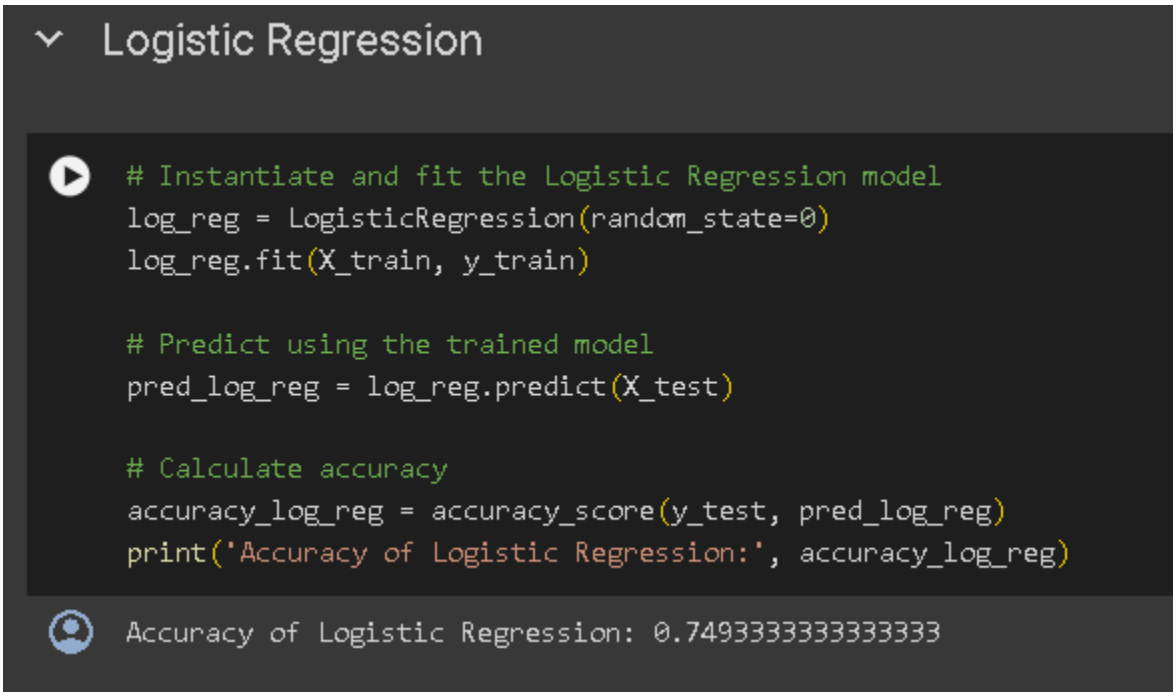
Model Development Phase Template

Date	15 March 2024
Team ID	738214
Project Title	Predicting Mental Health Illness Of Working Professionals Using Machine Learning.
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:



KNeighborsClassifier

```
[ ]  # Instantiate and fit the KNN model
     knn_classifier = KNeighborsClassifier()
     knn_classifier.fit(X_train, y_train)

     # Predict using the trained model
     pred_knn = knn_classifier.predict(X_test)

     # Calculate accuracy
     accuracy_knn = accuracy_score(y_test, pred_knn)
     print('Accuracy of K-Nearest Neighbors:', round(accuracy_knn,4)*100)
```

```
Accuracy of K-Nearest Neighbors: 65.07
```

DecisionTreeClassifier

```
[ ]  # Instantiate and fit the Decision Tree Classifier model
     dt_classifier = DecisionTreeClassifier(random_state=49)
     dt_classifier.fit(X_train, y_train)

     # Predict using the trained model
     pred_dt = dt_classifier.predict(X_test)

     # Calculate accuracy
     accuracy_dt = accuracy_score(y_test, pred_dt)
     print('Accuracy of Decision Tree Classifier:', round(accuracy_dt,4)*100)
```

```
Accuracy of Decision Tree Classifier: 69.87
```

Random Forest Classifier

```
[ ]  # Instantiate and fit the Random Forest model
     random_forest = RandomForestClassifier(random_state=49)
     random_forest.fit(X_train, y_train)

     # Predict using the trained model
     pred_rf = random_forest.predict(X_test)

     # Calculate accuracy
     accuracy_rf = accuracy_score(y_test, pred_rf)
     print('Accuracy of Random Forest Classifier:', round(accuracy_rf,4)*100)
```

```
Accuracy of Random Forest Classifier: 76.8
```

AdaBoost Classifier

```
[ ]  # Instantiate and fit the AdaBoost Classifier model
     adaboostClassifier = AdaBoostClassifier(random_state=49)
     adaboostClassifier.fit(X_train, y_train)

     # Predict using the trained model
     pred_abc = adaboostClassifier.predict(X_test)

     # Calculate accuracy
     accuracy_abc = accuracy_score(y_test, pred_abc)
     print('Accuracy of AdaBoost Classifier:', round(accuracy_abc,4)*100)
```

```
Accuracy of AdaBoost Classifier: 78.67
```

Gradient Boosting Classifier

```
# Instantiate and fit the Gradient Boosting model
gradientBoostingClassifier = GradientBoostingClassifier(random_state=49)
gradientBoostingClassifier.fit(X_train, y_train)

# Predict using the trained model
pred_gbc = gradientBoostingClassifier.predict(X_test)

# Calculate accuracy
accuracy_gbc = accuracy_score(y_test, pred_gbc)
print('Accuracy of Gradient Boosting Classifier:', round(accuracy_gbc,4)*100)
```

```
Accuracy of Gradient Boosting Classifier: 78.4
```

XGB Classifier

```
# Instantiate and fit the Random Forest model
xgb_classifier = XGBClassifier(random_state=49)
xgb_classifier.fit(X_train, y_train)

# Predict using the trained model
pred_xgb = xgb_classifier.predict(X_test)

# Calculate accuracy
accuracy_xgb = accuracy_score(y_test, pred_xgb)
print('Accuracy of XGB Classifier:', round(accuracy_xgb,4)*100)
```

```
Accuracy of XGB Classifier: 72.53
```

Model Validation and Evaluation Report:

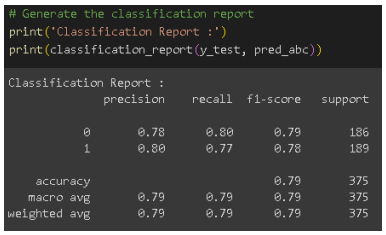
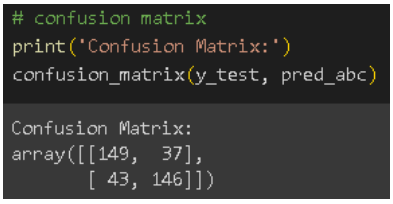
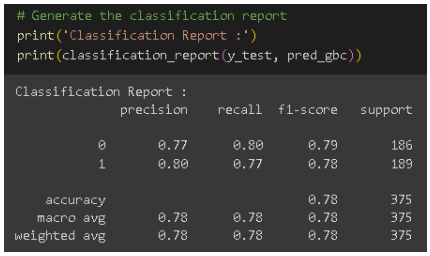
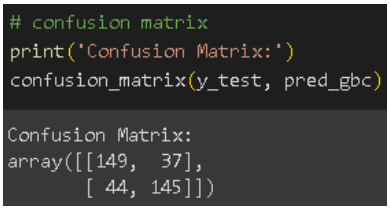
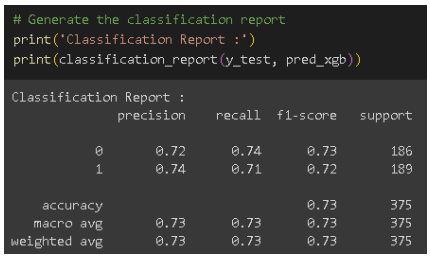
Model	Classification Report	Accuracy	Confusion Matrix

Logistic Regression	`# Generate the classification report`
	`print('Classification Report :')`
	`print(classification_report(y_test, pred_log_reg))`
	
	Classification Report :
	precision recall f1-score support
	0 0.73 0.78 0.76 186
	1 0.77 0.71 0.74 189
	accuracy 0.75 375
	macro avg 0.75 0.75 0.75 375
	weighted avg 0.75 0.75 0.75 375
		74.93%	`# confusion matrix`
			`print('Confusion Matrix:')`
			`print(confusion_matrix(y_test, pred_log_reg))`
			
			Confusion Matrix:
			[[146 40]
			[54 135]]

Kneighbors Classifier	`# Generate the classification report`
	`print('Classification Report :')`
	`print(classification_report(y_test, pred_knn))`
	
	Classification Report :
	precision recall f1-score support
	0 0.63 0.72 0.67 186
	1 0.68 0.59 0.63 189
	accuracy 0.65 375
	macro avg 0.65 0.65 0.65 375
	weighted avg 0.65 0.65 0.65 375
		65.07%	`# confusion matrix`
			`print('Confusion Matrix:')`
			`confusion_matrix(y_test, pred_knn)`
			
			Confusion Matrix:
			array([[133, 53],
			[78, 111]])

Decision Tree Classifier	`# Generate the classification report`
	`print('Classification Report :')`
	`print(classification_report(y_test, pred_dt))`
	
	Classification Report :
	precision recall f1-score support
	0 0.68 0.74 0.71 186
	1 0.72 0.66 0.69 189
	accuracy 0.70 375
	macro avg 0.70 0.70 0.70 375
	weighted avg 0.70 0.70 0.70 375
		69.87%	`# confusion matrix`
			`print('Confusion Matrix:')`
			`confusion_matrix(y_test, pred_dt)`
			
			Confusion Matrix:
			array([[138, 48],
			[65, 124]])

Random Forest Classifier	`# Generate the classification report`
	`print('Classification Report :')`
	`print(classification_report(y_test, pred_rf))`
	
	Classification Report :
	precision recall f1-score support
	0 0.75 0.79 0.77 186
	1 0.78 0.75 0.76 189
	accuracy 0.77 375
	macro avg 0.77 0.77 0.77 375
	weighted avg 0.77 0.77 0.77 375
		76.8%	`# confusion matrix`
			`print('Confusion Matrix:')`
			`confusion_matrix(y_test, pred_rf)`
			
			Confusion Matrix:
			array([[147, 39],
			[48, 141]])

AdaBoost Classifier		78.67%	
Gradient Boosting Classifier		78.4%	
XGB Classifier		72.53%