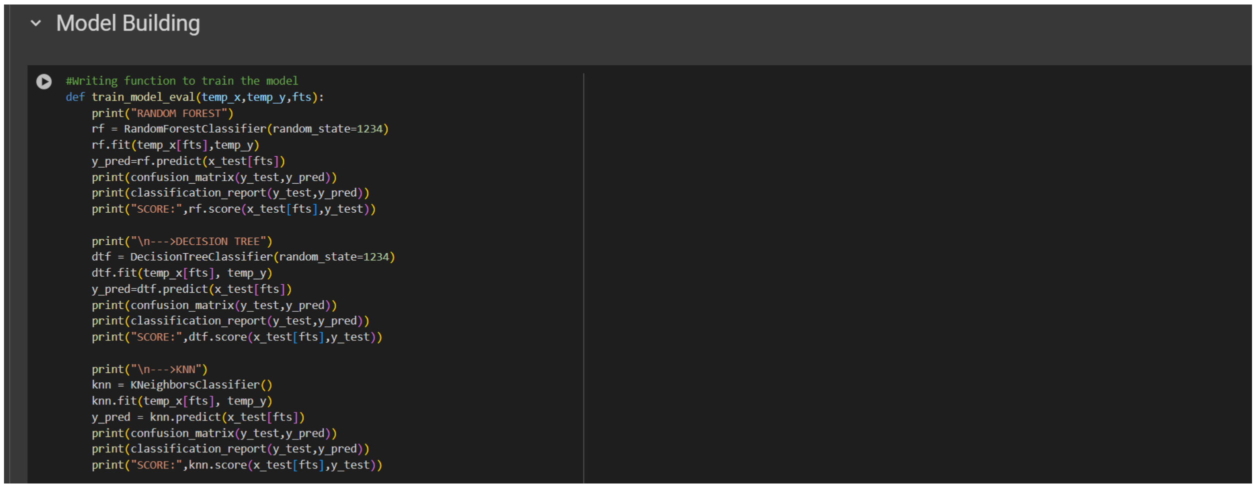
Model Development Phase Template

Date	15 July 2024
Team ID	739935
Project Title	Panic Disorder Detection
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

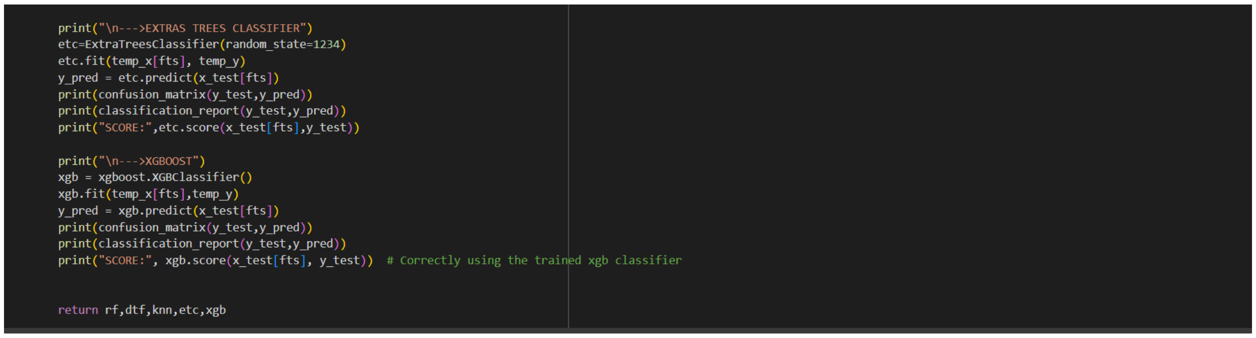
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

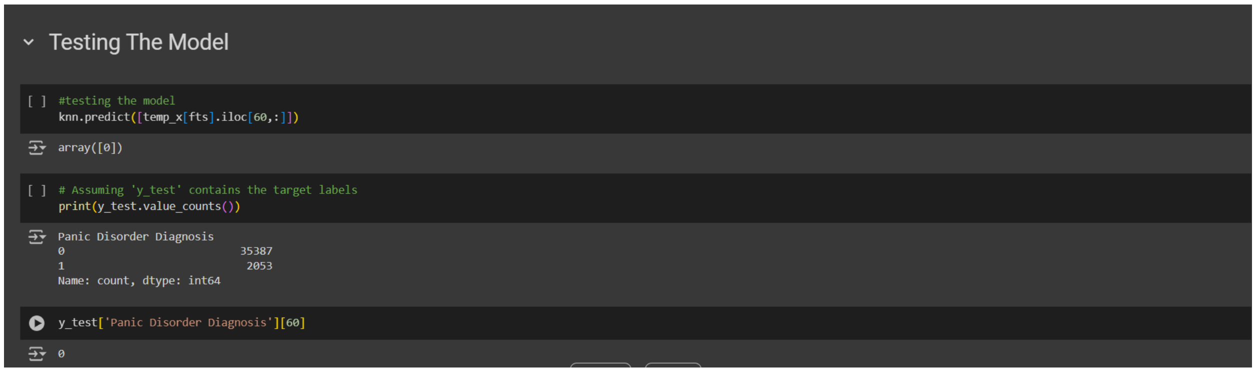


Model Building

```
#Writing function to train the model
def train_model_eval(temp_x,temp_y,fts):
    print("RANDOM FOREST")
    rf = RandomForestClassifier(random_state=1234)
    rf.fit(temp_x[fts],temp_y)
    y_pred=rf.predict(x_test[fts])
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print("SCORE:",rf.score(x_test[fts],y_test))

    print("\n--->DECISION TREE")
    dtf = DecisionTreeClassifier(random_state=1234)
    dtf.fit(temp_x[fts], temp_y)
    y_pred=dtf.predict(x_test[fts])
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print("SCORE:",dtf.score(x_test[fts],y_test))

    print("\n--->KNN")
    knn = KNeighborsClassifier()
    knn.fit(temp_x[fts], temp_y)
    y_pred = knn.predict(x_test[fts])
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print("SCORE:",knn.score(x_test[fts],y_test))
```



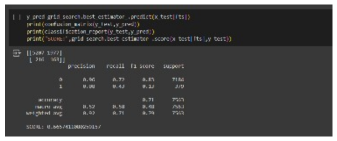
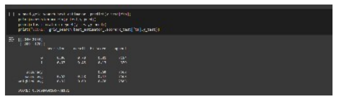
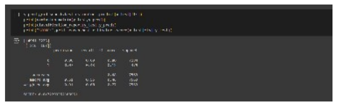
```
    print("\n--->EXTRAS TREES CLASSIFIER")
    etc=ExtraTreesClassifier(random_state=1234)
    etc.fit(temp_x[fts], temp_y)
    y_pred = etc.predict(x_test[fts])
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print("SCORE:",etc.score(x_test[fts],y_test))

    print("\n--->XGBOOST")
    xgb = xgboost.XGBClassifier()
    xgb.fit(temp_x[fts],temp_y)
    y_pred = xgb.predict(x_test[fts])
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print("SCORE:", xgb.score(x_test[fts], y_test))  # Correctly using the trained xgb classifier


    return rf,dtf,knn,etc,xgb
```



Testing The Model

```
[ ] #testing the model
    knn.predict([temp_x[fts].iloc[60,:]])

    array([0])

[ ] # Assuming 'y_test' contains the target labels
    print(y_test.value_counts())

    Panic Disorder Diagnosis
    0                            35387
    1                             2053
    Name: count, dtype: int64

    y_test['Panic Disorder Diagnosis'][60]

    0
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Decision Tree	[illegible]	0.7893	
Random Forest	[illegible]	0.7973	 [illegible]
XG Boost	[illegible]	0.77600	[illegible]
KNN	[illegible]	0.7499	 [illegible]

```
[ ] y_pred grid_search.best_estimator_.predict(x_test[fts])
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print("score:",grid_search.best_estimator_.score(x_test[fts],y_test))

    [[5207 1977]
     [ 216  163]]
                  precision    recall  f1 score   support

               0       0.96      0.72      0.83      7184
               1       0.08      0.43      0.13       379

        accuracy                           0.71      7563
       macro avg       0.52      0.58      0.48      7563
    weighted avg       0.92      0.71      0.79      7563

    SCORE: 0.6657411080259157
```