Model Optimization and Tuning Phase Template

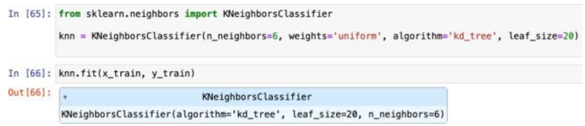
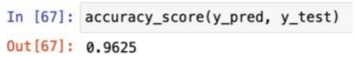
Date	15 MARCH 2024
Team ID	LTVIP2024TMID25011
Project Title	Early Prediction Of Chronic Kidney Disease Using Machine Learning
Maximum Marks	10 Marks

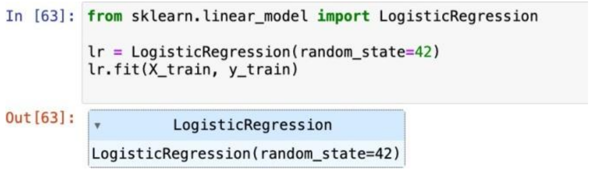
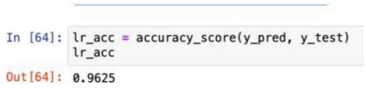
Model Optimization and Tuning Phase

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):

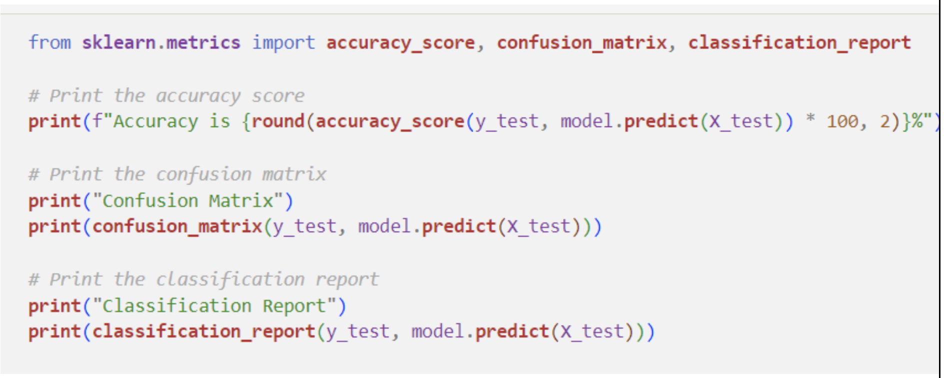
Model	Tuned Hyperparameters	Optimal Values

KNN	
	In [65]: `from sklearn.neighbors import KNeighborsClassifier`
	`knn = KNeighborsClassifier(n_neighbors=6, weights='uniform', algorithm='kd_tree', leaf_size=20)`
	In [66]: `knn.fit(x_train, y_train)`
	Out[66]: KNeighborsClassifier
	`KNeighborsClassifier(algorithm='kd_tree', leaf_size=20, n_neighbors=6)`
		
		In [67]: `accuracy_score(y_pred, y_test)`
		Out[67]: 0.9625

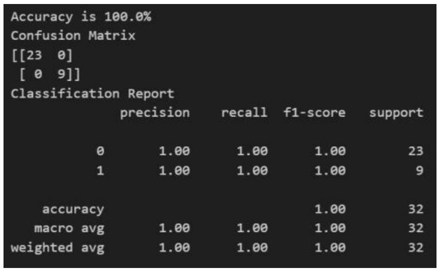
LOGISTIC
REGRESSION
	
	In [63]: `from sklearn.linear_model import LogisticRegression`
	`lr = LogisticRegression(random_state=42)`
	`lr.fit(X_train, y_train)`
	Out[63]: LogisticRegression
	`LogisticRegression(random_state=42)`
		
		In [64]: `lr_acc = accuracy_score(y_pred, y_test)`
		`lr_acc`
		Out[64]: 0.9625

Performance Metrics Comparison Report (2 Marks):

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr><td>KNN</td><td>



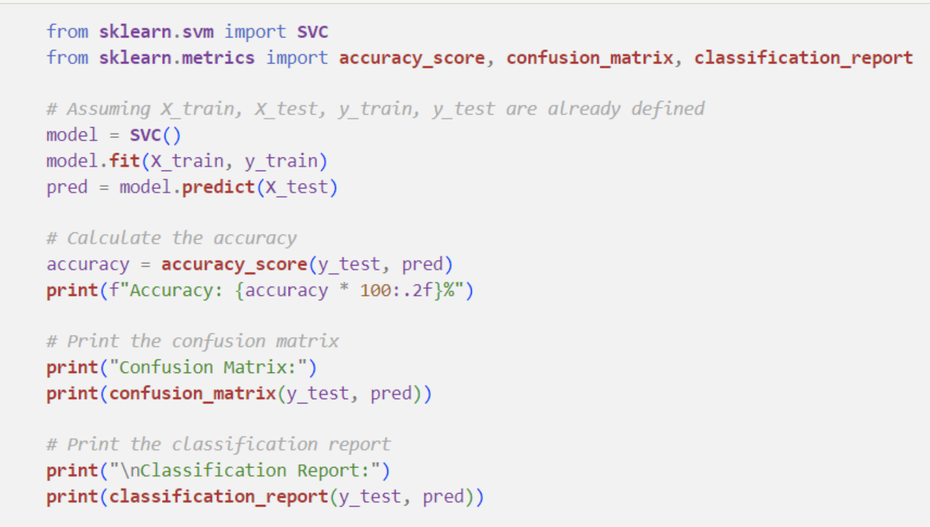
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Print the accuracy score
print(f"Accuracy is {round(accuracy_score(y_test, model.predict(X_test)) * 100, 2)}%")

# Print the confusion matrix
print("Confusion Matrix")
print(confusion_matrix(y_test, model.predict(X_test)))

# Print the classification report
print("Classification Report")
print(classification_report(y_test, model.predict(X_test)))
```



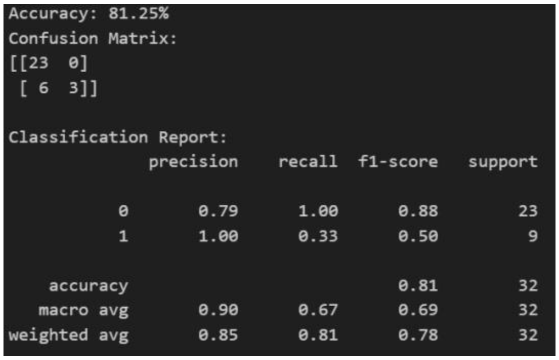
```
Accuracy is 100.0%
Confusion Matrix
[[23  0]
 [ 0  9]]
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00         9

    accuracy                           1.00        32
   macro avg       1.00      1.00      1.00        32
weighted avg       1.00      1.00      1.00        32
```

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

SVM



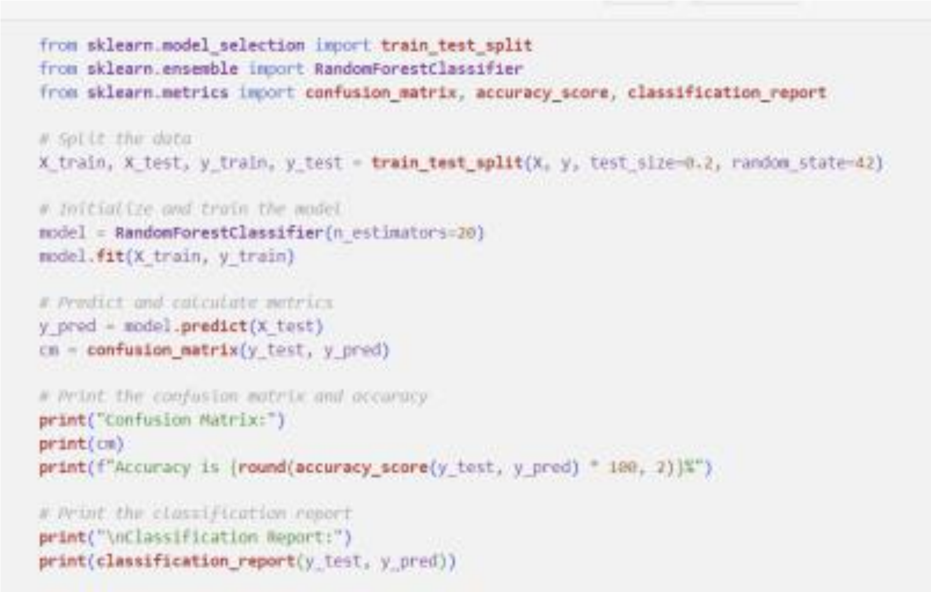
```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Assuming X_train, X_test, y_train, y_test are already defined
model = SVC()
model.fit(X_train, y_train)
pred = model.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, pred))

# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, pred))
```



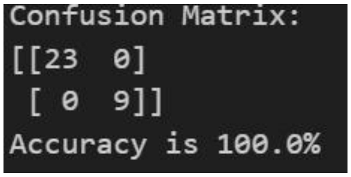
```
Accuracy: 81.25%
Confusion Matrix:
[[23  0]
 [ 6  3]]

Classification Report:
              precision    recall  f1-score   support

           0       0.79      1.00      0.88        23
           1       1.00      0.33      0.50         9

    accuracy                           0.81        32
   macro avg       0.90      0.67      0.69        32
weighted avg       0.85      0.81      0.78        32
```

RANDOM

FOREST



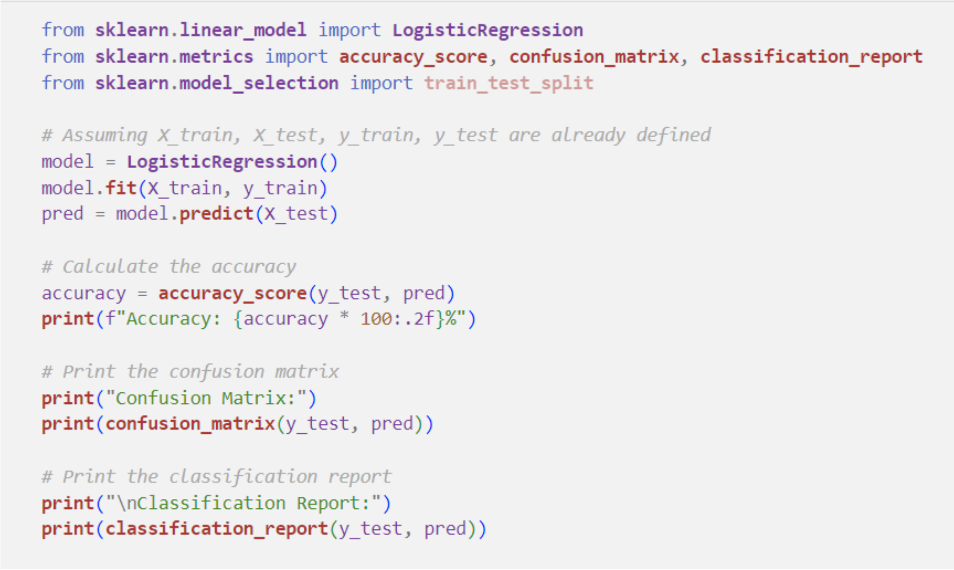
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the model
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)

# Predict and calculate metrics
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

# Print the confusion matrix and accuracy
print("Confusion Matrix:")
print(cm)
print(f"Accuracy is {round(accuracy_score(y_test, y_pred) * 100, 2)}%")

# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```



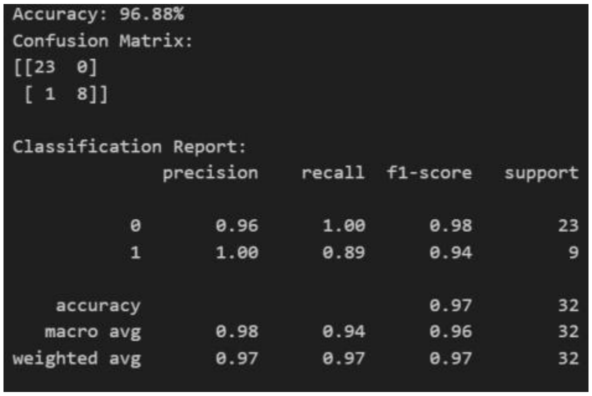
```
Confusion Matrix:
[[23  0]
 [ 0  9]]
Accuracy is 100.0%
```

LOGISTIC

REGRESSION

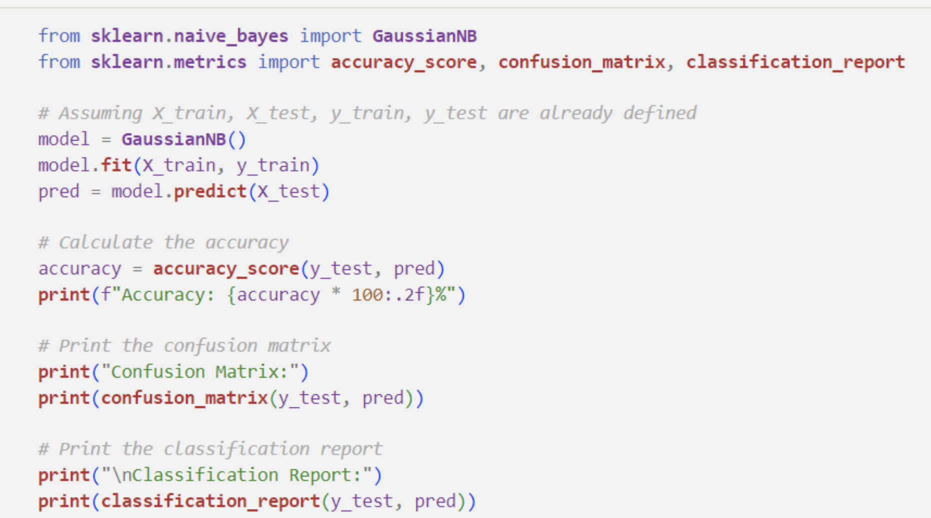


```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

# Assuming X_train, X_test, y_train, y_test are already defined
model = LogisticRegression()
model.fit(X_train, y_train)
pred = model.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, pred))

# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, pred))
```

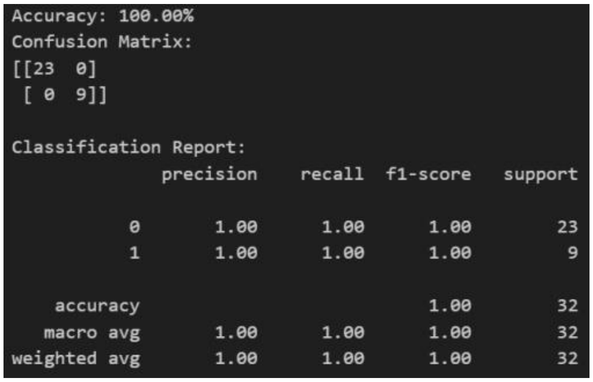


```
Accuracy: 96.88%
Confusion Matrix:
[[23  0]
 [ 1  8]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        23
           1       1.00      0.89      0.94         9

    accuracy                           0.97        32
   macro avg       0.98      0.94      0.96        32
weighted avg       0.97      0.97      0.97        32
```

NAIVE BAYES



```python
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Assuming X_train, X_test, y_train, y_test are already defined
model = GaussianNB()
model.fit(X_train, y_train)
pred = model.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, pred))

# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, pred))
```



```
Accuracy: 100.00%
Confusion Matrix:
[[23  0]
 [ 0  9]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00         9

    accuracy                           1.00        32
   macro avg       1.00      1.00      1.00        32
weighted avg       1.00      1.00      1.00        32
```

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Gradient Boosting	The Gradient Boosting model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.