GENERALIZED ANXIETY DISORDER

Table of Contents

Introduction

Public health plays an important role in promoting people's well-being, maintaining safety and protection from infectious diseases and environmental risks, and ensuring people's safety and quality of treatment. The Internet is transforming the economy, education, government, medical care, and even the way we communicate with loved ones daily, making it one of the major drivers of social change. People started using the incredible internet and spent a lot of time playing online games on it for the development of the internet. People's mental and mental health is compromised by online games, causing a variety of mental illnesses. This dataset contains information collected from gamers' global surveys. In the survey, psychologists usually asked questions to ask people with anxiety, social phobia, and little or no life satisfaction. The questionnaire consists of a series of questions asked during a psychological test.

Use different features of the model when analyzing the impact of online games on generalized anxiety disorder, including GAD, games, playstyles, platforms to play, age, gender, working hours, place of work, and place of residence. This project is demonstrated by developing a GUI-based application that represents end-to-end modeling using three machine learning algorithms: a random forest classifier, a decision tree, and a support vector machine.

Dataset cleanup, exploratory data analysis, preprocessing, modeling, model comparison, GUI creation, Power Point presentation preparation, group report creation, and GUI demo generation were part of the collaboration.

Description of a Data Set

The dataset used is retrieved from Kaggle which has educational and professional records of various people who has completed their training in a company. The dataset has 14250 observations and 55 features, most features are categorical (Nominal, Ordinal, Binary) and some with high cardinality. The dataset is imbalanced, and many features has missing values.

Features are as described:

Data columns (total 55 columns):

# Column	Non-Null Count Dtype
--- ------	-------------- -----
0 S. No.	13464 non-null int64
1 Timestamp	13464 non-null float64
2 GAD1	13464 non-null int64
3 GAD2	13464 non-null int64
4 GAD3	13464 non-null int64
5 GAD4	13464 non-null int64
6 GAD5	13464 non-null int64
7 GAD6	13464 non-null int64
8 GAD7	13464 non-null int64
9 GADE	12815 non-null object
10 SWL1	13464 non-null int64
11 SWL2	13464 non-null int64
12 SWL3	13464 non-null int64
13 SWL4	13464 non-null int64
14 SWL5	13464 non-null int64
15 Game	13464 non-null object
16 Platform	13464 non-null object
17 Hours	13434 non-null float64
18 earnings	13464 non-null object
19 whyplay	13464 non-null object
20 League	11626 non-null object
21 highestleague	0 non-null float64
22 streams	13364 non-null float64
23 SPIN1	13340 non-null float64
24 SPIN2	13310 non-null float64
25 SPIN3	13324 non-null float64
26 SPIN4	13305 non-null float64
27 SPIN5	13298 non-null float64

28 SPIN6	13308 non-null float64
29 SPIN7	13326 non-null float64
30 SPIN8	13320 non-null float64
31 SPIN9	13306 non-null float64
32 SPIN10	13304 non-null float64
33 SPIN11	13277 non-null float64
34 SPIN12	13296 non-null float64
35 SPIN13	13277 non-null float64
36 SPIN14	13308 non-null float64
37 SPIN15	13317 non-null float64
38 SPIN16	13317 non-null float64
39 SPIN17	13289 non-null float64
40 Narcissism	13441 non-null float64
41 Gender	13464 non-null object
42 Age	13464 non-null int64
43 Work	13426 non-null object
44 Degree	13464 non-null object
45 Birthplace	13464 non-null object
46 Residence	13464 non-null object
47 Reference	13449 non-null object
48 Playstyle	13464 non-null object
49 accept	13050 non-null object
50 GAD_T	13464 non-null int64
51 SWL_T	13464 non-null int64
52 SPIN_T	12814 non-null float64
53 Residence_ISO3	13354 non-null object
54 Birthplace_ISO3	13343 non-null object

Background

- Age and Gender
- Country of origin
- Country of residence
- Employment status
- Highest Degree earned

Gaming habits

- Main game played (+ ranking, if applicable)
- Hours played per week
- Platform
- Motivation (fun, improvement, competition,)
- Sociality (singleplayer, multiplayer, etc...)

Validated Scales

- Social Phobia Inventory (SPIN)
- Generalized Anxiety Disorder Screener (GAD-7)
- Satisfaction with Life Scale (SWL)
- Single Item Narcissism Scale (SINS)

DESCRIPTION OF DATA MINING ALGORITHMS

Decision Tree Classifier:

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed based on features of the given dataset.
- **Entropy:** It is defined as a measure of impurity present in the data. The entropy is almost zero when the sample attains homogeneity but is one when it is equally divided. Entropy with the lowest value makes a model better in terms of prediction as it segregates the classes better.

Random Forest Classifier:

- The Random Forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then. It collects the votes from different decision trees to decide the final prediction
- Moreover, a random forest technique has a capability to focus both on observations and variables of a training data for developing individual decision trees and take maximum voting for classification and the total average for regression problem respectively. It also uses a bagging technique that takes observations in a random manner and selects all columns which are incapable of representing significant variables at the root for all decision trees. In this manner, a random forest makes trees only which are dependent on each other by penalizing accuracy. We have a thumb rule which can be implemented for selecting sub-samples from observations using random forest. If we consider 2/3 of observations for training data and p be the number of columns, then
  - For classification, we take sqrt(p) number of columns
  - For regression, we take p/3 number of columns.
- The above thumb rule can be tuned in case of increasing the accuracy of the model **Random Forest pseudocode:**
  - Randomly select “k” features from total “m” features. a. Where k << m
  - Among the “k” features, calculate the node “d” using the best split point.
  - Split the node into daughter nodes using the best split.
  - Repeat 1 to 3 steps until “l” number of nodes has been reached.
- Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

- The Scikit Learn package of python has been used for the model development of random forest algorithm. The Sklearn package has the library named ensemble which incorporates random forest algorithm to be used. After the data being fetched from the dataset manually from the source, the imbalanced data is being label encoded using Label encoder function. Label encoding has been used as the columns were in string format and for the better understanding and for encoding it to numeric form.

Support Vector Machine:

- SVMs (support vector machines) are a class of supervised learning algorithms for classification and regression. It is, however, mostly employed to solve categorization difficulties. The decision function of an SVM is based on a collection of training data. It also enables a 'one-versus-one' technique for multi-class classification, in which the receiver operator characteristic for each class is calculated separately.
- The SVM based classifier is called the SVC (Support Vector Classifier) and which can be used it in classification problems.

Kernel Functions:

- Kernel functions can also be regarded as the tuning parameters in an SVM model. They are responsible for removing the computational requirement to achieve the higher dimensional vector space and deal with the non-linear separable data
- The kernel has been set to **linear** for our data by default.

PROJECT SETUP

Data Cleaning and Pre-processing:

The Generalized Anxiety Disorder dataset consists of null values as mentioned below:

S. No. 0	
Timestamp	0
GAD1	0
GAD2	0
GAD3	0
GAD4 0	
GAD5	0
GAD6	0
GAD7	0
GADE 649	
SWL1	0
SWL2	0
SWL3	0
SWL4	0
SWL5	0
Game	0
Platform	0
Hours	30
earnings	0
whyplay	0
League	1838
highestleague	13464
streams	100
SPIN1 124	
SPIN2 154	
SPIN3 140	
SPIN4 159	
SPIN5 166	
SPIN6 156	

SPIN7 138	
SPIN8 144	
SPIN9 158	
SPIN10	160
SPIN11	187
SPIN12	168
SPIN13	187
SPIN14	156
SPIN15	147
SPIN16	147
SPIN17	175
Narcissism	23
Gender	0
Age	0
Work	38
Degree	0
Birthplace	0
Residence	0
Reference	15
Playstyle	0
accept 414	
GAD_T	0
SWL_T	0
SPIN_T	650
Residence_ISO3	110
Birthplace_ISO3	121

- To remove the missing values fill.na () function has been used for the imputation:

  df = df.apply(lambda x: x.fillna(x.value_counts().index[0]))

- Dropped columns:

  df=df.drop(columns=['Narcissism','streams','SPIN1','SPIN2','SPIN3','SPIN4','SPIN5','SPIN6','SPIN7','SPIN8','SPIN9','SPIN10','SPIN11','SPIN12','SPIN13','SPIN14','SPIN15','SPIN16','SPIN17','Timestamp','accept','League','Birthplace','Reference','Birthplace_ISO3','highestleague','SWL1','SWL2','SWL3','SWL4','SWL5','earnings','whyplay','Birthplace_ISO3','Residence_ISO3'])

  df.drop(df[(df['Playstyle']!='Singleplayer') & (df['Playstyle']!='Multiplayer - online - with strangers') & (df['Playstyle']!='Multiplayer - online - with online acquaintances or teammates') & (df['Playstyle']!='Multiplayer - online - with real life friends') & (df['Playstyle']!='Multiplayer - offline (people in the same room)') & (df['Playstyle']!='all of the above')].index,axis=0,inplace=True)

- Converted the variables to numeric:

```
df["Age"] = pd.to_numeric(df["Age"])
df["Hours"] = pd.to_numeric(df["Hours"])
df["streams"] = pd.to_numeric(df["streams"])
df["Hours"]=pd.to_numeric(df['Hours'])
df["GAD_T"]=pd.to_numeric(df['GAD_T'])
```

- Standardized the data to remove outliers using z-score:

```
df = df[(-3 < zscore(df['Hours'])) & (zscore(df['Hours']) < 3)]
df = df[(-3 < zscore(df['Age'])) & (zscore(df['Age']) < 3)]
df = df[(-3 < zscore(df['GAD_T'])) & (zscore(df['GAD_T']) < 3)]
df = df[(-3 < zscore(df['SWL_T'])) & (zscore(df['SWL_T']) < 3)]
```

Label Encoding:

Label encoding has been applied to the target variable. Label Encoder is applied to the features as our use being the classification problem. Encoding was done to decide in a better way on how these labels must be operated, and labels are converted into numeric form.

```
features_list = ['GAD5','GAD6','GADE','SPIN_T','SWL_T','Game','Playstyle','Platform',
'Gender','Age','Hours','Work','Residence']
 X = features_list1.values
 y = data['GAD_T'].values
 class_le = LabelEncoder()
 class_names = class_le.fit_transform(y)
```

Label Binarizer:

Label Binarizer has been applied to the target variable to convert multi-class to binary values and Calculate ROC.

```
 class_names1 = [0,1,2]
 y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
  n_classes = y_test_bin.shape[1]
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
            fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_pred_score[:, i])
                        roc_auc[i] = auc(fpr[i], tpr[i])
```

EDA Analysis:

To visualize the dataset graphically histogram, scatter plots and heatmap have been used.

- Histogram has been used the understand the distribution of each variable in the dataset

```
self.current_features.value_counts().plot(kind='bar', ax=self.ax1)
self.ax1.set_title('Histogram of : ' + x_a)
self.ax1.set_xlabel(x_a)
self.ax1.set_ylabel('frequency')
```

- Scatter Plots have been used to understand the relationship among the variables.

```
self.ax1.scatter(X_1,y_1)

if self.checkbox1.isChecked():
    b, m = polyfit(X_1, y_1, 1)
    self.ax1.plot(X_1, b + m * X_1, '-', color="orange")

vtitle = "GAD_T vrs "+ cat1+ "Gaming study"
self.ax1.set_title(vtitle)
self.ax1.set_xlabel("Level of Generalized anxiety Disorder")
self.ax1.set_ylabel(cat1)
self.ax1.grid(True)
```

- Heat maps are used to find the correlation between the variables.

```
self.groupBox2 = QGroupBox('Correlation Plot')
self.groupBox2Layout= QVBoxLayout()
self.groupBox2.setLayout(self.groupBox2Layout)
self.groupBox2Layout.addWidget(self.canvas)
```

The selection of more than one variable can be restricted using the following command. This is implemented to make sure that therewill be no more than one variable plotted for Histogram and not more than two variables for the scatter plot.

```
def Message(self):
  QMessageBox.about(self, "Warning", " You can't exceed more than 1 feature")
```

Decision Tree Classifier:

- The packages that are used for modeling is Scikit Learn and we have imported decision tree classifier library to perform the functions of the algorithm.

  from sklearn.tree import DecisionTreeClassifier

- To prepare the data for model training, the data is encoded using the Label encoder function described above. The processed data is then separated into train and test halves at a 70:30 ratio by default, however this can be changed according to the user's needs. The model is trained and tested using the gini and entropy criteria, as well as the user-supplied max-depth.
- Once the model is initially trained and tested the dashboard of the decision tree classifier gets updated with the performance measures like confusion matrix, classification report, Accuracy score and ROC value.The below code snippet shows the above defined performance measures for entropy model.

```
# # decision Tree by MB
cols = df2[['GAD5','GAD6','GADE','SPIN_T','SWL_T','Game','Playstyle','Platform', 'Gender','Age','Hours','Work','Residence']]
x = cols.values
y = df2['GAD_T'].values
from sklearn.preprocessing import label_binarize
class_le = LabelEncoder()

y = class_le.fit_transform(y)

y1 = label_binarize(y, classes=[0,1,2])

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
x_train1, x_test1, y_train1, y_test1 = train_test_split(x, y1, test_size=0.3, random_state=1)

# Fit dt to the training set
rf1 = DecisionTreeClassifier(max_depth=3,criterion='entropy',random_state=0)
# Fit dt to the training set
rf1.fit(x_train,y_train)
# y_train_pred = rf1.predict(x_train)
y_test_pred = rf1.predict(x_test)
y_pred_score = rf1.predict_proba(x_test)


rf2 = OneVsRestClassifier(DecisionTreeClassifier(max_depth=3,criterion='entropy'))
# Fit dt to the training set
rf2.fit(x_train1,y_train1)
# y_train_pred = rf1.predict(x_train)
y_test_pred1 = rf2.predict(x_test1)
```

```python
# Evaluate test-set accuracy
print('test set evaluation: ')
print("Accuracy score: ",accuracy_score(y_test, y_test_pred)*100)
print("Confusion Matrix: \n",confusion_matrix(y_test, y_test_pred))
print("Classification report:\n",classification_report(y_test, y_test_pred))

from sklearn.metrics import roc_curve, auc

n_classes=3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test1[:, i], y_pred_score1[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
    print(f'AUC value of {i} class:{roc_auc[i]}')

# Plot of a ROC curve for a specific class
for i in range(n_classes):
    plt.figure()
    plt.plot(fpr[i], tpr[i], label='ROC curve (area = %0.2f)' % roc_auc[i])
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Decision Tree ROC')
    plt.legend(loc="lower right")
```

Random Forest Classifier:

- The packages that have been used for the modelling is Scikit Learn and we have imported random forest classifier library to perform the functions of the algorithm.

    from sklearn.ensemble import RandomForestClassifier

- The data is encoded by using Label encoder function provided above to prepare the data for model training. The processed data is then split into train and test split at the ratio of 70:30 by default which is subject to change as per the user's requirements. The model gets trained and tested with the number of estimators provided by the user.

- Once the model is initially trained and tested the dashboard of the random forest classifier gets updated with the performance measures like confusion matrix, classification report, Accuracy score and ROC value. The below code snippet shows the above defined performance measures for entropy model and the same can be implemented for gini model as well.
- The important features of the model can be viewed using Imp_Features option on the dashboard:

```
#========================== random forest ==========================
#%%
#Bresh

from sklearn.ensemble import RandomForestClassifier
from sklearn.multiclass import OneVsRestClassifier
# Instantiate dtree
rf1 = RandomForestClassifier(n_estimators=100)
# Fit dt to the training set
rf1.fit(x_train,y_train)
# y_train_pred = rf1.predict(x_train)
y_test_pred = rf1.predict(x_test)
y_pred_score = rf1.predict_proba(x_test)


rf2 = OneVsRestClassifier(RandomForestClassifier(n_estimators=100))
# Fit dt to the training set
rf2.fit(x_train1,y_train1)
# y_train_pred = rf1.predict(x_train)
y_test_pred1 = rf2.predict(x_test1)
y_pred_score1 = rf2.predict_proba(x_test1)

print('Random forest results')
```

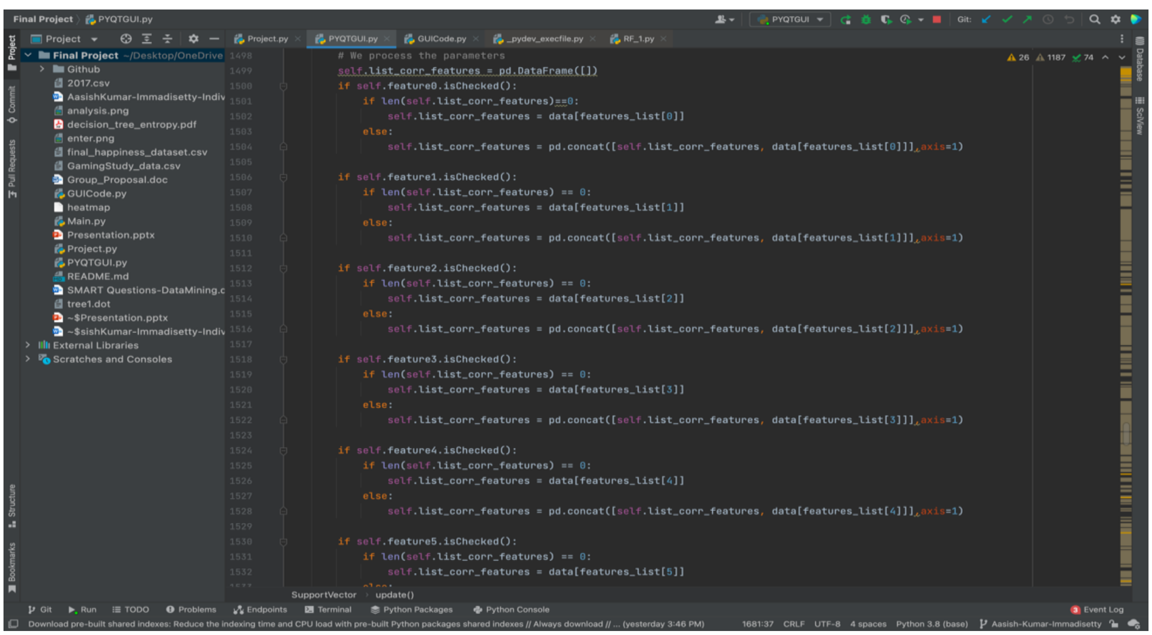
```
# Evaluate test-set accuracy
print('test set evaluation: ')
print("Accuracy score: ",accuracy_score(y_test, y_test_pred)*100)
print("Confusion Matrix: \n",confusion_matrix(y_test, y_test_pred))
print("Classification report:\n",classification_report(y_test, y_test_pred))

from sklearn.metrics import roc_curve, auc

n_classes=3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test1[:, i], y_pred_score1[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
    print(f'AUC value of {i} class:{roc_auc[i]}')
```

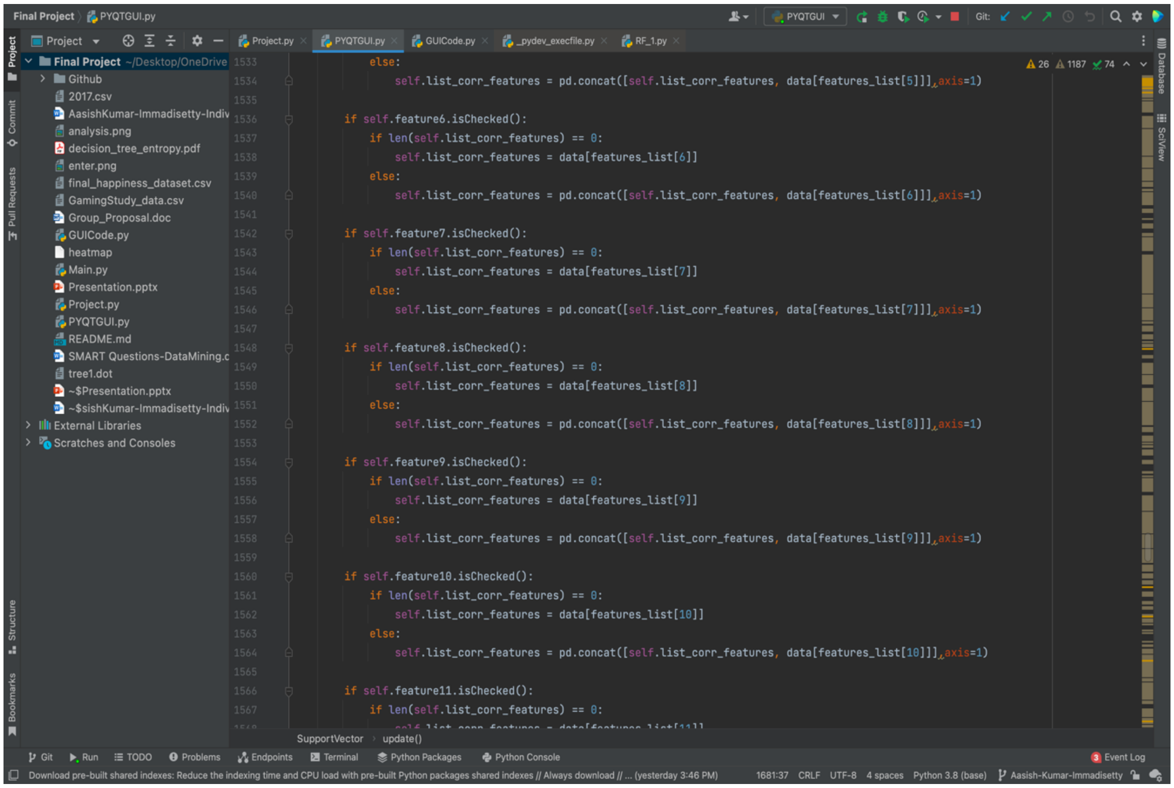
```python
# Plot of a ROC curve for a specific class
for i in range(n_classes):
    plt.figure()
    plt.plot(fpr[i], tpr[i], label='ROC curve (area = %0.2f)' % roc_auc[i])
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Random Forest ROC')
    plt.legend(loc="lower right")
    plt.show()
```

Support Vector Machine:

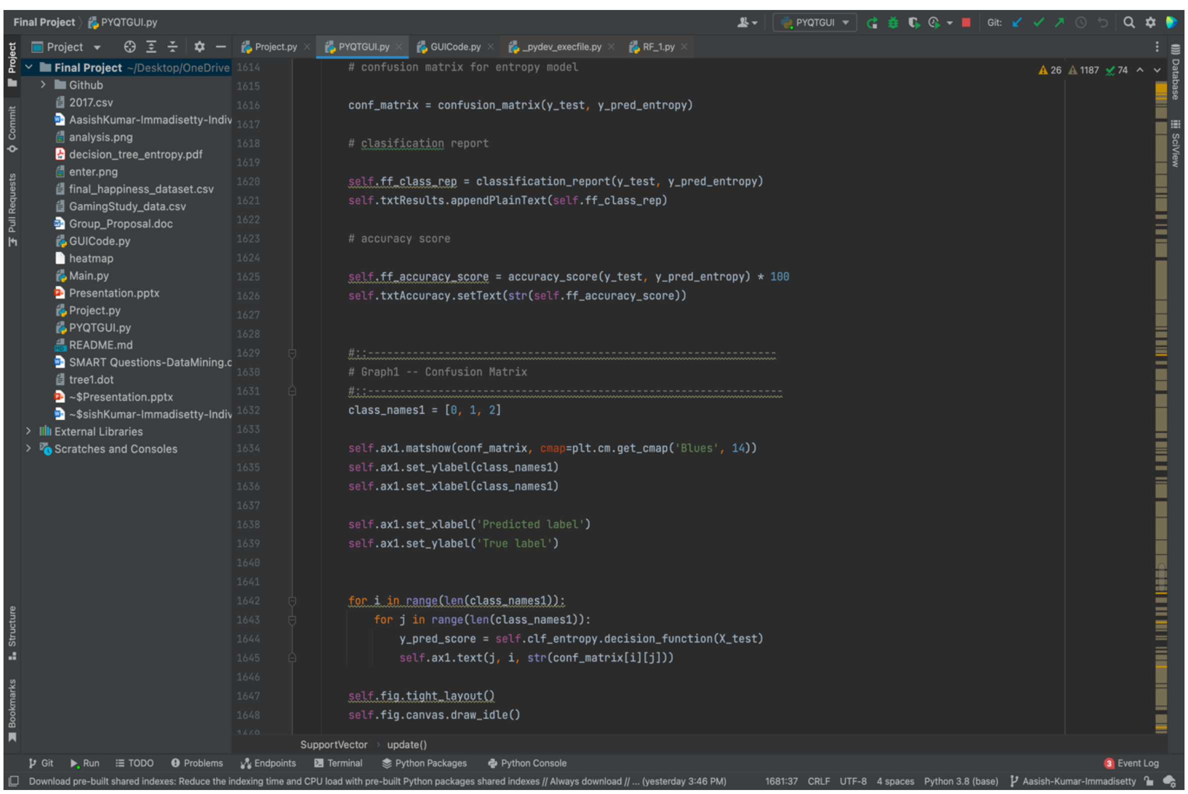
- The packages that have been used for the modelling is Scikit Learn and we have imported Support Vector Classifier library to perform the functions of the algorithm.
  from sklearn.svm import SVC

- Label Encoder and Label Binarizer function provided above to prepare the data for model training and calculating auc_roc scores. The processed data is then split at the ratio 70:30 by default which is subject to change as per the user's requirements. The model gets trained and tested based on kernel" linear "along with the feature's selection provided by the user.

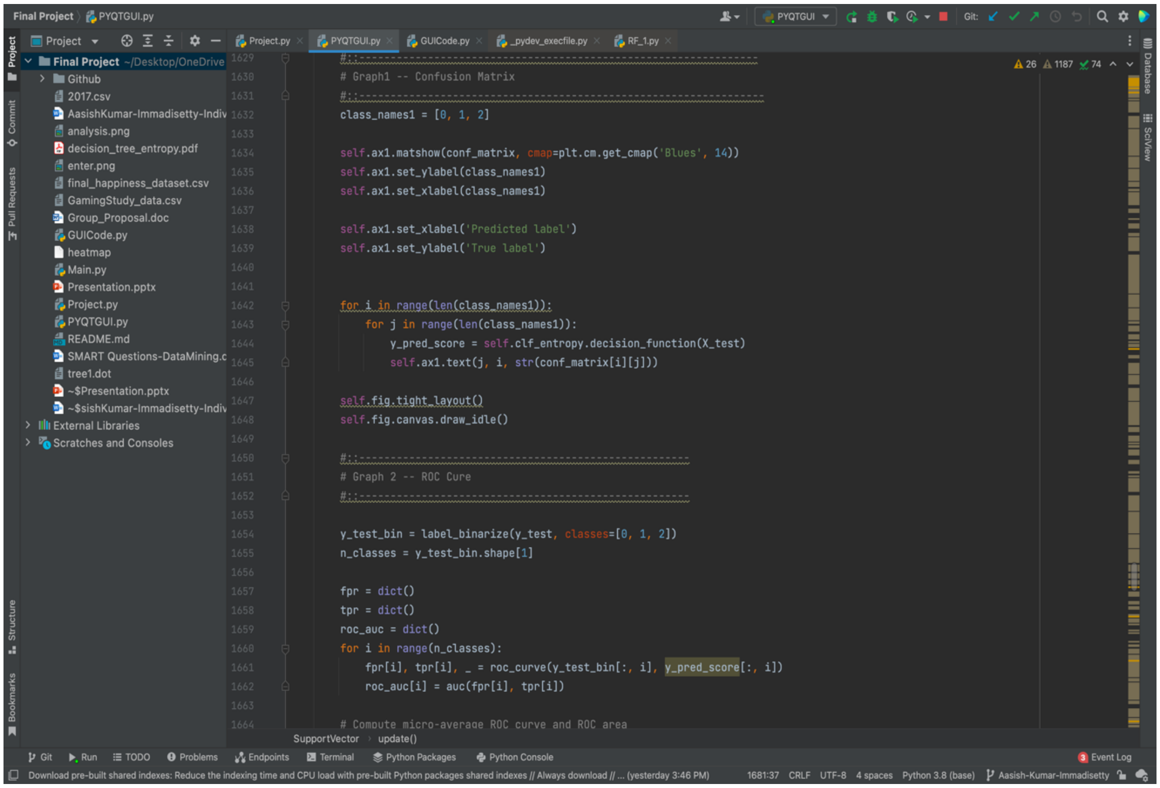




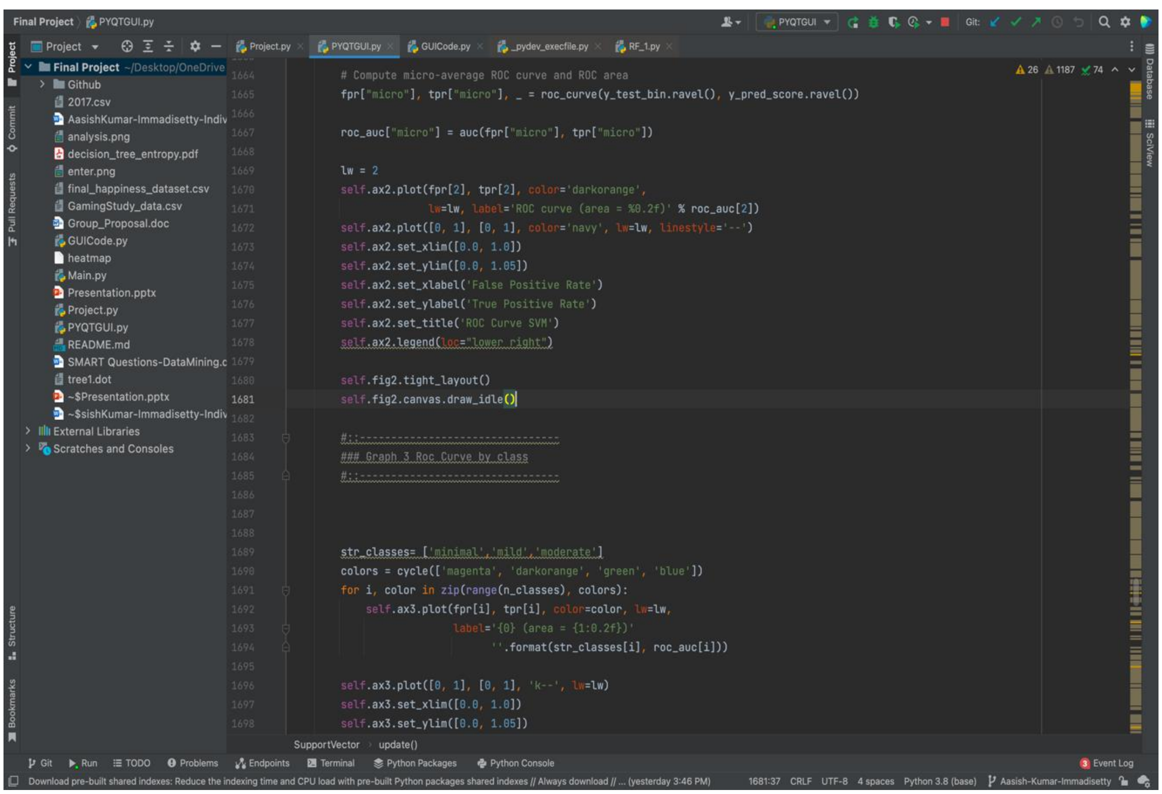
Final Project
PYQTGUI.py
Project.py
PYQTGUI.py
GUICode.py
_pydev_execfile.py
RF_1.py
else:
self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[5]]],axis=1)
if self.feature6.isChecked():
if len(self.list_corr_features) == 0:
self.list_corr_features = data[features_list[6]]
else:
self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[6]]],axis=1)
if self.feature7.isChecked():
if len(self.list_corr_features) == 0:
self.list_corr_features = data[features_list[7]]
else:
self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[7]]],axis=1)
if self.feature8.isChecked():
if len(self.list_corr_features) == 0:
self.list_corr_features = data[features_list[8]]
else:
self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[8]]],axis=1)
if self.feature9.isChecked():
if len(self.list_corr_features) == 0:
self.list_corr_features = data[features_list[9]]
else:
self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[9]]],axis=1)
if self.feature10.isChecked():
if len(self.list_corr_features) == 0:
self.list_corr_features = data[features_list[10]]
else:
self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[10]]],axis=1)
if self.feature11.isChecked():
if len(self.list_corr_features) == 0:
SupportVector
update()



Final Project
PYQTGUI.py
Project.py
PYQTGUI.py
GUICode.py
_pydev_execfile.py
RF_1.py
# confusion matrix for entropy model
conf_matrix = confusion_matrix(y_test, y_pred_entropy)
# clasification report
self.ff_class_rep = classification_report(y_test, y_pred_entropy)
self.txtResults.appendPlainText(self.ff_class_rep)
# accuracy score
self.ff_accuracy_score = accuracy_score(y_test, y_pred_entropy) * 100
self.txtAccuracy.setText(str(self.ff_accuracy_score))
#::---
Graph1 -- Confusion Matrix
#::--
class_names1 = [0, 1, 2]
self.ax1.matshow(conf_matrix, cmap=plt.cm.get_cmap('Blues', 14))
self.ax1.set_ylabel(class_names1)
self.ax1.set_xlabel(class_names1)
self.ax1.set_xlabel('Predicted label')
self.ax1.set_ylabel('True label')
for i in range(len(class_names1)):
for j in range(len(class_names1)):
y_pred_score = self.clf_entropy.decision_function(X_test)
self.ax1.text(j, i, str(conf_matrix[i][j]))
self.fig.tight_layout()
self.fig.canvas.draw_idle()
SupportVector
update()

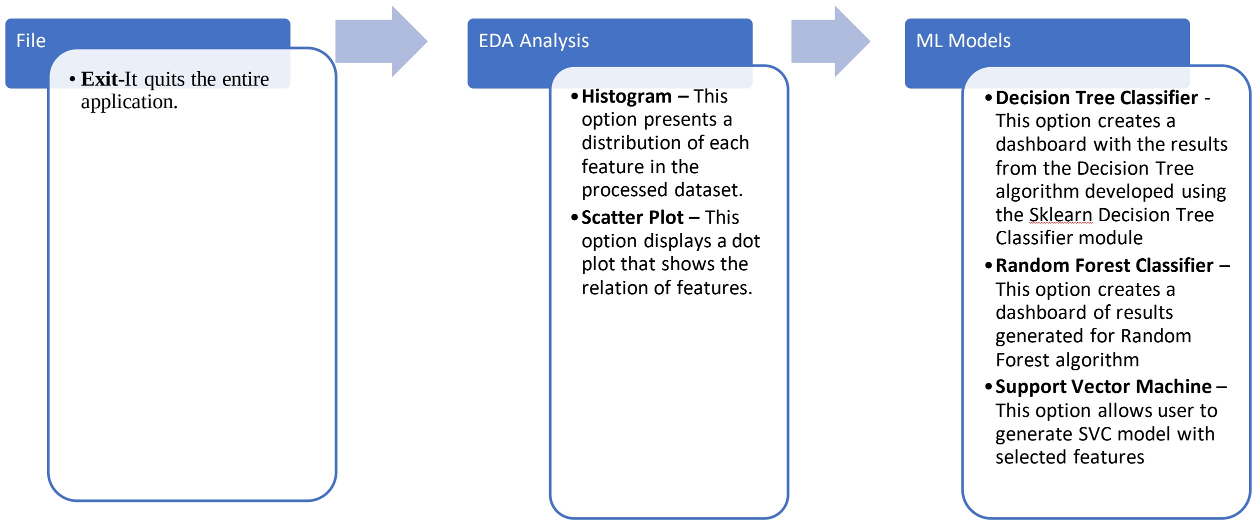


```python
#::-----------------------------------------------------------------
# Graph1 -- Confusion Matrix
#::------------------------------------------------------------------
class_names1 = [0, 1, 2]

self.ax1.matshow(conf_matrix, cmap=plt.cm.get_cmap('Blues', 14))
self.ax1.set_ylabel(class_names1)
self.ax1.set_xlabel(class_names1)

self.ax1.set_xlabel('Predicted label')
self.ax1.set_ylabel('True label')


for i in range(len(class_names1)):
    for j in range(len(class_names1)):
        y_pred_score = self.clf_entropy.decision_function(X_test)
        self.ax1.text(j, i, str(conf_matrix[i][j]))

self.fig.tight_layout()
self.fig.canvas.draw_idle()

#::------------------------------------------------------
# Graph 2 -- ROC Cure
#::------------------------------------------------------

y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
n_classes = y_test_bin.shape[1]

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_pred_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
```



```python
# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), y_pred_score.ravel())

roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

lw = 2
self.ax2.plot(fpr[2], tpr[2], color='darkorange',
              lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
self.ax2.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
self.ax2.set_xlim([0.0, 1.0])
self.ax2.set_ylim([0.0, 1.05])
self.ax2.set_xlabel('False Positive Rate')
self.ax2.set_ylabel('True Positive Rate')
self.ax2.set_title('ROC Curve SVM')
self.ax2.legend(loc="lower right")

self.fig2.tight_layout()
self.fig2.canvas.draw_idle()

#::-------------------------------
### Graph 3 Roc Curve by class
#::-------------------------------



str_classes= ['minimal','mild','moderate']
colors = cycle(['magenta', 'darkorange', 'green', 'blue'])
for i, color in zip(range(n_classes), colors):
    self.ax3.plot(fpr[i], tpr[i], color=color, lw=lw,
                  label='{0} (area = {1:0.2f})'
                        ''.format(str_classes[i], roc_auc[i]))

self.ax3.plot([0, 1], [0, 1], 'k--', lw=lw)
self.ax3.set_xlim([0.0, 1.0])
self.ax3.set_ylim([0.0, 1.05])
```

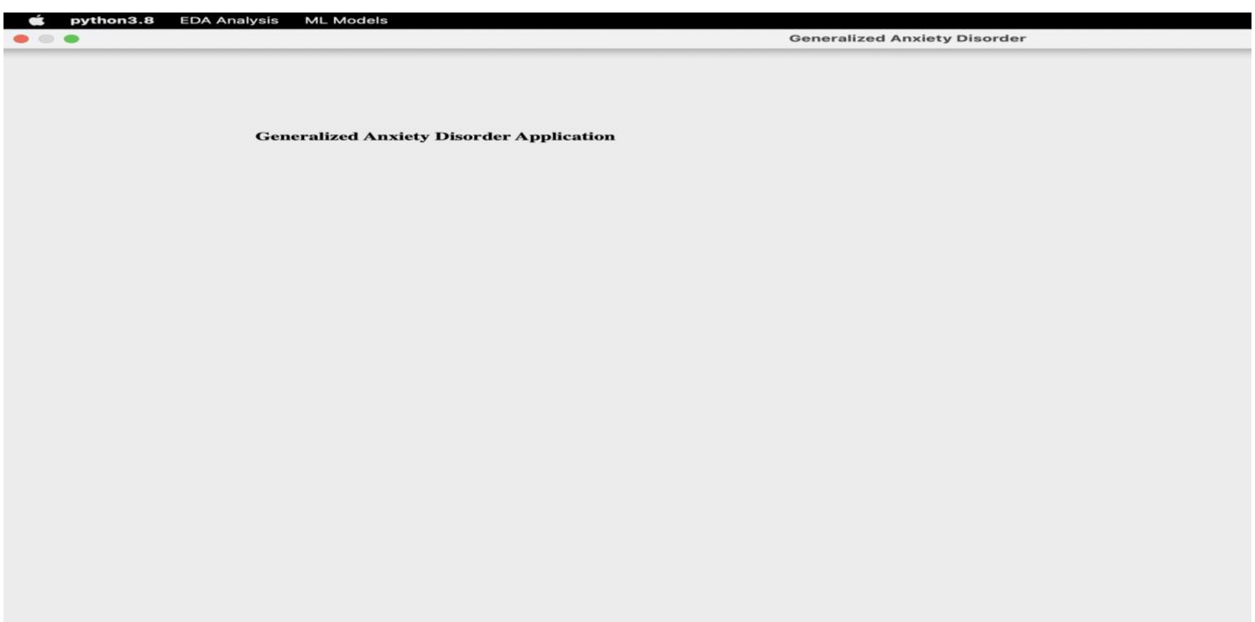
Structure of the application:



RESULTS

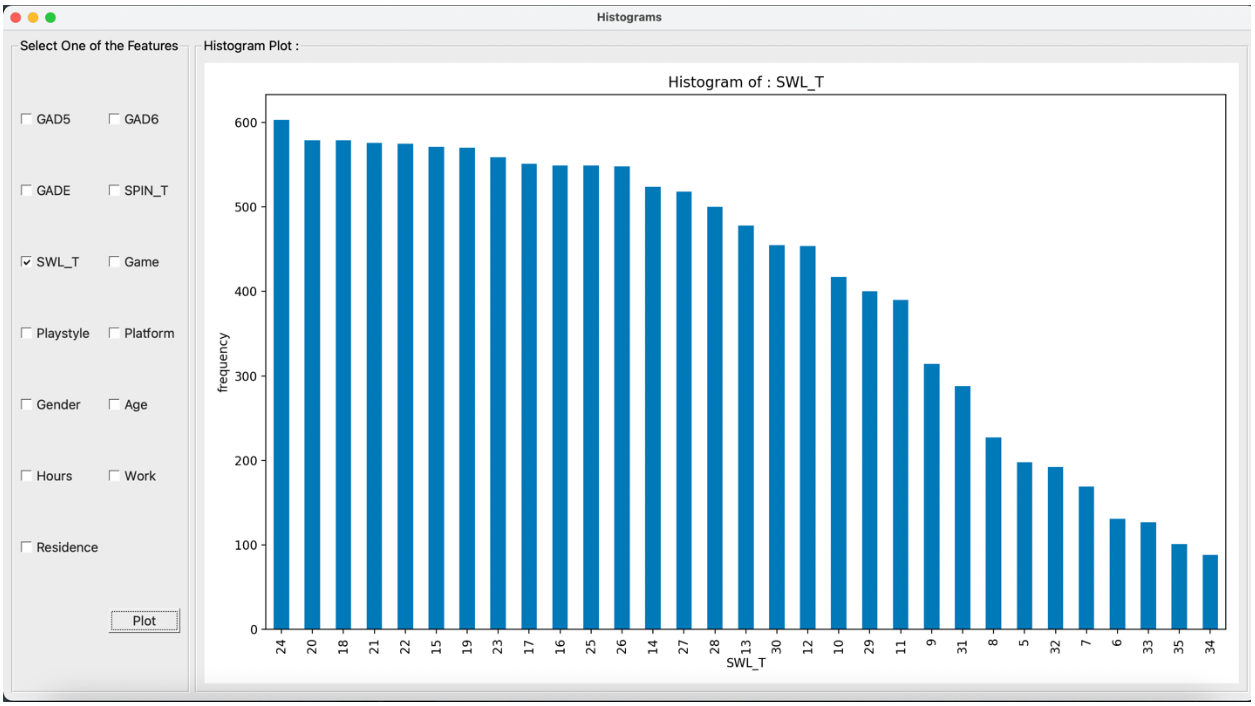
Initial UI Window:

- This is how the Generalized Anxiety Disorder looks, it has three buttons one for File (Exit), EDA Analysis & ML Models.



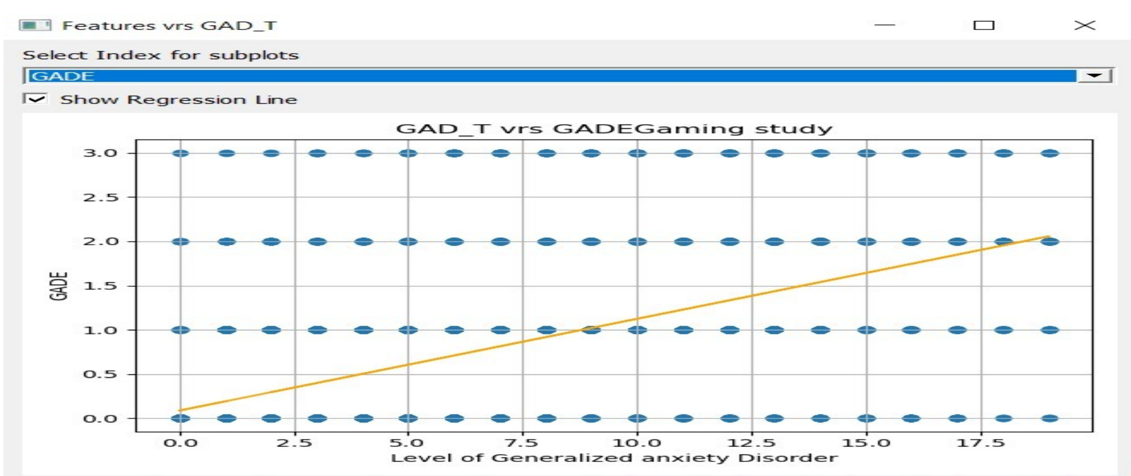
Histogram:

- The image explains the pictorial representation of histogram and the various features provided on the left grid and after pushing the plot button the histogram gets displayed on the canvas.



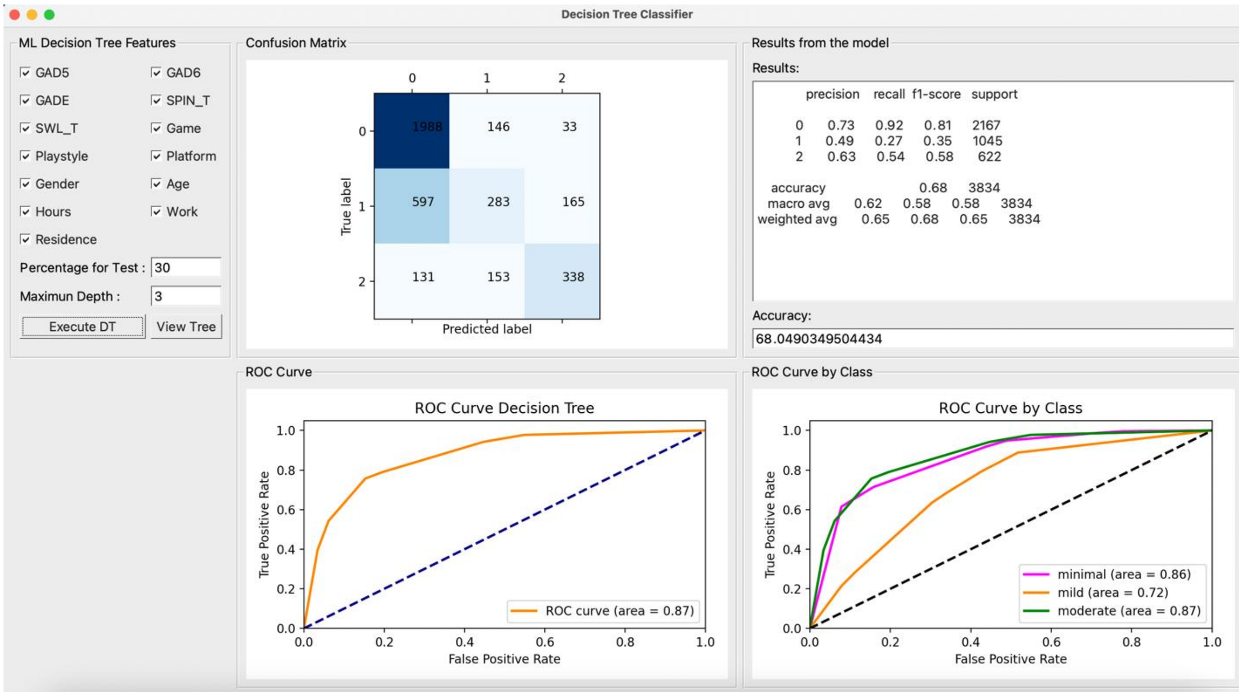
Scatter Plot:

- This image describes the scatter plot graphical representation of the various features selected; left grid represents the variables to be selected for y axis and upper grid represents the variable to be selected for the x-axis.



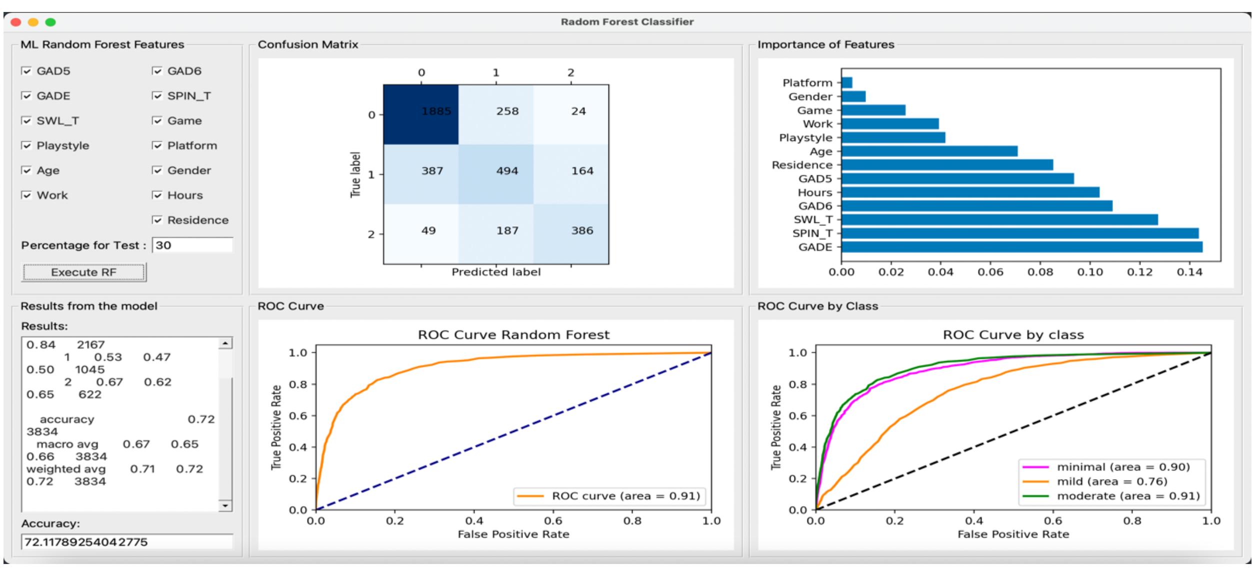
Decision Tree Classifier:

- The image displays the decision tree dashboard with confusion matrix, results from the model, ROC curve and ROC curve by class



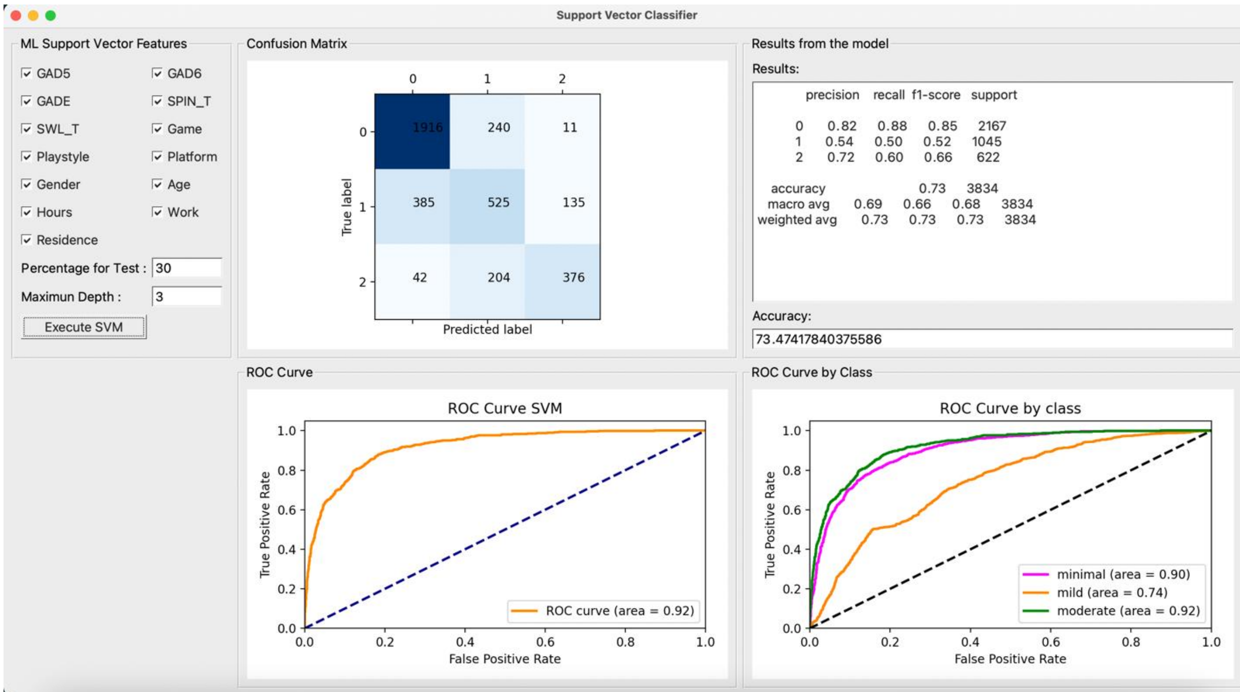
Random Forest Classifier:

- The image displays the Random Forest Classifier dashboard with confusion matrix, results from the model, ROC curve and ROC curve by class, importance of features.



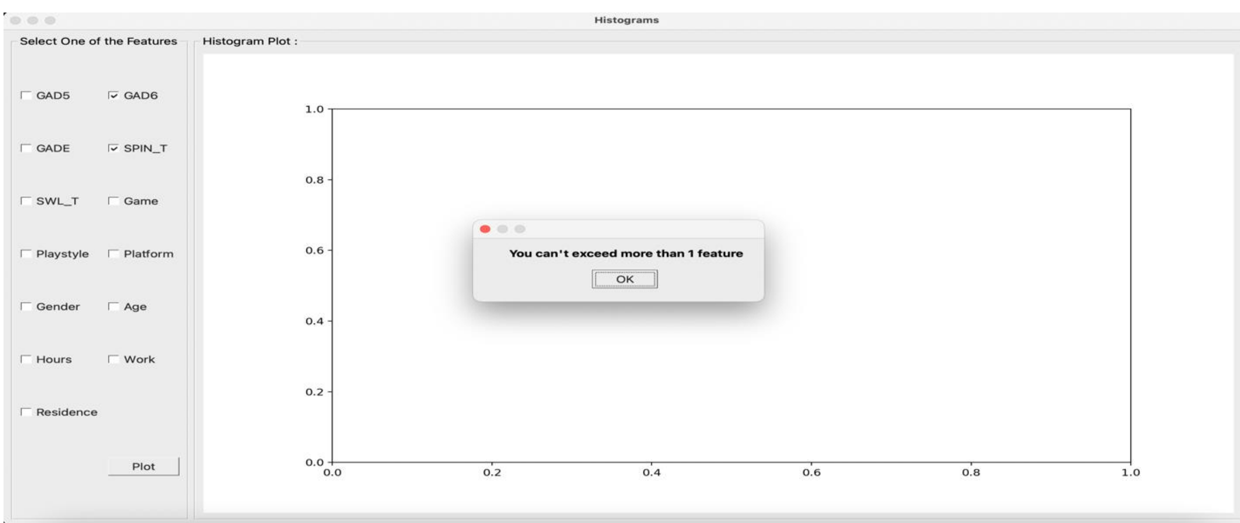
Support Vector Classifier:

- The image displays the Support Vector Machine dashboard with confusion matrix, results from the model, ROC curve and ROC curve by class, importance of features.



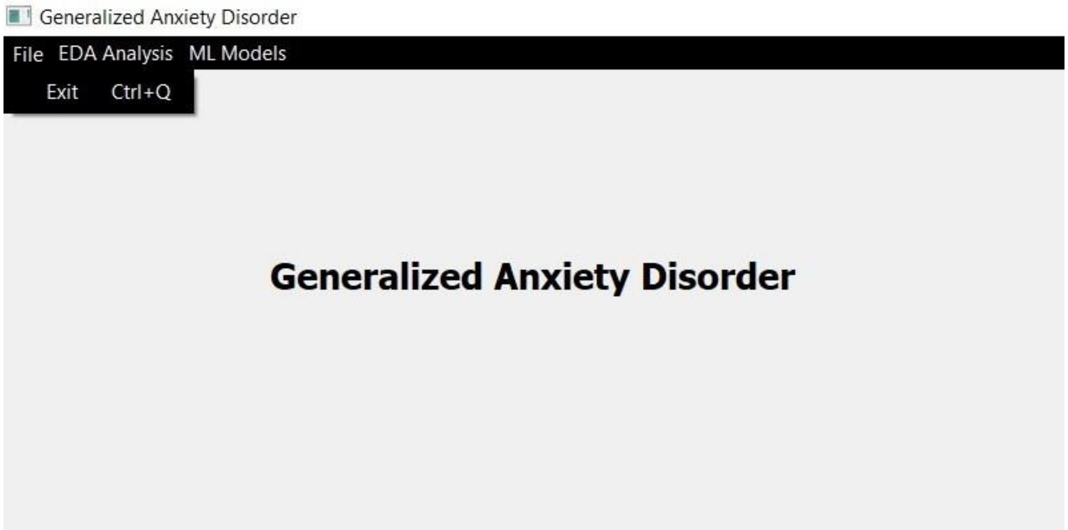
Error Window:

- The validation has been given on graphs to make sure that the feature selection should not exceed more than required variables respectively.



Closing Window:

- A menu bar with a closing file option has been given on the dashboard of the application.



SUMMARY

SVM Model:

- Accuracy of SVM Model, Accuracy = 73.47%
- From the classification report of SVM model:
  - F1 score for 0’s: 0.85, 1’s: 0.52, 2’s: 0.66
  - Precision for 0’s: 0.82, 1’s: 0.54, 2’s: 0.72
  - Recall for 0’s: 0.88, 1’s: 0.50, 2’s: 0.60
- Area under curve for SVM Model is 0.92. An ideal model should have AUC value above 8 which is why we can say model is decent.
- The features with high importance are:

  'GAD5','GAD6','GADE','SPIN_T','SWL_T','Game','Playstyle','Platform', 'Gender','Age','Hours','Work','Residence'

Decision Tree Classifier:

- Accuracy of Decision Tree, Accuracy = 68%
- From the classification report of Decision Tree model:
  - F1 score for 0 : 0.81 ,for 1: 0.35 , for 2: 0.58
  - Precision for 0 :0.73, for 1:0.49 , for 2 :0.63
  - Recall for 0: 0.92, for 1:0.27, for 2:0.54
- Area under the ROC curve for the decision tree is 0.87.
- The features with high importance are:

  'GAD5','GAD6','GADE','SPIN_T','SWL_T','Game','Playstyle','Platform', 'Gender','Age','Hours','Work','Residence'

Random Forest Classifier:

- Accuracy of Random Forest Classifier Model, Accuracy = 72.11%
- From the classification report of SVM model:
  - F1 score for 0’s: 0.87, 1’s: 0.47, 2’s: 0.62
  - Precision for 0’s: 0.84, 1’s: 0.50, 2’s: 0.65
  - Recall for 0’s: 0.81, 1’s: 0.53, 2’s: 0.67
- Area under curve for Random Forest Model is 0.91.
- The features with high importance are:

  'GAD5','GAD6','GADE','SPIN_T','SWL_T','Game','Playstyle','Platform', 'Gender','Age','Hours','Work','Residence'

CONCLUSION

- Because it has the highest accuracy, f1-score, and precision of the three distinct classifiers, the Support Vector Machine model is the best. It also has the highest AUC of the three classifiers.

- Because it has slightly less accuracy and other results than the Support Vector Machine model, the Random Forest model is the second-best of the three classifiers. Increase the number of estimators in the model to increase accuracy and other outputs, but this will demand more processing resources.

- The Decision Tree Model is the least popular of the three classifiers, with less accuracy, f1 score, and precision comparatively. However, its AUC value of 0.87 which indicates that the model is accurate.

REFERENCES

- https://doc.bccnsoft.com/docs/PyQt5/
- https://pyqt5.files.wordpress.com/2017/06/pyqt5tutorial.pdf
- https://numpy.org/doc/
- https://matplotlib.org/stable/users/index.html
- https://matplotlibguide.readthedocs.io/en/latest/
- https://sklearn.org/user_guide.html
- https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
- https://towardsdatascience.com/explain-your-machine-learning-with-feature-importance-774cd72abe
- https://www.mygreatlearning.com/blog/introduction-to-support-vector-machine/
- https://www.javatpoint.com/machine-learning-random-forest-algorithm
- https://www.mygreatlearning.com/blog/decision-tree-algorithm/
- https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm