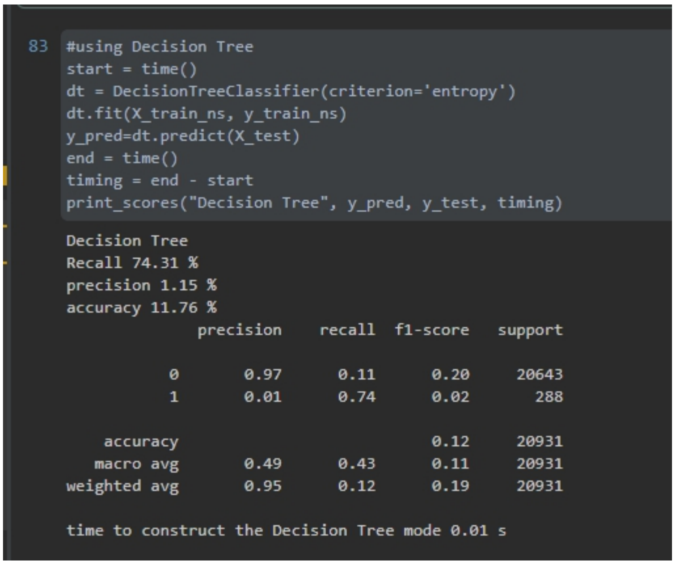


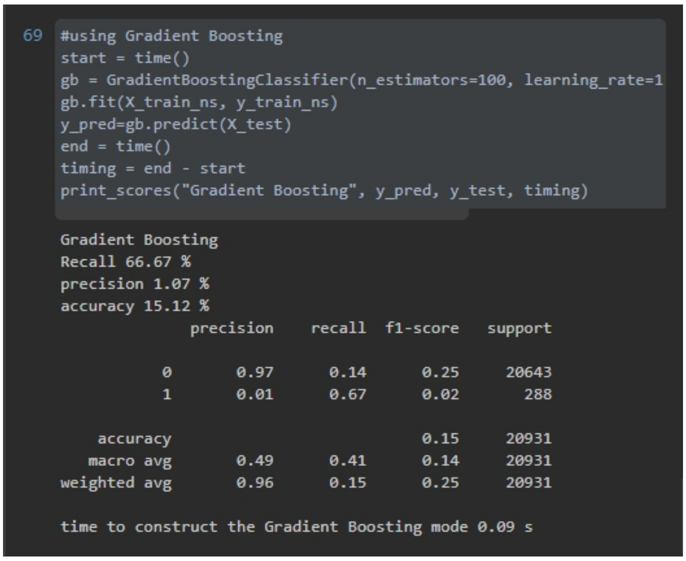
```
83  #using Decision Tree
start = time()
dt = DecisionTreeClassifier(criterion='entropy')
dt.fit(X_train_ns, y_train_ns)
y_pred=dt.predict(X_test)
end = time()
timing = end - start
print_scores("Decision Tree", y_pred, y_test, timing)
```

```
Decision Tree
Recall 74.31 %
precision 1.15 %
accuracy 11.76 %
              precision    recall  f1-score   support

           0       0.97      0.11      0.20     20643
           1       0.01      0.74      0.02       288

    accuracy                           0.12     20931
   macro avg       0.49      0.43      0.11     20931
weighted avg       0.95      0.12      0.19     20931

time to construct the Decision Tree mode 0.01 s
```



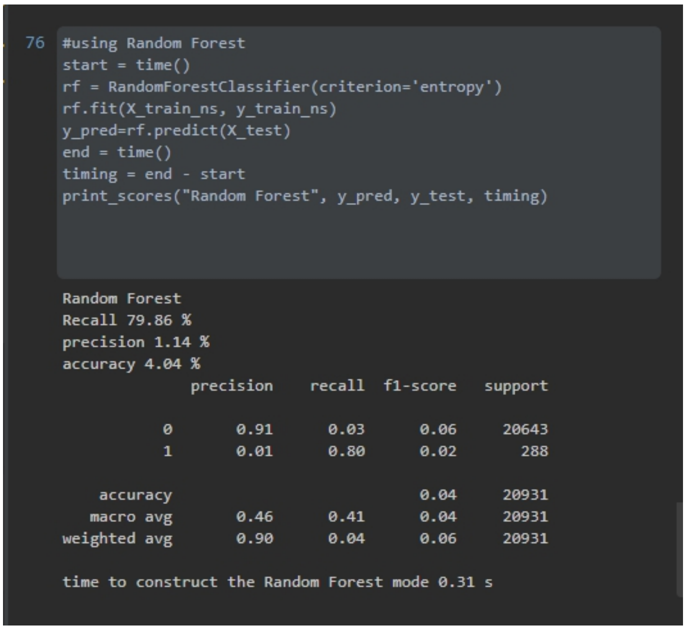
```
69  #using Gradient Boosting
start = time()
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=1
gb.fit(X_train_ns, y_train_ns)
y_pred=gb.predict(X_test)
end = time()
timing = end - start
print_scores("Gradient Boosting", y_pred, y_test, timing)
```

```
Gradient Boosting
Recall 66.67 %
precision 1.07 %
accuracy 15.12 %
              precision    recall  f1-score   support

           0       0.97      0.14      0.25     20643
           1       0.01      0.67      0.02       288

    accuracy                           0.15     20931
   macro avg       0.49      0.41      0.14     20931
weighted avg       0.96      0.15      0.25     20931

time to construct the Gradient Boosting mode 0.09 s
```



```
76  #using Random Forest
start = time()
rf = RandomForestClassifier(criterion='entropy')
rf.fit(X_train_ns, y_train_ns)
y_pred=rf.predict(X_test)
end = time()
timing = end - start
print_scores("Random Forest", y_pred, y_test, timing)
```

```
Random Forest
Recall 79.86 %
precision 1.14 %
accuracy 4.04 %
              precision    recall  f1-score   support

           0       0.91      0.03      0.06     20643
           1       0.01      0.80      0.02       288

    accuracy                           0.04     20931
   macro avg       0.46      0.41      0.04     20931
weighted avg       0.90      0.04      0.06     20931

time to construct the Random Forest mode 0.31 s
```

Before we using under sampling, all three algorithms have a high recall, precision and accuracy, but since we know that is because of the unbalance label we just ignore it. But after we built the model (after undersampling), we find all the three algorithms have the similar result, low precision, a little high recall and low accuracy. We know in this model:

Precision =TP/(TP+FP). Low precision means our FP is too large. We know that we have 20643 not fatal cases and only 288 fatal cases. Therefore, it means in this model, we cannot predict if the cases if fatal based on the features we have under the samples we have. It might be we get some noisy or unnecessary features influence our prediction.

Recall = TP/(TP+FN). Our recall is 66%-80% which is a little high. Therefore, the FN should be lower than TP, almost 3 times. Like we pick 75%. TP/(TP+FN)=3/4= 3/(3+1). We know TP should not very large since we only have 288 fatal cases. So based on our feature, the model think in majority of cases, the individual is fatal. That is the same as what FP shown. That means the condition of fatal cases are more changeable than not fatal cases.

So we know the when the precision(FP is too large) and recall (FN is too less) is nearly same, the accuracy can see which algorithms is better. We also will need to consider the time to construct the model.

So we have
DT: 11.76%, 0.01s
GS: 15.12%, 0.09s
RF: 4.04%, 0.31s

Therefore, I would rank Gradient Boosting as 1 since it has the best accuracy and acceptable construction time. Decision Tree will be 2. And Random Forest be the 3 for the lowest accuracy.