Model Development Phase Template

Date	10 JULY 2024
Team ID	SWTID1720151909
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:

1) Step 1: Data Preparation

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load the dataset
file_path = '/mnt/data/panic_disorder_dataset_testing.csv'
data = pd.read_csv(file_path)

# Selected features and target variable
selected_features = [
    'Age', 'Gender', 'Family History', 'Personal History',
    'Current Stressors', 'Symptoms', 'Severity', 'Impact on Life', 'Demographics',
    'Medical History', 'Psychiatric History', 'Substance Use', 'Coping Mechanisms',
    'Social Support', 'Lifestyle Factors'
]
target = 'Panic Disorder Diagnosis'

# Handle missing values if any (example: fill with mean or mode)
data = data.fillna(data.mean())

# Split the data into features and target
X = data[selected_features]
y = data[target]

# Encode categorical variables
X = pd.get_dummies(X, drop_first=True)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
```

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

2) TRAIN MULTIPLE MODELS

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

# Initialize the models
logistic_model = LogisticRegression()
random_forest_model = RandomForestClassifier()
svm_model = SVC()
knn_model = KNeighborsClassifier()
naive_bayes_model = GaussianNB()

# Train the models
logistic_model.fit(X_train, y_train)
random_forest_model.fit(X_train, y_train)
svm_model.fit(X_train, y_train)
knn_model.fit(X_train, y_train)
naive_bayes_model.fit(X_train, y_train
```

3) Step 3: Model Validation and Evaluation

```
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Function to evaluate the model
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
    print('Classification Report:')
    print(classification_report(y_test, y_pred))
    print('Confusion Matrix:')
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d')
    plt.show()

# Evaluate Logistic Regression
print("Logistic Regression")
evaluate_model(logistic_model, X_test, y_test)

# Evaluate Random Forest
print("Random Forest")
evaluate_model(random_forest_model, X_test, y_test)

# Evaluate SVM
print("Support Vector Machine")
evaluate_model(svm_model, X_test, y_test)

# Evaluate KNN
print("K-Nearest Neighbors")
evaluate_model(knn_model, X_test, y_test)

# Evaluate Naive Bayes
print("Naive Bayes")
evaluate_model(naive_bayes_model, X_test, y_test)
```

4) Step 4: Model Comparison

```
# Store accuracy results
model_names = ['Logistic Regression', 'Random Forest', 'SVM', 'KNN', 'Naive Bayes']
accuracies = [
    accuracy_score(y_test, logistic_model.predict(X_test)),
    accuracy_score(y_test, random_forest_model.predict(X_test)),
    accuracy_score(y_test, svm_model.predict(X_test)),
    accuracy_score(y_test, knn_model.predict(X_test)),
    accuracy_score(y_test, naive_bayes_model.predict(X_test))
]

# Create a DataFrame to display the results
results = pd.DataFrame({
    'Model': model_names,
    'Accuracy': accuracies
}).sort_values(by='Accuracy', ascending=False)

print("Model Comparison")
print(results)
```

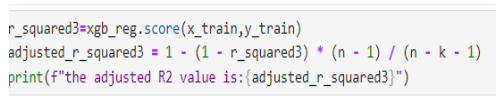
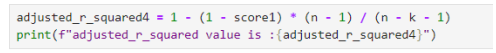
Model Validation and Evaluation Report:

Model	Classification Report (Adjusted R2 Score)	R2 score	Mean Squared Error (MSE) And Mean Absolute Error

Linear Regression	`adjusted_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - k - 1)`
	`print(adjusted_r_squared)`
	`0.0002331292697371756 3`
		0.00024	

Ridge Regression	`adjusted_r_squared1 = 1 - (1 - r_squared1) * (n - 1) / (n - k - 1)`
	`print(f"adjusted R squared value is:{adjusted_r_squared1}")`
	`adjusted R squared value is:0.0002331292697262954 4`
		0.00026	

Decision Tree Regressor	`r_squared2=tree_random.score(x_train,y_train)`
	`print(f"R2 score value is: {r_squared2}")`
	`adjusted_r_squared2 = 1 - (1 - r_squared2) * (n - 1) / (n - k - 1)`
	`print('adjusted R2 score value is :',adjusted_r_squared2)`
	`R2 score value is: 0.8565162068304613`
	`adjusted R2 score value is : 0.8565149385006436`
		0.75935	

XG boost Regression		0.32488	
Random Forest Regressor		0.98307	