

Model Optimization and Tuning Phase Report

Date	16th June 2025
Team ID	SWTID1749621188
Project Title	Anemia Sense Leveraging-Machine Learning For-Precise Anemia Recognition
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
Logistic Regression	<pre># 1. Logistic Regression model lr_classifier = LogisticRegression() # Define the hyperparameters and their possible values for tuning param_grid = { 'penalty': ['l1', 'l2', 'elasticnet', None], 'C': [0.01, 0.1, 1, 10, 100], 'solver': ['lbfgs', 'liblinear', 'saga'], 'max_iter': [100, 200, 500] }</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: {best_params}') print(f'Accuracy on Test Set: {accuracy}')</pre>
Naïve Bayes	<pre># 2. Naive Bayes classifier nb_classifier = GaussianNB() # Define the hyperparameters and their possible values for tuning param_grid = { 'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5] }</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: {best_params}') print(f'Accuracy on Test Set: {accuracy}')</pre>

Random Forest Classifier	<pre># 3. Random Forest Classifier rf_classifier = RandomForestClassifier() # Define the hyperparameters and their possible values for tuning param_grid = { 'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False] }</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: {best_params}') print(f'Accuracy on Test Set: {accuracy}')</pre>
SVM	<pre># 4. SVM classifier svm_classifier = SVC() # Define the hyperparameters and their possible values for tuning param_grid = { 'C': [0.1, 1, 10, 100], 'kernel': ['linear', 'rbf', 'poly', 'sigmoid'], 'gamma': ['scale', 'auto'], 'degree': [2, 3, 4] }</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: {best_params}') print(f'Accuracy on Test Set: {accuracy}')</pre>
Gradient Boosting	<pre># 5. Gradient Boosting classifier gb_classifier = GradientBoostingClassifier() # Define the hyperparameters and their possible values for tuning param_grid = { 'n_estimators': [50, 100, 200], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 5, 7], 'subsample': [0.6, 0.8, 1.0], 'min_samples_split': [2, 5, 10] }</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: {best_params}') print(f'Accuracy on Test Set: {accuracy}')</pre>
Lasso	<pre># 6. Lasso Logistic Regression model lasso_classifier = LogisticRegression(penalty='l1', solver='saga') # Define the hyperparameters and their possible values for tuning param_grid = { 'C': [0.01, 0.1, 1, 10, 100], 'max_iter': [100, 200, 500], 'tol': [1e-4, 1e-3, 1e-2] }</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: {best_params}') print(f'Accuracy on Test Set: {accuracy}')</pre>
Decision Tree Classifier	<pre># 7. Decision Tree classifier dt_classifier = DecisionTreeClassifier() # Define the hyperparameters and their possible values for tuning 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] }</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: {best_params}') print(f'Accuracy on Test Set: {accuracy}')</pre>

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Decision Tree	<pre> Confusion Matrix: [[157 0] [0 128]] Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 157 1 1.00 1.00 1.00 128 accuracy 1.00 macro avg 1.00 weighted avg 1.00 </pre>
Random Forest	<pre> Confusion Matrix: [[157 0] [0 128]] Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 157 1 1.00 1.00 1.00 128 accuracy 1.00 macro avg 1.00 weighted avg 1.00 </pre>
Naïve Bayes	<pre> Confusion Matrix: [[150 7] [7 121]] Classification Report: precision recall f1-score support 0 0.96 0.96 0.96 157 1 0.95 0.95 0.95 128 accuracy 0.95 macro avg 0.95 weighted avg 0.95 </pre>
Gradient Boosting	<pre> Confusion Matrix: [[157 0] [0 128]] Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 157 1 1.00 1.00 1.00 128 accuracy 1.00 macro avg 1.00 weighted avg 1.00 </pre>

Logistic Regression	<pre> Confusion Matrix: [[154 3] [0 128]] Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 157 1 0.98 1.00 0.99 128 accuracy 0.99 macro avg 0.99 weighted avg 0.99 </pre>
SVM	<pre> Confusion Matrix: [[152 5] [0 128]] Classification Report: precision recall f1-score support 0 1.00 0.97 0.98 157 1 0.96 1.00 0.98 128 accuracy 0.98 macro avg 0.98 weighted avg 0.98 </pre>
Lasso	<pre> Confusion Matrix: [[155 2] [0 128]] Classification Report: precision recall f1-score support 0 1.00 0.99 0.99 157 1 0.98 1.00 0.99 128 accuracy 0.99 macro avg 0.99 weighted avg 0.99 </pre>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	The Random Forest 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.