Model Optimization and Tuning Phase Template

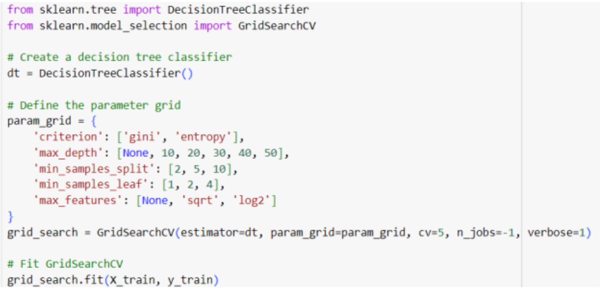
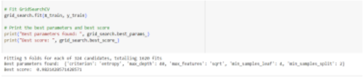
Date	16 JULY 2024
Team ID	SWTID1720075199
Project Title	Early Prediction Of Chronic Kidney Disease Using Machine Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

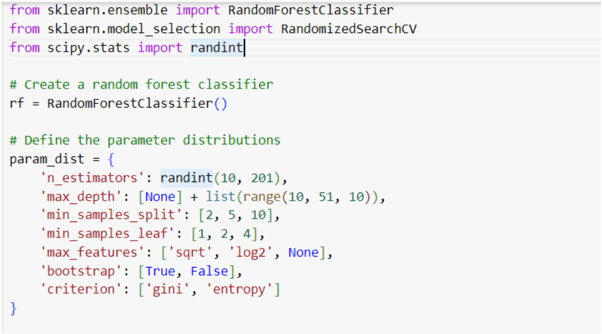
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

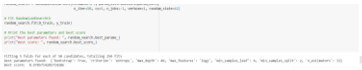
Model	Tuned Hyperparameters	Optimal Values

Decision Trees	
	from sklearn.tree import DecisionTreeClassifier
	from sklearn.model_selection import GridSearchCV
	
	# Create a decision tree classifier
	dt = DecisionTreeClassifier()
	
	# Define the parameter grid
	param_grid = {
	    'criterion': ['gini', 'entropy'],
	    'max_depth': [None, 10, 20, 30, 40, 50],
	    'min_samples_split': [2, 5, 10],
	    'min_samples_leaf': [1, 2, 4],
	    'max_features': [None, 'sqrt', 'log2']
	}
	grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=5, n_jobs=-1, verbose=1)
	
	# Fit GridSearchCV
	grid_search.fit(X_train, y_train)
		
		# Fit GridSearchCV
		grid_search.fit(X_train, y_train)
		
		# Print the best parameters and best score
		print("Best parameters found: ", grid_search.best_params_)
		print("Best score: ", grid_search.best_score_)
		
		Fitting 5 folds for each of 324 candidates, totalling 1620 fits
		Best parameters found: {'criterion': 'entropy', 'max_depth': 40, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2}
		Best score: 0.9821428571428571

Random Forest

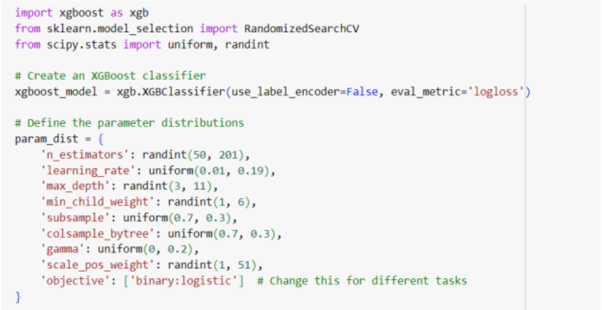


```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

# Create a random forest classifier
rf = RandomForestClassifier()

# Define the parameter distributions
param_dist = {
    'n_estimators': randint(10, 201),
    'max_depth': [None] + list(range(10, 51, 10)),
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True, False],
    'criterion': ['gini', 'entropy']
}
```

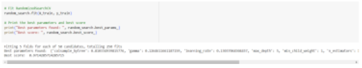


```
                                   n_iter=50, cv=5, n_jobs=-1, verbose=1, random_state=42)

# Fit RandomizedSearchCV
random_search.fit(X_train, y_train)

# Print the best parameters and best score
print("Best parameters found: ", random_search.best_params_)
print("Best score: ", random_search.best_score_)

Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best parameters found:  {'bootstrap': True, 'criterion': 'entropy', 'max_depth': 40, 'max_features': 'log2', 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': [illegible]}
Best score:  0.9785714285714286
```

XGboost



```
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform, randint

# Create an XGBoost classifier
xgboost_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Define the parameter distributions
param_dist = {
    'n_estimators': randint(50, 201),
    'learning_rate': uniform(0.01, 0.19),
    'max_depth': randint(3, 11),
    'min_child_weight': randint(1, 6),
    'subsample': uniform(0.7, 0.3),
    'colsample_bytree': uniform(0.7, 0.3),
    'gamma': uniform(0, 0.2),
    'scale_pos_weight': randint(1, 51),
    'objective': ['binary:logistic']  # Change this for different tasks
}
```

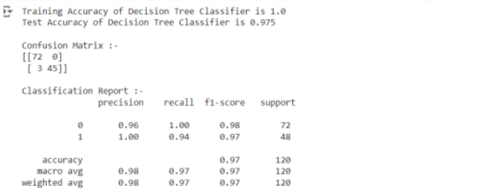


```
# Fit RandomizedSearchCV
random_search.fit(X_train, y_train)

# Print the best parameters and best score
print("Best parameters found: ", random_search.best_params_)
print("Best score: ", random_search.best_score_)

Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best parameters found:  {'colsample_bytree': [illegible], 'gamma': [illegible], 'learning_rate': [illegible], 'max_depth': 5, 'min_child_weight': 1, 'n_estimators': [illegible]
Best score:  0.9714285714285715
```

Performance Metrics Comparison Report (2 Marks):

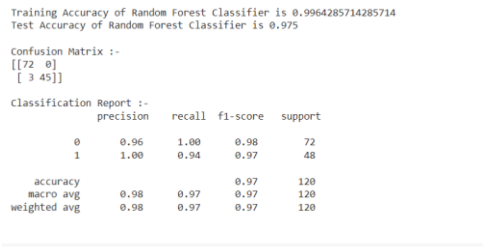
Model	Optimized Metric

Decision Tree

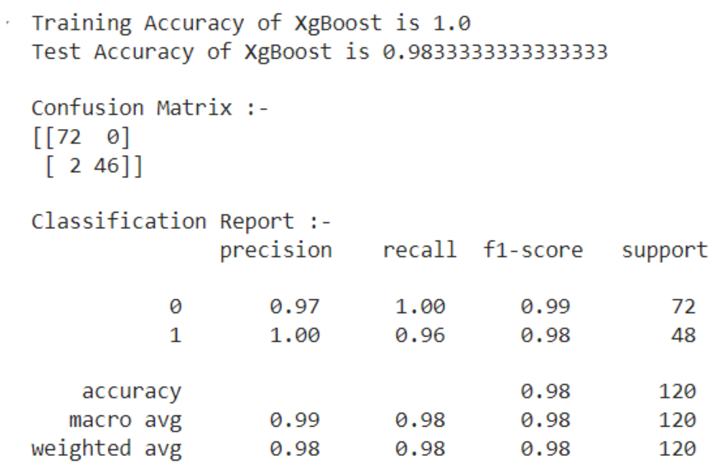


```
Training Accuracy of Decision Tree Classifier is 1.0
Test Accuracy of Decision Tree Classifier is 0.975

Confusion Matrix :-
[[72  0]
 [ 3 45]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.94      0.97        48

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```

Random Forest



```
Training Accuracy of Random Forest Classifier is 0.9964285714285714
Test Accuracy of Random Forest Classifier is 0.975

Confusion Matrix :-
[[72  0]
 [ 3 45]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.94      0.97        48

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```

XGBoost	

```
Training Accuracy of XgBoost is 1.0
Test Accuracy of XgBoost is 0.9833333333333333

Confusion Matrix :-
[[72  0]
 [ 2 46]]

Classification Report :-
               precision    recall  f1-score   support

           0       0.97      1.00      0.99        72
           1       1.00      0.96      0.98        48

    accuracy                           0.98       120
   macro avg       0.99      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
XgBoost	XgBoost was chosen as final model because it has highest accuracy and to minimise overfitting during hyperparameter training.