Model Optimization and Tuning Phase Template

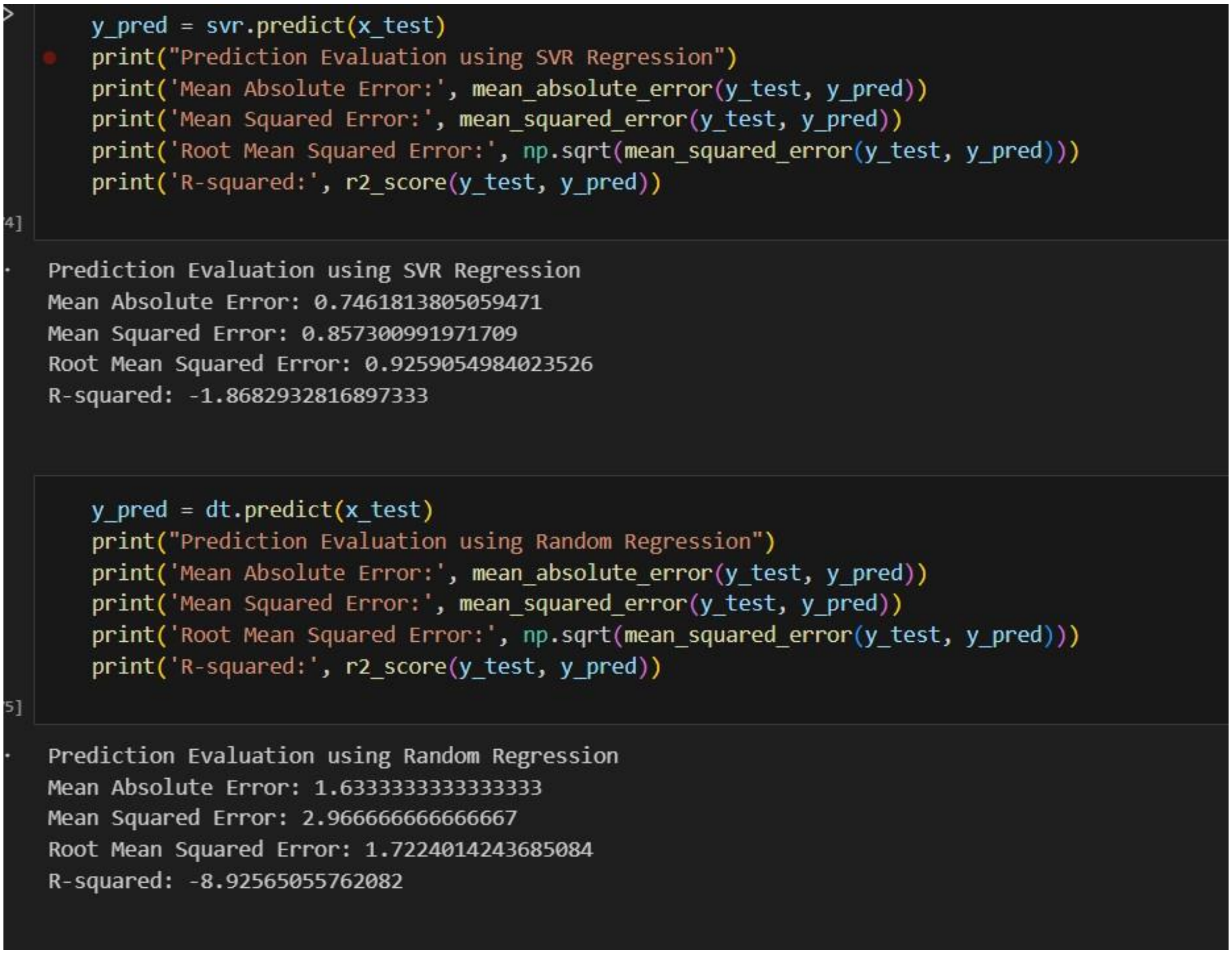
Date	09 JULY 2024
Team ID	740660
Project Title	Evolving efficient classification patterns in Lymphography
Maximum Marks	10 Marks

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):

crop

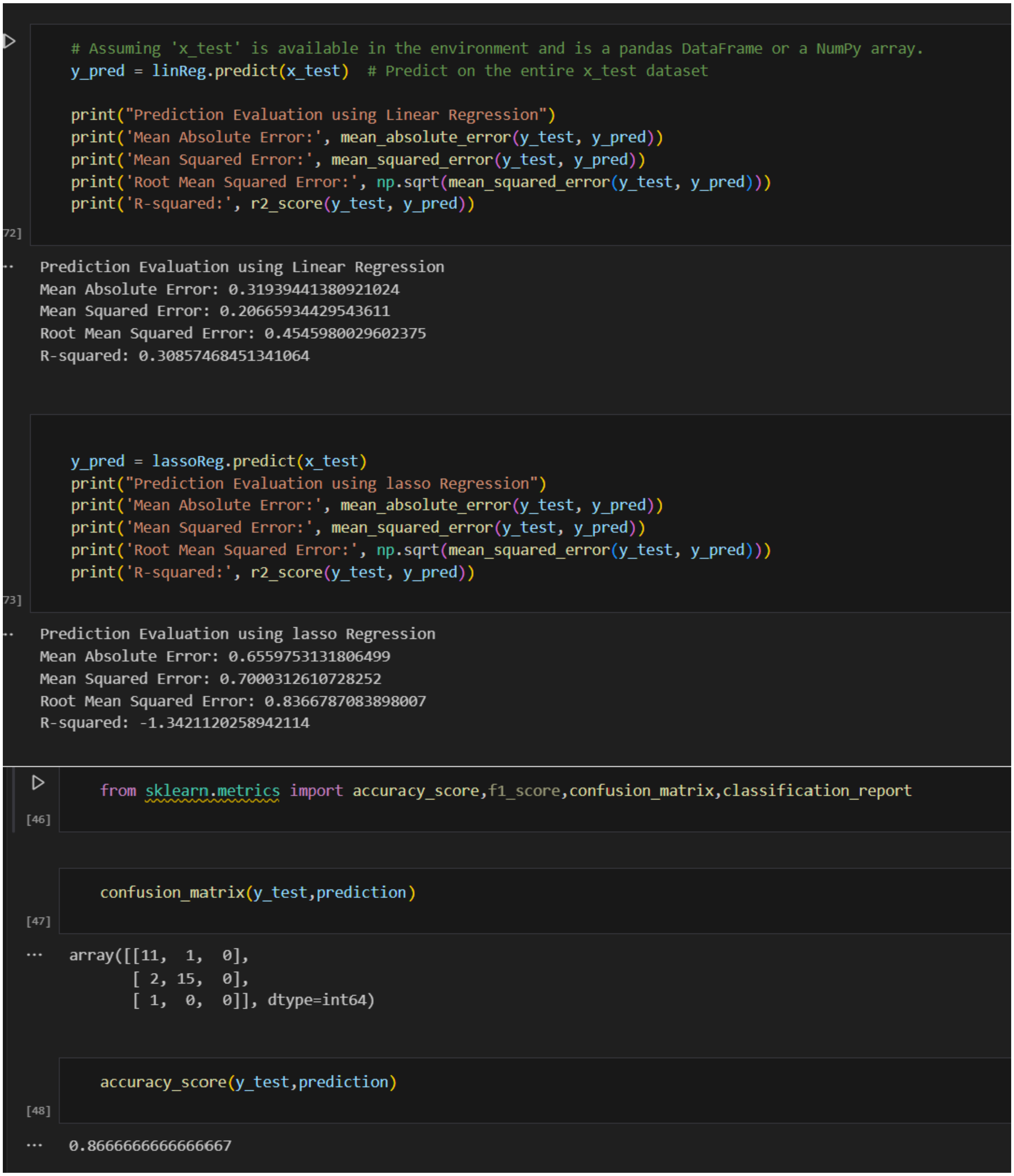


```
y_pred = svr.predict(x_test)
print("Prediction Evaluation using SVR Regression")
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using SVR Regression
Mean Absolute Error: 0.7461813805059471
Mean Squared Error: 0.857300991971709
Root Mean Squared Error: 0.9259054984023526
R-squared: -1.8682932816897333
```

```
y_pred = dt.predict(x_test)
print("Prediction Evaluation using Random Regression")
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using Random Regression
Mean Absolute Error: 1.6333333333333333
Mean Squared Error: 2.966666666666667
Root Mean Squared Error: 1.7224014243685084
R-squared: -8.92565055762082
```



```python
# Assuming 'x_test' is available in the environment and is a pandas DataFrame or a NumPy array.
y_pred = linReg.predict(x_test)  # Predict on the entire x_test dataset

print("Prediction Evaluation using Linear Regression")
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using Linear Regression
Mean Absolute Error: 0.31939441380921024
Mean Squared Error: 0.20665934429543611
Root Mean Squared Error: 0.4545980029602375
R-squared: 0.30857468451341064
```

```python
y_pred = lassoReg.predict(x_test)
print("Prediction Evaluation using lasso Regression")
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared:', r2_score(y_test, y_pred))
```

```
Prediction Evaluation using lasso Regression
Mean Absolute Error: 0.6559753131806499
Mean Squared Error: 0.7000312610728252
Root Mean Squared Error: 0.8366787083898007
R-squared: -1.3421120258942114
```

```python
from sklearn.metrics import accuracy_score,f1_score,confusion_matrix,classification_report
```

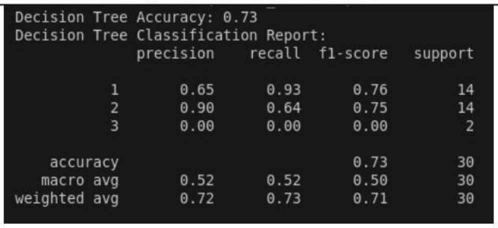
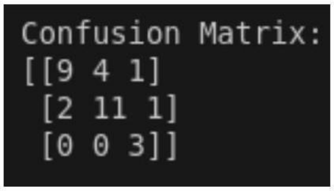
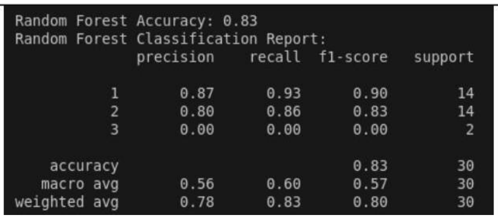
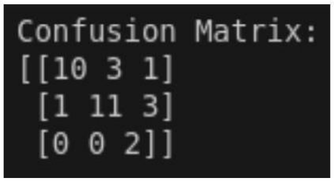
```python
confusion_matrix(y_test,prediction)
```

```
array([[11,  1,  0],
       [ 2, 15,  0],
       [ 1,  0,  0]], dtype=int64)
```

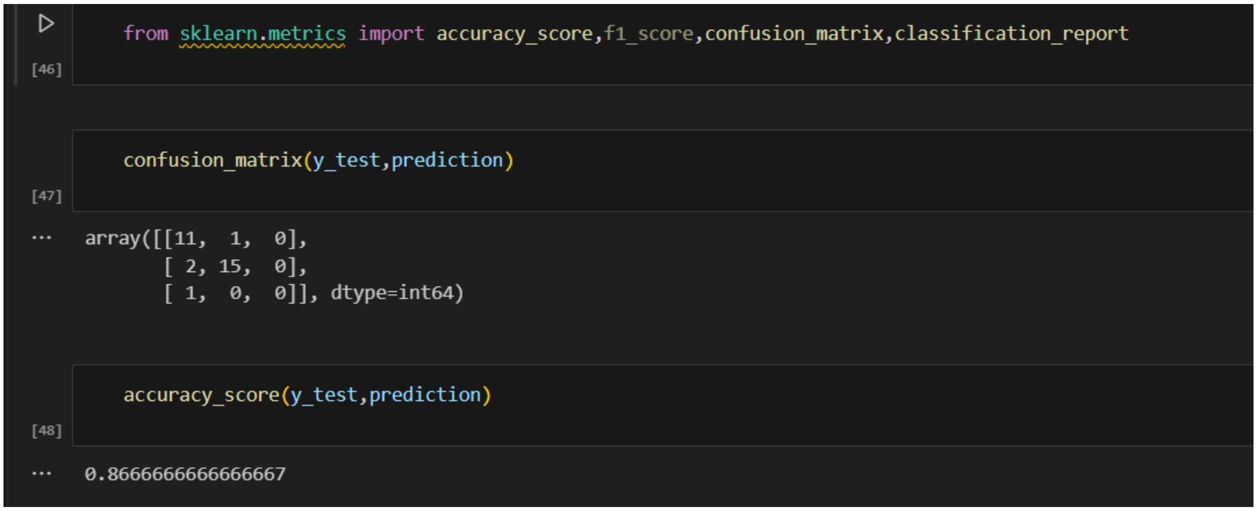
```python
accuracy_score(y_test,prediction)
```

```
0.8666666666666667
```

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric	Confusion Matrix
Decision Tree	 Decision Tree Accuracy: 0.73 Decision Tree Classification Report: precision recall f1-score support 1 0.65 0.93 0.76 14 2 0.90 0.64 0.75 14 3 0.00 0.00 0.00 2 accuracy 0.73 30 macro avg 0.52 0.52 0.50 30 weighted avg 0.72 0.73 0.71 30	 Confusion Matrix: [[9 4 1] [2 11 1] [0 0 3]]
Random Forest	 Random Forest Accuracy: 0.83 Random Forest Classification Report: precision recall f1-score support 1 0.87 0.93 0.90 14 2 0.80 0.86 0.83 14 3 0.00 0.00 0.00 2 accuracy 0.83 30 macro avg 0.56 0.60 0.57 30 weighted avg 0.78 0.83 0.80 30	 Confusion Matrix: [[10 3 1] [1 11 3] [0 0 2]]

Final Model Selection Justification (2 Marks):



```
from sklearn.metrics import accuracy_score,f1_score,confusion_matrix,classification_report
```

```
confusion_matrix(y_test,prediction)
```

```
array([[11,  1,  0],
       [ 2, 15,  0],
       [ 1,  0,  0]], dtype=int64)
```

```
accuracy_score(y_test,prediction)
```

```
0.8666666666666667
```