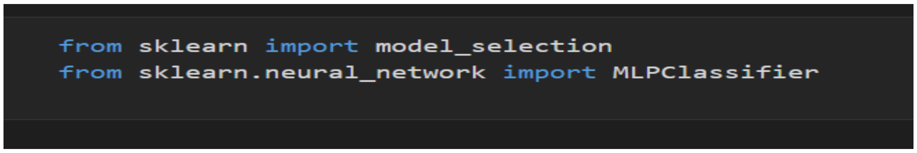
# Testing Model With Multiple Evaluation Metrics

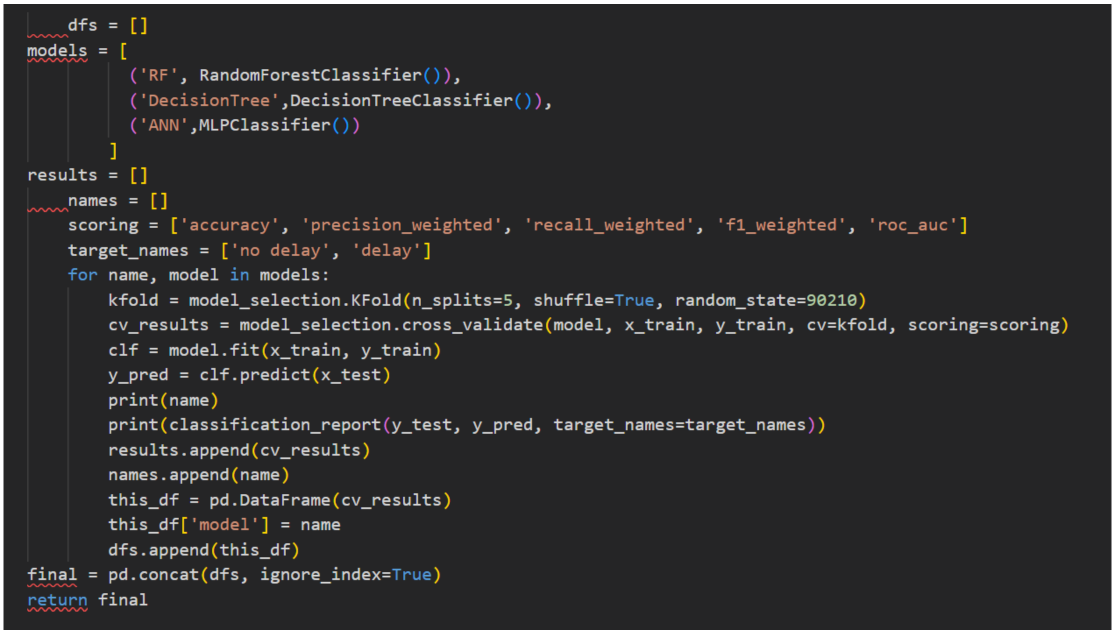
Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision,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

## Compare The Model

For comparing the above three models

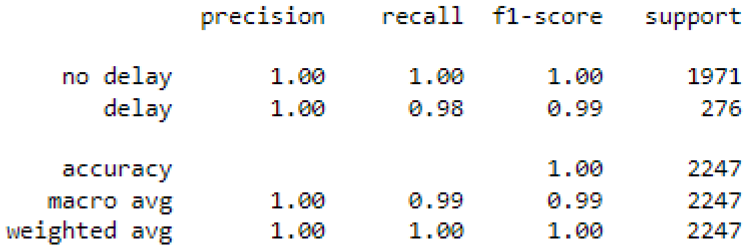


```python
from sklearn import model_selection
from sklearn.neural_network import MLPClassifier
```



```python
    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

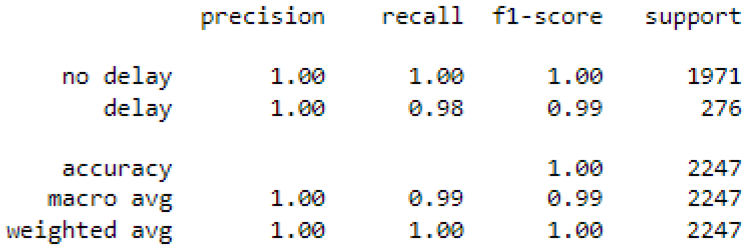


```
              precision    recall  f1-score   support

    no delay       1.00      1.00      1.00      1971
       delay       1.00      0.98      0.99       276

    accuracy                           1.00      2247
   macro avg       1.00      0.99      0.99      2247
weighted avg       1.00      1.00      1.00      2247
```

DecisionTree

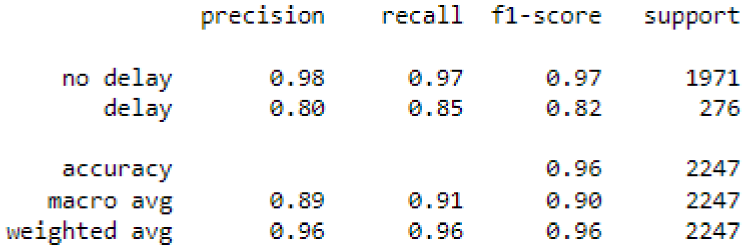


```
              precision    recall  f1-score   support

    no delay       1.00      1.00      1.00      1971
       delay       1.00      0.98      0.99       276

    accuracy                           1.00      2247
   macro avg       1.00      0.99      0.99      2247
weighted avg       1.00      1.00      1.00      2247
```

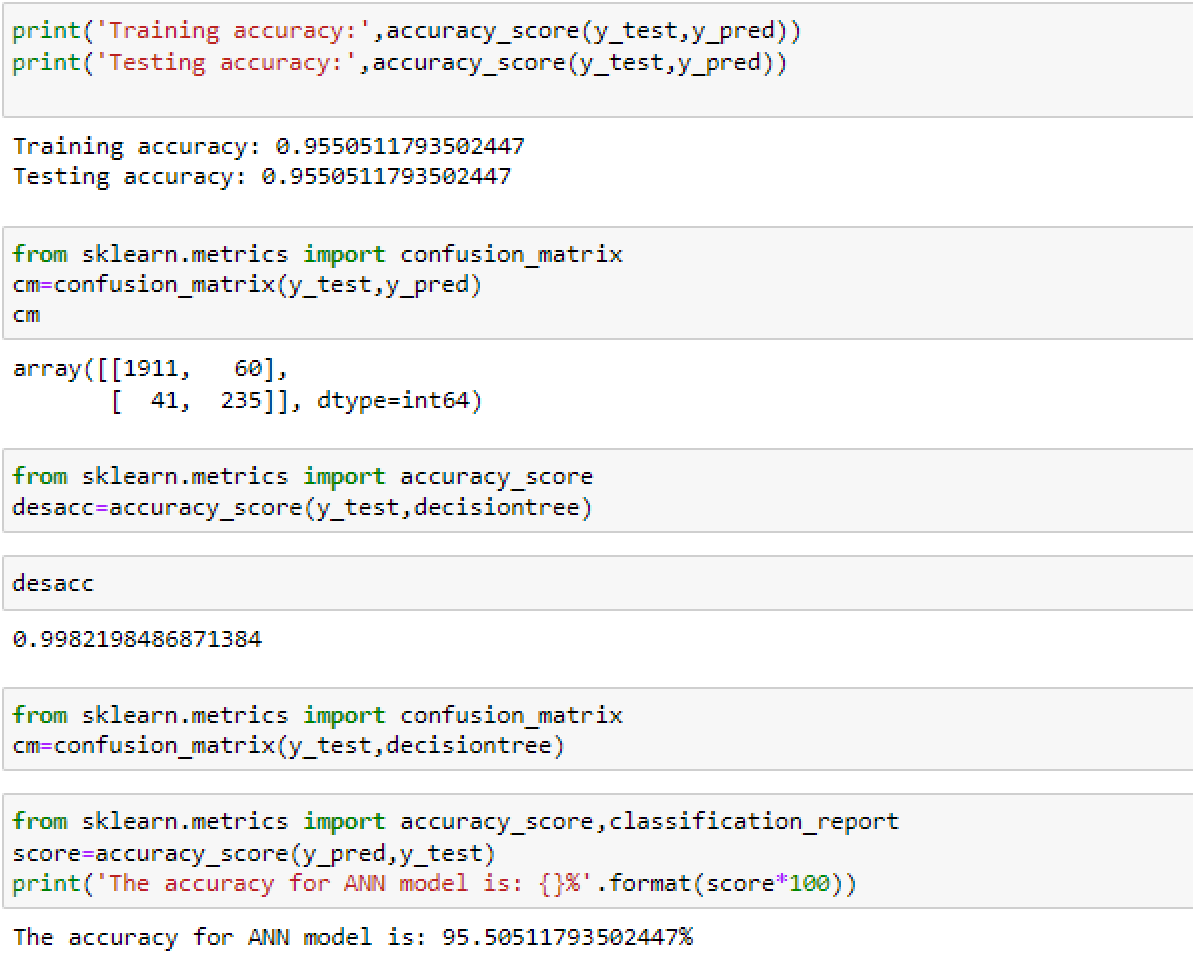
ANN



```
              precision    recall  f1-score   support

    no delay       0.98      0.97      0.97      1971
       delay       0.80      0.85      0.82       276

    accuracy                           0.96      2247
   macro avg       0.89      0.91      0.90      2247
weighted avg       0.96      0.96      0.96      2247
```



```
print('Training accuracy:',accuracy_score(y_test,y_pred))
print('Testing accuracy:',accuracy_score(y_test,y_pred))
```

```
Training accuracy: 0.9550511793502447
Testing accuracy: 0.9550511793502447
```

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

```
array([[1911,   60],
       [  41,  235]], dtype=int64)
```

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

```
desacc
```

```
0.9982198486871384
```

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

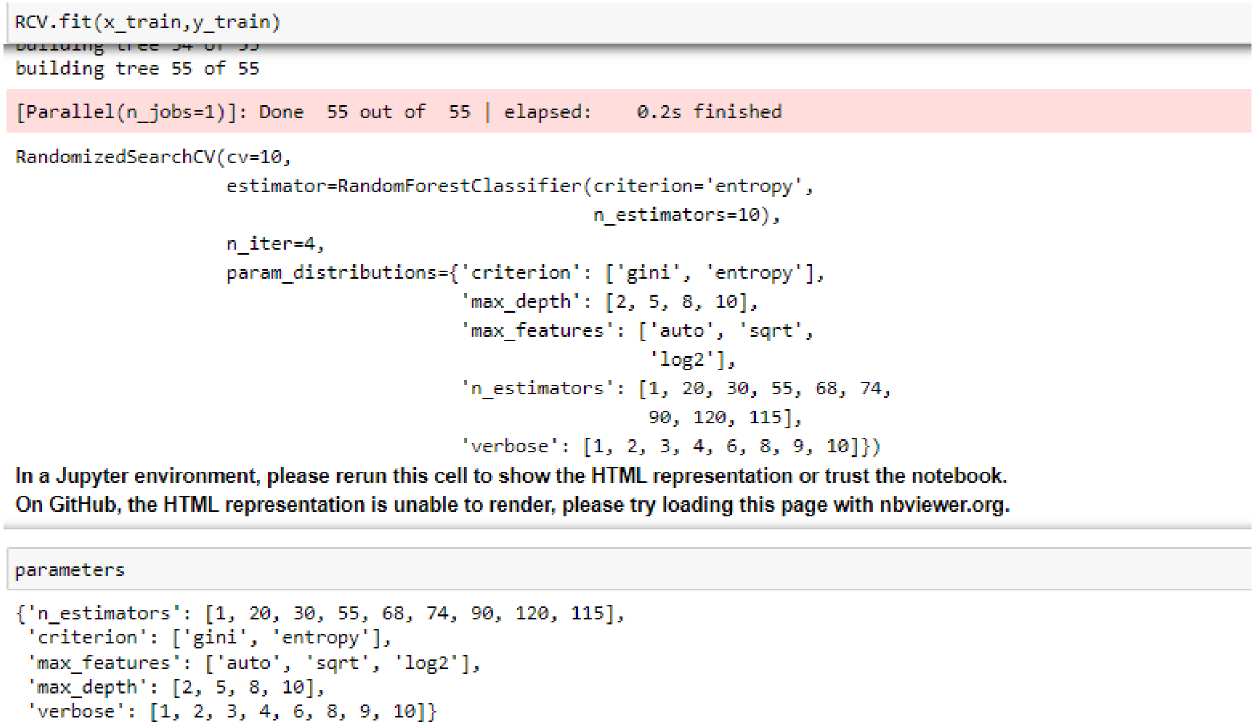
```
from sklearn.metrics import accuracy_score,classification_report
score=accuracy_score(y_pred,y_test)
print('The accuracy for ANN model is: {}%'.format(score*100))
```

```
The accuracy for ANN model is: 95.50511793502447%
```

## 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



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

```
building tree 54 of 55
building tree 55 of 55
```

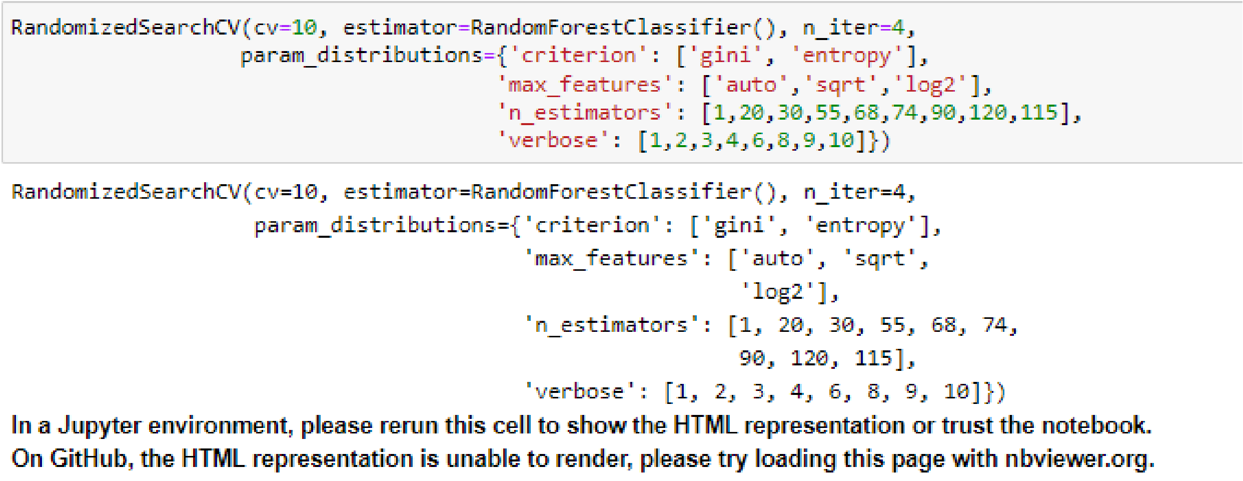
```
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:    0.2s finished
```

```
RandomizedSearchCV(cv=10,
                   estimator=RandomForestClassifier(criterion='entropy',
                                                    n_estimators=10),
                   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]})
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
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]}
```



```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_iter=4,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        '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]})
```

```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_iter=4,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        '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]})
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**