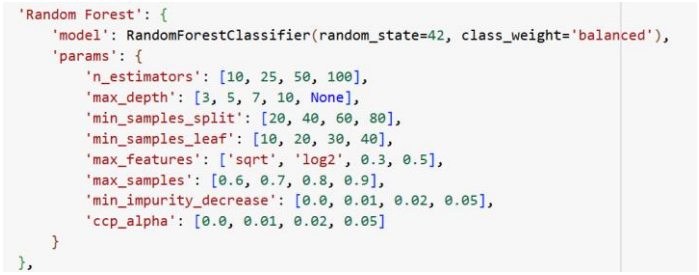
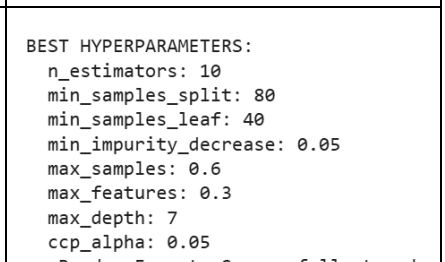
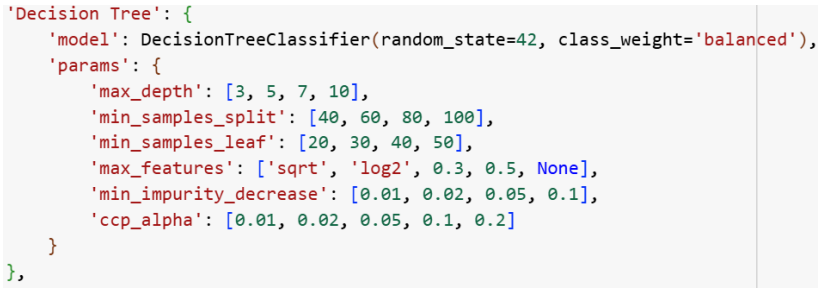
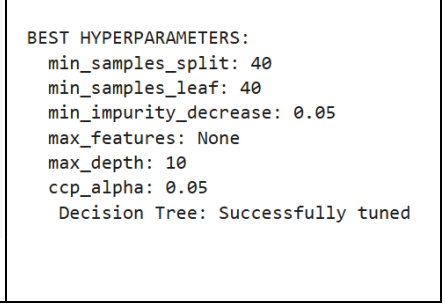
Model Optimization and Tuning Phase Template

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		
Decision Tree		

Gradient Boosting	'Gradient Boosting': {
	'model': GradientBoostingClassifier(random_state=42),
	'params': {
	'n_estimators': [10, 25, 50, 100],
	'learning_rate': [0.01, 0.05, 0.1, 0.2],
	'max_depth': [2, 3, 4, 5],
	'min_samples_split': [40, 60, 80],
	'min_samples_leaf': [20, 30, 40],
	'subsample': [0.6, 0.7, 0.8, 0.9],
	'max_features': ['sqrt', 'log2', 0.3, 0.5],
	'min_impurity_decrease': [0.01, 0.02, 0.05],
	'ccp_alpha': [0.0, 0.01, 0.02]
	}
	},
		BEST HYPERPARAMETERS:
		subsample: 0.6
		n_estimators: 100
		min_samples_split: 80
		min_samples_leaf: 40
		min_impurity_decrease: 0.05
		max_features: log2
		max_depth: 2
		learning_rate: 0.01
		ccp_alpha: 0.0
		Gradient Boosting: Successfully tuned

XGBoost	'XGBoost': {
	'model': xgb.XGBClassifier(random_state=42, eval_metric='logloss'),
	'params': {
	'n_estimators': [50, 100,150,200],
	'learning_rate': [0.1, 0.15, 0.2, 0.3],
	'max_depth': [5,6,7,8],
	'min_child_weight': [3,5,7, 10],
	'subsample': [0.7, 0.8, 0.9, 1],
	'colsample_bytree': [0.6, 0.7, 0.8, 0.9],
	'reg_alpha': [0, 0.01, 0.1, 1],
	'reg_lambda': [0.1, 1, 5, 10],
	'gamma': [0, 0.1, 0.5, 1],
	'scale_pos_weight': [1, 2, 3] # Handle class imbalance
	}
	},
		BEST HYPERPARAMETERS:
		subsample: 0.9
		scale_pos_weight: 1
		reg_lambda: 10
		reg_alpha: 0
		n_estimators: 200
		min_child_weight: 7
		max_depth: 8
		learning_rate: 0.15
		gamma: 0.5
		colsample_bytree: 0.6
		XGBoost: Successfully tuned

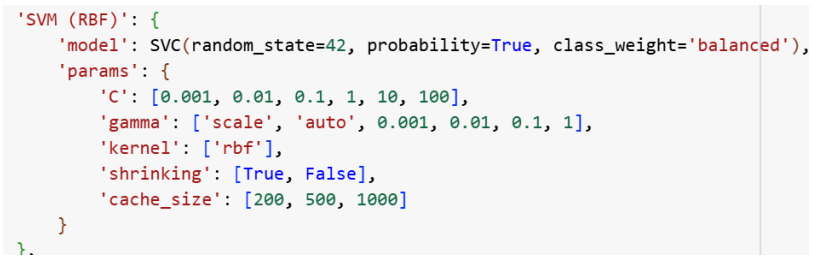
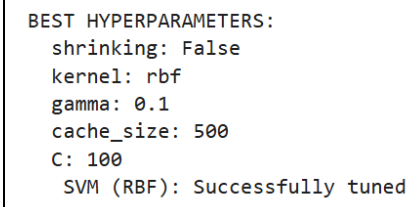
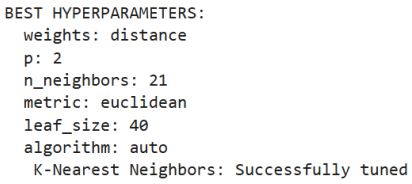
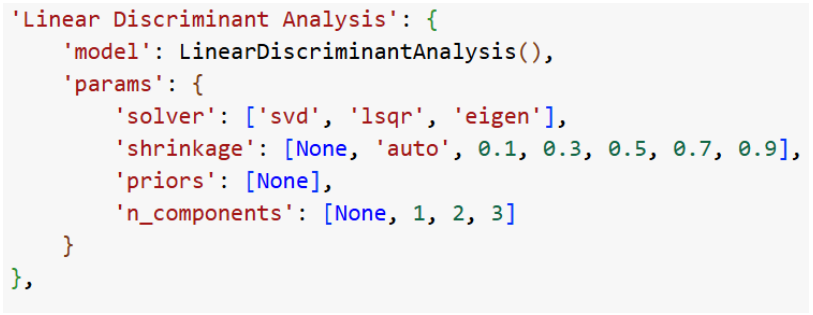
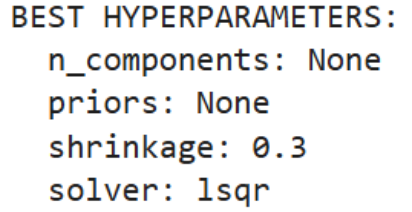
LightGBM	'LightGBM': {
	'model': lgb.LGBMClassifier(random_state=42, verbose=-1),
	'params': {
	'n_estimators': [10, 25, 50, 100],
	'learning_rate': [0.01, 0.05, 0.1, 0.2],
	'max_depth': [2, 3, 4, 5],
	'num_leaves': [7, 15, 31, 63],
	'min_child_samples': [10, 20, 30, 40],
	'min_split_gain': [0.01, 0.1, 0.5, 1],
	'subsample': [0.6, 0.7, 0.8, 0.9],
	'colsample_bytree': [0.3, 0.5, 0.7, 0.9],
	'reg_alpha': [0, 0.01, 0.1, 1],
	'reg_lambda': [0.1, 1, 5, 10],
	'min_child_weight': [0.001, 0.01, 0.1, 1],
	'class_weight': ['balanced', None]
	}
	},
		BEST HYPERPARAMETERS:
		subsample: 0.6
		reg_lambda: 0.1
		reg_alpha: 0.01
		num_leaves: 15
		n_estimators: 10
		min_split_gain: 1
		min_child_weight: 0.001
		min_child_samples: 30
		max_depth: 2
		learning_rate: 0.2
		colsample_bytree: 0.5
		class_weight: balanced
		LightGBM: Successfully tuned

CatBoost	`'CatBoost': {`
	`'model': CatBoostClassifier(random_state=42, verbose=False),`
	`'params': {`
	`'iterations': [10, 25, 50, 100],`
	`'learning_rate': [0.01, 0.05, 0.1, 0.2],`
	`'depth': [2, 3, 4, 5],`
	`'min_data_in_leaf': [10, 20, 30, 40],`
	`'l2_leaf_reg': [1, 3, 5, 10, 20],`
	`'subsample': [0.6, 0.7, 0.8, 0.9],`
	`'colsample_bylevel': [0.3, 0.5, 0.7, 0.9],`
	`'border_count': [32, 64, 128],`
	`'bagging_temperature': [0, 0.5, 1],`
	`'class_weights': [[1, 1], [1, 2], [1, 3]]  # Handle imbalance`
	`}`
	`},`
		BEST HYPERPARAMETERS:
		subsample: 0.6
		min_data_in_leaf: 40
		learning_rate: 0.2
		l2_leaf_reg: 5
		iterations: 10
		depth: 4
		colsample_bylevel: 0.5
		class_weights: [1, 3]
		border_count: 64
		bagging_temperature: 0
		CatBoost: Successfully tuned

AdaBoost	`# ADABOOST`
	`'AdaBoost': {`
	`'model': AdaBoostClassifier(random_state=42),`
	`'params': {`
	`'n_estimators': [10, 25, 50, 100],`
	`'learning_rate': [0.01, 0.1, 0.5, 1.0, 2.0],`
	`'algorithm': ['SAMME', 'SAMME.R']`
	`}`
	`},`
		BEST HYPERPARAMETERS:
		algorithm: SAMME
		learning_rate: 0.01
		n_estimators: 10
		AdaBoost: Successfully tuned

Logistic Regression	`'Logistic Regression': {`
	`'model': LogisticRegression(random_state=42, max_iter=1000, class_weight='balanced'),`
	`'params': {`
	`'C': [0.001, 0.01, 0.1, 1, 10, 100],`
	`'penalty': ['l1', 'l2', 'elasticnet'],`
	`'solver': ['liblinear', 'saga'],`
	`'l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9],`
	`'fit_intercept': [True, False]`
	`}`
	`},`
		BEST HYPERPARAMETERS:
		solver: liblinear
		penalty: l2
		l1_ratio: 0.5
		fit_intercept: False
		C: 1
		Logistic Regression: Successfully tuned

SGD Classifier	`'SGD Classifier': {`
	`'model': SGDClassifier(random_state=42, max_iter=1000, class_weight='balanced'),`
	`'params': {`
	`'alpha': [0.0001, 0.001, 0.01, 0.1, 1],`
	`'penalty': ['l1', 'l2', 'elasticnet'],`
	`'l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9],`
	`'learning_rate': ['constant', 'optimal', 'invscaling', 'adaptive'],`
	`'eta0': [0.001, 0.01, 0.1, 1],`
	`'early_stopping': [True, False],`
	`'validation_fraction': [0.1, 0.2, 0.3]`
	`}`
	`},`
		BEST HYPERPARAMETERS:
		validation_fraction: 0.3
		penalty: l1
		learning_rate: constant
		l1_ratio: 0.3
		eta0: 0.01
		early_stopping: True
		alpha: 0.01
		SGD Classifier: Successfully tuned

SVM (RBF)		
KNN		
Linear		

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Random Forest	precision recall f1-score support 0 1.00 0.88 0.94 50 1 0.83 1.00 0.91 30 accuracy 0.93 80 macro avg 0.92 0.94 0.92 80 weighted avg 0.94 0.93 0.93 80 Test Accuracy: 0.9250 Precision: 0.9375 Recall: 0.9250 F1 Score: 0.9260 ROC AUC: 0.9933 PR AUC: 0.9889
XGBoost	precision recall f1-score support 0 0.96 0.98 0.97 50 1 0.97 0.93 0.95 30 accuracy 0.96 80 macro avg 0.96 0.96 0.96 80 weighted avg 0.96 0.96 0.96 80

<table>
<tr><td></td><td>

Test Accuracy: 0.9625
Precision: 0.9626
Recall: 0.9625
F1 Score: 0.9624
ROC AUC: 0.9933
PR AUC: 0.9905

</td></tr>
<tr><td>LightGBM</td><td>

	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	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

</td></tr>
<tr><td>Logistic Regression</td><td>

	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	1.00	80
weighted avg	1.00	1.00	1.00	80

</td></tr>
</table>

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

	Test Accuracy: 0.9500 Precision: 0.9500 Recall: 0.9500 F1 Score: 0.9500 ROC AUC: 0.9947 PR AUC: 0.9920
CatBoost	precision recall f1-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 0.96 80 weighted avg 0.97 0.96 0.96 80 Test Accuracy: 0.9625 Precision: 0.9659 Recall: 0.9625 F1 Score: 0.9628 ROC AUC: 0.9973 PR AUC: 0.9958
Decision Tree	precision recall f1-score support 0 0.95 0.84 0.89 50 1 0.78 0.93 0.85 30 accuracy 0.88 80 macro avg 0.87 0.89 0.87 80 weighted avg 0.89 0.88 0.88 80

	Test Accuracy: 0.8750 Precision: 0.8883 Recall: 0.8750 F1 Score: 0.8767 ROC AUC: 0.8867 PR AUC: 0.7509
KNN	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 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
Linear Discriminant Analysis	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 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
AdaBoost	precision recall f1-score support 0 0.95 0.84 0.89 50 1 0.78 0.93 0.85 30 accuracy 0.88 80 macro avg 0.87 0.89 0.87 80 weighted avg 0.89 0.88 0.88 80 Test Accuracy: 0.8750 Precision: 0.8883 Recall: 0.8750 F1 Score: 0.8767 ROC AUC: 0.8867 PR AUC: 0.7509
SGD Classifier	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 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
Gaussian Naive Bayes	precision recall f1-score support 0 1.00 0.84 0.91 50 1 0.79 1.00 0.88 30 accuracy 0.90 80 macro avg 0.89 0.92 0.90 80 weighted avg 0.92 0.90 0.90 80 Test Accuracy: 0.9000 Precision: 0.9211 Recall: 0.9000 F1 Score: 0.9015 ROC AUC: 1.0000 PR AUC: 1.0000
Bernoulli Naive Bayes	precision recall f1-score support 0 1.00 0.96 0.98 50 1 0.94 1.00 0.97 30 accuracy 0.97 80 macro avg 0.97 0.98 0.97 80 weighted avg 0.98 0.97 0.98 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
CatBoost	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 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
```