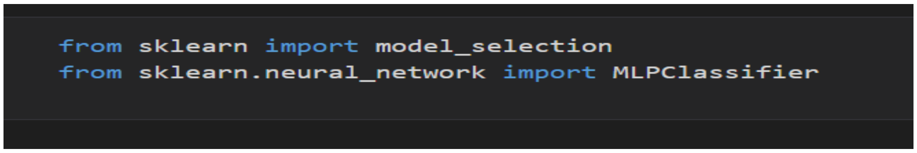
Testing Model With Multiple Evaluation Metrics

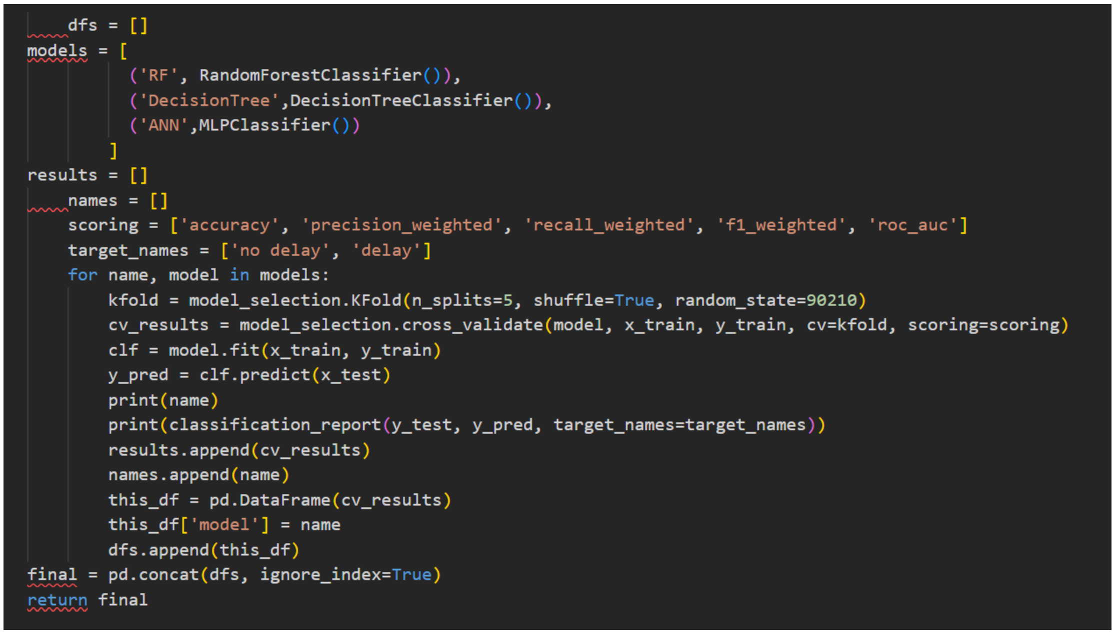
Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Compare The Model

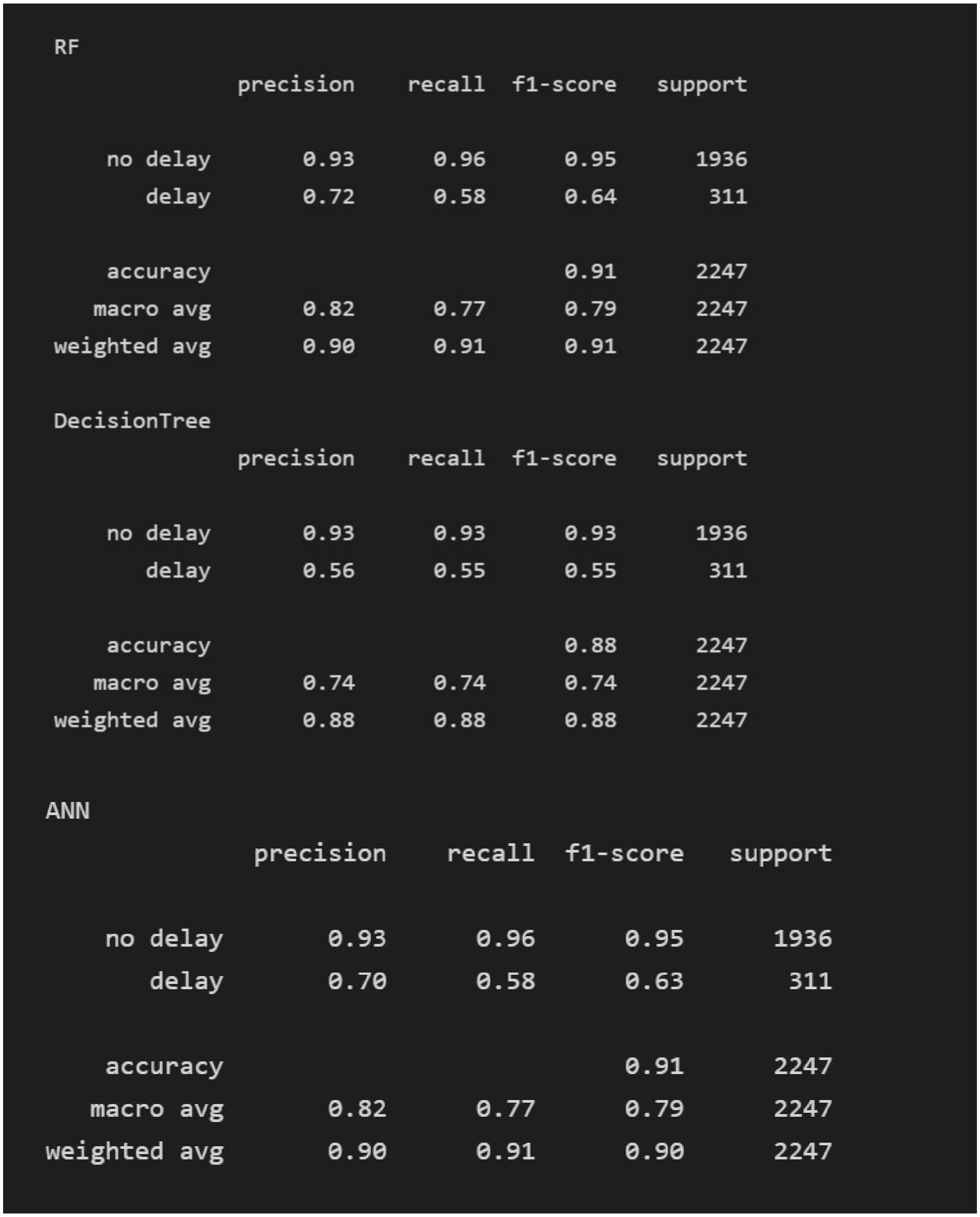
For comparing the above three models



```
from sklearn import model_selection
from sklearn.neural_network import MLPClassifier
```

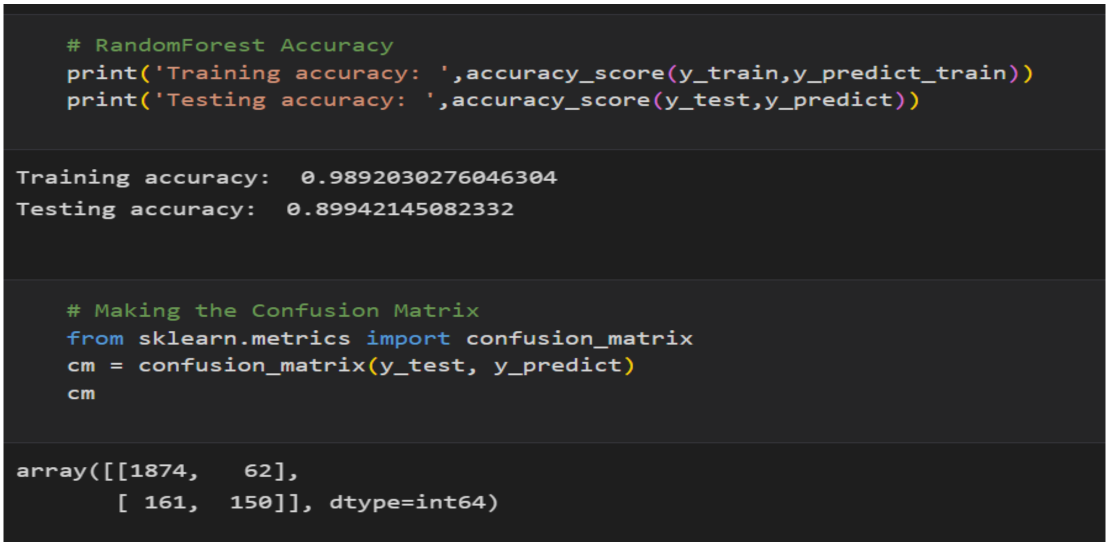


```
    dfs = []
models = [
            ('RF', RandomForestClassifier()),
            ('DecisionTree',DecisionTreeClassifier()),
            ('ANN',MLPClassifier())
        ]
results = []
    names = []
    scoring = ['accuracy', 'precision_weighted', 'recall_weighted', 'f1_weighted', 'roc_auc']
    target_names = ['no delay', 'delay']
    for name, model in models:
        kfold = model_selection.KFold(n_splits=5, shuffle=True, random_state=90210)
        cv_results = model_selection.cross_validate(model, x_train, y_train, cv=kfold, scoring=scoring)
        clf = model.fit(x_train, y_train)
        y_pred = clf.predict(x_test)
        print(name)
        print(classification_report(y_test, y_pred, target_names=target_names))
        results.append(cv_results)
        names.append(name)
        this_df = pd.DataFrame(cv_results)
        this_df['model'] = name
        dfs.append(this_df)
final = pd.concat(dfs, ignore_index=True)
return final
```



```
RF
              precision    recall  f1-score   support

    no delay       0.93      0.96      0.95      1936
       delay       0.72      0.58      0.64       311

    accuracy                           0.91      2247
   macro avg       0.82      0.77      0.79      2247
weighted avg       0.90      0.91      0.91      2247

DecisionTree
              precision    recall  f1-score   support

    no delay       0.93      0.93      0.93      1936
       delay       0.56      0.55      0.55       311

    accuracy                           0.88      2247
   macro avg       0.74      0.74      0.74      2247
weighted avg       0.88      0.88      0.88      2247
```

```
ANN
              precision    recall  f1-score   support

    no delay       0.93      0.96      0.95      1936
       delay       0.70      0.58      0.63       311

    accuracy                           0.91      2247
   macro avg       0.82      0.77      0.79      2247
weighted avg       0.90      0.91      0.90      2247
```

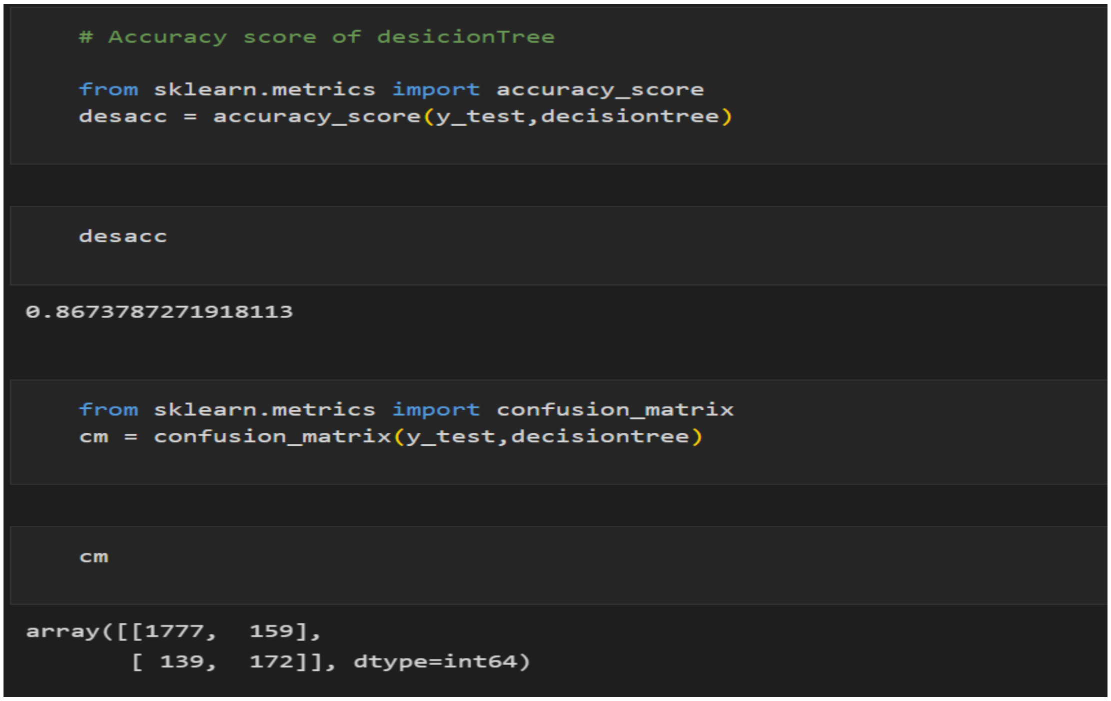


```
# RandomForest Accuracy
print('Training accuracy: ',accuracy_score(y_train,y_predict_train))
print('Testing accuracy: ',accuracy_score(y_test,y_predict))
```

```
Training accuracy:  0.9892030276046304
Testing accuracy:  0.89942145082332
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predict)
cm
```

```
array([[1874,   62],
       [ 161,  150]], dtype=int64)
```



```
# Accuracy score of desicionTree

from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)
```

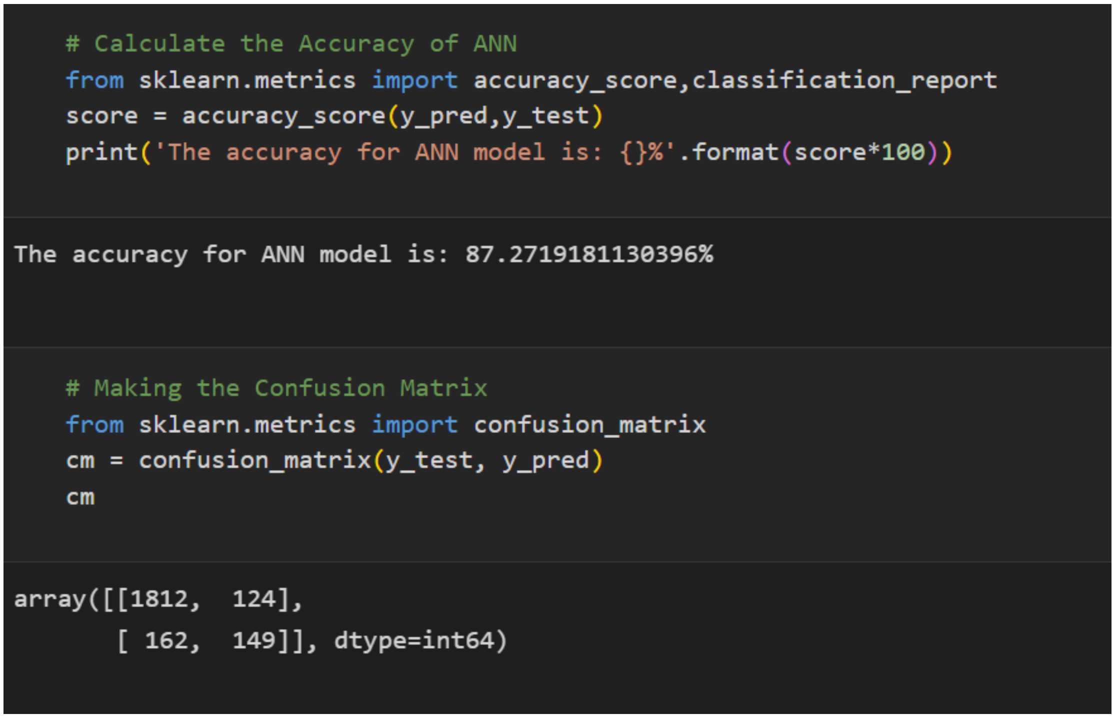
```
desacc
```

```
0.8673787271918113
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,decisiontree)
```

```
cm
```

```
array([[1777,  159],
       [ 139,  172]], dtype=int64)
```



```
# Calculate the Accuracy of ANN
from sklearn.metrics import accuracy_score,classification_report
score = accuracy_score(y_pred,y_test)
print('The accuracy for ANN model is: {}%'.format(score*100))
```

```
The accuracy for ANN model is: 87.2719181130396%
```

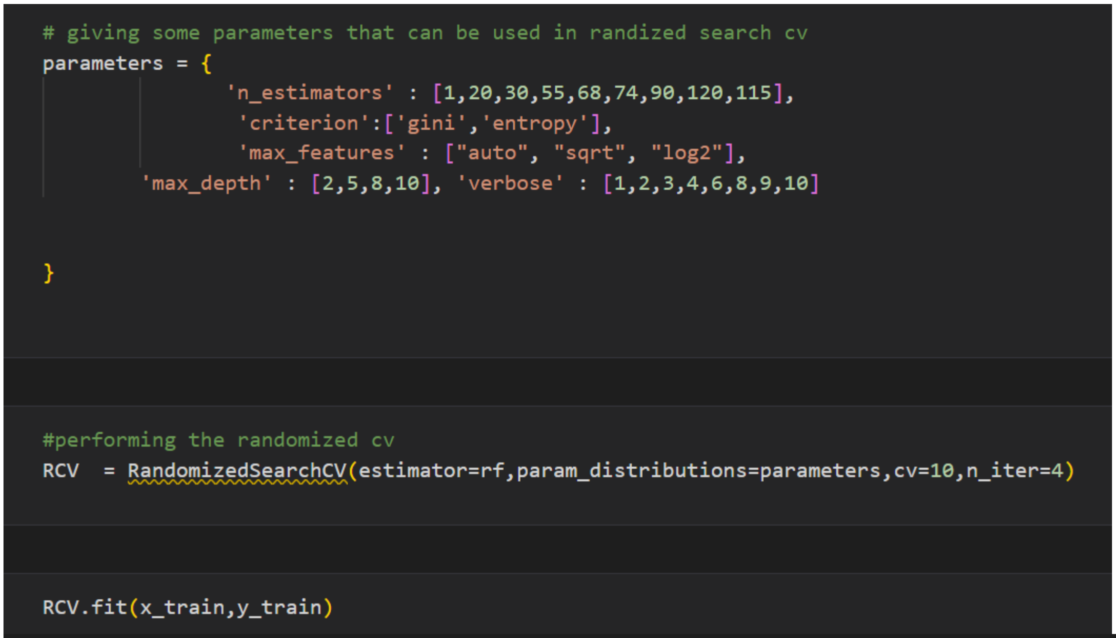
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1812,  124],
       [ 162,  149]], dtype=int64)
```

Comparing Model Accuracy Before & After Applying Hyperparameter Tuning

Evaluating performance of the model From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

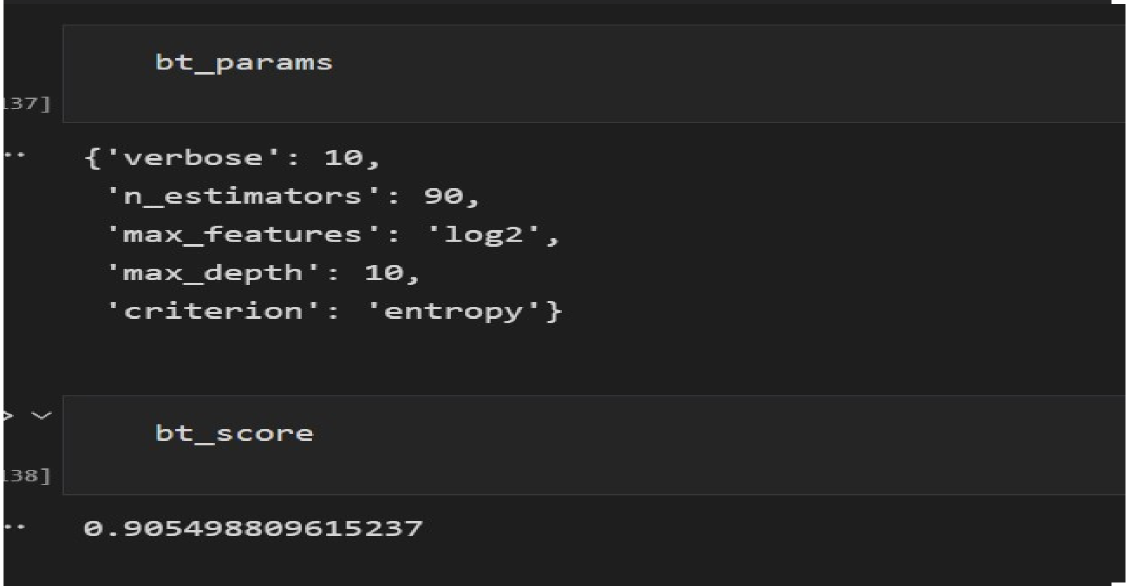
Note: To understand cross validation, refer to this link



```
# giving some parameters that can be used in randized search cv
parameters = {
                'n_estimators' : [1,20,30,55,68,74,90,120,115],
                 'criterion':['gini','entropy'],
                 'max_features' : ["auto", "sqrt", "log2"],
        'max_depth' : [2,5,8,10], 'verbose' : [1,2,3,4,6,8,9,10]


}
```

```
#performing the randomized cv
RCV  = RandomizedSearchCV(estimator=rf,param_distributions=parameters,cv=10,n_iter=4)
```

```
RCV.fit(x_train,y_train)
```

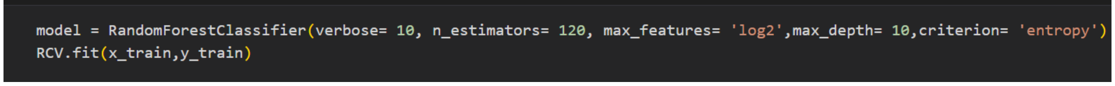


```
    bt_params
37]
```

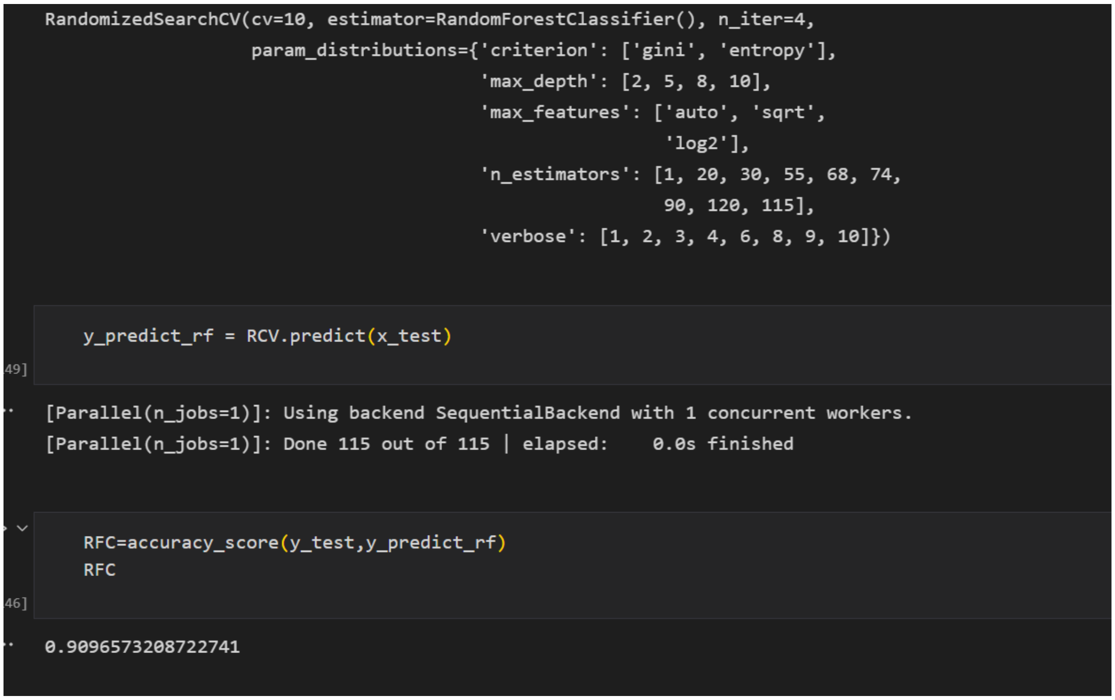
```
{'verbose': 10,
 'n_estimators': 90,
 'max_features': 'log2',
 'max_depth': 10,
 'criterion': 'entropy'}
```

```
    bt_score
38]
```

```
0.905498809615237
```



```
model = RandomForestClassifier(verbose= 10, n_estimators= 120, max_features= 'log2',max_depth= 10,criterion= 'entropy')
RCV.fit(x_train,y_train)
```



```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_iter=4,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [2, 5, 8, 10],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'n_estimators': [1, 20, 30, 55, 68, 74,
                                                         90, 120, 115],
                                        'verbose': [1, 2, 3, 4, 6, 8, 9, 10]})
```

```
y_predict_rf = RCV.predict(x_test)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 115 out of 115 | elapsed:    0.0s finished
```

```
RFC=accuracy_score(y_test,y_predict_rf)
RFC
```

```
0.9096573208722741
```