.0  0.0  0.0  0.0   0      0   No
4           1.0     1.0  1.0  3.0  3.0   0      7  Yes
```

```
[5 rows x 45 columns]
```

[205]:

```
# List variable names
variable_names = IMD_data.columns.tolist()

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

```
['patid', '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', 'stin2vntr_', 'httlpr1', 'HTTLPRrs35531',
'rs35531', 'ptsd', 'gad', 'mdd', 'pc', 'intdis', 'IMDs']
```

[206]:

```
#Dropping Non-Important and Redundant Variables
# List of variables to drop
#Give DummyClass
#columns_to_drop = ['patid', 'height_m','motherali1', 'fatherali1',␣
 ↪'rounded_bmi', 'BMI','ses', 'ptsd', 'gad', 'mdd', 'pc', 'intdis']
#Gives best score, with AdaBoost
#columns_to_drop = ['patid', 'height_m','motherali1', 'fatherali1',␣
 ↪'rounded_bmi', 'BMI','ses', 'pc', 'intdis']
#Removing
columns_to_drop = ['patid', 'height_m','motherali1', 'fatherali1',␣
 ↪'rounded_bmi', 'BMI','ses', 'pc', 'intdis']

#columns_to_drop = ['patid', 'height_m','motherali1', 'fatherali1',␣
 ↪'rounded_bmi', 'BMI','ses', 'pc', 'intdis']

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

Note: Removing MDD, PTSD, and GAD from the data drops performace from 90s% to 70s %

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.

[207]:

```
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: 36
The dimension is: (736, 36)
```

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.

[208]:
```
IMD_data.dtypes
```

[208]:
```
age                    float64
agecatak                object
AgeCategory             object
sex1                    object
religion1               object
childeduc1              object
heightst1              float64
weightst1              float64
BMI_category            object
childtrib1              object
orphanhood              object
ses_cat                 object
livelihood1             object
sexualever1             object
childstay1              object
childartk1              object
childworst1             object
childpremt1             object
chilborhiv              object
tobacco_status          object
alcohol1                object
Stress                  object
GroupCategory           object
cd4takeoff1            float64
CD4_category            object
Viralload              float64
Viralload_Category      object
tlbase                 float64
stin2vntr_             float64
httlpr1                float64
HTTLPRrs35531          float64
rs35531                float64
ptsd                   float64
gad                    float64
mdd                    float64
IMDs                    object
dtype: object
```

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

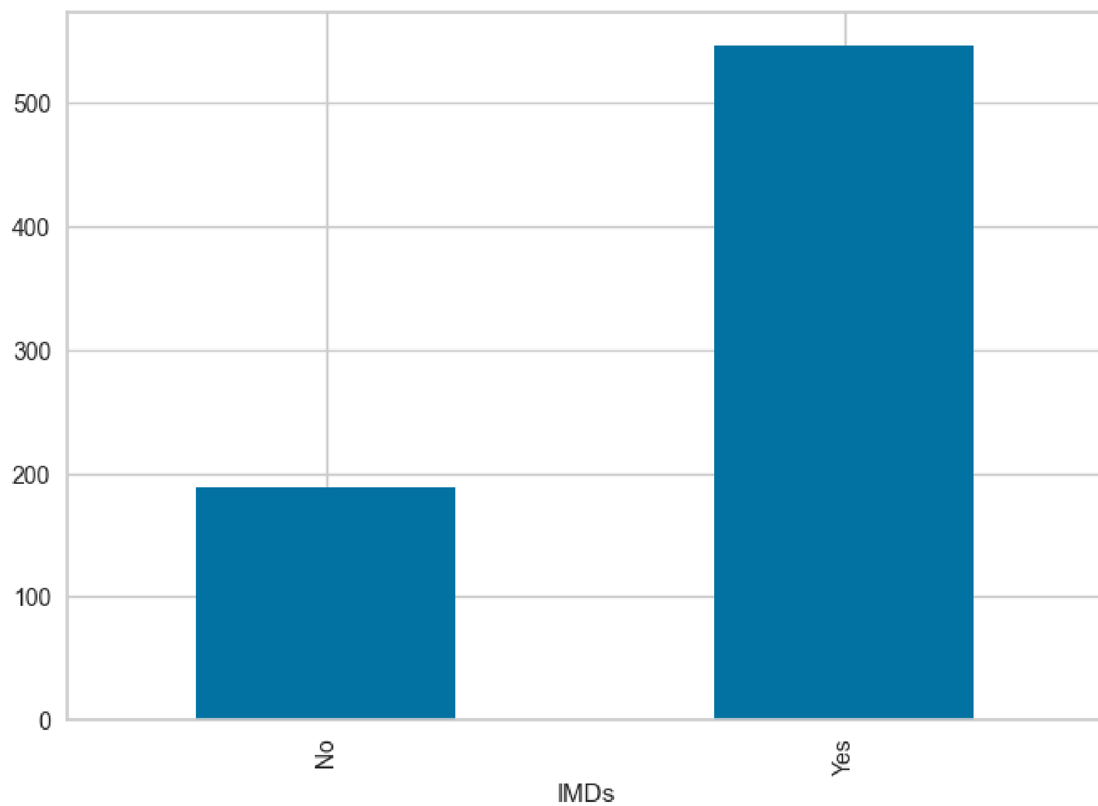
[209]:
```
IMD_data.describe().T
```

[209]:
```
               count          mean            std        min         25%  \
age            696.0     10.574136       3.141758   4.525667    7.874744
heightst1      733.0    131.050887      22.690718  12.500000  119.000000
weightst1      736.0     29.878111      11.915348  10.000000   21.000000
cd4takeoff1    732.0    945.527322     533.852160  11.000000  568.750000
Viralload      611.0  30247.690671  155631.745971   0.000000    0.000000
tlbase         613.0      1.147457       0.362307   0.019943    0.914130
stin2vntr_     658.0      2.474164       0.643503   1.000000    2.000000
httlpr1        698.0      1.399713       0.583710   1.000000    1.000000
HTTLPRrs35531  685.0      2.486131       1.765954   1.000000    1.000000
rs35531        685.0      1.379562       0.550537   1.000000    1.000000
ptsd           728.0      0.453297       0.873271   0.000000    0.000000
gad            722.0      2.308864       2.668490   0.000000    0.000000
mdd            722.0      1.537396       2.361119   0.000000    0.000000

                      50%          75%           max
age             10.368240    13.000000  1.711431e+01
heightst1      130.600000   144.000000  4.272000e+02
weightst1       27.000000    36.000000  1.530000e+02
cd4takeoff1    888.000000  1224.250000  3.924000e+03
Viralload       10.000000   801.500000  2.621863e+06
tlbase           1.133582     1.358633  2.179359e+00
stin2vntr_       3.000000     3.000000  3.000000e+00
httlpr1          1.000000     2.000000  3.000000e+00
HTTLPRrs35531    2.000000     3.000000  7.000000e+00
rs35531          1.000000     2.000000  3.000000e+00
ptsd             0.000000     1.000000  5.000000e+00
gad              2.000000     4.000000  1.800000e+01
mdd              0.000000     2.000000  1.500000e+01
```

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.

[210]:
```
##Plotting the distribution of the dependent variable (DV).
IMD_data.groupby('IMDs').size().plot(kind='bar')
```

[210]: <Axes: xlabel='IMDs'>



[211]:
```
#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

[212]:
```
#Import classification model from pycaret
from pycaret.classification import *
```

[213]:
```python
# 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 0x284892b2e10>
```

[214]:
```python
#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)
```

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

<pandas.io.formats.style.Styler at 0x28486640f90>

Processing:   0%|          | 0/61 [00:00<?, ?it/s]

                                    Model  Accuracy  AUC  Recall   Prec.  \
ada                 Ada Boost Classifier    0.9980  0.0  0.9980  0.9981
lightgbm  Light Gradient Boosting Machine    0.9980  0.0  0.9980  0.9981
gbc             Gradient Boosting Classifier    0.9922  0.0  0.9922  0.9924
qda       Quadratic Discriminant Analysis    0.9864  0.0  0.9864  0.9869
dt                  Decision Tree Classifier    0.9845  0.0  0.9845  0.9850
rf                  Random Forest Classifier    0.9805  0.0  0.9805  0.9814
lr                       Logistic Regression    0.9532  0.0  0.9532  0.9586
et                    Extra Trees Classifier    0.8639  0.0  0.8639  0.8674
ridge                       Ridge Classifier    0.8037  0.0  0.8037  0.7979
lda          Linear Discriminant Analysis    0.8036  0.0  0.8036  0.8065
dummy                       Dummy Classifier    0.7437  0.0  0.7437  0.5531
nb                               Naive Bayes    0.7340  0.0  0.7340  0.6804
knn                   K Neighbors Classifier    0.6970  0.0  0.6970  0.6468
svm                      SVM - Linear Kernel    0.6312  0.0  0.6312  0.6213

              F1   Kappa     MCC  TT (Sec)
ada       0.9980  0.9947  0.9948     0.271
lightgbm  0.9980  0.9947  0.9948     0.405
gbc       0.9921  0.9791  0.9796     0.333
qda       0.9862  0.9630  0.9642     0.211
dt        0.9845  0.9593  0.9600     0.188
rf        0.9801  0.9468  0.9489     0.312
lr        0.9542  0.8830  0.8870     0.304
```

```
et         0.8510  0.5893  0.6190     0.294
ridge      0.7936  0.4422  0.4557     0.186
lda        0.8013  0.4773  0.4846     0.183
dummy      0.6344  0.0000  0.0000     0.190
nb         0.6795  0.1087  0.1321     0.186
knn        0.6577  0.0498  0.0601     0.200
svm        0.5934  0.0232  0.0331     0.182
```

[215]:
```
#Print the best model
print(best)
```

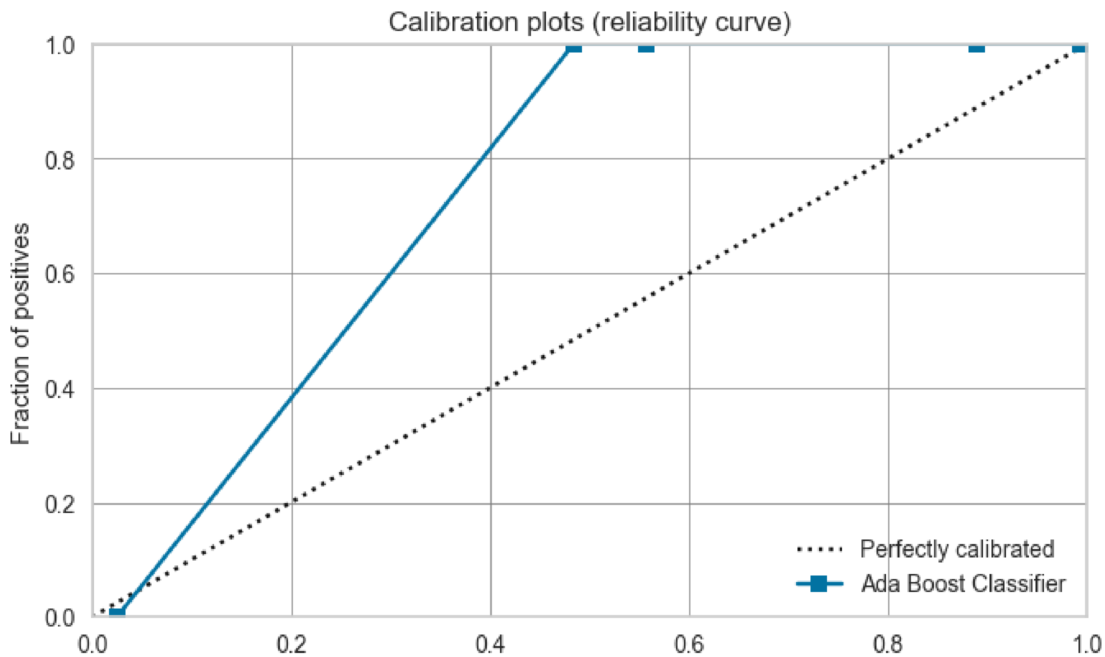
```
AdaBoostClassifier(algorithm='SAMME.R', estimator=None, learning_rate=1.0,
                   n_estimators=50, random_state=123)
```

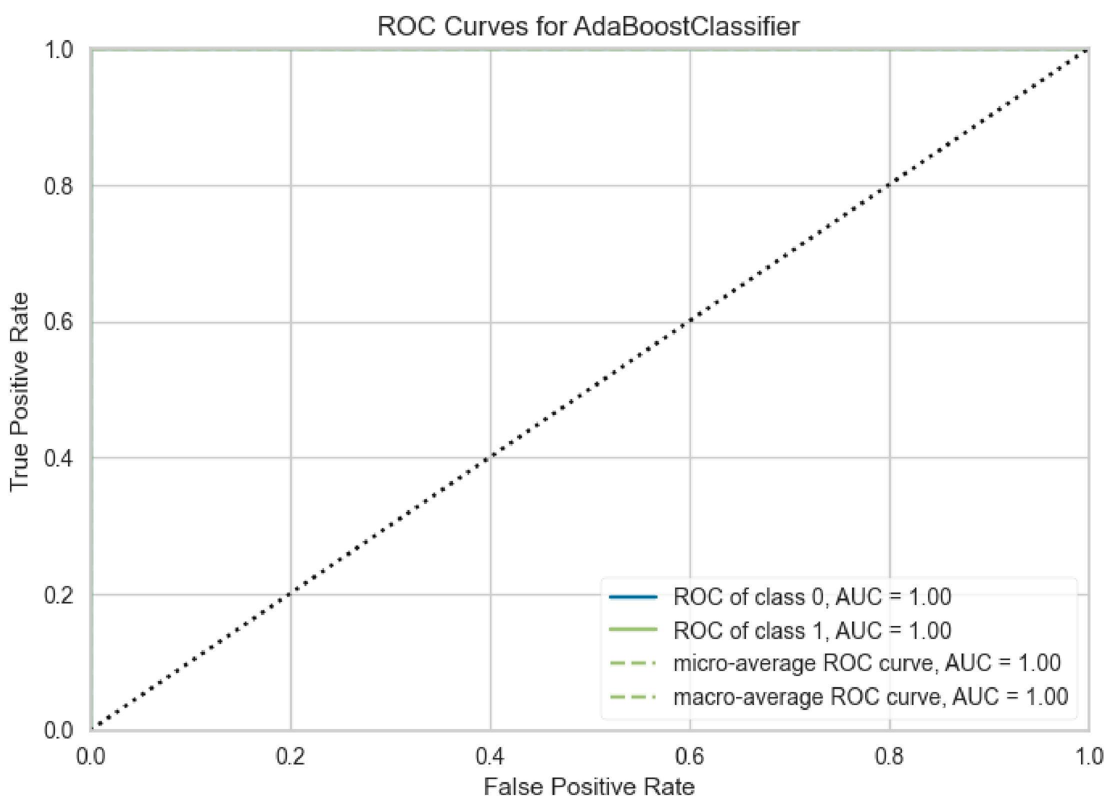
[216]:
```
s.evaluate_model(best)
```

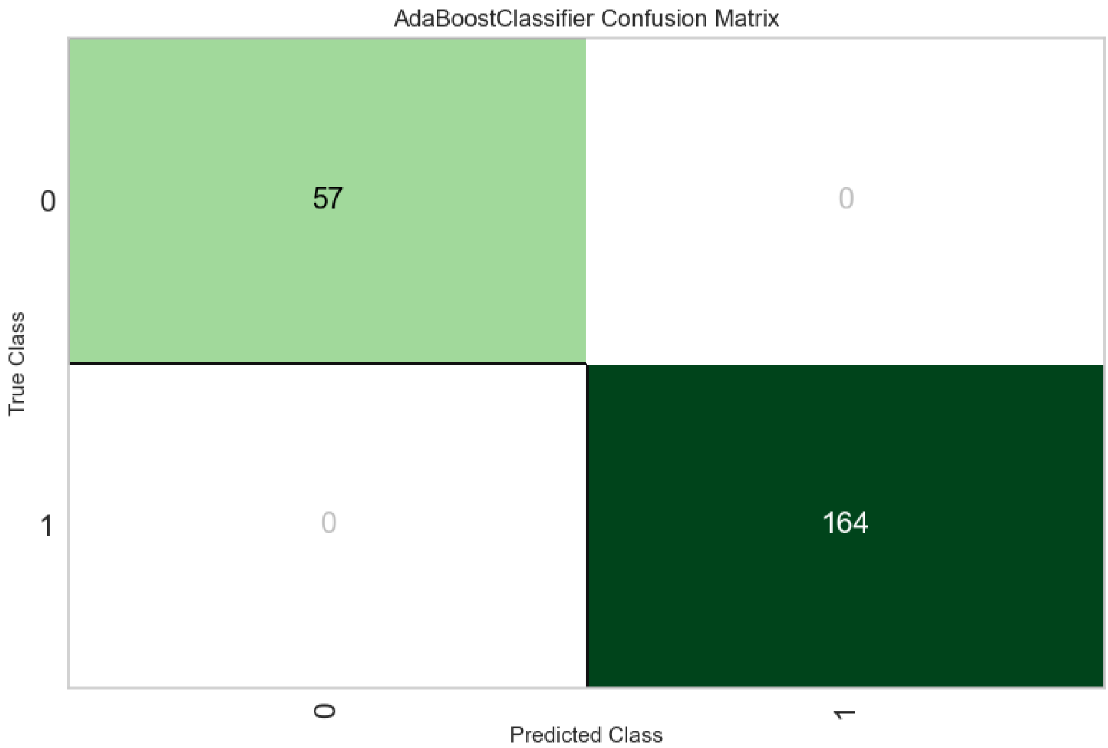
```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',),␣
 ↪options=(('Pipeline Plot', 'pipelin…
```

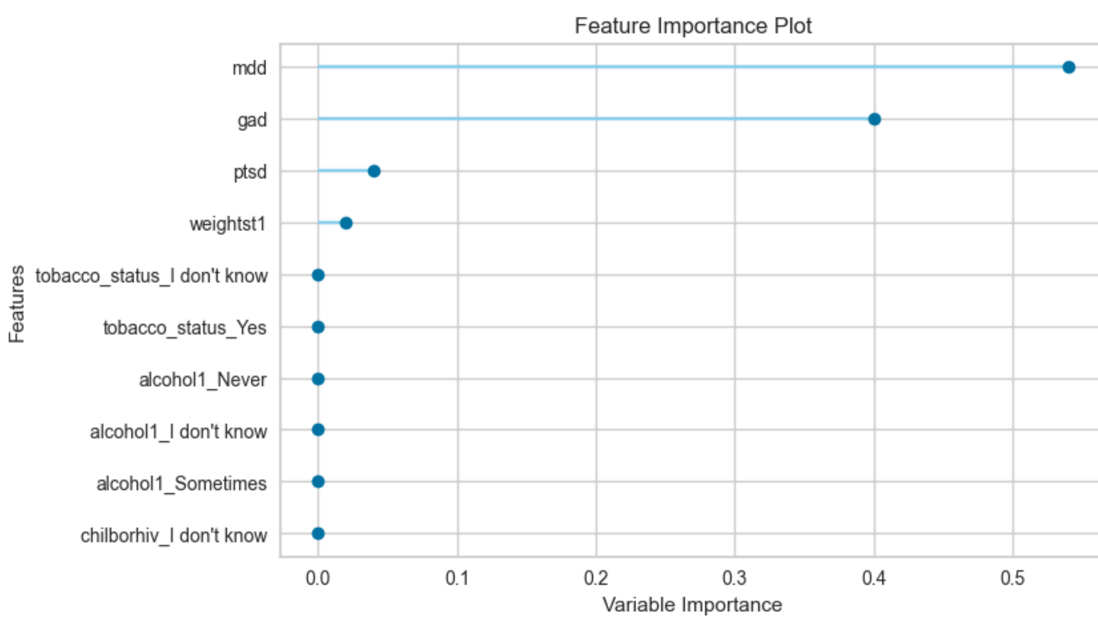
[217]:
```
# Save specific plots using plot_model with save=True

# 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)
```

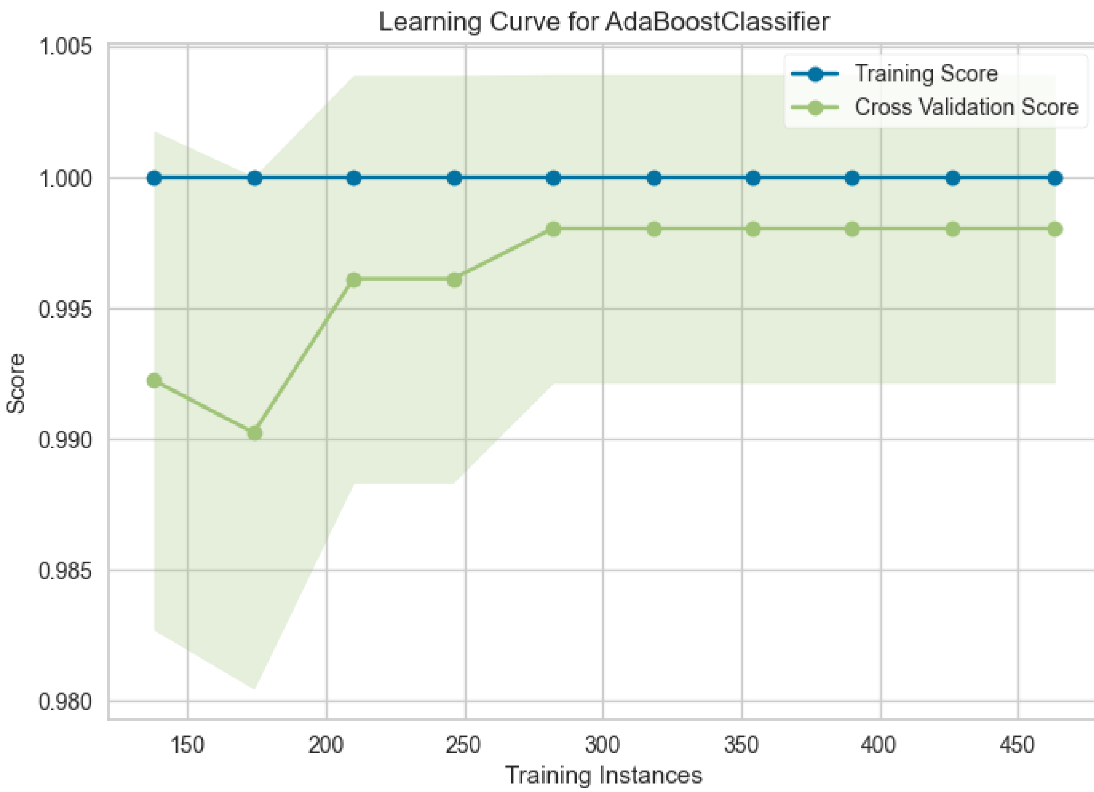
[218]:
```
#DISPLAY THE PLOTS
#Import the package
from IPython.display import Image, display

# 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"))
```


Calibration plots (reliability curve)
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated
Ada Boost Classifier


ROC Curves for AdaBoostClassifier
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
ROC of class 0, AUC = 1.00
ROC of class 1, AUC = 1.00
micro-average ROC curve, AUC = 1.00
macro-average ROC curve, AUC = 1.00


AdaBoostClassifier Confusion Matrix
57
0
0
164
True Class
0
1
0
1
Predicted Class


Feature Importance Plot
mdd
gad
ptsd
weightst1
tobacco_status_I don't know
tobacco_status_Yes
alcohol1_Never
alcohol1_I don't know
alcohol1_Sometimes
chilborhiv_I don't know
Features
0.0
0.1
0.2
0.3
0.4
0.5
Variable Importance



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.

[219]:
```
#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
```

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

                     Model  Accuracy  AUC  Recall  Prec.   F1  Kappa  MCC
0  Ada Boost Classifier       1.0  1.0     1.0    1.0  1.0    1.0  1.0
```

[220]:
```
# 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)
```

```python
# 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>

[220]:

```
         age  agecatak AgeCategory    sex1   religion1   childeduc1  \
0  11.668720   9 to 12  Adolescent  Female  Born Again  Pre-primary
1  15.000000  13 to 17  Adolescent    Male  Protestant  Pre-primary
2   6.020534   9 to 12  Adolescent  Female      Muslim  Pre-primary
3   5.242984    5 to 8    Children  Female    Catholic  Pre-primary
4  11.696100   9 to 12  Adolescent  Female      Muslim  Pre-primary

    heightst1  weightst1   BMI_category                 childtrib1  …  \
0  134.000000       31.0    Underweight  Non-Munganda but Ugandan  …
1  137.000000       35.0  Normal weight  Non-Munganda but Ugandan  …
2  120.099998       19.0    Underweight                    Muganda  …
3   99.000000       15.0    Underweight  Non-Munganda but Ugandan  …
4  131.600006       30.0    Underweight                    Muganda  …

     tlbase stin2vntr_ httlpr1 HTTLPRrs35531 rs35531 ptsd  gad  mdd  \
0  1.085469        NaN     3.0           NaN     NaN  0.0  0.0  1.0
1  1.206384        3.0     3.0           7.0     1.0  0.0  4.0  6.0
2  1.251031        1.0     1.0           2.0     2.0  0.0  0.0  3.0
3  1.682923        3.0     2.0           6.0     2.0  0.0  0.0  0.0
4  1.131768        3.0     1.0           1.0     1.0  1.0  3.0  3.0

  prediction_label prediction_score
0              Yes           0.6422
1              Yes           0.9123
2              Yes           0.7084
3               No           0.6625
4              Yes           0.9782

[5 rows x 37 columns]
```