Model Optimization and Tuning Phase Report

Date	14 JULY 2024
Team ID	FACULTY
Project Title	Fetal AI: Using Machine Learning To Predict And Monitor Fetal Health.
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 (6Marks):

Model	Tuned Hyperparameters	Optimal Values

Random Forest	**Define the Random Forest Classifier**
	`[66] rf_classifier = RandomForestClassifier()`
	Define the hyperparameters and posiible values
	`[68] param_grid = { 'n_estimators': [100, 200, 300], 'max_depth': [5, 10, 15], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 5, 10], 'criterion': ['gini', 'entropy'], }`
		`[33] print("For the amounts of training data is:", x_train_smote.shape[0]) # Assumir print("Accuracy of RandomForestClassifier:",RF_model.score(x_test,y_test)) cm=confusion_matrix(y_test,predictions) cm_display=ConfusionMatrixDisplay(cm).plot() plt.show()`
		For the amounts of training data is: 3477
		Accuracy of RandomForestClassifier: 0.9373040752351097
		`printf(f'Optimal Hyperparameters:{best_params}')`
		Optimal Hyperparameters:{ 'n_estimators':100,'max_depth':5 'min_samples_split':2'min_samples_leaf':5 'criterion': 'gini'}

Decision Tree	**Define the Decision tree Classifier**
	`[57] dt_classifier = DecisionTreeClassifier()`
	Define the hyperparameters and posiible values
	`[58] param_grid = { 'max_depth': [3, 5, 10], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 5, 10], 'criterion': ['gini', 'entropy'], 'splitter':['best','random'] }`
		`[38] print("For the amounts of training data is:",x_train_smote.shape[0]) print("Accuracy of DecisionTreeClassifier:",DT_model.score(x_test,y_test)) cm=confusion_matrix(y_test,predictions) cm_display=ConfusionMatrixDisplay(cm).plot() plt.show()`
		For the amounts of training data is: 3477
		Accuracy of DecisionTreeClassifier: 0.9075235109717869
		`printf(f'Optimal Hyperparameters:{best_params}')`
		Optimal Hyperparameters:{ 'max_depth','3','min_samples_split','2' 'min_samples_leaf''5' ,'criterion': 'gini','splitter':'best'}

Logistic Regression	**Define the LogisticRegression**
	`[71] lr_classifier = LogisticRegression()`
	Define the hyperparameters and posiible values
	`[72] param_grid = { 'penalty': ['l1', 'l2', 'elasticnet', 'none'], # Regularization type 'C': [0.001, 0.01, 0.1, 1, 10, 100], # Inverse of regularization strength 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], # Algorithm to use 'max_iter': [100, 200, 500] # Maximum number of iterations }`
		`[42] print("For the amounts of training data is:",x_train_smote.shape[0]) print("Accuracy of LogisticRegression:",LR_model.score(x_test,y_test)) cm=confusion_matrix(y_test,predictions) cm_display=ConfusionMatrixDisplay(cm).plot() plt.show()`
		For the amounts of training data is: 3477
		Accuracy of LogisticRegression: 0.771159746081505
		`printf(f'Optimal Hyperparameters:{best_params}')`
		Optimal Hyperparameters: { 'penalty':l2,elasticnet:none 'C':0.1,'solver':sag,'max_iter':100}

KNN
KNeighbors
Classifier
	Define the KNN KNeighborsClassifier
	`[69] knn_classifier = KNeighborsClassifier()`
	Define the hyperparameters and posiible values
	`[75] param_grid={ 'n_neighbors': [3, 5, 7, 9, 11], # number of nearest neighbors to co 'weights': ['uniform', 'distance'], # weight function to use 'p': [1, 2], # exponent for Minkowski distance }`
		`[46] print("For the amounts of training data is:",x_train_smote.shape[0]) print("Accuracy of KNeighborsClassifier:",KNN_model.score(x_test,y_test)) cm=confusion_matrix(y_test,predictions) cm_display=ConfusionMatrixDisplay(cm).plot() plt.show()`
		For the amounts of training data is: 3477
		Accuracy of KNeighborsClassifier: 0.8369905956112853
		`printf(f'Optimal Hyperparameters:{best_params}')`
		Optimal Hyperparameters:{'n_neighbors': 3,'weights':uniform,'p':1}

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

Random Forest	`print(classification_report(y_test,predictions))`
	`precision recall f1-score support`
	`1.0 0.96 0.84 0.90 496`
	`2.0 0.53 0.81 0.64 101`
	`3.0 0.71 0.83 0.76 41`
	`accuracy 0.84 638`
	`macro avg 0.73 0.83 0.77 638`
	`weighted avg 0.88 0.84 0.85 638`
	`[36] confusion_matrix(y_test,predictions)`
	`array([[477, 17, 2], [ 15, 82, 4], [ 2, 0, 39]])`

Decision Tree	[39] print(classification_report(y_test,predictions)) precision recall f1-score support 1.0 0.97 0.93 0.95 496 2.0 0.73 0.80 0.76 101 3.0 0.73 0.88 0.80 41 accuracy 0.91 638 macro avg 0.81 0.87 0.84 638 weighted avg 0.91 0.91 0.91 638 [40] confusion_matrix(y_test,predictions) array([[462, 29, 5], [12, 81, 8], [4, 1, 36]])
Logistic Regression	[43] print(classification_report(y_test,predictions)) precision recall f1-score support 1.0 0.97 0.77 0.86 496 2.0 0.44 0.73 0.55 101 3.0 0.49 0.88 0.63 41 accuracy 0.77 638 macro avg 0.63 0.79 0.68 638 weighted avg 0.85 0.77 0.79 638 [44] confusion_matrix(y_test,predictions) array([[382, 92, 22], [12, 74, 15], [1, 4, 36]])
KNN KNeighbors Classifier	[47] print(classification_report(y_test,predictions)) precision recall f1-score support 1.0 0.96 0.84 0.90 496 2.0 0.53 0.81 0.64 101 3.0 0.71 0.83 0.76 41 accuracy 0.84 638 macro avg 0.73 0.83 0.77 638 weighted avg 0.88 0.84 0.85 638 [48] confusion_matrix(y_test,predictions) array([[418, 69, 9], [14, 82, 5], [4, 3, 34]])

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 hyper parameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.