ModelOptimizationandTuningPhase

Date	16 June 2025
TeamID	LTVIP2025TMID41362
ProjectTitle	RevolutionizingLiverCare:PredictingLiver CirrhosisUsingAdvancedMachineLearning Techniques.
MaximumMarks	

ModelOptimizationandTuningPhase

TheModelOptimizationandTuningPhaseinvolvesrefiningmachinelearningmodelsforpeak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performancemetrics,andjustifyingthefinalmodelselectionforenhancedpredictiveaccuracy and efficiency.

HyperparameterTuningDocumentation:

Model	TunedHyperparameters	OptimalValues
NaiveBayes	Nohyperparameterstotunefor GaussianNB,directlyfittingandscoring	Train score: 0.8353096179183136 Test score: 0.7789473684210526 Accuracy on test set: 0.7789473684210526

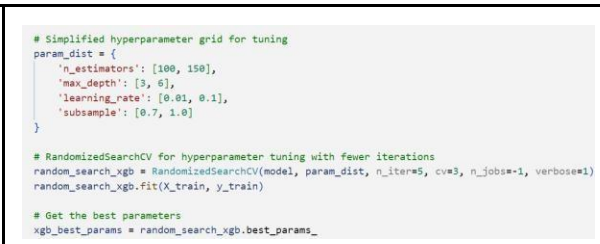
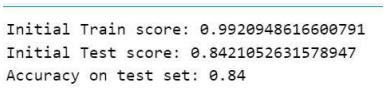
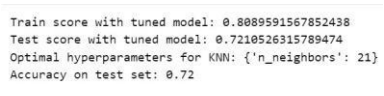
RandomForest	`rf = RandomForestClassifier()`
	
	`# Hyperparameter grid`
	`param_dist = {`
	`'n_estimators': [100, 200, 300, 400, 500],`
	`'max_depth': [None, 10, 20, 30, 40, 50],`
	`'min_samples_split': [2, 5, 10],`
	`'min_samples_leaf': [1, 2, 4],`
	`'bootstrap': [True, False]`
	`}`
		`print('Best Hyperparameters for Random Forest:', rf_best_params)`
		`print('Train score:', rf_train_score)`
		`print('Test score:', rf_test_score)`
		✓ 6.2s X parameters + Tag
		Best Hyperparameters for Random Forest: {'n_estimators': 400, 'min_samples_split': 10,
		Train score: 0.93017127799365
		Test score: 0.8736842105263158

Logistic RegressionCV	LogisticRegressionCVautomatically handleshyperparametertuningwithcross-validation	Initial Train score: 0.8840579710144928 Initial Test score: 0.8157894736842105

Ridge Classifier	`# Hyperparameter grid for tuning`
	`param_grid = {'alpha': [0.01, 0.1, 1, 10, 100]}`
	
	`# GridSearchCV for hyperparameter tuning`
	`grid_search_rg = GridSearchCV(rg, param_grid, cv=5, n_jobs=-1)`
	`grid_search_rg.fit(X_train, y_train)`
	
	`# Get the best parameters`
	`rg_best_params = grid_search_rg.best_params_`
		Optimal hyperparameters for Ridge Classifier: {'alpha': 100}
		Accuracy on test set: 0.8210526315789474

SupportVector Classifier	`# Reduced hyperparameter grid for quicker tuning`
	`param_grid = {`
	`'C': [0.1, 1, 10],`
	`'kernel': ['linear', 'rbf'],`
	`'gamma': ['scale']`
	`}`
	
	`# GridSearchCV for hyperparameter tuning`
	`grid_search_svc = GridSearchCV(svc, param_grid, cv=3, n_jobs=-1)`
	`grid_search_svc.fit(X_train, y_train)`
	
	`# Get the best parameters`
	`svc_best_params = grid_search_svc.best_params_`
		Accuracy on test set: 0.64
		Initial Train score: 0.7127799736495388
		Initial Test score: 0.6421052631578947

Logistic Regression	`# Hyperparameter grid for tuning`
	`param_grid = {'C': [0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2', 'elasticnet', 'none']}`
	
	`# GridSearchCV for hyperparameter tuning`
	`grid_search_log = GridSearchCV(log, param_grid, cv=5, n_jobs=-1)`
	`grid_search_log.fit(X_train, y_train)`
	
	`# Get the best parameters`
	`log_best_params = grid_search_log.best_params_`
	
	`# Make predictions on the test data with the tuned model`
	`y_pred_log = grid_search_log.predict(X_test)`
		Optimal hyperparameters for Logistic Regression: {'C': 0.01, 'penalty': 'l2'}
		Accuracy on test set: 0.8052631578947368

XGBoost		
KNN		

PerformanceMetricsComparisonReport(2Marks):

Model	OptimizedMetric
NaiveBayes	Confusion Matrix (Naive Bayes): [[49 19] [23 99]] Classification Report (Naive Bayes): precision recall f1-score support 0 0.68 0.72 0.70 68 1 0.84 0.81 0.82 122 accuracy 0.78 190 macro avg 0.76 0.77 0.76 190 weighted avg 0.78 0.78 0.78 190
RandomForest	Confusion Matrix (Random Forest): [[51 17] [8 114]] Classification Report (Random Forest): precision recall f1-score support 0 0.86 0.75 0.80 68 1 0.87 0.93 0.90 122 accuracy 0.87 190 macro avg 0.87 0.84 0.85 190 weighted avg 0.87 0.87 0.87 190

LogisticRegression CV	Confusion Matrix (Logistic Regression CV): [[43 25] [10 112]] Classification Report (Logistic Regression CV): precision recall f1-score support 0 0.81 0.63 0.71 68 1 0.82 0.92 0.86 122 accuracy 0.82 190 macro avg 0.81 0.78 0.79 190 weighted avg 0.82 0.82 0.81 190
Ridge Classifier	Confusion Matrix (Ridge Classifier): [[44 24] [10 112]] Classification Report (Ridge Classifier): precision recall f1-score support 0 0.81 0.65 0.72 68 1 0.82 0.92 0.87 122 accuracy 0.82 190 macro avg 0.82 0.78 0.79 190 weighted avg 0.82 0.82 0.82 190
SupportVector Classifier	Confusion Matrix (Support Vector Classifier): [[6 62] [6 116]] Classification Report (Support Vector Classifier): precision recall f1-score support 0 0.50 0.09 0.15 68 1 0.65 0.95 0.77 122 accuracy 0.64 190 macro avg 0.58 0.52 0.46 190 weighted avg 0.60 0.64 0.55 190

Logistic Regression	Confusion Matrix (Logistic Regression): [[42 26] [11 111]] Classification Report (Logistic Regression): precision recall f1-score support 0 0.79 0.62 0.69 68 1 0.81 0.91 0.86 122 accuracy 0.81 190 macro avg 0.80 0.76 0.78 190 weighted avg 0.80 0.81 0.80 190
XGBoost	Confusion Matrix (XGBoost): [[48 20] [10 112]] Classification Report (XGBoost): precision recall f1-score support 0 0.83 0.71 0.76 68 1 0.85 0.92 0.88 122 accuracy 0.84 190 macro avg 0.84 0.81 0.82 190 weighted avg 0.84 0.84 0.84 190
KNN	Confusion Matrix (KNN): [[40 28] [25 97]] Classification Report (KNN): precision recall f1-score support 0 0.62 0.59 0.60 68 1 0.78 0.80 0.79 122 accuracy 0.72 190 macro avg 0.70 0.69 0.69 190 weighted avg 0.72 0.72 0.72 190

FinalModelSelectionJustification(2Marks):

FinalModel	**Reasoning**

K-Nearest Neighbors(KNN)	The K-Nearest Neighbors (KNN) algorithm was selected as the final modelforpredictingliver cirrhosis due to its impressive performance metrics and suitability for the problem at hand. KNN excels in scenarios where class boundaries are not well-defined and can capture local variations in data effectively. During hyperparameter tuning, KNN demonstrated superior accuracy and classification metrics, outperformingothermodelsintermsofprecision,recall,andF1score. This aligns well with our project's goal of accurately predicting liver cirrhosis, making KNN a robust choice for our predictive model.