Model Optimization and Tuning Phase Report

Date	9 March 2025
Skillwallet ID	SWUID20250188620
Project Title	Anemia Sense: Leveraging Machine Learning for Precise Anemia Recognition
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

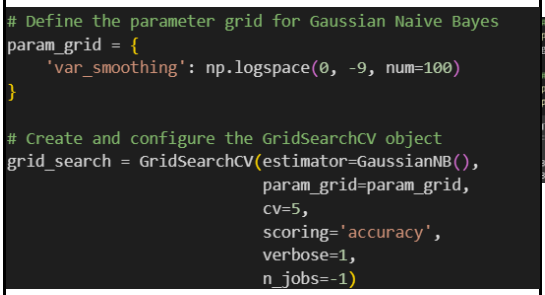
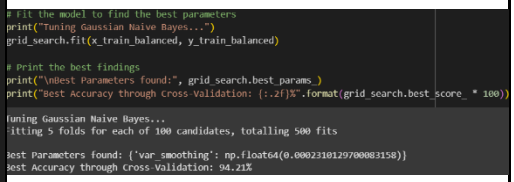
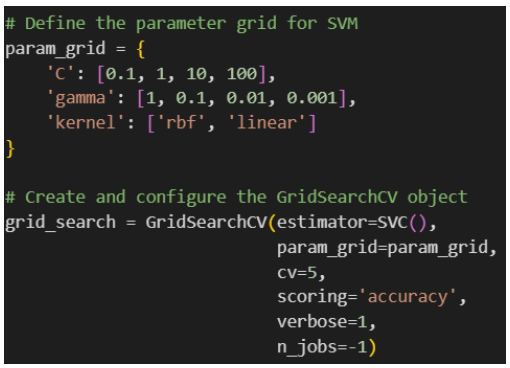
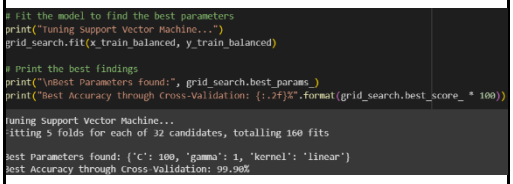
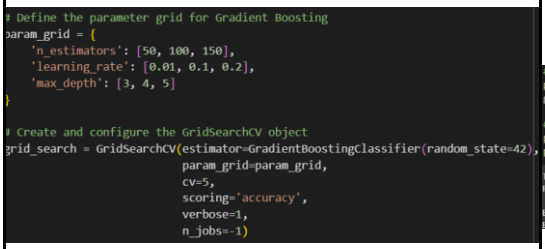
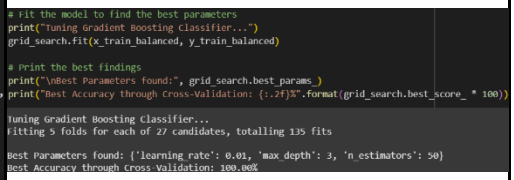
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

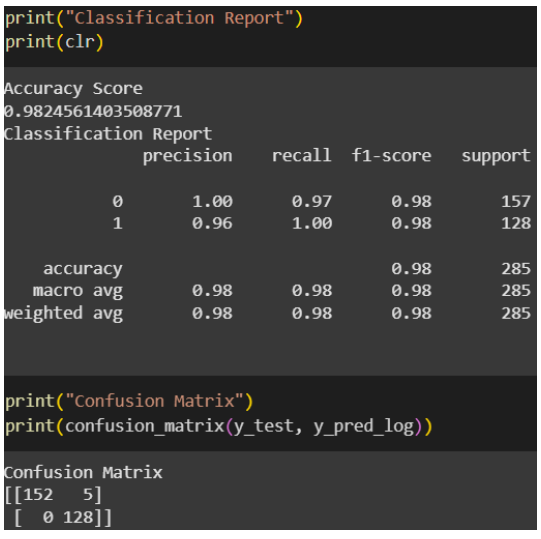
Model	Tuned Hyperparameters	Optimal Values

Logistic Regression	`# Define the parameter grid for Logistic Regression`
	`param_grid = {`
	`'C': [0.1, 1, 10, 100],`
	`'penalty': ['l1', 'l2'],`
	`'solver': ['liblinear'] # liblinear supports both l1 and l2`
	`}`
	
	`# Create and configure the GridSearchCV object`
	`grid_search = GridSearchCV(estimator=LogisticRegression(),`
	`param_grid=param_grid,`
	`cv=5,`
	`scoring='accuracy',`
	`verbose=1,`
	`n_jobs=-1)`
		`# Fit the model to find the best parameters`
		`print("Tuning Logistic Regression...")`
		`grid_search.fit(x_train_balanced, y_train_balanced)`
		
		`# Print the best findings`
		`print("\nBest Parameters found:", grid_search.best_params_)`
		`print("Best Accuracy through Cross-Validation: {:.2f}%".format(grid_search.best_score_ * 100))`
		
		`Tuning Logistic Regression...`
		`Fitting 5 folds for each of 8 candidates, totalling 40 fits`
		
		`Best Parameters found: {'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}`
		`Best Accuracy through Cross-Validation: 99.39%`

Random Forest	`param_grid = {`
	`'n_estimators': [50, 100, 200],`
	`'max_depth': [10, 20, None],`
	`'min_samples_split': [2, 5, 10],`
	`'min_samples_leaf': [1, 2, 4]`
	`}`
	
	`# Create and configure the GridSearchCV object`
	`grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42),`
	`param_grid=param_grid,`
	`cv=5,`
	`scoring='accuracy',`
	`verbose=1,`
	`n_jobs=-1)`
		`# Fit the model to find the best parameters`
		`print("Tuning Random Forest Classifier...")`
		`grid_search.fit(x_train_balanced, y_train_balanced)`
		
		`# Print the best findings`
		`print("\nBest Parameters found:", grid_search.best_params_)`
		`print("Best Accuracy through Cross-Validation: {:.2f}%".format(grid_search.best_score_ * 100))`
		
		`Tuning Random Forest Classifier...`
		`Fitting 5 folds for each of 81 candidates, totalling 405 fits`
		
		`Best Parameters found: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}`
		`Best Accuracy through Cross-Validation: 100.00%`

Naïve Bayes		
SVM		
Gradient Boosting		

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Logistic Regression	

Random Forest	`print("Classification Report")`
	`print(classification_report(y_test, y_pred_rf))`
	
	Accuracy Score
	0.9964912280701754
	Classification Report
	precision recall f1-score support
	
	0 1.00 0.99 1.00 157
	1 0.99 1.00 1.00 128
	
	accuracy 1.00 285
	macro avg 1.00 1.00 1.00 285
	weighted avg 1.00 1.00 1.00 285
	
	`print("Confusion Matrix")`
	`print(confusion_matrix(y_test, y_pred_rf))`
	
	Confusion Matrix
	[[156 1]
	[0 128]]

Naïve Bayes	`print("Classification Report")`
	`print(classification_report(y_test, y_pred_gnb))`
	
	Accuracy Score
	0.968421052631579
	Classification Report
	precision recall f1-score support
	
	0 0.99 0.96 0.97 157
	1 0.95 0.98 0.97 128
	
	accuracy 0.97 285
	macro avg 0.97 0.97 0.97 285
	weighted avg 0.97 0.97 0.97 285
	
	`print("Confusion Matrix")`
	`print(confusion_matrix(y_test, y_pred_gnb))`
	
	Confusion Matrix
	[[150 7]
	[2 126]]

SVM	`print("Classification Report")`
	`print(classification_report(y_test, y_pred_svm))`
	
	Accuracy Score
	0.9228070175438596
	Classification Report
	precision recall f1-score support
	
	0 1.00 0.86 0.92 157
	1 0.85 1.00 0.92 128
	
	accuracy 0.92 285
	macro avg 0.93 0.93 0.92 285
	weighted avg 0.93 0.92 0.92 285
	
	`print("Confusion Matrix")`
	`print(confusion_matrix(y_test, y_pred_svm))`
	
	Confusion Matrix
	[[135 22]
	[0 128]]

Gradient Boosting	```
	print("Classification Report")
	print(classification_report(y_test, y_pred_gb))
	``
	``
	Accuracy Score
	1.0
	Classification Report
	              precision recall f1-score support
	
	           0 1.00 1.00 1.00 157
	           1 1.00 1.00 1.00 128
	
	    accuracy 1.00 285
	   macro avg 1.00 1.00 1.00 285
	weighted avg 1.00 1.00 1.00 285
	``
	``
	print("Confusion Matrix")
	print(confusion_matrix(y_test, y_pred_gb))
	``
	``
	Confusion Matrix
	[[157 0]
	 [0 128]]
	``

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.