Model Optimization and Tuning Phase Report

Date	8 August 2025
Skill wallet ID	SWUID20250185217
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 focuses on refining machine learning models to achieve peak predictive performance. This stage involves implementing optimized model code, systematically fine-tuning hyperparameters, and evaluating multiple configurations to identify the most effective setup. Performance metrics such as Accuracy, F1-Score, Precision, Recall, and ROC-AUC will be compared across tuned models. The process culminates in a clear justification for the final model selection, ensuring enhanced predictive accuracy, computational efficiency, and overall robustness in real-world deployment.

Hyperparameter Tuning Documentation (6Marks):

Model	Tuned Hyperparameters	Optimal Values

Logistic Regression	`param_grid_lr = {`
	`    'penalty': ['l1', 'l2'],`
	`    'C': [0.01, 0.1, 1, 10, 100],`
	`    'solver': ['liblinear'],`
	`    'max_iter': [1000, 3000, 5000]`
	`}`
	`grid_lr = GridSearchCV(`
	`    LogisticRegression(),`
	`    param_grid_lr,`
	`    cv=5,`
	`    scoring='accuracy'`
	`)`
		`grid_lr.fit(X_train_scaled, y_train)`
		
		`print("Best Params:", grid_lr.best_params_)`
		`print("Best Accuracy:", grid_lr.best_score_)`
		
		`Best Params: {'C': 100, 'max_iter': 1000, 'penalty': 'l1', 'solver': 'liblinear'}`
		`Best Accuracy: 1.0`

Random Forest	`param_grid_rf = {`
	`    'n_estimators': [50, 100, 200, 300],`
	`    'max_depth': [None, 10, 20, 30],`
	`    'min_samples_split': [2, 5, 10],`
	`    'min_samples_leaf': [1, 2, 4],`
	`    'bootstrap': [True, False]`
	`}`
	
	`grid_rf = GridSearchCV(`
	`    estimator=RandomForestClassifier(random_state=42),`
	`    param_grid=param_grid_rf,`
	`    cv=5,`
	`    scoring='accuracy',`
	`    n_jobs=-1,`
	`    verbose=2`
	`)`
		`print("Best Random Forest Parameters:", grid_rf.best_params_)`
		`print("Best Cross-Validation Accuracy:", grid_rf.best_score_)`
		
		`Fitting 5 folds for each of 288 candidates, totalling 1440 fits`
		`Best Random Forest Parameters: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}`
		`Best Cross-Validation Accuracy: 1.0`

Naïve Bayes

```python
param_grid_nb = {
    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5]
}

grid_nb = GridSearchCV(
    GaussianNB(),
    param_grid=param_grid_nb,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=2
)
```

```python
print("Best NB Parameters:", grid_nb.best_params_)
print("Best CV Accuracy:", grid_nb.best_score_)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best NB Parameters: {'var_smoothing': 1e-09}
Best CV Accuracy: 0.9324298258971625
```

SVM

```python
param_grid_svc = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'gamma': ['scale', 'auto'],
    'degree': [2, 3, 4]  # Only used in 'poly' kernel
}

grid_svc = GridSearchCV(
    SVC(probability=True, random_state=42),
    param_grid=param_grid_svc,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=2
)
```

```python
print("Best SVC Parameters:", grid_svc.best_params_)
print("Best CV Accuracy:", grid_svc.best_score_)
```

```
Fitting 5 folds for each of 96 candidates, totalling 480 fits
Best SVC Parameters: {'C': 100, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'}
Best CV Accuracy: 1.0
```

Gradient Boosting

```python
param_grid_gbc = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 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]
}

grid_gbc = GridSearchCV(
    GradientBoostingClassifier(random_state=42),
    param_grid=param_grid_gbc,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=2
)
```

```python
print("Best GBC Parameters:", grid_gbc.best_params_)
print("Best CV Accuracy:", grid_gbc.best_score_)
```

```
Fitting 5 folds for each of 648 candidates, totalling 3240 fits
Best GBC Parameters: {'learning_rate': 0.01, 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100, 'subsample': 0.8}
Best CV Accuracy: 1.0
```

Performance Metrics Comparison Report (2 Marks):

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr><td>Logistic Regression</td><td><pre>
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_lr))
print("\nClassification Report:\n", classification_report(y_test, y_pred_lr))

Confusion Matrix:
 [[113   0]
 [0 135]]

Classification Report:
              precision   recall f1-score   support

           0   1.00   1.00   1.00   113
           1   1.00   1.00   1.00   135

    accuracy   1.00   248
   macro avg   1.00   1.00   1.00   248
weighted avg   1.00   1.00   1.00   248
</pre></td></tr>
<tr><td>Random Forest</td><td><pre>
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
Classification Report:
              precision   recall f1-score   support

           0   1.00   1.00   1.00   113
           1   1.00   1.00   1.00   135

    accuracy   1.00   248
   macro avg   1.00   1.00   1.00   248
weighted avg   1.00   1.00   1.00   248

Confusion Matrix:
 [[113   0]
 [0 135]]
</pre></td></tr>
</table>

Naïve Bayes

```
print("\nClassification Report:\n", classification_report(y_test, y_pred_nb))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_nb))
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.96      0.98       113
           1       0.97      0.99      0.98       135

    accuracy                           0.98       248
   macro avg       0.98      0.98      0.98       248
weighted avg       0.98      0.98      0.98       248


Confusion Matrix:
 [[109   4]
 [  1 134]]
```

SVM

```
print("\nClassification Report:\n", classification_report(y_test, y_pred_svc))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_svc))
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       113
           1       1.00      1.00      1.00       135

    accuracy                           1.00       248
   macro avg       1.00      1.00      1.00       248
weighted avg       1.00      1.00      1.00       248


Confusion Matrix:
 [[113   0]
 [  0 135]]
```

Gradient Boosting

```
print("\nClassification Report:\n", classification_report(y_test, y_pred_gbc))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_gbc))
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       113
           1       1.00      1.00      1.00       135

    accuracy                           1.00       248
   macro avg       1.00      1.00      1.00       248
weighted avg       1.00      1.00      1.00       248


Confusion Matrix:
 [[113   0]
 [  0 135]]
```

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.