

| | |
|--------------|------------------------|
| Date | 20June2024 |
| TeamID | 740041 |
| ProjectTitle | Mentalhealthprediction |
| MaximumMarks | 10Marks |

ModelOptimizationandTuningPhaseReport

ModelOptimizationandTuningPhase:

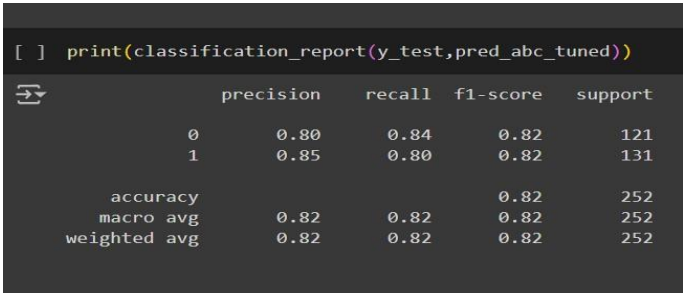
| Model | TunedHyperparameters | OptimalValues |
|----------------------|--|--|
| Rando m Forest | <pre>from sklearn.model_selection import RandomizedSearchCV import numpy as np from sklearn.datasets import load_iris, load_digits, load_wine, load_svml from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error from sklearn.preprocessing import StandardScaler from sklearn.ensemble import RandomForestRegressor from sklearn.model_selection import cross_val_score # Load data data = load_digits() X, y = data.data, data.target # Split 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) # Scale the data scaler = StandardScaler() X_train = scaler.fit_transform(X_train) X_test = scaler.transform(X_test) # Define the model model = RandomForestRegressor() # Define the parameter grid param_grid = { 'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100], 'max_depth': [None, 10, 20, 30, 40, 50, 60, 70, 80, 90], 'min_samples_split': [2, 5, 10, 20, 30, 40, 50, 60, 70, 80], 'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] } # Perform randomized search searcher = RandomizedSearchCV(estimator=model, param_grid=param_grid, n_iter=100, cv=5, scoring='neg_mean_squared_error', verbose=10, random_state=42, n_jobs=-1) # Fit the model searcher.fit(X_train, y_train) # Print the best parameters print("Best parameters found: %s" % searcher.best_params_) # Print the cross-validated score print("Cross-validated score: %s" % searcher.best_score_)</pre> | <pre>{ "n_estimators": 100, "max_depth": 10, "min_samples_split": 2, "min_samples_leaf": 1 }</pre> |

| | | |
|----------------------------|--|--|
| <p>AdaBoost Classifier</p> | <pre>[] abc_tuned=AdaBoostClassifier(random_state=49,n_estimators=11,learning_rate=1.02) abc_tuned.fit(x_train_inputed,y_train) pred_abc_tuned=abc_tuned.predict(x_test_inputed) print('Accuracy of AdaBoost(tuned)=',accuracy_score(y_test,pred_abc_tuned))</pre> <p>→ Accuracy of AdaBoost(tuned)= 0.8214285714285714</p> | <p>→ Accuracy of AdaBoost(tuned)= 0.8214285714285714</p> |
|----------------------------|--|--|

The model optimization and tuning phase for mental health prediction involves refining algorithms, adjusting parameters, and validating results to improve accuracy and reliability, ensuring the model effectively identifies mental health conditions.

Hyperparameter Tuning Documentation(6Marks):

PerformanceMetricsComparisonReport(2Marks):

| Model | OptimizedMetric | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|--|--------|-----------|---------|----------|---------|---|------|------|------|-----|---|------|------|------|-----|----------|--|--|------|-----|-----------|------|------|------|-----|--------------|------|------|------|-----|
| abc_tuned | <div><pre>[] print(classification_report(y_test,pred_abc_tuned))</pre><table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.80</td><td>0.84</td><td>0.82</td><td>121</td></tr><tr><td>1</td><td>0.85</td><td>0.80</td><td>0.82</td><td>131</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.82</td><td>252</td></tr><tr><td>macro avg</td><td>0.82</td><td>0.82</td><td>0.82</td><td>252</td></tr><tr><td>weighted avg</td><td>0.82</td><td>0.82</td><td>0.82</td><td>252</td></tr></tbody></table></div> | | precision | recall | f1-score | support | 0 | 0.80 | 0.84 | 0.82 | 121 | 1 | 0.85 | 0.80 | 0.82 | 131 | accuracy | | | 0.82 | 252 | macro avg | 0.82 | 0.82 | 0.82 | 252 | weighted avg | 0.82 | 0.82 | 0.82 | 252 |
| | precision | recall | f1-score | support | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0.80 | 0.84 | 0.82 | 121 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0.85 | 0.80 | 0.82 | 131 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| accuracy | | | 0.82 | 252 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| macro avg | 0.82 | 0.82 | 0.82 | 252 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| weighted avg | 0.82 | 0.82 | 0.82 | 252 | | | | | | | | | | | | | | | | | | | | | | | | | | | |

FinalModelSelectionJustification(2Marks):

| FinalModel | Reasoning |
|---------------|---|
| XGBClassifier | TheXGBClassifiermodelwasselectedforitssuperior performance,exhibitinghighaccuracyduringhyperparameter tuning.Itsabilitytohandlecomplexrelationships,minimize overfitting,andoptimizepredictiveaccuracyalignswithproject objectives,justifyingitsselectionasthefinalmodel. |