



## **Model Optimization and Tuning Phase Template**

Date	19 June 2025
Team ID	SWTID174971392
Project Title	Early prediction for chronic kidney disease detection: A progressive approach to health management
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
Random Forest	'Random Forest': {     'model': RandomForestClassifier(random_state=42, class_weight='balanced'),     'params': {         'n_estimators': [10, 25, 50, 100],         'max_depth': [3, 5, 7, 10, None],         'min_samples_split': [20, 40, 60, 80],         'min_samples_split': [20, 40, 60, 80],         'max_features': ['sqrt', 'log2', 0.3, 0.5],         'max_samples': [0.6, 0.7, 0.8, 0.9],         'min_impurity_decrease': [0.0, 0.01, 0.02, 0.05],         'ccp_alpha': [0.0, 0.01, 0.02, 0.05] } },	BEST HYPERPARAMETERS:  n_estimators: 10  min_samples_split: 80  min_samples_leaf: 40  min_impurity_decrease: 0.05  max_samples: 0.6  max_features: 0.3  max_depth: 7  ccp_alpha: 0.05  Random Forest: Successfully tuned
Decision Tree	'Decision Tree': {     'model': DecisionTreeClassifier(random_state=42, class_weight='balanced'),     'params': {         'max_depth': [3, 5, 7, 10],         'min_samples_split': [40, 60, 80, 100],         'min_samples_leaf': [20, 30, 40, 50],         'max_features': ['sqrt', 'log2', 0.3, 0.5, None],         'min_impurity_decrease': [0.01, 0.02, 0.05, 0.1],         'ccp_alpha': [0.01, 0.02, 0.05, 0.1, 0.2]    } },	BEST HYPERPARAMETERS: min_samples_split: 40 min_samples_leaf: 40 min_impurity_decrease: 0.05 max_features: None max_depth: 10 ccp_alpha: 0.05 Decision Tree: Successfully tuned





```
'Gradient Boosting': {
                          'model': GradientBoostingClassifier(random_state=42),
                          'params': {
                               'n_estimators': [10, 25, 50, 100],
                                                                                           BEST HYPERPARAMETERS:
                                                                                            subsample: 0.6
                               'learning_rate': [0.01, 0.05, 0.1, 0.2],
                                                                                            n estimators: 100
                               'max_depth': [2, 3, 4, 5],
                                                                                            min_samples_split: 80
Gradient
                                                                                            min_samples_leaf: 40
                               'min_samples_split': [40, 60, 80],
                                                                                            min_impurity_decrease: 0.05
                               'min_samples_leaf': [20, 30, 40],
                                                                                            max features: log2
Boosting
                                                                                            max depth: 2
                               'subsample': [0.6, 0.7, 0.8, 0.9],
                                                                                            learning_rate: 0.01
                               'max_features': ['sqrt', 'log2', 0.3, 0.5],
                                                                                            ccp_alpha: 0.0
                                                                                             Gradient Boosting: Successfully tuned
                               'min_impurity_decrease': [0.01, 0.02, 0.05],
                               'ccp_alpha': [0.0, 0.01, 0.02]
                          }
                     },
                    'XGBoost': {
                        'model': xgb.XGBClassifier(random_state=42, eval_metric='logloss'),
                                                                                           BEST HYPERPARAMETERS:
                                                                                             subsample: 0.9
                            'n_estimators': [50, 100,150,200],
                                                                                             scale_pos_weight: 1
                            'learning_rate': [0.1, 0.15, 0.2, 0.3],
                                                                                             reg_lambda: 10
                            'max_depth': [5,6,7,8],
                                                                                             reg_alpha: 0
                            'min_child_weight': [ 3,5,7, 10],
                                                                                             n_estimators: 200
                            'subsample': [0.7, 0.8, 0.9, 1],
XGBoost
                                                                                             min_child_weight: 7
                            'colsample_bytree': [0.6, 0.7, 0.8, 0.9],
                                                                                             max depth: 8
                            'reg_alpha': [0, 0.01, 0.1, 1],
                                                                                             learning_rate: 0.15
                            'reg_lambda': [0.1, 1, 5, 10],
                                                                                             gamma: 0.5
                            'gamma': [0, 0.1, 0.5, 1],
                                                                                             colsample bytree: 0.6
                            'scale_pos_weight': [1, 2, 3] # Handle class imbalance
                                                                                              XGBoost: Successfully tuned
                    },
                    'LightGBM': {
                         'model': lgb.LGBMClassifier(random_state=42, verbose=-1),
                         'params': {
                             'n_estimators': [10, 25, 50, 100],
                                                                                           BEST HYPERPARAMETERS:
                                                                                             subsample: 0.6
                             'learning_rate': [0.01, 0.05, 0.1, 0.2],
                                                                                             reg_lambda: 0.1
                             'max_depth': [2, 3, 4, 5],
                                                                                             reg_alpha: 0.01
                             'num_leaves': [7, 15, 31, 63],
                                                                                             num_leaves: 15
                             'min_child_samples': [10, 20, 30, 40],
                                                                                             n_estimators: 10
                                                                                             min_split_gain: 1
                             'min_split_gain': [0.01, 0.1, 0.5, 1],
LightGBM
                                                                                             min_child_weight: 0.001
                             'subsample': [0.6, 0.7, 0.8, 0.9],
                                                                                             min_child_samples: 30
                             'colsample_bytree': [0.3, 0.5, 0.7, 0.9],
                                                                                             max_depth: 2
                                                                                             learning_rate: 0.2
                             'reg_alpha': [0, 0.01, 0.1, 1],
                                                                                             colsample_bytree: 0.5
                             'reg_lambda': [0.1, 1, 5, 10],
                                                                                             class_weight: balanced
                             'min_child_weight': [0.001, 0.01, 0.1, 1],
                                                                                              LightGBM: Successfully tuned
                             'class_weight': ['balanced', None]
                        }
                    },
```





```
'model': CatBoostClassifier(random_state=42, verbose=False),
                                                                                                       BEST HYPERPARAMETERS:
                            'params': {
                                                                                                         subsample: 0.6
                                'iterations': [10, 25, 50, 100],
                                                                                                         min_data_in_leaf: 40
                                 'learning_rate': [0.01, 0.05, 0.1, 0.2],
                                                                                                         learning_rate: 0.2
                                 'depth': [2, 3, 4, 5],
                                                                                                         12_leaf_reg: 5
                                'min_data_in_leaf': [10, 20, 30, 40],
                                                                                                         iterations: 10
                                '12_leaf_reg': [1, 3, 5, 10, 20],
CatBoost
                                                                                                         depth: 4
                                 'subsample': [0.6, 0.7, 0.8, 0.9],
                                                                                                         colsample_bylevel: 0.5
                                'colsample_bylevel': [0.3, 0.5, 0.7, 0.9],
                                                                                                         class_weights: [1, 3]
                                 'border_count': [32, 64, 128],
                                                                                                         border_count: 64
                                'bagging_temperature': [0, 0.5, 1],
                                                                                                         bagging_temperature: 0
                                'class_weights': [[1, 1], [1, 2], [1, 3]] # Handle imbalance
                                                                                                          CatBoost: Successfully tuned
                            }
                       },
                        # ADABOOST
                        'AdaBoost': {
                             'model': AdaBoostClassifier(random_state=42),
                                                                                                       BEST HYPERPARAMETERS:
                             'params': {
                                                                                                         algorithm: SAMME
                                  'n_estimators': [10, 25, 50, 100],
AdaBoost
                                                                                                         learning_rate: 0.01
                                  'learning_rate': [0.01, 0.1, 0.5, 1.0, 2.0],
                                                                                                         n_estimators: 10
                                                                                                          AdaBoost: Successfully tuned
                                  'algorithm': ['SAMME', 'SAMME.R']
                             }
                       },
                       'Logistic Regression': {
                            'model': LogisticRegression(random_state=42, max_iter=1000, class_weight='balanced'),
                                                                                                       BEST HYPERPARAMETERS:
                            'params': {
                                                                                                         solver: liblinear
                              'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2', 'elasticnet'],
Logistic
                                                                                                         penalty: 12
                                                                                                         11_ratio: 0.5
                              'solver': ['liblinear', 'saga'],
                                                                                                         fit_intercept: False
Regression
                              'l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9],
                                                                                                         C: 1
                              'fit_intercept': [True, False]
                                                                                                          Logistic Regression: Successfully tuned
                          }
                       },
                        'SGD Classifier': {
                           'model': SGDClassifier(random_state=42, max_iter=1000, class_weight='balanced'),
                                                                                                       BEST HYPERPARAMETERS:
                            'params': {
                                                                                                         validation_fraction: 0.3
                               'alpha': [0.0001, 0.001, 0.01, 0.1, 1],
                                                                                                         penalty: 11
SGD
                               'penalty': ['l1', 'l2', 'elasticnet'],
                                                                                                          learning_rate: constant
                               'l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9],
                                                                                                         11 ratio: 0.3
                              'learning_rate': ['constant', 'optimal', 'invscaling', 'adaptive'],
                                                                                                         eta0: 0.01
                               'eta0': [0.001, 0.01, 0.1, 1],
Classifier
                                                                                                         early stopping: True
                               'early_stopping': [True, False],
                                                                                                         alpha: 0.01
                               'validation_fraction': [0.1, 0.2, 0.3]
                                                                                                          SGD Classifier: Successfully tuned
                           }
                       },
```





```
'SVM (RBF)': {
                                                                                          BEST HYPERPARAMETERS:
                        'model': SVC(random_state=42, probability=True, class_weight='balanced'),
                                                                                            shrinking: False
                        'params': {
                                                                                            kernel: rbf
                           'C': [0.001, 0.01, 0.1, 1, 10, 100],
                            'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1],
                                                                                            gamma: 0.1
SVM (RBF)
                            'kernel': ['rbf'],
                                                                                            cache_size: 500
                           'shrinking': [True, False],
                                                                                            C: 100
                            'cache_size': [200, 500, 1000]
                                                                                             SVM (RBF): Successfully tuned
                        }
                    'K-Nearest Neighbors': {
                         'model': KNeighborsClassifier(),
                         'params': {
                                                                                         BEST HYPERPARAMETERS:
                             'n_neighbors': [3, 5, 7, 9, 11, 15, 21, 31],
                                                                                           weights: distance
                             'weights': ['uniform', 'distance'],
                                                                                           p: 2
                                                                                           n_neighbors: 21
                             'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
KNN
                                                                                           metric: euclidean
                             'metric': ['euclidean', 'manhattan', 'minkowski'],
                                                                                           leaf_size: 40
                                                                                           algorithm: auto
                             'p': [1, 2, 3],
                                                                                            K-Nearest Neighbors: Successfully tuned
                             'leaf_size': [10, 20, 30, 40, 50]
                        }
                    },
                    'Linear Discriminant Analysis': {
                         'model': LinearDiscriminantAnalysis(),
                                                                                          BEST HYPERPARAMETERS:
                         'params': {
                             'solver': ['svd', 'lsqr', 'eigen'],
Linear
                                                                                             n_components: None
                             'shrinkage': [None, 'auto', 0.1, 0.3, 0.5, 0.7, 0.9],
                                                                                             priors: None
                             'priors': [None],
                                                                                             shrinkage: 0.3
                             'n_components': [None, 1, 2, 3]
                         }
                                                                                             solver: lsqr
                    },
```





# **Performance Metrics Comparison Report (2 Marks):**

Model	Optimized Metric					
		precision	recall f	1-score su	pport	
	0	1.00	0.88	0.94	50	
	1	0.83	1.00	0.91	30	
	accuracy			0.93	80	
Random Forest	macro avg weighted avg	0.92 0.94	0.94 0.93	0.92 0.93	80 80	
	Test Accuracy Precision: 0.9 Recall: 0.925 F1 Score: 0.9 ROC AUC: 0.99 PR AUC: 0.988	9375 0 260 33				
		precision	recal	l f1-score	support	
	0	0.96	0.9	8 0.97	50	
	1	0.97	0.9	0.95	30	
XGBoost	accuracy			0.96	80	
	macro avg	0.96	0.9		80	
	weighted avg				80	





	Test Accuracy: 0 Precision: 0.962 Recall: 0.9625 F1 Score: 0.9624 ROC AUC: 0.9933 PR AUC: 0.9905	6			
		precision	recall	f1-score	support
	0	1.00	1.00	1.00	50
	1	1.00	1.00	1.00	30
	accuracy			1.00	80
	macro avg	1.00	1.00		
	weighted avg				
LightGBM					
	Test Accuracy: 1. Precision: 1.0000 Recall: 1.0000 F1 Score: 1.0000 ROC AUC: 1.0000 PR AUC: 1.0000				
		precision	recall	f1-score	support
	0	1.00	1.00	1.00	50
	1	1.00	1.00	1.00	30
Logistic Regression	accuracy			1.00	80
	macro avg	1.00	1.00	1.00	80
	weighted avg	1.00	1.00	1.00	80





	Test Accuracy: 1. Precision: 1.0000 Recall: 1.0000 F1 Score: 1.0000 ROC AUC: 1.0000 PR AUC: 1.0000				
		recision	recall	f1-score	support
	0	1.00	1.00	1.00	50
	1	1.00	1.00	1.00	30
	accuracy			1.00	80
	macro avg	1.00	1.00		
SVM(RBF)	weighted avg	1.00			80
	Test Accuracy: 1 Precision: 1.0000 Recall: 1.0000 F1 Score: 1.0000 ROC AUC: 1.0000 PR AUC: 1.0000	9			
		orecision	recall	f1-score	support
	0	0.96	0.96	0.96	50
	1	0.93	0.93	0.93	30
Gradient Boosting					
	accuracy			0.95	80
	macro avg	0.95	0.95	0.95	80
	weighted avg	0.95	0.95	0.95	80





	Test Accuracy: Precision: 0.9 Recall: 0.9500 F1 Score: 0.950 ROC AUC: 0.994 PR AUC: 0.9920	500 00 7				
		precision	recall	†1-score	support	
	0	1.00	0.94	0.97	50	
	1	0.91	1.00	0.95	30	
	accuracy			0.96	80	
	macro avg	0.95	0.97		80	
	weighted avg				80	
CatBoost						
	Test Accuracy: Precision: 0.9 Recall: 0.9625 F1 Score: 0.96 ROC AUC: 0.997 PR AUC: 0.9958	659 528 73				
		precision	recall	f1-score	support	
	0	0.95	0.84	0.89	50	
	1		0.93		30	
Decision Tree	accuracy	,		0.88	80	
	macro avg		0.89		80	
	weighted avg		0.88		80	





	Test Accuracy: Precision: 0.8 Recall: 0.8750 F1 Score: 0.87 ROC AUC: 0.886 PR AUC: 0.7509	883 67 7			
		precision	recal	l f1-score	support
	e	1.00	1.00	0 1.00	50
	1		1.00		
	accuracy	,		1.00	80
	macro avg	1.00	1.00	0 1.00	80
KNN	weighted avg	1.00	1.00	0 1.00	80
	Test Accur Precision: Recall: 1. F1 Score: ROC AUC: 1 PR AUC: 1.	0000 1.0000 0000			
		precision	recall	f1-score	support
Linear Discriminant	0	1.00	1.00	1.00	50
Analysis	1	1.00	1.00	1.00	30
	accuracy			1.00	80
	macro avg	1.00	1.00	1.00	80
	weighted avg	1.00	1.00	1.00	80





	Test Accuracy Precision: 1 Recall: 1.00 F1 Score: 1.0 ROC AUC: 1.00 PR AUC: 1.00	. 0000 00 0000 000			
		precision	recall	f1-score	support
	0	0.95	0.84	0.89	50
	1		0.93		
	accuracy			0.88	80
	macro avg	0.87	0.89	0.87	80
AdaBoost	weighted avg	0.89	0.88	0.88	80
	Test Accuracy: Precision: 0.88 Recall: 0.8750 F1 Score: 0.876 ROC AUC: 0.8867 PR AUC: 0.7509	7			
		precision	recall	f1-score	support
	0	1.00	1.00	1.00	50
SGD Classifier	1		1.00		30
	accuracy			1.00	80
	macro avg	1.00	1.00	1.00	80
	weighted avg	1.00	1.00	1.00	80





	Test Accuracy: 1.0000 Precision: 1.0000 Recall: 1.0000 F1 Score: 1.0000 ROC AUC: 1.0000 PR AUC: 1.0000						
		pr	recision	recall	f1-score	support	
		0	1.00	0.84	0.91	50	
		1	0.79	1.00	0.88	30	
	accurac	V			0.90	80	
	macro av		0.89	0.92	0.90	80	
	weighted av	g	0.92	0.90	0.90	80	
Gaussian Naive Bayes							
	Test Acc Precision Recall: F1 Score ROC AUC: PR AUC:	on: 0 0.90 e: 0.	9015 9016 9000 900				
			precision	reca	all f1-sc	ore sup	pport
		0	1.00	0.	.96 @	.98	50
		1	0.94	1.	.00 0	.97	30
	accur	acy			e	.97	80
Bernoulli Naive Bayes	macro	_	0.97	0.		.97	80
	weighted		0.98			.98	80





Test Accuracy: 0.9750

Precision: 0.9766 Recall: 0.9750 F1 Score: 0.9751 ROC AUC: 1.0000

PR AUC: 1.0000

## **Final Model Selection Justification (2 Marks):**

Final Model	Reasoning
	We aimed to identify the model with the lowest risk under each version. For this, we carefully noted the specific hyperparameters that had been tuned for each model so we could recreate and retrain them consistently. After rebuilding the models using those same hyperparameters, we trained them on the original dataset to reproduce the same performance results. Once that was done, we tested these models on a new, valid dataset containing data that wasn't part of the original training set.  Among all models tested in Version 1—which included both the original selected features and feature-engineered ones—the CatBoost model performed the best. It was chosen as the top model based on its ability to generalize well (low overfitting) and its high recall, which is especially important in healthcare applications where minimizing false negatives is
CatBoost	critical.





\_\_\_\_\_

FINAL LEADERBOARD - RANKED BY ACCURACY (VERSION 1)

-----

1 Rank 1: CatBoost - 1.0000 (100.00%)
2 Rank 2: XGBoost - 0.9600 (96.00%)
3 Rank 3: Gradient Boosting - 0.9000 (90.00%)
. Rank 4: Random Forest - 0.8400 (84.00%)

\_\_\_\_\_

#### Leaderboard as DataFrame:

	Rank	Model	Accuracy	Accuracy_Percentage
0	1	CatBoost	1.00	100.0
1	2	XGBoost	0.96	96.0
2	3	Gradient Boosting	0.90	90.0
3	4	Random Forest	0.84	84.0