



MODEL OPTIMIZATION AND TUNING PHASE TEMPLATE

Date	27th July 2005
Team ID	LTVIP2025TMID60634
Project Title	Revolutionizing Liver Care: Predicting Liver Cirrhosis Using Advanced Machine Learning Techniques.
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
Naive Bayes	No hyperparameters to tune for GaussianNB, directly fitting and scoring	Train score: 0.8353096179183136 Test score: 0.7789473684210526 Accuracy on test set: 0.7789473684210526
Random Forest	<pre>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] }</pre>	print('Best Hyperparameters for Bandom Forest:', rf_best_params) print('Train_score', rf_train_score') print('Train_score'; rf_test_score) / 6b x parameters for Random Forest: {'n_estimators': 480, 'min_samples_split': 18, Train_score: 8.939171277997365 Test_score: 8.833684186183158





Logistic Regression CV	Logistic Regression CV automatically handles hyperparameter tuning with cross-validation	Initial Train score: 0.8840579710144928 Initial Test score: 0.8157894736842105
Ridge Classifier	<pre># 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_</pre>	Optimal hyperparameters for Ridge Classifier: {'alpha': 100} Accuracy on test set: 0.8210526315789474
Support Vector Classifier	<pre># 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_</pre>	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': ['ll', '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': '12') Accuracy on test set: 0.8052631578947368
XG Boost	<pre># Simplified hyperparameter grid for tuning param_dist = { 'n_estimators': [100, 150], 'max_depth': [3, 6], 'learning_rate': [0.01, 0.1], 'subsample': [0.7, 1.0] } # RandomizedSearchCV for hyperparameter tuning with fewer iterations random_search_xgb = RandomizedSearchCV(model, param_dist, n_iter=5, cv=3, n_jobs=-1, verbos=1) random_search_xgb.fit(%_train, y_train) # Get the best parameters xgb_best_params = random_search_xgb.best_params_</pre>	Initial Train score: 0.9920948616600791 Initial Test score: 0.8421052631578947 Accuracy on test set: 0.84





HYPERPARAMETER TUNING

k = np.random.randint(1,50,60)
params = {'n_neighbors' : k}

random_search = RandomizedSearchCV(knn, params, n_iter=5, cv=5, n_jobs=-1, verbose = 0)
random_search = fix_tx_train, y_train)

print('train_score - '+ str(random_search.score(X_train, y_train)))
print('test_score- '+ str(random_search.score(X_test,y_test)))

knn.get_params()

Performance Metrics Comparison Report (2 Marks):

Model	(Optimiz	ed Met	ric		
ve Bayes	Confusion Matr [[49 19] [23 99]] Classification 0 1 accuracy macro avg weighted avg		ive Bayes): recall 0.72 0.81		68 122 190 190 190	
	Confusion Matri [[51 17] [8 114]] Classification	x (Random F	Forest):			
		precision	recall	f1-score	support	
dom Forest	0 1	0.86 0.87	0.75 0.93	0.80 0.90	68 122	
	accuracy macro avg weighted avg	0.87 0.87	0.84 0.87	0.87 0.85 0.87	190 190 190	
	Confusion Matrix [[43 25] [10 112]]				vv.	
istia Dagmassian				f1-score		
isuc Regression	0 1	0.81 0.82	0.63 0.92	0.71 0.86	68 122	
	accuracy macro avg weighted avg	0.81 0.82	0.78 0.82	0.82 0.79 0.81	190 190 190	
dom Forest	[8 114]] Classification 0 1 accuracy macro avg weighted avg Confusion Matrix [[43 25] [10 112]] Classification R p 0 1 accuracy macro avg	precision 0.86 0.87 0.87 0.87 (Logistic eport (Log recision 0.81 0.82 0.81	e Regressi gistic Regrecall 0.63 0.92	f1-score 0.80 0.90 0.87 0.85 0.87 ion CV): gression C f1-score 0.71 0.86 0.82 0.79	68 122 190 190 190 V): support 68 122 190 190	





Confusion Matrix (Ridge Classifier):	[44 24							
[4 24	[44 24] [101 12] Classification Report (Ridge Classifier) : precision recall f1-score support		Confusion Matri	v (Ridge Cl	assifier):			
Classifier Classification Report (Ridge Classifier):	Classifier Classifier Classifier Classifier Classifier Classifier Classifier Classifier Precision Precisio			.x (Niuge Ci				
Ridge Classifier	Classifier Classifier Precision Precall f1-score support							
Ridge Classifier e	Ridge Classifier		75 Table 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Report (Ric	ge Classif	ier):		
Ridge Classifier	Ridge Classifier						support	
1 0.82 0.92 0.87 122	Confusion Matrix (Support Vector Classifier): [6 62]	D: 1 C1 : C					6-3-*r*00090 0	
Confusion Matrix (Support Vector Classifier): [[6 62]	Confusion Matrix (Support Vector Classifier): [6 62]	Ridge Classifier	0	0.81	0.65	0.72	68	
Confusion Matrix (Support Vector Classifier): [6 62] Classification Report (Support Vector Classifier): [6 62] Classification Report (Support Vector Classifier): precision recall f1-score support	Confusion Matrix (Support Vector Classifier): [[6 62]		1	0.82	0.92	0.87	122	
Confusion Matrix (Support Vector Classifier): [6 62] Classification Report (Support Vector Classifier): [6 62] Classification Report (Support Vector Classifier): precision recall f1-score support	Confusion Matrix (Support Vector Classifier): [[6 62]							
Confusion Matrix (Support Vector Classifier): [6 62]	Confusion Matrix (Support Vector Classifier): [[6 62]		,					
Confusion Matrix (Support Vector Classifier):	Confusion Matrix (Support Vector Classifier): [[6 62]							
[Classifier Cla		weighted avg	0.82	0.82	0.82	190	
[Classifier Cla							
[Classifier Cla							
[Classifier Cla							
Support Vector Classification Report (Support Vector Classifier):	Classification Report (Support Vector Classifier): Support Vector		Confusion Matri	k (Support	Vector Cla	ssifier):		
Classification Report (Support Vector Classifier): precision recall f1-score support	Classification Report (Support Vector Classifier): precision recall f1-score support		[[6 62]					
Classifier Precision Pre	Support Vector							
Classifier 0	Classifier		Classification R	Report (Sup	port Vecto	or Classif	ier):	
Classifier 0	Classifier	Cymnost Vastas						
Classifier 1	Classifier	Support vector						
Confusion Matrix (Logistic Regression): [[42 26]	Confusion Matrix (Logistic Regression):	~	0	0.50	0.09	0.15	68	
Confusion Matrix (Logistic Regression): [42 26	Confusion Matrix (Logistic Regression):	Classifier	1	0.65	0.95	0.77	122	
Confusion Matrix (Logistic Regression): [42 26	Confusion Matrix (Logistic Regression):							
Confusion Matrix (Logistic Regression): [42 26	Confusion Matrix (Logistic Regression): [[42 26]		accuracy			0.64	190	
Confusion Matrix (Logistic Regression): [[42 26]	Confusion Matrix (Logistic Regression):		macro avg	0.58	0.52	0.46	190	
Confusion Matrix (XGBoost):	Classification Report (Logistic Regression):		weighted avg	0.60	0.64	0.55	190	
Confusion Matrix (XGBoost):	Classification Report (Logistic Regression):							
Confusion Matrix (XGBoost):	Classification Report (Logistic Regression):							
Confusion Matrix (XGBoost):	Classification Report (Logistic Regression):							
Confusion Matrix (XGBoost):	Classification Report (Logistic Regression):		Confusion Matri	ix (Logistic	Regression	1):		
Classification Report (Logistic Regression): precision recall f1-score support	Confusion Matrix (XGBoost): [48 20			rx (rogratio	Kegi coolon	.,.		
Logistic Regression 0 0.79 0.62 0.69 68 1 0.81 0.91 0.86 122	Description Precision Pr							
Logistic Regression 0 0.79 0.62 0.69 68 1 0.81 0.91 0.86 122	Logistic Regression 0			Report (Log	istic Regre	ession):		
Accuracy 0.81 190 macro avg 0.80 0.76 0.78 190 weighted avg 0.80 0.81 0.80 190 Confusion Matrix (XGBoost): [[48 20] [10 112]] Classification Report (XGBoost): precision recall f1-score support XG Boost 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	Confusion Matrix (XGBoost): [48 20] [10 112]] Classification Report (XGBoost): precision recall f1-score supposed by the suppose of the suppos			precision	recall f	1-score	support	
Accuracy 0.81 190 macro avg 0.80 0.76 0.78 190 weighted avg 0.80 0.81 0.80 190 Confusion Matrix (XGBoost): [[48 20] [10 112]] Classification Report (XGBoost): precision recall f1-score support XG Boost 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	Confusion Matrix (XGBoost): [48 20] [10 112]] Classification Report (XGBoost): precision recall f1-score supposed by the suppose of the suppos	La gistia Dagmassian						
Confusion Matrix (XGBoost): [[48	Confusion Matrix (XGBoost): [[48 20]	Logistic Regression						
Confusion Matrix (XGBoost): [[48 20] [10 112]] Classification Report (XGBoost):	Confusion Matrix (XGBoost): [48 20] [10 112]] Classification Report (XGBoost): precision recall f1-score support in the suppor		1,	0.81	0.91	0.86	122	
Confusion Matrix (XGBoost): [[48 20] [10 112]] Classification Report (XGBoost):	Confusion Matrix (XGBoost): [48 20] [10 112]] Classification Report (XGBoost): precision recall f1-score support in the suppor		accuracy			0.81	190	
Confusion Matrix (XGBoost): [[48 20]	Confusion Matrix (XGBoost): [[48 20]		(5)	0.80	0.76			
Confusion Matrix (XGBoost): [[48 20]	Confusion Matrix (XGBoost): [[48 20]							
[[48 20] [10 112]] Classification Report (XGBoost):	[[48 20] [10 112]] Classification Report (XGBoost):		5					
[[48 20]	[[48 20] [10 112]] Classification Report (XGBoost):							J
[[48 20] [10 112]] Classification Report (XGBoost):	[[48 20] [10 112]] Classification Report (XGBoost):							
[[48 20]	[[48 20] [10 112]] Classification Report (XGBoost):							
[10 112]] Classification Report (XGBoost):	[10 112]] Classification Report (XGBoost):			(XGBoost):			
Classification Report (XGBoost):	Classification Report (XGBoost):		[[48 20]					
Classification Report (XGBoost):	Classification Report (XGBoost):		[10 112]]					
XG Boost 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	XG Boost			Report (XG	Boost):			
XG Boost 0 0.83 0.71 0.76 68 1 0.85 0.92 0.88 122 accuracy macro avg 0.84 0.81 0.82 190	XG Boost 0 0.83 0.71 0.76 1 0.85 0.92 0.88 1 accuracy 0.84 1 macro avg 0.84 0.81 0.82 1					f1-sco	re support	
accuracy 0.84 190 macro avg 0.84 0.81 0.82 190	accuracy 0.84 1 macro avg 0.84 0.81 0.82 1		•				000 00 00 00 00 00 00 00 00 00 00 00 00	
accuracy 0.84 190 macro avg 0.84 0.81 0.82 190	accuracy 0.84 1 macro avg 0.84 0.81 0.82 1	XG Boost	a	0 83	a 71	a 7	5 68	
accuracy 0.84 190 macro avg 0.84 0.81 0.82 190	accuracy 0.84 1 macro avg 0.84 0.81 0.82 1							
macro avg 0.84 0.81 0.82 190	macro avg 0.84 0.81 0.82 1		1	0.85	0.92	0.8	5 122	
macro avg 0.84 0.81 0.82 190	macro avg 0.84 0.81 0.82 1							
	ADDITION OF THE SECOND		(5)					
weighted avg 0.84 0.84 190	weighted avg 0.84 0.84 1							
			weighted avg	0.84	0.84	0.8	4 190	J
								J
	1							





	Confusion Matrix [[40 28] [25 97]]		17.		
	Classification R			f1-score	support
KNN	Ø 1	0.62 0.78	0.59 0.80	0.60 0.79	68 122
	accuracy			0.72	190
	macro avg	0.70	0.69	0.69	190
	weighted avg	0.72	0.72	0.72	190

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
K-Nearest Neighbors (KNN)	The K-Nearest Neighbors (KNN) algorithm was selected as the final model for predicting liver 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, outperforming other models in terms of precision, recall, and F1 score. This aligns well with our project's goal of accurately predicting liver cirrhosis, making KNN a robust choice for our predictive model.