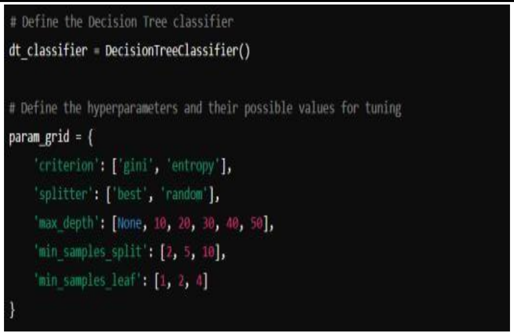
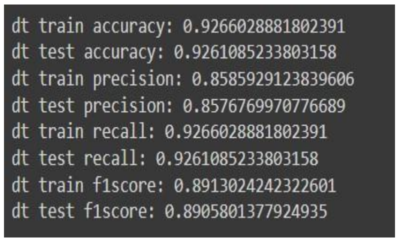
Model Optimization and Tuning Phase Report

Date	15 July 2024
Team ID	740116
Project Title	Sepsis Survival Minimal Clinical Records
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

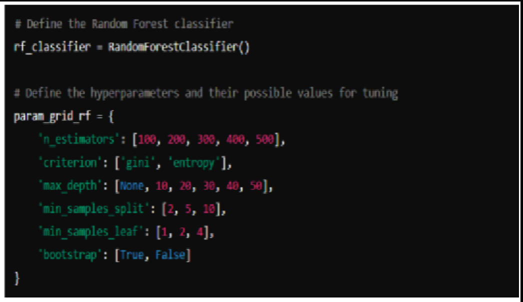
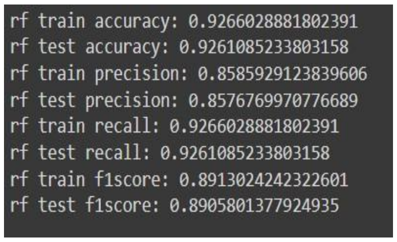
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values
Decision Tree		

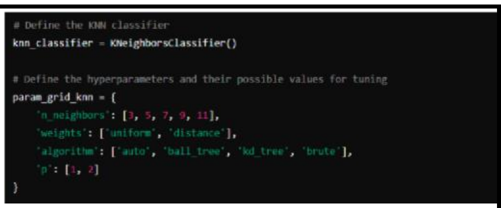
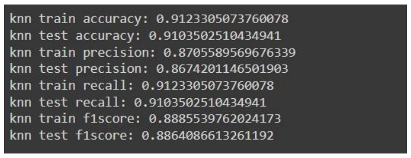
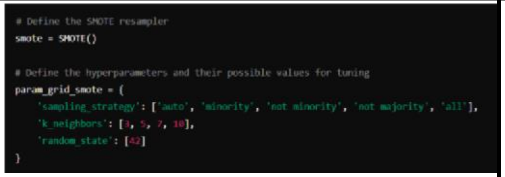
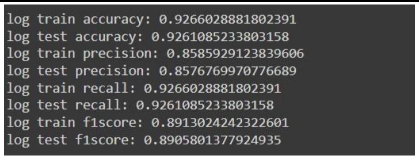
```
# Define the Decision Tree classifier
dt_classifier = DecisionTreeClassifier()


# Define the hyperparameters and their possible values for tuning
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
dt train accuracy: 0.9266028881802391
dt test accuracy: 0.9261085233803158
dt train precision: 0.8585929123839606
dt test precision: 0.8576769970776689
dt train recall: 0.9266028881802391
dt test recall: 0.9261085233803158
dt train f1score: 0.8913024242322601
dt test f1score: 0.8905801377924935
```

Random Forest		

```
# Define the Random Forest classifier
rf_classifier = RandomForestClassifier()

# Define the hyperparameters and their possible values for tuning
param_grid_rf = {
    'n_estimators': [100, 200, 300, 400, 500],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
```

```
rf train accuracy: 0.9266028881802391
rf test accuracy: 0.9261085233803158
rf train precision: 0.8585929123839606
rf test precision: 0.8576769970776689
rf train recall: 0.9266028881802391
rf test recall: 0.9261085233803158
rf train f1score: 0.8913024242322601
rf test f1score: 0.8905801377924935
```

KNN		
Logistic Regresson		

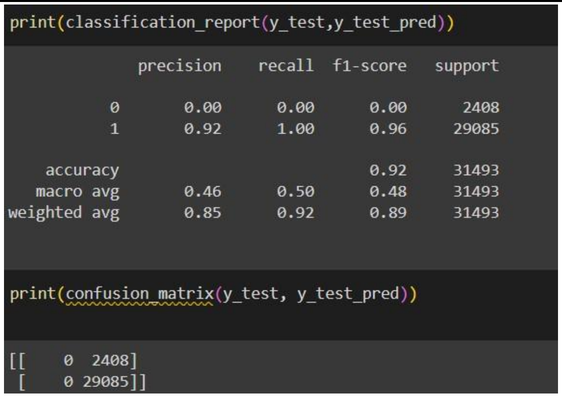
```
# Define the KNN classifier
knn_classifier = KNeighborsClassifier()

# Define the hyperparameters and their possible values for tuning
param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'p': [1, 2]
}
```

```
knn train accuracy: 0.9123305073760078
knn test accuracy: 0.9103502510434941
knn train precision: 0.8705589569676339
knn test precision: 0.8674201146501903
knn train recall: 0.9123305073760078
knn test recall: 0.9103502510434941
knn train f1score: 0.8885539762024173
knn test f1score: 0.8864086613261192
```

```
# Define the SMOTE resampler
smote = SMOTE()

# Define the hyperparameters and their possible values for tuning
param_grid_smote = {
    'sampling_strategy': ['auto', 'minority', 'not minority', 'not majority', 'all'],
    'k_neighbors': [3, 5, 7, 10],
    'random_state': [42]
}
```

```
log train accuracy: 0.9266028881802391
log test accuracy: 0.9261085233803158
log train precision: 0.8585929123839606
log test precision: 0.8576769970776689
log train recall: 0.9266028881802391
log test recall: 0.9261085233803158
log train f1score: 0.8913024242322601
log test f1score: 0.8905801377924935
```

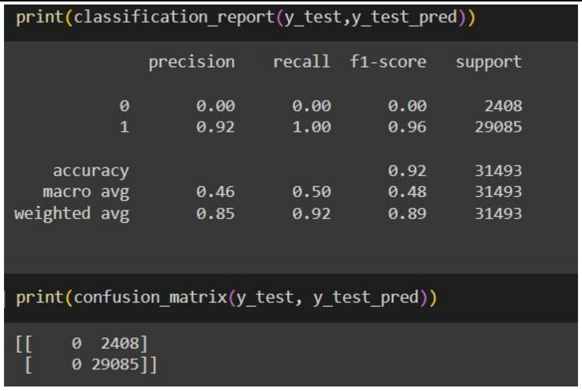
Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Decision Tree	

```
print(classification_report(y_test,y_test_pred))

              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2408
           1       0.92      1.00      0.96     29085

    accuracy                           0.92     31493
   macro avg       0.46      0.50      0.48     31493
weighted avg       0.85      0.92      0.89     31493
```

```
print(confusion_matrix(y_test, y_test_pred))

[[    0  2408]
 [    0 29085]]
```

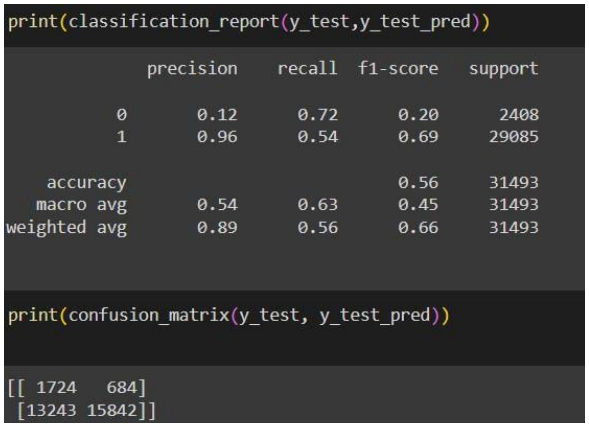
Random Forest



```
print(classification_report(y_test,y_test_pred))

              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2408
           1       0.92      1.00      0.96     29085

    accuracy                           0.92     31493
   macro avg       0.46      0.50      0.48     31493
weighted avg       0.85      0.92      0.89     31493
```

```
print(confusion_matrix(y_test, y_test_pred))

[[    0  2408]
 [    0 29085]]
```

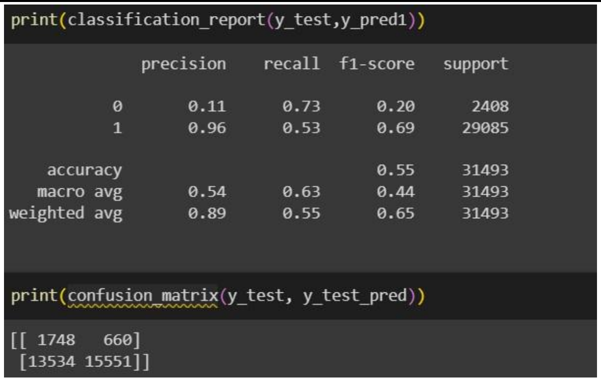
KNN



```
print(classification_report(y_test,y_test_pred))

              precision    recall  f1-score   support

           0       0.12      0.72      0.20      2408
           1       0.96      0.54      0.69     29085

    accuracy                           0.56     31493
   macro avg       0.54      0.63      0.45     31493
weighted avg       0.89      0.56      0.66     31493
```

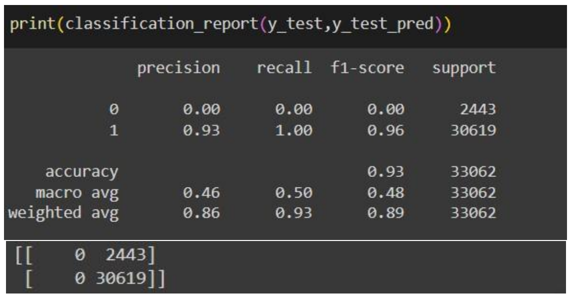
```
print(confusion_matrix(y_test, y_test_pred))

[[ 1724   684]
 [13243 15842]]
```

SMOTE



```
print(classification_report(y_test,y_pred1))

              precision    recall  f1-score   support

           0       0.11      0.73      0.20      2408
           1       0.96      0.53      0.69     29085

    accuracy                           0.55     31493
   macro avg       0.54      0.63      0.44     31493
weighted avg       0.89      0.55      0.65     31493
```

```
print(confusion_matrix(y_test, y_test_pred))

[[ 1748   660]
 [13534 15551]]
```

Logistic Regression	

```
print(classification_report(y_test,y_test_pred))
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2443
           1       0.93      1.00      0.96     30619

    accuracy                           0.93     33062
   macro avg       0.46      0.50      0.48     33062
weighted avg       0.86      0.93      0.89     33062

[[    0  2443]
 [    0 30619]]
```

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Random Forest	The decision to select the Random Forest model after applying SMOTE was driven by the need to address class imbalance, leverage the robustness and performance of an ensemble method, gain insights into feature importance, and ensure reliable and correct predictions. This combination aims to enhance the overall predictive capability of the sepsis survival prediction system, contributing to better clinical outcomes.