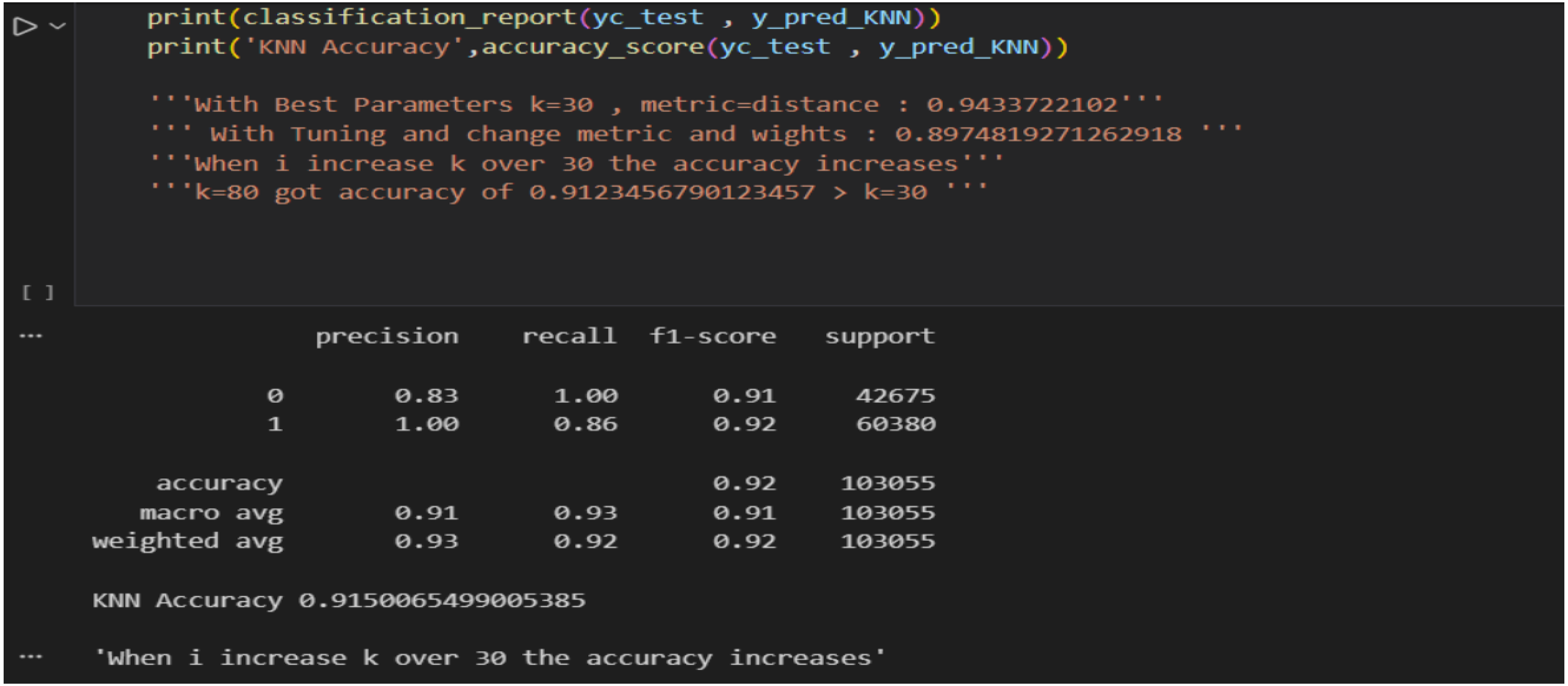
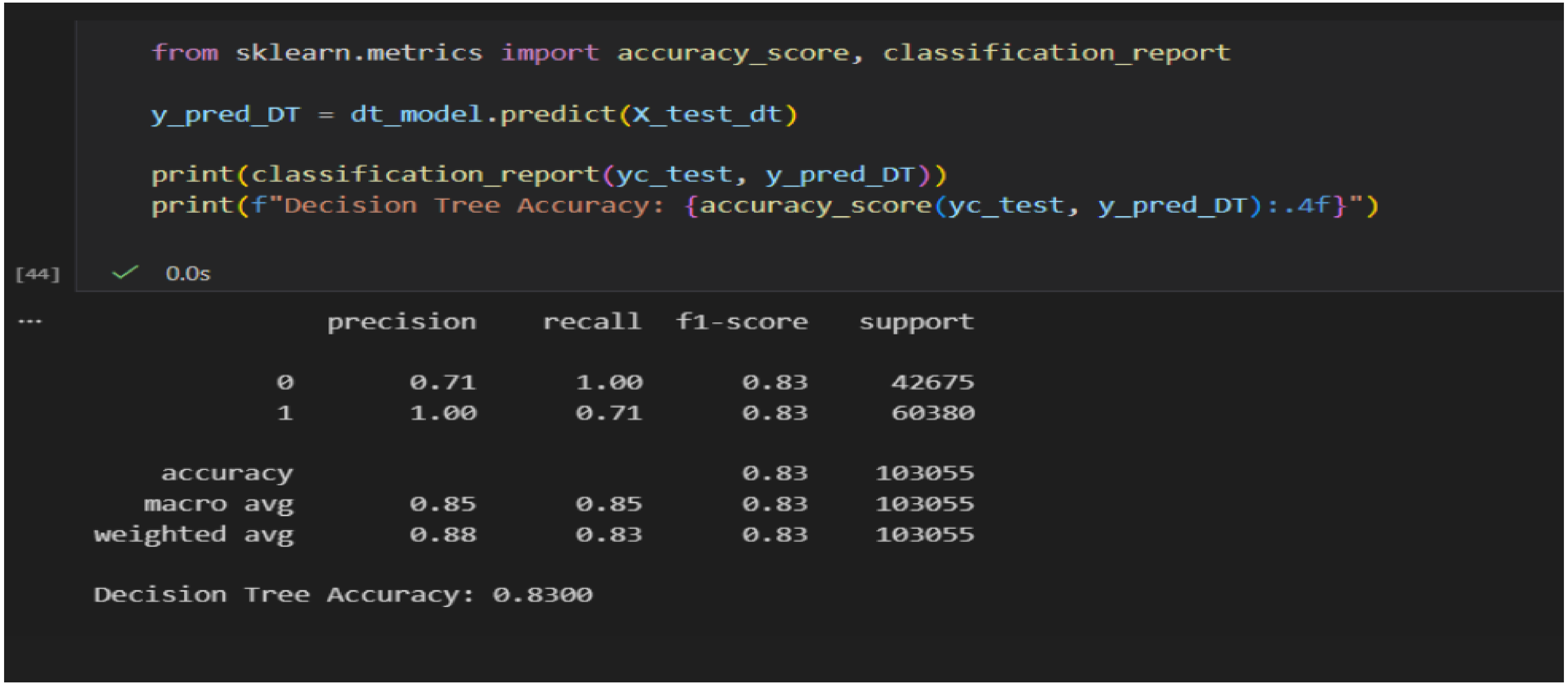
KNN ACC



```
print(classification_report(yc_test , y_pred_KNN))
print('KNN Accuracy',accuracy_score(yc_test , y_pred_KNN))

'''With Best Parameters k=30 , metric=distance : 0.9433722102'''
''' With Tuning and change metric and wights : 0.8974819271262918 '''
'''When i increase k over 30 the accuracy increases'''
'''k=80 got accuracy of 0.9123456790123457 > k=30 '''
```

```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91     42675
           1       1.00      0.86      0.92     60380

    accuracy                           0.92    103055
   macro avg       0.91      0.93      0.91    103055
weighted avg       0.93      0.92      0.92    103055

KNN Accuracy 0.9150065499005385

'When i increase k over 30 the accuracy increases'
```

Decision Tree ACC

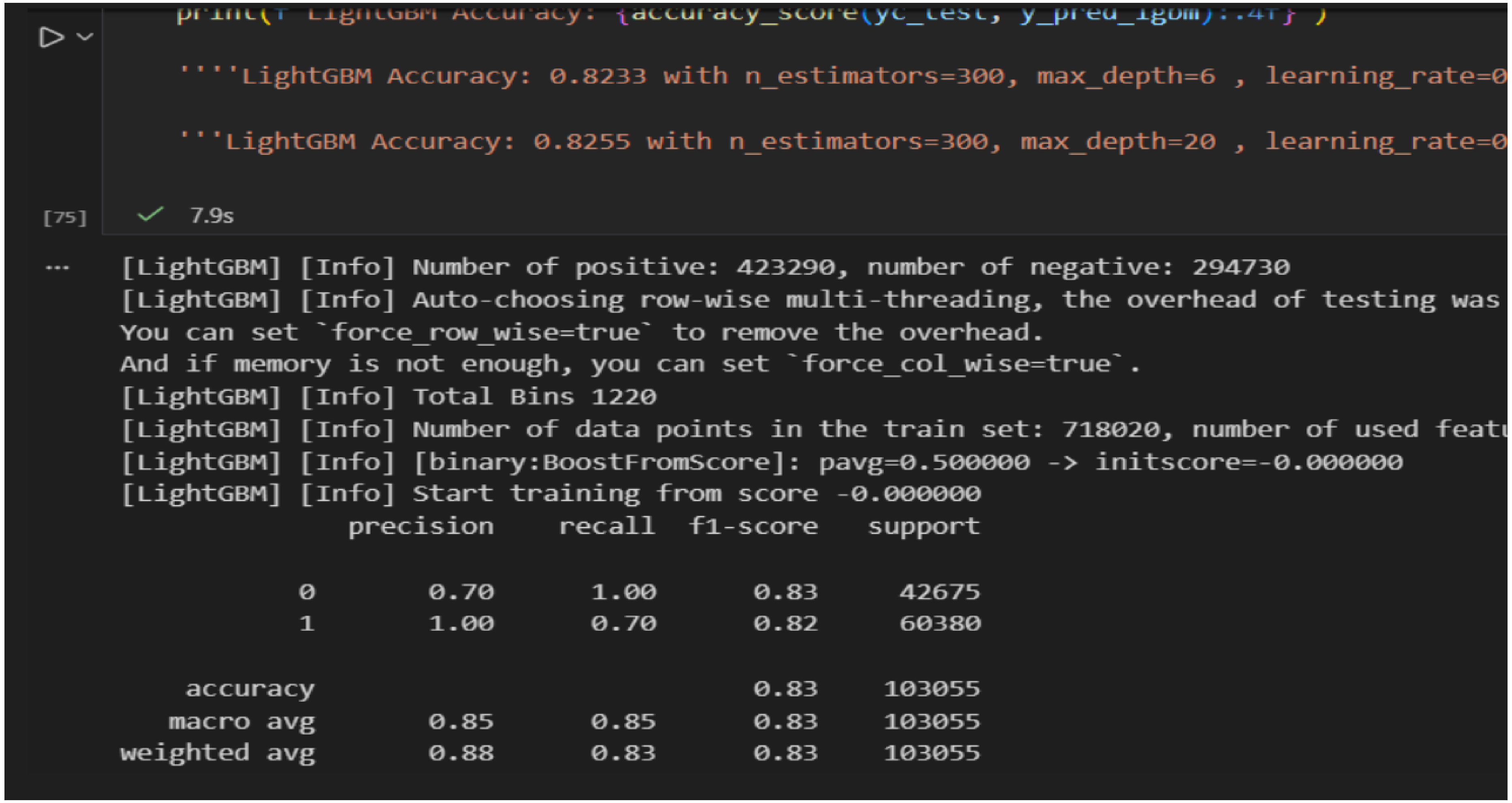


```
from sklearn.metrics import accuracy_score, classification_report

y_pred_DT = dt_model.predict(X_test_dt)

print(classification_report(yc_test, y_pred_DT))
print(f"Decision Tree Accuracy: {accuracy_score(yc_test, y_pred_DT):.4f}")
```

[44] ✓ 0.0s

```
              precision    recall  f1-score   support

           0       0.71      1.00      0.83     42675
           1       1.00      0.71      0.83     60380

    accuracy                           0.83    103055
   macro avg       0.85      0.85      0.83    103055
weighted avg       0.88      0.83      0.83    103055

Decision Tree Accuracy: 0.8300
```

LightGBM ACC

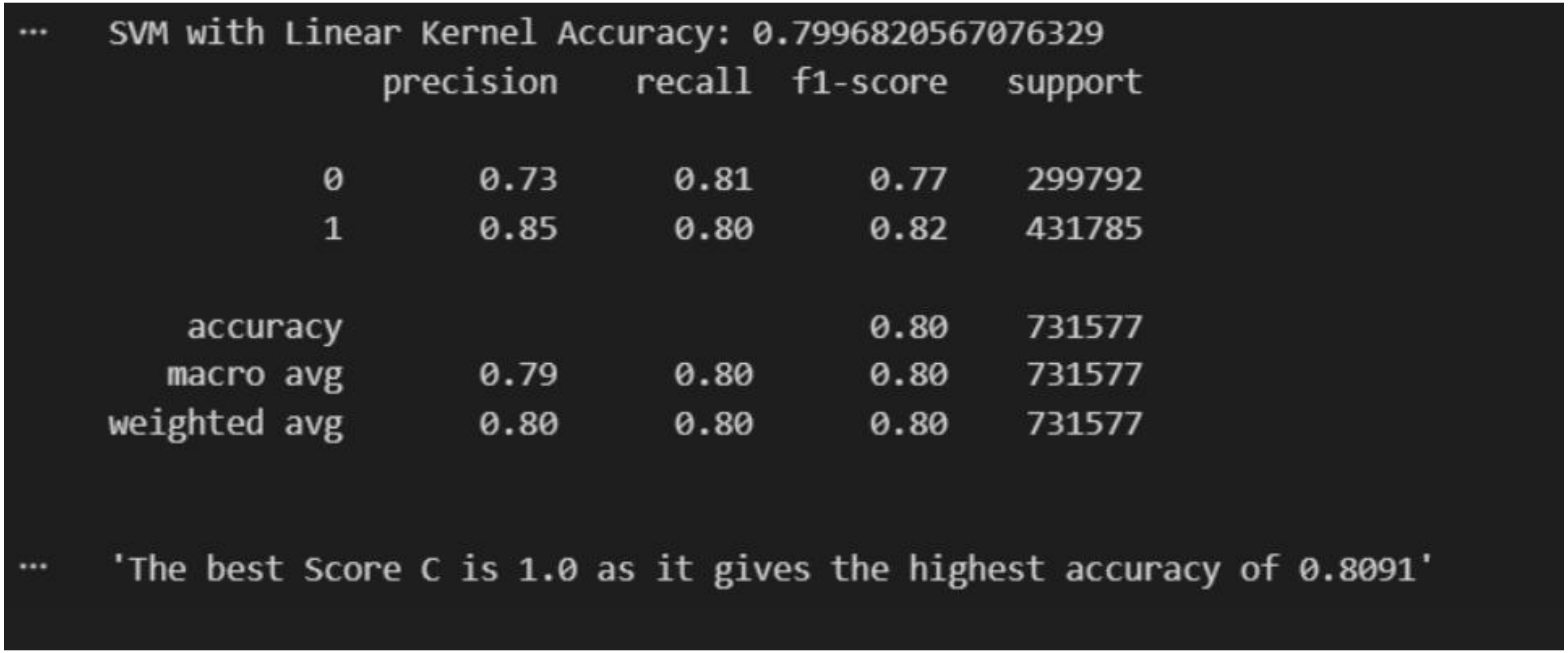


```
print(f"LightGBM Accuracy: {accuracy_score(yc_test, y_pred_lgbm):.4f}")

''''LightGBM Accuracy: 0.8233 with n_estimators=300, max_depth=6 , learning_rate=0

'''LightGBM Accuracy: 0.8255 with n_estimators=300, max_depth=20 , learning_rate=0
```

[75] 7.9s

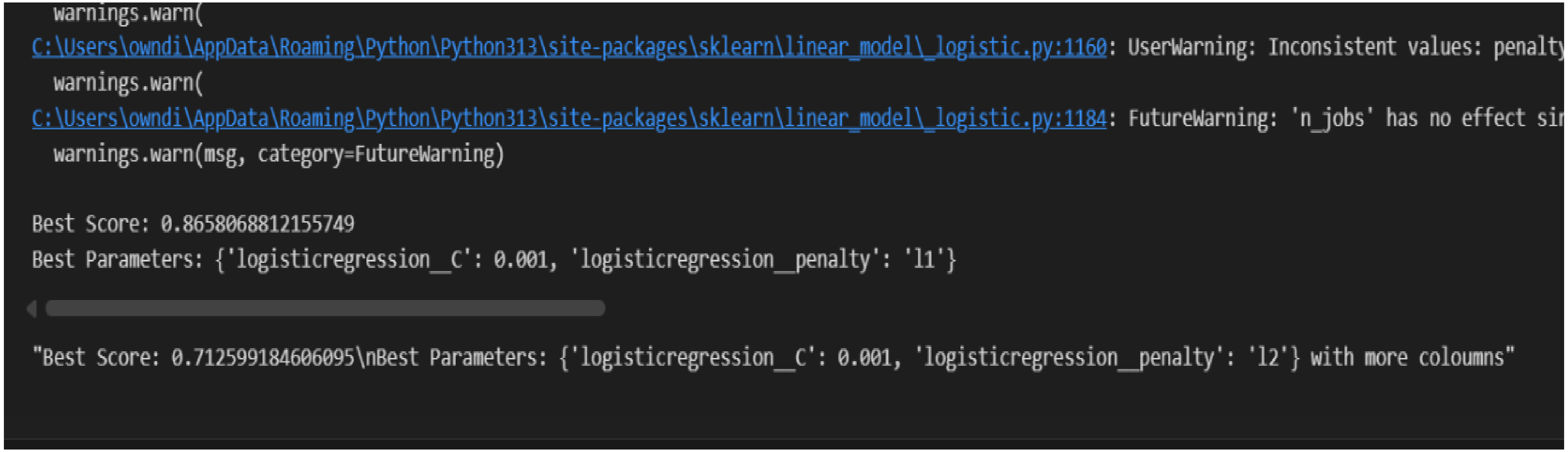
```
[LightGBM] [Info] Number of positive: 423290, number of negative: 294730
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1220
[LightGBM] [Info] Number of data points in the train set: 718020, number of used featu
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=-0.000000
[LightGBM] [Info] Start training from score -0.000000
              precision    recall  f1-score   support

           0       0.70      1.00      0.83     42675
           1       1.00      0.70      0.82     60380

    accuracy                           0.83    103055
   macro avg       0.85      0.85      0.83    103055
weighted avg       0.88      0.83      0.83    103055
```

SVM Linear kernal ACC



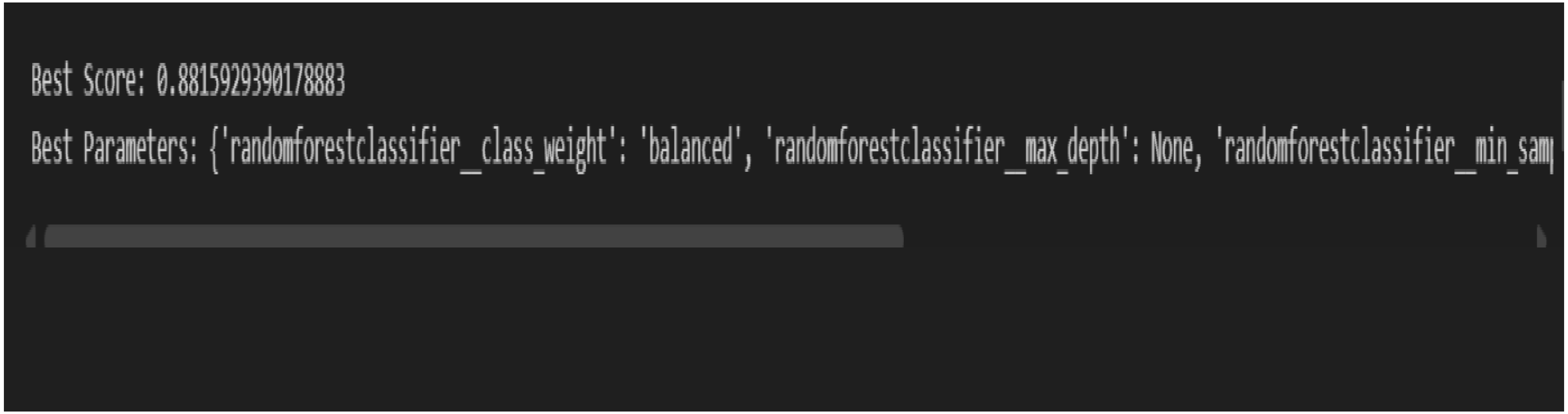
```
...   SVM with Linear Kernel Accuracy: 0.7996820567076329
                   precision    recall  f1-score   support

              0       0.73      0.81      0.77    299792
              1       0.85      0.80      0.82    431785

       accuracy                           0.80    731577
      macro avg       0.79      0.80      0.80    731577
   weighted avg       0.80      0.80      0.80    731577


...   'The best Score C is 1.0 as it gives the highest accuracy of 0.8091'
```

Logistic Regression ACC



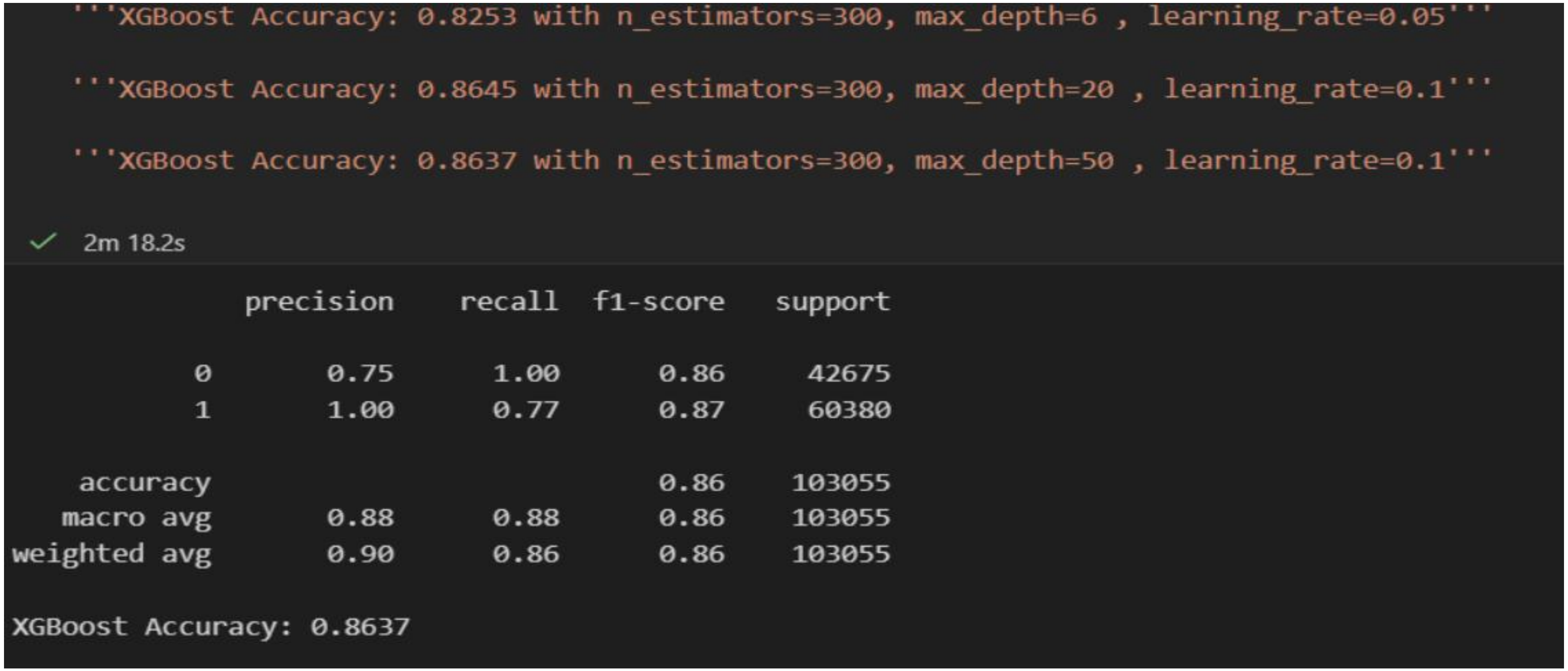
```
  warnings.warn(
C:\Users\owndi\AppData\Roaming\Python\Python313\site-packages\sklearn\linear_model\_logistic.py:1160: UserWarning: Inconsistent values: penalty
  warnings.warn(
C:\Users\owndi\AppData\Roaming\Python\Python313\site-packages\sklearn\linear_model\_logistic.py:1184: FutureWarning: 'n_jobs' has no effect si
  warnings.warn(msg, category=FutureWarning)

Best Score: 0.8658068812155749
Best Parameters: {'logisticregression__C': 0.001, 'logisticregression__penalty': 'l1'}

"Best Score: 0.712599184606095\nBest Parameters: {'logisticregression__C': 0.001, 'logisticregression__penalty': 'l2'} with more coloumns"
```

Random Forest



```
Best Score: 0.8815929390178883
Best Parameters: {'randomforestclassifier__class_weight': 'balanced', 'randomforestclassifier__max_depth': None, 'randomforestclassifier__min_samp
```

XG Boost ACC



```
    '''XGBoost Accuracy: 0.8253 with n_estimators=300, max_depth=6 , learning_rate=0.05'''

    '''XGBoost Accuracy: 0.8645 with n_estimators=300, max_depth=20 , learning_rate=0.1'''

    '''XGBoost Accuracy: 0.8637 with n_estimators=300, max_depth=50 , learning_rate=0.1'''
```

✓ 2m 18.2s

```
              precision    recall  f1-score   support

           0       0.75      1.00      0.86     42675
           1       1.00      0.77      0.87     60380

    accuracy                           0.86    103055
   macro avg       0.88      0.88      0.86    103055
weighted avg       0.90      0.86      0.86    103055

XGBoost Accuracy: 0.8637
```