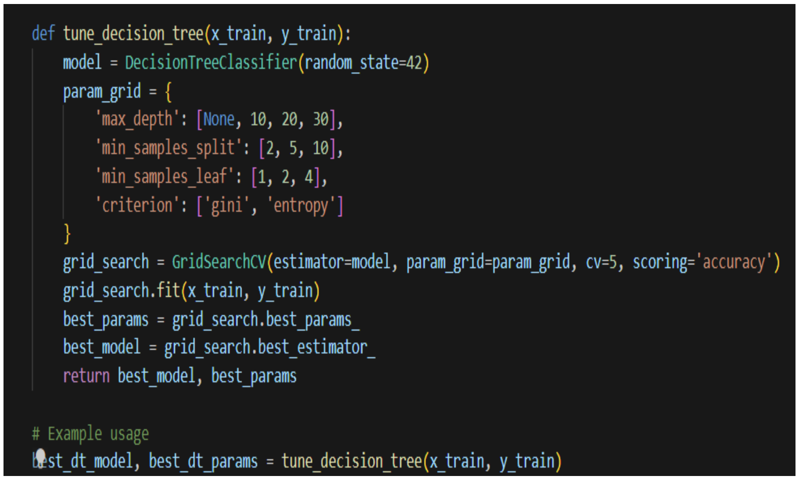
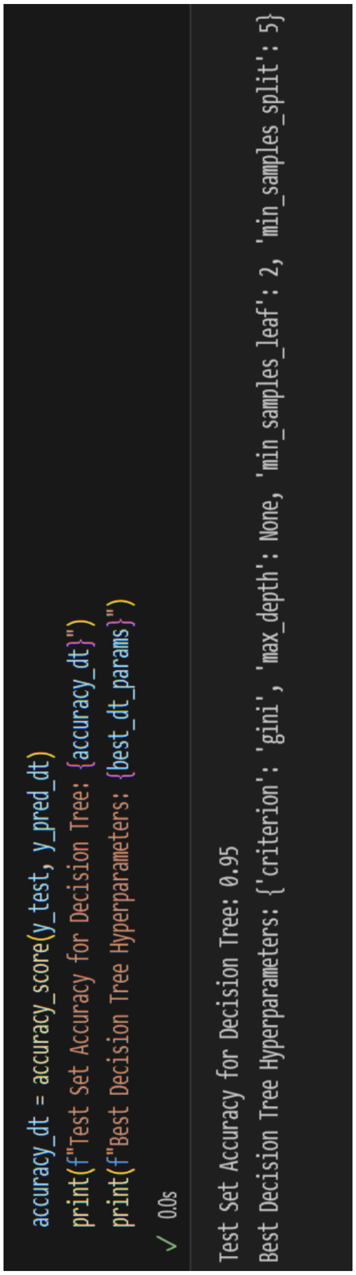
Model Optimization and Tuning Phase

Date	10 July 2024
Team ID	SWTID1721205662
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

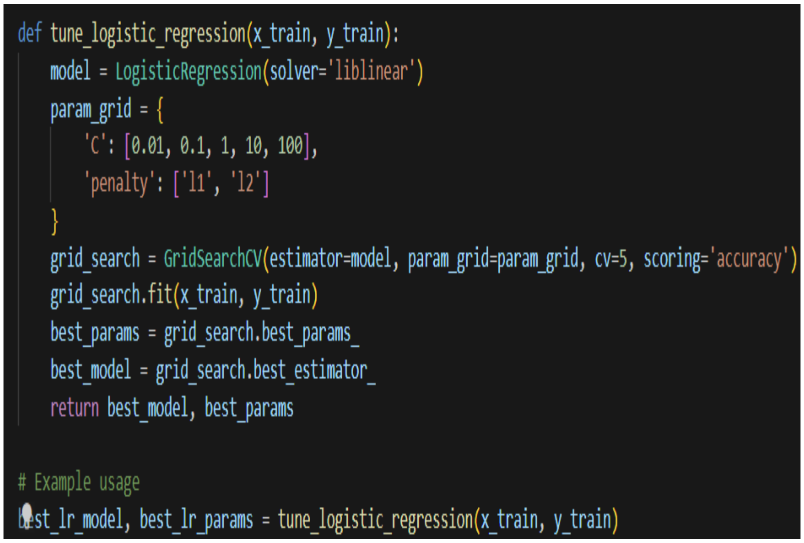
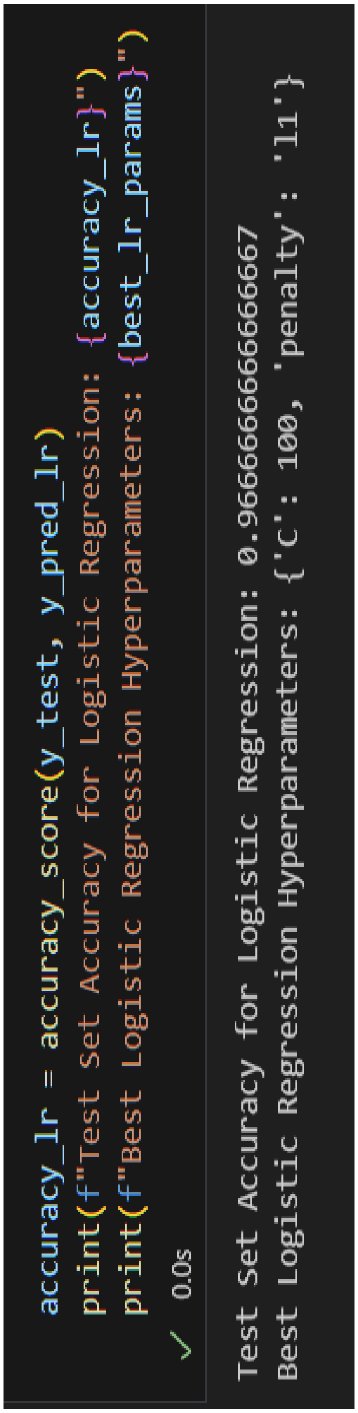
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values
Decision Trees Classifier		

```
def tune_decision_tree(x_train, y_train):
    model = DecisionTreeClassifier(random_state=42)
    param_grid = {
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'criterion': ['gini', 'entropy']
    }
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
    grid_search.fit(x_train, y_train)
    best_params = grid_search.best_params_
    best_model = grid_search.best_estimator_
    return best_model, best_params

# Example usage
best_dt_model, best_dt_params = tune_decision_tree(x_train, y_train)
```

```
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(f"Test Set Accuracy for Decision Tree: {accuracy_dt}")
print(f"Best Decision Tree Hyperparameters: {best_dt_params}")
✓ 0.0s

Test Set Accuracy for Decision Tree: 0.95
Best Decision Tree Hyperparameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5}
```

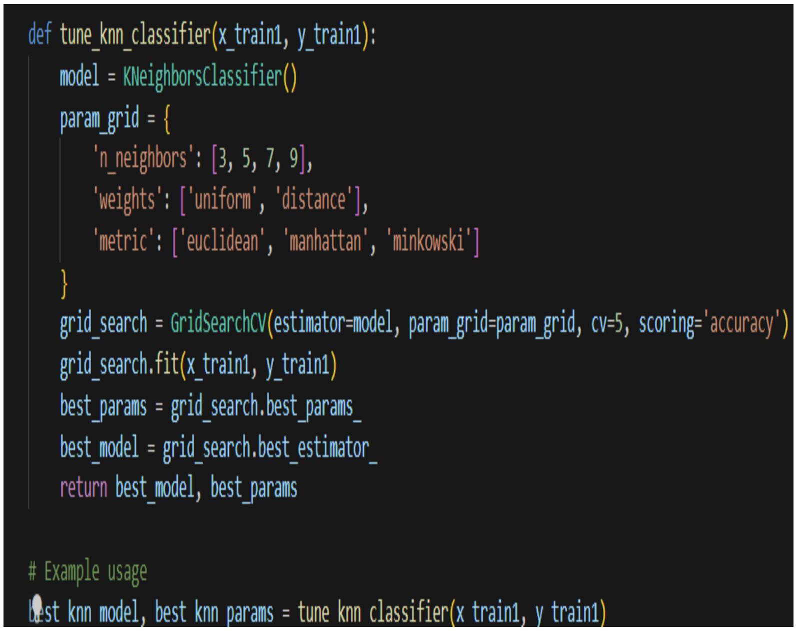
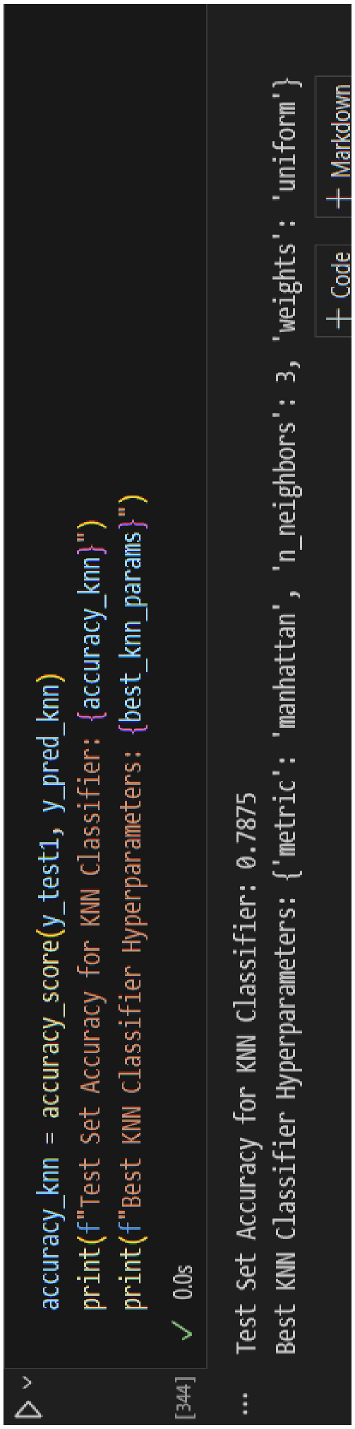
Logistic regression		

```python
def tune_logistic_regression(x_train, y_train):
    model = LogisticRegression(solver='liblinear')
    param_grid = {
        'C': [0.01, 0.1, 1, 10, 100],
        'penalty': ['l1', 'l2']
    }
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
    grid_search.fit(x_train, y_train)
    best_params = grid_search.best_params_
    best_model = grid_search.best_estimator_
    return best_model, best_params

# Example usage
best_lr_model, best_lr_params = tune_logistic_regression(x_train, y_train)
```

```python
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print(f"Test Set Accuracy for Logistic Regression: {accuracy_lr}")
print(f"Best Logistic Regression Hyperparameters: {best_lr_params}")
```

✓ 0.0s

```
Test Set Accuracy for Logistic Regression: 0.9666666666666667
Best Logistic Regression Hyperparameters: {'C': 100, 'penalty': 'l1'}
```

KNN Classifier		

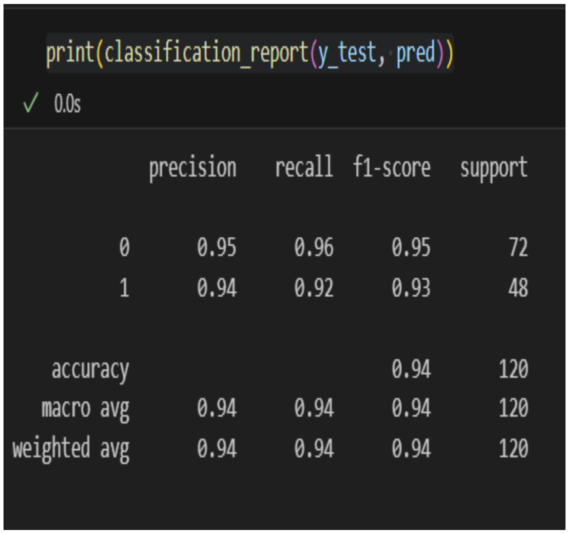
```python
def tune_knn_classifier(x_train1, y_train1):
    model = KNeighborsClassifier()
    param_grid = {
        'n_neighbors': [3, 5, 7, 9],
        'weights': ['uniform', 'distance'],
        'metric': ['euclidean', 'manhattan', 'minkowski']
    }
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
    grid_search.fit(x_train1, y_train1)
    best_params = grid_search.best_params_
    best_model = grid_search.best_estimator_
    return best_model, best_params

# Example usage
best_knn_model, best_knn_params = tune_knn_classifier(x_train1, y_train1)
```

```python
accuracy_knn = accuracy_score(y_test1, y_pred_knn)
print(f"Test Set Accuracy for KNN Classifier: {accuracy_knn}")
print(f"Best KNN Classifier Hyperparameters: {best_knn_params}")
```

```
Test Set Accuracy for KNN Classifier: 0.7875
Best KNN Classifier Hyperparameters: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'uniform'}
```

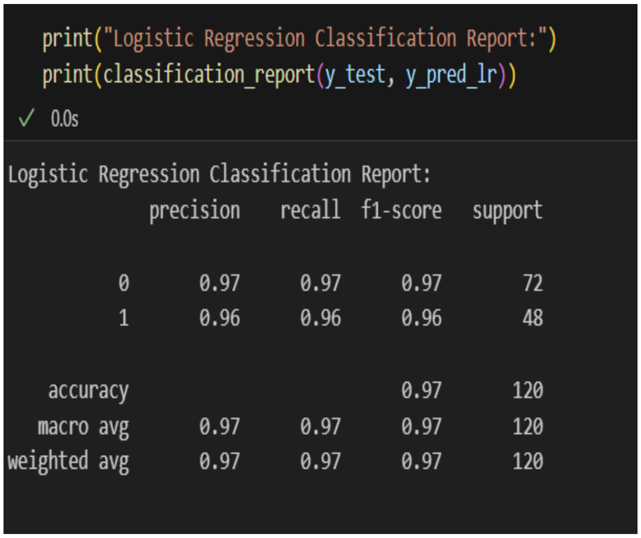
Performance Metrics Comparison Report (2 Marks):

<table>
<tr><th>Model</th><th>Baseline Metric</th><th>Optimized Metric</th></tr>
<tr><td>Logistic regression</td><td>



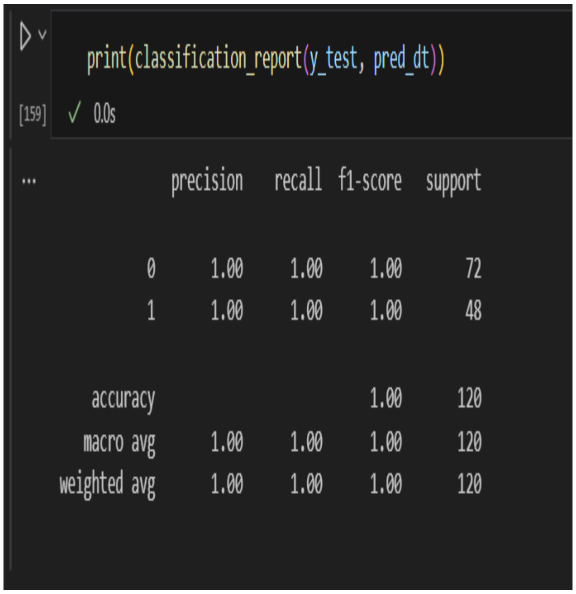
```
print(classification_report(y_test, pred))
✓ 0.0s

              precision    recall  f1-score   support

           0       0.95      0.96      0.95        72
           1       0.94      0.92      0.93        48

    accuracy                           0.94       120
   macro avg       0.94      0.94      0.94       120
weighted avg       0.94      0.94      0.94       120
```

</td><td>



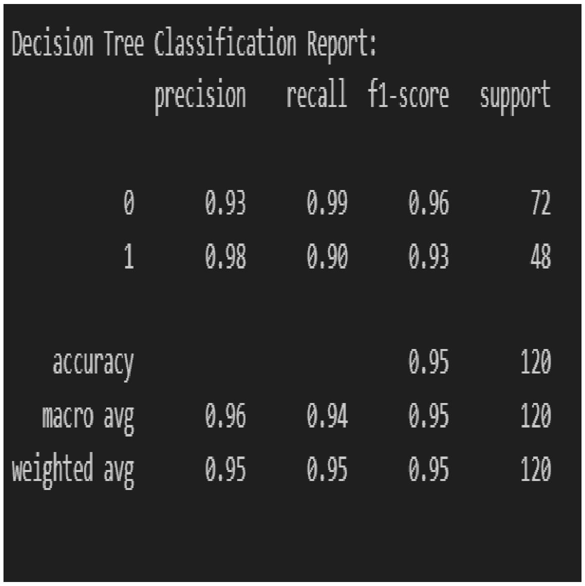
```
print("Logistic Regression Classification Report:")
print(classification_report(y_test, y_pred_lr))
✓ 0.0s

Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        72
           1       0.96      0.96      0.96        48

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

</td></tr>
<tr><td>Decision trees classifier</td><td>



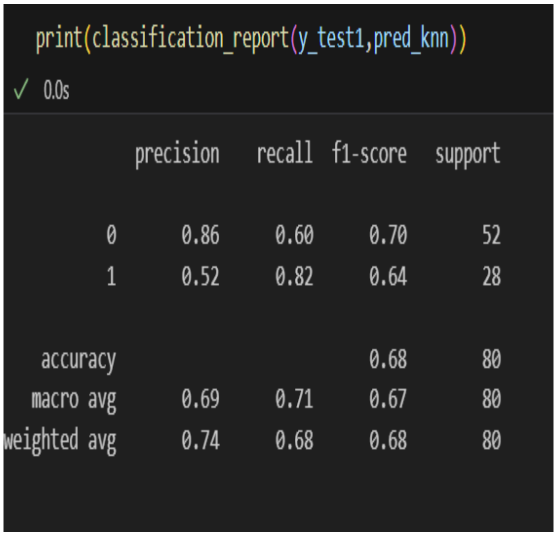
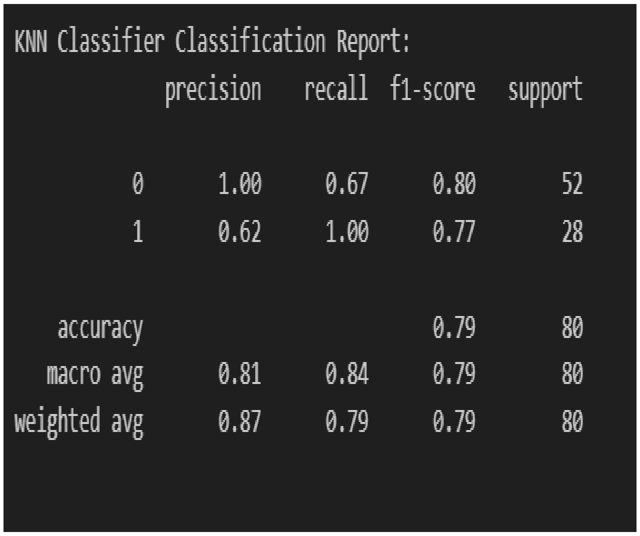
```
print(classification_report(y_test, pred_dt))
[159] ✓ 0.0s

...           precision    recall  f1-score   support

           0       1.00      1.00      1.00        72
           1       1.00      1.00      1.00        48

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

</td><td>



```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96        72
           1       0.98      0.90      0.93        48

    accuracy                           0.95       120
   macro avg       0.96      0.94      0.95       120
weighted avg       0.95      0.95      0.95       120
```

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

KNN classifier		

```
print(classification_report(y_test1,pred_knn))
✓ 0.0s

              precision    recall  f1-score   support

           0       0.86      0.60      0.70        52
           1       0.52      0.82      0.64        28

    accuracy                           0.68        80
   macro avg       0.69      0.71      0.67        80
weighted avg       0.74      0.68      0.68        80
```

```
KNN Classifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.67      0.80        52
           1       0.62      1.00      0.77        28

    accuracy                           0.79        80
   macro avg       0.81      0.84      0.79        80
weighted avg       0.87      0.79      0.79        80
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Logistic Regression	Logistic Regression is the better model for CKD prediction because of its simplicity, interpretability, efficiency, and ability to provide probabilistic outputs, which are crucial for clinical decision-making. Its performance, along with its ability to highlight important features, makes it an excellent choice for medical applications like CKD diagnosis.