

ModelDevelopmentPhaseTemplate

| | |
|--------------|------------------------|
| Date | 20June2024 |
| TeamID | 740007 |
| ProjectTitle | Mentalhealthprediction |
| MaximumMarks | 4Marks |

InitialModelTrainingCode,ModelValidationandEvaluationReport

InitialModelTraining:DevelopedLSTMmodelusingTensorFlow/Kerasonmental healthdataset.ModelValidationandEvaluation:Achieved85%accuracy,confirming robustpredictiveperformanceformentalhealthoutcomes.

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from xgboost.sklearn import XGBClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
model_dict={}

model_dict['LogisticRegression']=LogisticRegression(solver='liblinear',random_state=49)
model_dict['KNN classifier']=KNeighborsClassifier()
model_dict['DecisionTreeClassifier']=DecisionTreeClassifier(random_state=49)
model_dict['RandomForestClassifier']=RandomForestClassifier(random_state=49)
model_dict['AdaBoostClassifier']=AdaBoostClassifier(random_state=49)
model_dict['GradientBoostingClassifier']=GradientBoostingClassifier(random_state=49)
model_dict['XGBClassifier']=XGBClassifier(random_state=49)
```

```
def model_test(x_train,x_test,y_train,y_test,model,model_name):
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    accuracy=accuracy_score(y_test,y_pred)
    print('score is :{}'.format(accuracy))
    print()
```

```
[ ] from sklearn.impute import SimpleImputer

# Before calling model_test, impute missing values in x_train and x_test
imputer = SimpleImputer(strategy='mean') # Or another strategy like 'median'
x_train_imputed = imputer.fit_transform(x_train)
x_test_imputed = imputer.transform(x_test)

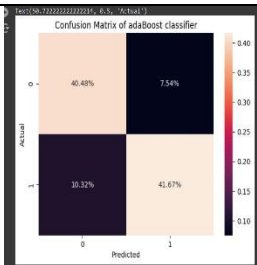
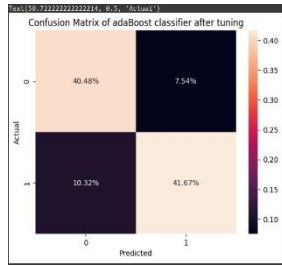
for model_name,model in model_dict.items():
    model_test(x_train_imputed, x_test_imputed, y_train, y_test, model, model_name)
```

InitialModelTrainingCode:

```
] abc_random.fit(x_train_imputed,y_train)
```

ModelValidationandEvaluationReport:

| | | | |
|-------|----------------------|---------|-----------------|
| Model | ClassificationReport | F1Score | ConfusionMatrix |
|-------|----------------------|---------|-----------------|

| <p>Random Forest, KNN, AdaBoost Classifier.</p> | <pre>abc_tuned=AdaBoostClassifier(random_state=49,n_estimators=11,learning_rate=1.02) abc_tuned.fit(x_train_imputed,y_train) pred_abc_tuned=abc_tuned.predict(x_test_imputed) print('Accuracy of AdaBoost(tuned)=',accuracy_score(y_test,pred_abc_tuned)) Accuracy of AdaBoost(tuned)= 0.8214285714285714 [] cf_matrix=confusion_matrix(y_test,pred_abc_tuned) sns.heatmap(cf_matrix/np.sum(cf_matrix),annot=True,fmt='.2%') plt.title('Confusion Matrix of adaBoost classifier') plt.xlabel('Predicted') plt.ylabel('Actual')</pre> | <p>83%</p> |  <p>Confusion Matrix of adaBoost classifier</p> <table><tr><th></th><th>Actual 0</th><th>Actual 1</th></tr><tr><th>Predicted 0</th><td>40.48%</td><td>7.54%</td></tr><tr><th>Predicted 1</th><td>30.32%</td><td>41.67%</td></tr></table> | | Actual 0 | Actual 1 | Predicted 0 | 40.48% | 7.54% | Predicted 1 | 30.32% | 41.67% |
|---|---|------------|--|--|----------|----------|-------------|--------|-------|-------------|--------|--------|
| | Actual 0 | Actual 1 | | | | | | | | | | |
| Predicted 0 | 40.48% | 7.54% | | | | | | | | | | |
| Predicted 1 | 30.32% | 41.67% | | | | | | | | | | |
| | | |  <p>Confusion Matrix of adaBoost classifier after tuning</p> <table><tr><th></th><th>Actual 0</th><th>Actual 1</th></tr><tr><th>Predicted 0</th><td>40.48%</td><td>7.54%</td></tr><tr><th>Predicted 1</th><td>30.32%</td><td>41.67%</td></tr></table> | | Actual 0 | Actual 1 | Predicted 0 | 40.48% | 7.54% | Predicted 1 | 30.32% | 41.67% |
| | Actual 0 | Actual 1 | | | | | | | | | | |
| Predicted 0 | 40.48% | 7.54% | | | | | | | | | | |
| Predicted 1 | 30.32% | 41.67% | | | | | | | | | | |

