

Model Optimization and Tuning Phase

Date	10 July 2024
Team ID	SWTID1720078683
Project Title	Anemia Sense: Leveraging Machine Learning for Precise Anemia Recognitions
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values
Decision Tree	<pre>[44]: from sklearn.tree import DecisionTreeClassifier from sklearn.model_selection import RandomizedSearchCV [45]: dec = DecisionTreeClassifier() [46]: param_grid = { 'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max_depth': [None, 10, 20, 30, 40, 50], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4] } [47]: dec = RandomizedSearchCV(dec, param_grid, cv=5) [48]: dec.fit(x_train, y_train) [49]: RandomizedSearchCV estimator: DecisionTreeClassifier DecisionTreeClassifier</pre>	<pre>RandomizedSearchCV Results: Best Score: 0.95 Best Parameters: {'criterion': 'entropy', 'max_depth': 30, 'min_samples_split': 5, 'min_samples_leaf': 4, 'splitter': 'best'}</pre>
Random Forest	<pre>from sklearn.ensemble import RandomForestClassifier from sklearn.model_selection import RandomizedSearchCV rf = RandomForestClassifier() param_grid = { 'n_estimators': [50, 100, 200], 'criterion': ['gini', 'entropy'], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], } rf = RandomizedSearchCV(rf, param_grid, cv=5) rf.fit(x_train, y_train) RandomizedSearchCV estimator: RandomForestClassifier RandomForestClassifier</pre>	<pre>RandomizedSearchCV Results: Best Score: 0.98 Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 5, 'min_samples_leaf': 4, 'n_estimators': 200}</pre>

Gradient Boosting	<pre> from sklearn.ensemble import GradientBoostingClassifier from sklearn.model_selection import RandomizedSearchCV GB = GradientBoostingClassifier() param_grid = { 'n_estimators': [50, 100, 200], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 4, 5], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'subsample': [0.8, 1.0] } GB = RandomizedSearchCV(GB, param_grid, cv=5) GB.fit(x_train,y_train) RandomizedSearchCV - estimator: GradientBoostingClassifier - GradientBoostingClassifier </pre>	<pre> print('Test accuracy: %.3f'%GB.score(x_test,y_test)) print('Test accuracy of best: %.3f'%GB.best_score_) Best parameters: {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100, 'subsample': 0.8} Best accuracy on test: 0.8 </pre>
-------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Decision Tree	<pre> rep_dc = classification_report(y_test,y_predict) print(rep_dc) precision recall f1-score support 0 1.00 1.00 1.00 123 1 1.00 1.00 1.00 125 accuracy 1.00 1.00 1.00 248 macro avg 1.00 1.00 1.00 248 weighted avg 1.00 1.00 1.00 248 confusion_matrix(y_test,y_predict) array([[123, 0], [0, 125]], dtype=int64) </pre>
Random Forest	<pre> rep_rf = classification_report(y_test,y_predict) print(rep_rf) precision recall f1-score support 0 1.00 1.00 1.00 123 1 1.00 1.00 1.00 125 accuracy 1.00 1.00 1.00 248 macro avg 1.00 1.00 1.00 248 weighted avg 1.00 1.00 1.00 248 confusion_matrix(y_test,y_predict) array([[123, 0], [0, 125]], dtype=int64) </pre>
Gradient Boosting	<pre> rep_GB = classification_report(y_test,y_predict) print(rep_GB) precision recall f1-score support 0 1.00 1.00 1.00 123 1 1.00 1.00 1.00 125 accuracy 1.00 1.00 1.00 248 macro avg 1.00 1.00 1.00 248 weighted avg 1.00 1.00 1.00 248 confusion_matrix(y_test,y_predict) array([[123, 0], [0, 125]], dtype=int64) </pre>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Gradient Boosting	The Gradient Boosting model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model