DATS 6103 – Intro to Data Mining

Individual Final Report
Group – 1
Dec 7, 2021
Aasish Kumar Immadisetty

Table of Contents

1. Introduction:

Public health has an important role in promoting the population's welfare, maintaining its security, and protecting it from infectious disease and environmental risks, and assisting in assuring the population's safety and quality treatment. The Internet is transforming business, education, government, healthcare, and even how we communicate with our loved ones on a daily, and it has emerged as one of the most important drivers of social change. People have started using the incredible internet and started online gaming, spending a lot of time on it because of the internet's evolution. People's mental and psychological health is being harmed by online gaming, which is leading to a variety of mental diseases. This dataset contains information gathered as part of a global survey of gamers. The survey asked questions that psychologists typically ask persons with anxiety, social phobia, and low to no life satisfaction. The questionnaire is made up of a series of questions that would be asked in a psychological investigation.

In analyzing the influence of online gaming on General Anxiety Disorder, we will use different features from the models, such as GAD, Game, Playstyle, Platform on which they play, Age, Gender, Hours, Work, and Residence. The project is illustrated by developing a GUI-based application that displays the end-to-end modeling utilizing three machine learning algorithms: Random Forest Classifier, Decision Tree, and Support Vector Machine, respectively.

Cleaning the dataset, exploratory data analysis, preprocessing, modeling, model comparison, GUI creation, preparing a power point presentation, writing the group report, and generating a demo of the GUI were all part of the shared effort.

2. Personal Contribution:

Code:

Pre-processing:

- Heat map was plotted to see whether there was a correlation between different features and to narrow down the dataset to only the features that were required.
- Because most of the variables with missing values are categorical data, I filled in the blank values using the most often occurring phrase in that column.
- The z-score is used to standardize data to improve the quality of the data in the dataset and to remove outliers.
- To convert multi-class labels to binary labels, the target variable was label binarized, which makes learning one regressor or binary classifier per class much easier.
- Divide the data set into two sections: training and testing.

EDA Analysis:

- Built scatter plot for variables Hours and Age to understand the relation between the variables.

Model Building:

- For classification and regression, created a Support Vector Machine model.

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 confusion matrix, classification report, accuracy, ROC curve, and Area under the Curve were calculated for the Support Vector Machine.

Presentation:

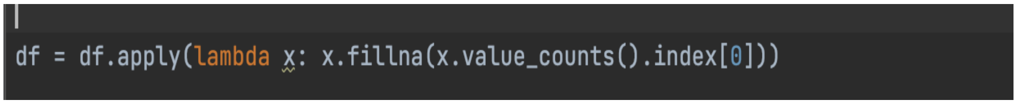
- Assisted with the preparation of presentations to explain the EDA Analysis and SVM Model.

Group Final Report:

- I worked on a few aspects of data pre-processing, EDA analysis, and the outputs of the Support Vector Machine (SVM) model.

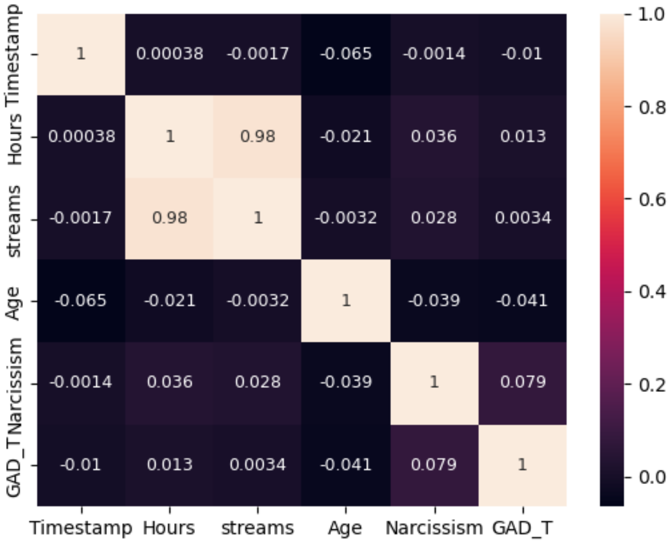
3. Personal Contribution in detail:

- Because most of the variables with missing values are categorical data, I filled in the blank values using the most often occurring phrase in that column

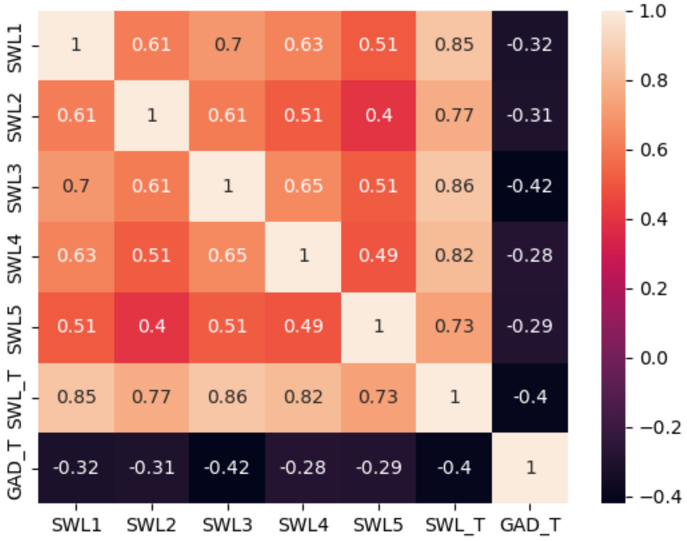


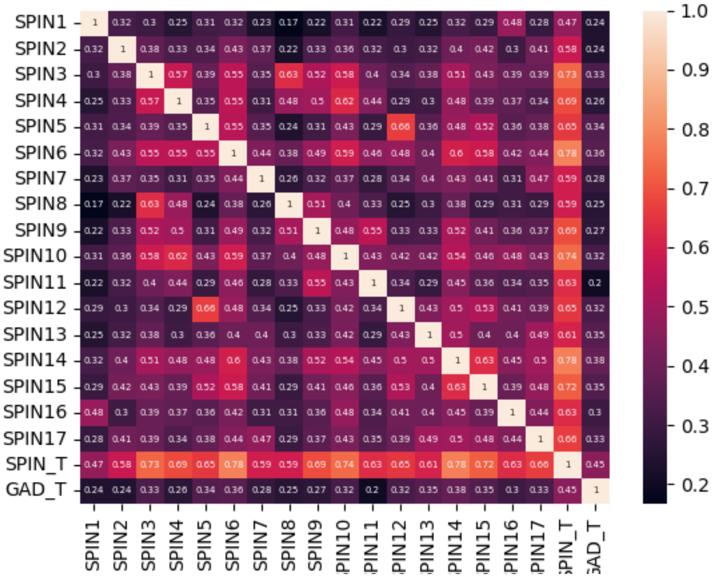
```
df = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

- Heat map was plotted to see whether there was a correlation between different features and to narrow down the dataset to only the features that were required.

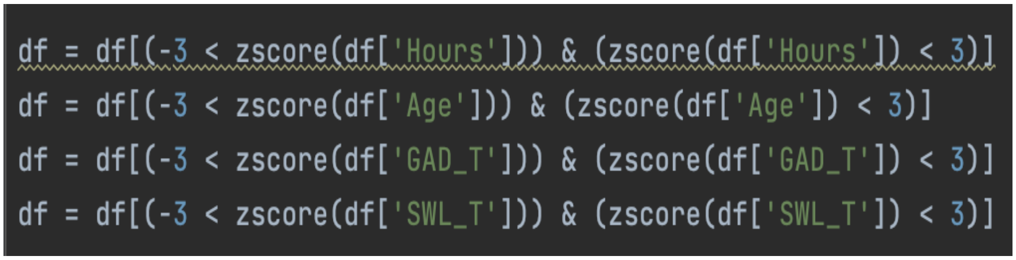


	Timestamp	Hours	streams	Age	Narcissism	GAD_T
Timestamp	1	0.00038	-0.0017	-0.065	-0.0014	-0.01
Hours	0.00038	1	0.98	-0.021	0.036	0.013
streams	-0.0017	0.98	1	-0.0032	0.028	0.0034
Age	-0.065	-0.021	-0.0032	1	-0.039	-0.041
Narcissism	-0.0014	0.036	0.028	-0.039	1	0.079
GAD_T	-0.01	0.013	0.0034	-0.041	0.079	1



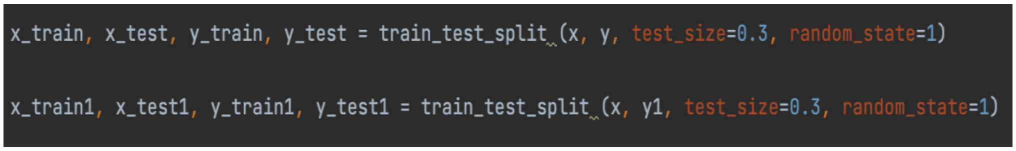


- The z-score is used to standardize data to improve the quality of the data in the dataset and to remove outliers.



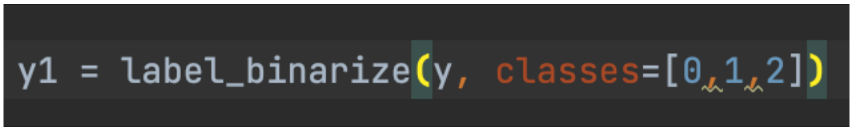
```
df = df[(-3 < zscore(df['Hours'])) & (zscore(df['Hours']) < 3)]
df = df[(-3 < zscore(df['Age'])) & (zscore(df['Age']) < 3)]
df = df[(-3 < zscore(df['GAD_T'])) & (zscore(df['GAD_T']) < 3)]
df = df[(-3 < zscore(df['SWL_T'])) & (zscore(df['SWL_T']) < 3)]
```

- Divide the data set into two sections: training and testing.



```
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)
```

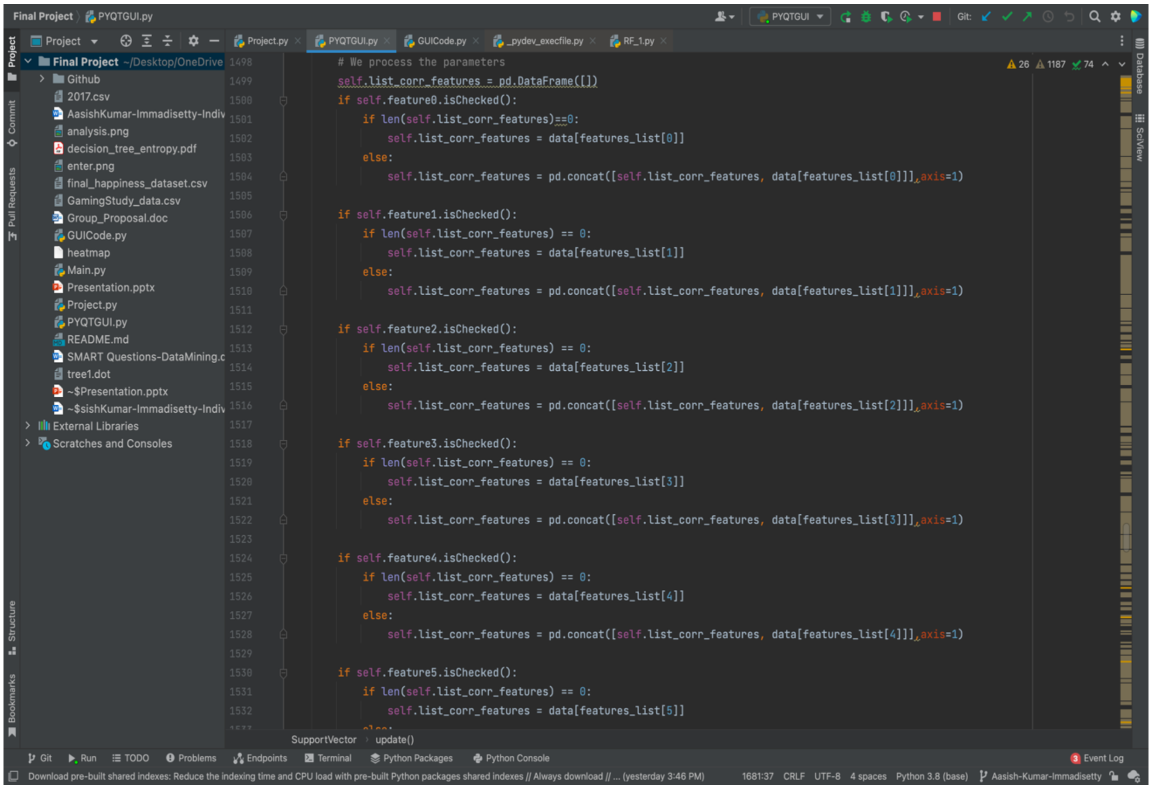
- To convert multi-class labels to binary labels, the target variable was label binarized, which makes learning one regressor or binary classifier per class much easier.



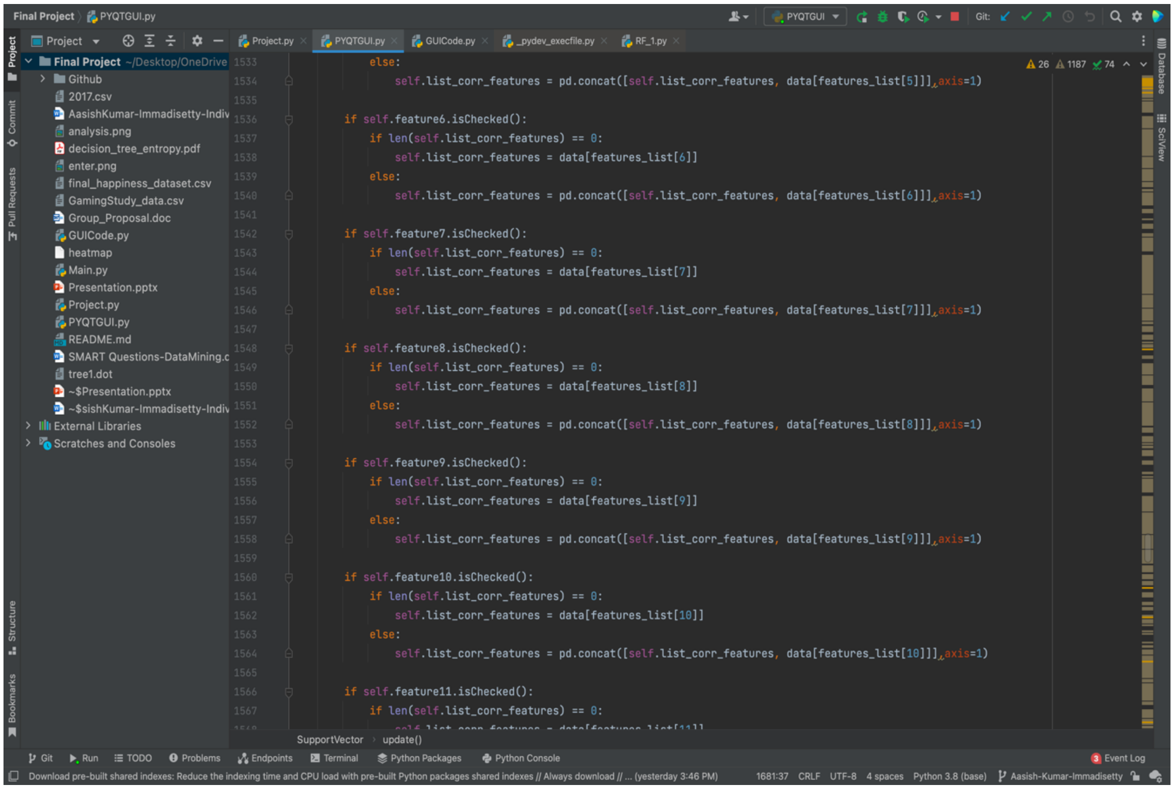
```
y1 = label_binarize(y, classes=[0,1,2])
```

Model Building (PYQT5 code):

- For classification and regression, created a Support Vector Machine model. The confusion matrix, classification report, accuracy, ROC curve, and Area under the Curve were calculated for the Support Vector Machine.

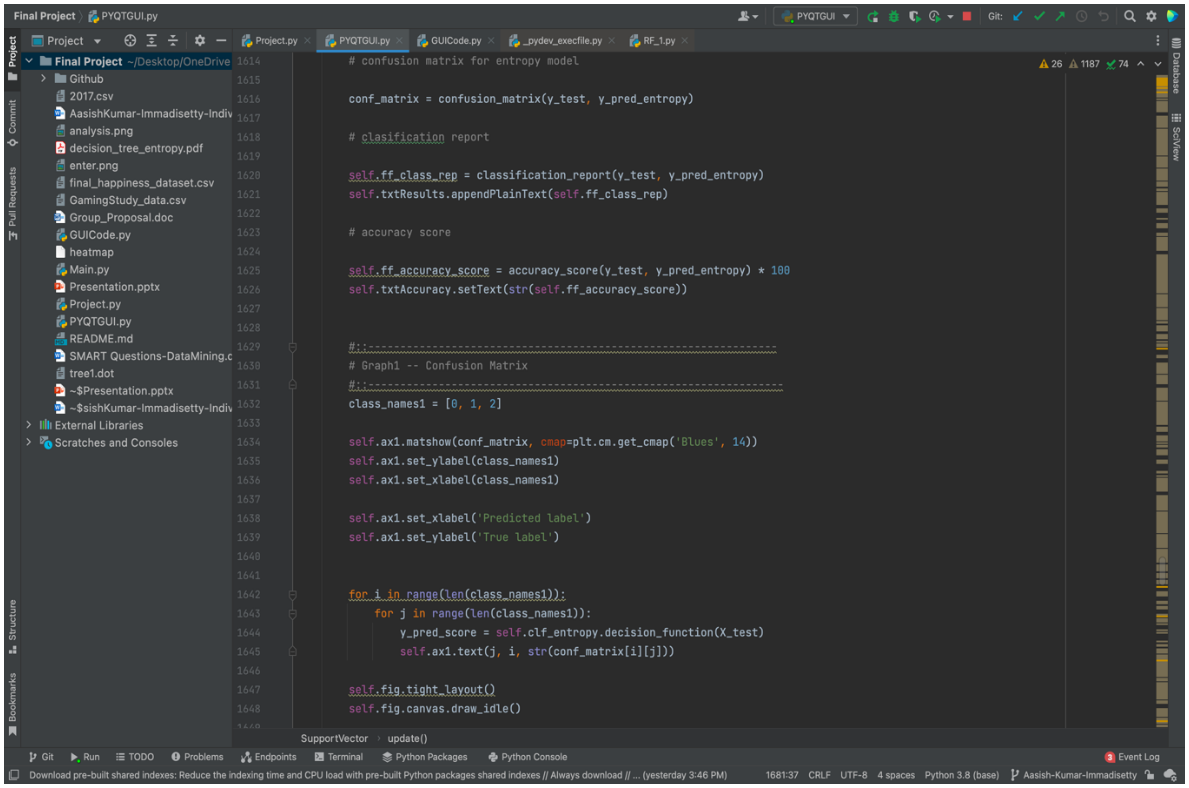


```
# We process the parameters
self.list_corr_features = pd.DataFrame([])
if self.feature0.isChecked():
    if len(self.list_corr_features)==0:
        self.list_corr_features = data[features_list[0]]
    else:
        self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[0]]],axis=1)

if self.feature1.isChecked():
    if len(self.list_corr_features) == 0:
        self.list_corr_features = data[features_list[1]]
    else:
        self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[1]]],axis=1)

if self.feature2.isChecked():
    if len(self.list_corr_features) == 0:
        self.list_corr_features = data[features_list[2]]
    else:
        self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[2]]],axis=1)

if self.feature3.isChecked():
    if len(self.list_corr_features) == 0:
        self.list_corr_features = data[features_list[3]]
    else:
        self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[3]]],axis=1)

if self.feature4.isChecked():
    if len(self.list_corr_features) == 0:
        self.list_corr_features = data[features_list[4]]
    else:
        self.list_corr_features = pd.concat([self.list_corr_features, data[features_list[4]]],axis=1)

if self.feature5.isChecked():
    if len(self.list_corr_features) == 0:
        self.list_corr_features = data[features_list[5]]
    else:
```



```
            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:
```

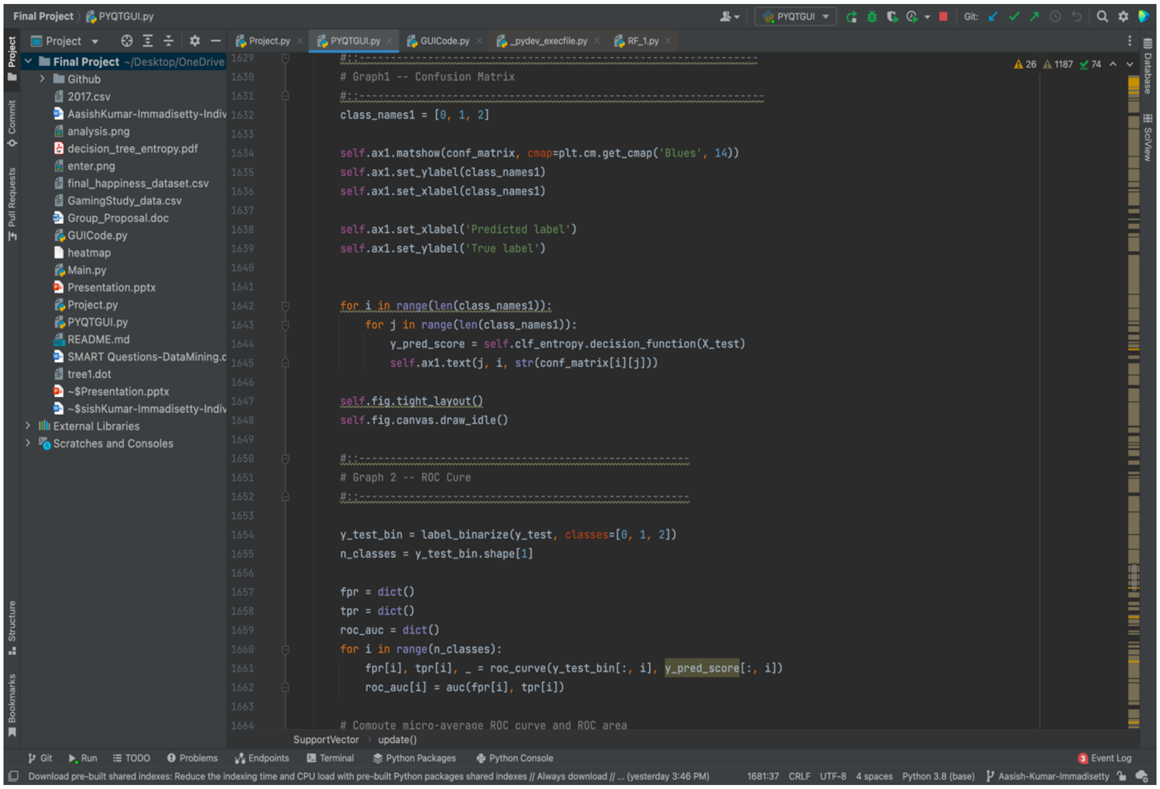
```
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // ... (yesterday 3:46 PM)    1681:37  CRLF  UTF-8  4 spaces  Python 3.8 (base)  Aasish-Kumar-Immadisetty
```



```
        # 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()
```

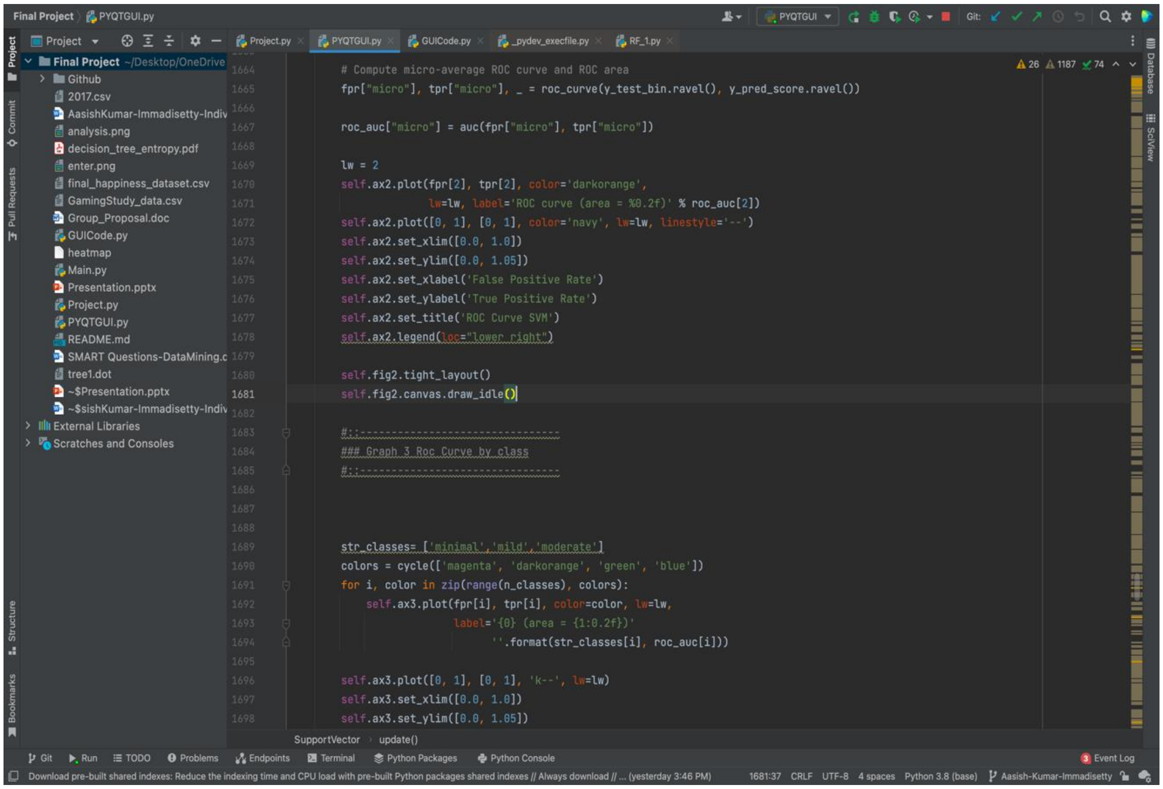
```
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // ... (yesterday 3:46 PM)    1681:37  CRLF  UTF-8  4 spaces  Python 3.8 (base)  Aasish-Kumar-Immadisetty
```

- The three images above show how data is put into X and Y variables for training, testing, and calculating roc, as well as visualizing roc, using Label Encoder and Label Binarizer.



```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
```

- The code in the image above can be used to generate a confusion matrix from the features chosen and plot the roc curve.



```
        # 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])
```

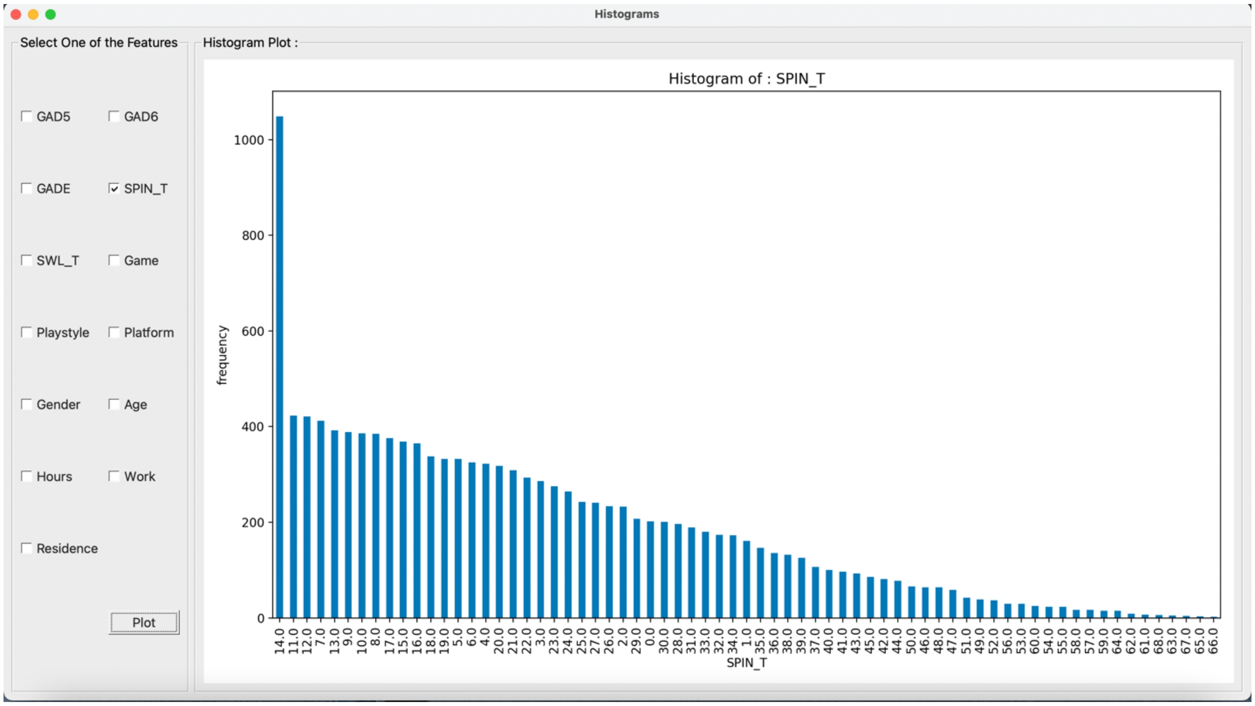
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // ... (yesterday 3:46 PM)

Because our target variable is multi-class, the above code will assist in plotting the roc curve for each class.

4. Results:

Histogram:

The graphic depicts the pictorial representation of the histogram as well as the numerous features available on the left grid, and the histogram is presented on the canvas after pressing the plot button.

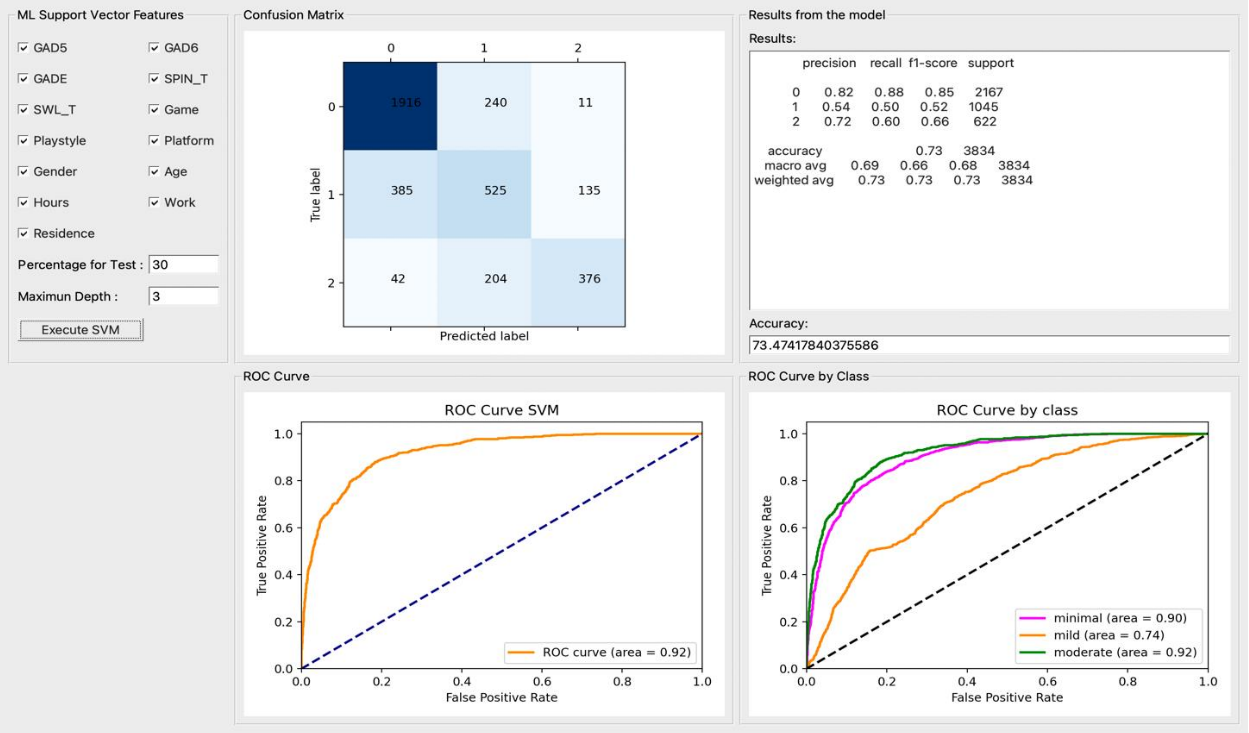


Model Building:

To display the roc curve and calculate the score, we first select all X-variables that are not target variables and load the target variable Y, which is the target variable and is label binarized as well as label encoded to convert labels to numeric values.

Support Vector Machine (SVM):

The image displays the SVM dashboard, and the user can manually change the percentage for test. The features can be selected as per user's choice and click on "Execute SVM" to run the model which displays confusion matrix, results from the model, ROC curve, ROC curve by each class.



ML Support Vector Features

☑ GAD5 ☑ GAD6

☑ GADE ☑ SPIN_T

☑ SWL_T ☑ Game

☑ Playstyle ☑ Platform

☑ Gender ☑ Age

☑ Hours ☑ Work

☑ Residence

Percentage for Test : 30

Maximun Depth : 3

Confusion Matrix

True label \ Predicted label	0	1	2
0	1916	240	11
1	385	525	135
2	42	204	376

Results from the model

Results:

```
          precision   recall  f1-score   support

   0        0.82       0.88     0.85      2167
   1        0.54       0.50     0.52      1045
   2        0.72       0.60     0.66       622

 accuracy                       0.73      3834
macro avg       0.69     0.66     0.68      3834
weighted avg    0.73     0.73     0.73      3834
```

Accuracy:

73.47417840375586

5. Summary and Conclusion:

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'

Conclusion:

- Since SVM has the highest accuracy and precision of the three possible classifiers, the Support Vector Machine is the best model. It also has the highest AUC of the three classifiers. In addition, we developed the model with the highest importance of features, making it the most efficient of the three classifiers in terms of accuracy.
- Overall, we can say that the Support Vector Machine model can accurately predict how online gaming will influence Generalized Anxiety Disorder with 73.47 percent accuracy.
- Using the dataset collected, we can investigate how online gaming affects other illnesses in the future.

6. Percent of Code:

The internet is utilized for only reference and none of the code is copied. Searching for Syntaxes to apply a function was done, but it was never downloaded from the internet. Al Of the code was written by hand, using skills learned through Prof. Amir Jafari’s Lectures and GitHub exercises.

7. References:

- 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.javatpoint.com/