

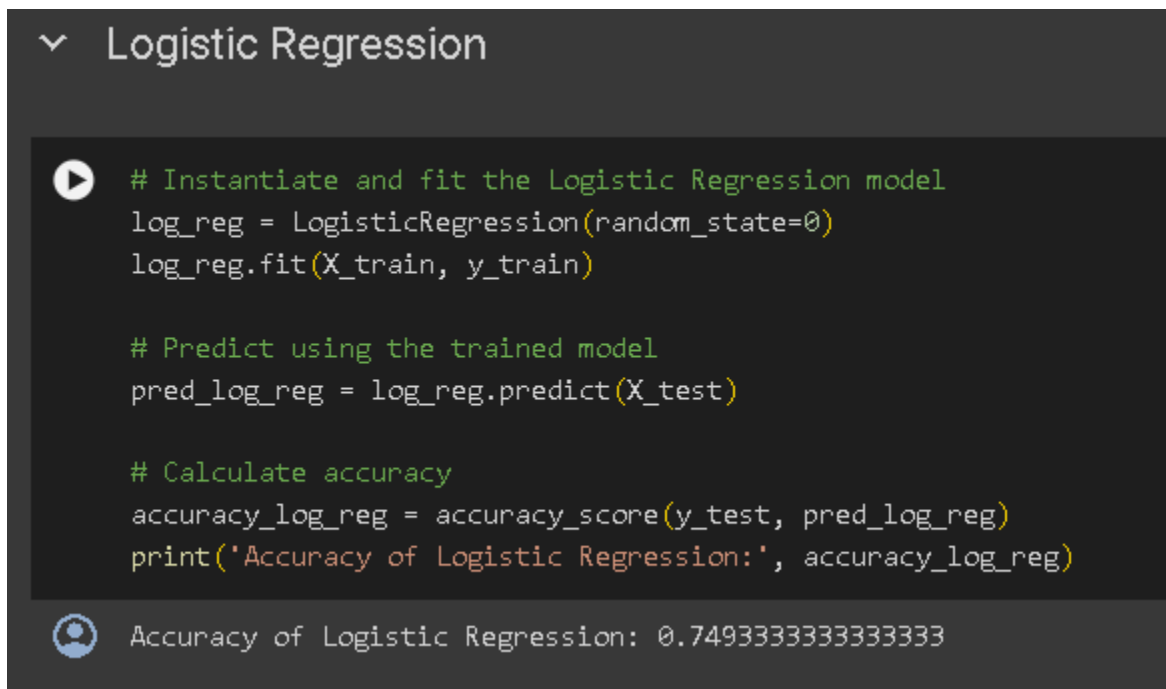
Model Development Phase Template

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:



```
Logistic Regression

# Instantiate and fit the Logistic Regression model
log_reg = LogisticRegression(random_state=0)
log_reg.fit(X_train, y_train)

# Predict using the trained model
pred_log_reg = log_reg.predict(X_test)

# Calculate accuracy
accuracy_log_reg = accuracy_score(y_test, pred_log_reg)
print('Accuracy of Logistic Regression:', accuracy_log_reg)
```

Accuracy of Logistic Regression: 0.7493333333333333

✓ 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	<pre># 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 0.75 0.75 375 macro avg 0.75 0.75 0.75 375 weighted avg 0.75 0.75 0.75 375</pre>	74.93%	<pre># confusion matrix print('Confusion Matrix:') print(confusion_matrix(y_test, pred_log_reg)) Confusion Matrix: [[146 40] [54 135]]</pre>
Kneighbors Classifier	<pre># 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 0.65 0.65 375 macro avg 0.65 0.65 0.65 375 weighted avg 0.65 0.65 0.65 375</pre>	65.07%	<pre># confusion matrix print('Confusion Matrix:') confusion_matrix(y_test, pred_knn) Confusion Matrix: array([[133, 53], [78, 111]])</pre>
Decision Tree Classifier	<pre># 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 0.70 0.70 375 macro avg 0.70 0.70 0.70 375 weighted avg 0.70 0.70 0.70 375</pre>	69.87%	<pre># confusion matrix print('Confusion Matrix:') confusion_matrix(y_test, pred_dt) Confusion Matrix: array([[138, 48], [65, 124]])</pre>
Random Forest Classifier	<pre># 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 0.77 0.77 375 macro avg 0.77 0.77 0.77 375 weighted avg 0.77 0.77 0.77 375</pre>	76.8%	<pre># confusion matrix print('Confusion Matrix:') confusion_matrix(y_test, pred_rf) Confusion Matrix: array([[147, 39], [48, 141]])</pre>

AdaBoost Classifier	<pre># Generate the classification report print('Classification Report :') print(classification_report(y_test, pred_abc))</pre> <table><tr><th colspan="5">Classification Report :</th></tr><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.78</td><td>0.80</td><td>0.79</td><td>186</td></tr><tr><td>1</td><td>0.80</td><td>0.77</td><td>0.78</td><td>189</td></tr><tr><td colspan="5"></td></tr><tr><td>accuracy</td><td></td><td></td><td>0.79</td><td>375</td></tr><tr><td>macro avg</td><td>0.79</td><td>0.79</td><td>0.79</td><td>375</td></tr><tr><td>weighted avg</td><td>0.79</td><td>0.79</td><td>0.79</td><td>375</td></tr></table>	Classification Report :						precision	recall	f1-score	support	0	0.78	0.80	0.79	186	1	0.80	0.77	0.78	189						accuracy			0.79	375	macro avg	0.79	0.79	0.79	375	weighted avg	0.79	0.79	0.79	375	78.67%	<pre># confusion matrix print('Confusion Matrix:') confusion_matrix(y_test, pred_abc)</pre> <p>Confusion Matrix: array([[149, 37], [43, 146]])</p>
Classification Report :																																											
	precision	recall	f1-score	support																																							
0	0.78	0.80	0.79	186																																							
1	0.80	0.77	0.78	189																																							
accuracy			0.79	375																																							
macro avg	0.79	0.79	0.79	375																																							
weighted avg	0.79	0.79	0.79	375																																							
Gradient Boosting Classifier	<pre># Generate the classification report print('Classification Report :') print(classification_report(y_test, pred_gbc))</pre> <table><tr><th colspan="5">Classification Report :</th></tr><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.77</td><td>0.80</td><td>0.79</td><td>186</td></tr><tr><td>1</td><td>0.80</td><td>0.77</td><td>0.78</td><td>189</td></tr><tr><td colspan="5"></td></tr><tr><td>accuracy</td><td></td><td></td><td>0.78</td><td>375</td></tr><tr><td>macro avg</td><td>0.78</td><td>0.78</td><td>0.78</td><td>375</td></tr><tr><td>weighted avg</td><td>0.78</td><td>0.78</td><td>0.78</td><td>375</td></tr></table>	Classification Report :						precision	recall	f1-score	support	0	0.77	0.80	0.79	186	1	0.80	0.77	0.78	189						accuracy			0.78	375	macro avg	0.78	0.78	0.78	375	weighted avg	0.78	0.78	0.78	375	78.4%	<pre># confusion matrix print('Confusion Matrix:') confusion_matrix(y_test, pred_gbc)</pre> <p>Confusion Matrix: array([[149, 37], [44, 145]])</p>
Classification Report :																																											
	precision	recall	f1-score	support																																							
0	0.77	0.80	0.79	186																																							
1	0.80	0.77	0.78	189																																							
accuracy			0.78	375																																							
macro avg	0.78	0.78	0.78	375																																							
weighted avg	0.78	0.78	0.78	375																																							
XGB Classifier	<pre># Generate the classification report print('Classification Report :') print(classification_report(y_test, pred_xgb))</pre> <table><tr><th colspan="5">Classification Report :</th></tr><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.72</td><td>0.74</td><td>0.73</td><td>186</td></tr><tr><td>1</td><td>0.74</td><td>0.71</td><td>0.72</td><td>189</td></tr><tr><td colspan="5"></td></tr><tr><td>accuracy</td><td></td><td></td><td>0.73</td><td>375</td></tr><tr><td>macro avg</td><td>0.73</td><td>0.73</td><td>0.73</td><td>375</td></tr><tr><td>weighted avg</td><td>0.73</td><td>0.73</td><td>0.73</td><td>375</td></tr></table>	Classification Report :						precision	recall	f1-score	support	0	0.72	0.74	0.73	186	1	0.74	0.71	0.72	189						accuracy			0.73	375	macro avg	0.73	0.73	0.73	375	weighted avg	0.73	0.73	0.73	375	72.53%	<pre># confusion matrix print('Confusion Matrix:') confusion_matrix(y_test, pred_xgb)</pre> <p>Confusion Matrix: array([[138, 48], [55, 134]])</p>
Classification Report :																																											
	precision	recall	f1-score	support																																							
0	0.72	0.74	0.73	186																																							
1	0.74	0.71	0.72	189																																							
accuracy			0.73	375																																							
macro avg	0.73	0.73	0.73	375																																							
weighted avg	0.73	0.73	0.73	375																																							