Model Optimization and Tuning Phase

Date	04 July 2024
Team ID	SWTID1720110187
Project Title	Revolutionizing Liver Care
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

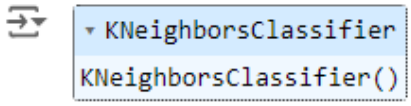
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

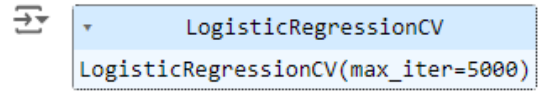
Hyperparameter Tuning Documentation (6 Marks):

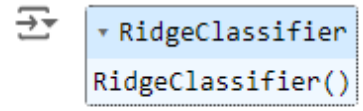
Model	Tuned Hyperparameters

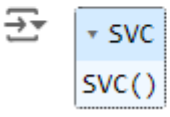
Random Forest Classifier	`[19] from sklearn.ensemble import RandomForestClassifier`
	`rf = RandomForestClassifier()`
	`rf.fit(x_train,y_train)`
	`RandomForestClassifier`
	`RandomForestClassifier()`

Logistic Regression	`[25] from sklearn.linear_model import LogisticRegression`
	`lr = LogisticRegression(max_iter=500)`
	`lr.fit(x_train,y_train)`
	`LogisticRegression`
	`LogisticRegression(max_iter=500)`

KNN	[28] `from sklearn.neighbors import KNeighborsClassifier`
	`knn = KNeighborsClassifier()`
	`knn.fit(x_train,y_train)`
	 

Logistic Regression CV	[36] `# Logistic Regression CV`
	`from sklearn.linear_model import LogisticRegressionCV`
	`lcv = LogisticRegressionCV(max_iter=5000)`
	`lcv.fit(x_train, y_train)`
	 

Ridge Classifier	[37] `# Ridge Classifier`
	`from sklearn.linear_model import RidgeClassifier`
	`rg = RidgeClassifier()`
	`rg.fit(x_train, y_train)`
	 

Support Vetor Classifier	[38] `# Support Vector Classifier`
	`from sklearn.svm import SVC`
	`svc = SVC()`
	`svc.fit(x_train, y_train)`
	 

XGBoost	[39] # XGBoost from xgboost import XGBClassifier xgb = XGBClassifier() xgb.fit(x_train, y_train) XGBClassifier XGBClassifier(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, objective='multi:softprob', ...)
AdaBoost Classifier	[42] ab.fit(x_train,y_train) AdaBoostClassifier AdaBoostClassifier()
Gradient Boosting Classifier	[43] gb.fit(x_train,y_train) GradientBoostingClassifier GradientBoostingClassifier()

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

Random Forest	`from sklearn.metrics import classification_report`
	
	`y_pred = rf.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	precision recall f1-score support
	
	0 0.95 0.99 0.97 177
	1 1.00 1.00 1.00 3
	2 0.00 0.00 0.00 10
	
	accuracy 0.94 190
	macro avg 0.65 0.66 0.66 190
	weighted avg 0.90 0.94 0.92 190

Linear Regression	`[ ] from sklearn.metrics import classification_report`
	
	`y_pred = lr.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	precision recall f1-score support
	
	0 0.95 0.97 0.96 177
	1 1.00 1.00 1.00 3
	2 0.00 0.00 0.00 10
	
	accuracy 0.92 190
	macro avg 0.65 0.66 0.65 190
	weighted avg 0.90 0.92 0.91 190

KNN	`[ ] from sklearn.metrics import classification_report`
	
	`y_pred = knn.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	precision recall f1-score support
	
	0 0.94 1.00 0.97 177
	1 1.00 0.67 0.80 3
	2 0.00 0.00 0.00 10
	
	accuracy 0.94 190
	macro avg 0.65 0.56 0.59 190
	weighted avg 0.89 0.94 0.92 190

Randomized Search	`[ ]` `from sklearn.metrics import classification_report`
	
	`y_pred = random_search.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	`precision recall f1-score support`
	
	`0 0.94 1.00 0.97 177`
	`1 1.00 0.67 0.80 3`
	`2 0.00 0.00 0.00 10`
	
	`accuracy 0.94 190`
	`macro avg 0.65 0.56 0.59 190`
	`weighted avg 0.89 0.94 0.92 190`

Logistic Regression CV	`[23]` `from sklearn.metrics import classification_report`
	
	`y_pred = lcv.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	`precision recall f1-score support`
	
	`0 0.95 1.00 0.97 177`
	`1 1.00 1.00 1.00 3`
	`2 0.00 0.00 0.00 10`
	
	`accuracy 0.95 190`
	`macro avg 0.65 0.67 0.66 190`
	`weighted avg 0.90 0.95 0.92 190`

Ridge Classifier	`from sklearn.metrics import classification_report`
	
	`y_pred = rg.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	`precision recall f1-score support`
	
	`0 0.94 1.00 0.97 177`
	`1 1.00 0.67 0.80 3`
	`2 0.00 0.00 0.00 10`
	
	`accuracy 0.94 190`
	`macro avg 0.65 0.56 0.59 190`
	`weighted avg 0.89 0.94 0.92 190`

Support Vector Mchine	[28] `from sklearn.metrics import classification_report`
	`y_pred = svc.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	precision recall f1-score support
	
	0 0.95 1.00 0.97 177
	1 1.00 1.00 1.00 3
	2 0.00 0.00 0.00 10
	
	accuracy 0.95 190
	macro avg 0.65 0.67 0.66 190
	weighted avg 0.90 0.95 0.92 190

XGBoost	[20] `from sklearn.metrics import classification_report`
	`y_pred = xgb.predict(x_test)`
	`print(classification_report(y_test, y_pred))`
	
	precision recall f1-score support
	
	0 0.97 0.97 0.97 177
	1 1.00 1.00 1.00 3
	2 0.44 0.40 0.42 10
	
	accuracy 0.94 190
	macro avg 0.80 0.79 0.80 190
	weighted avg 0.94 0.94 0.94 190

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
XGBoost	The model XGBoost has the highest precision and accuracy. Hence XGBoost is chosen as the final model.