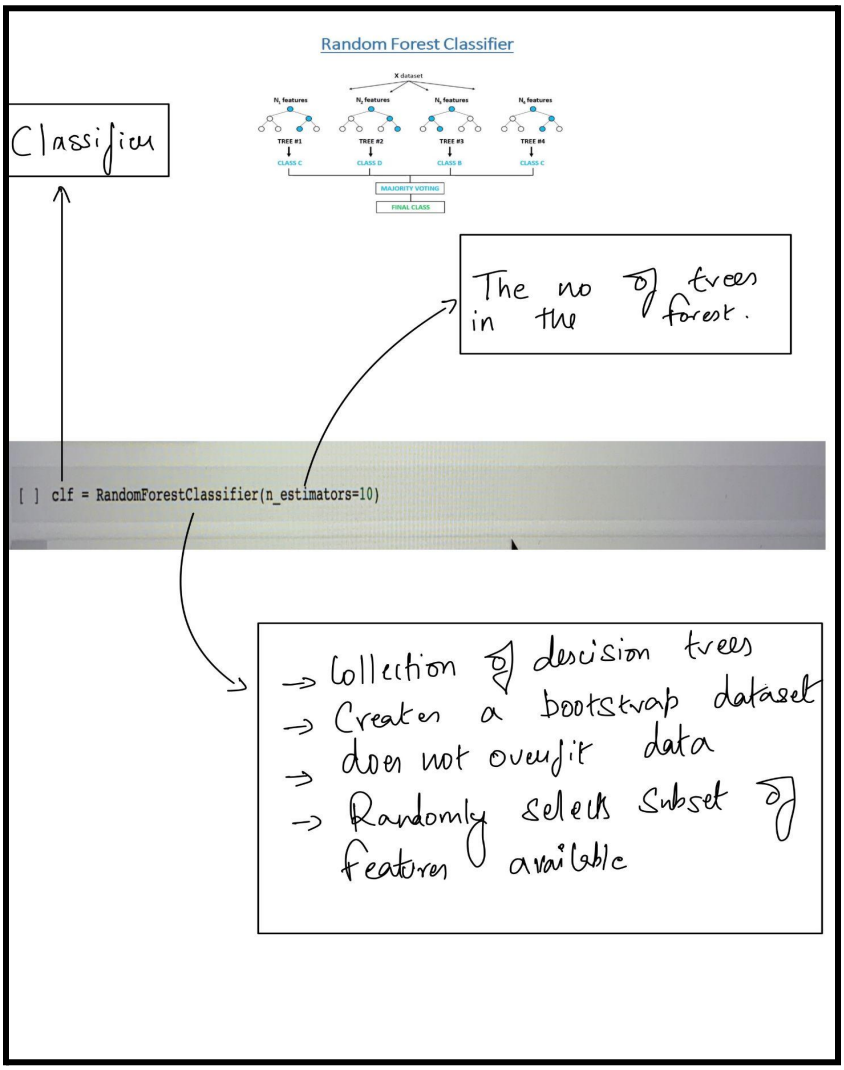
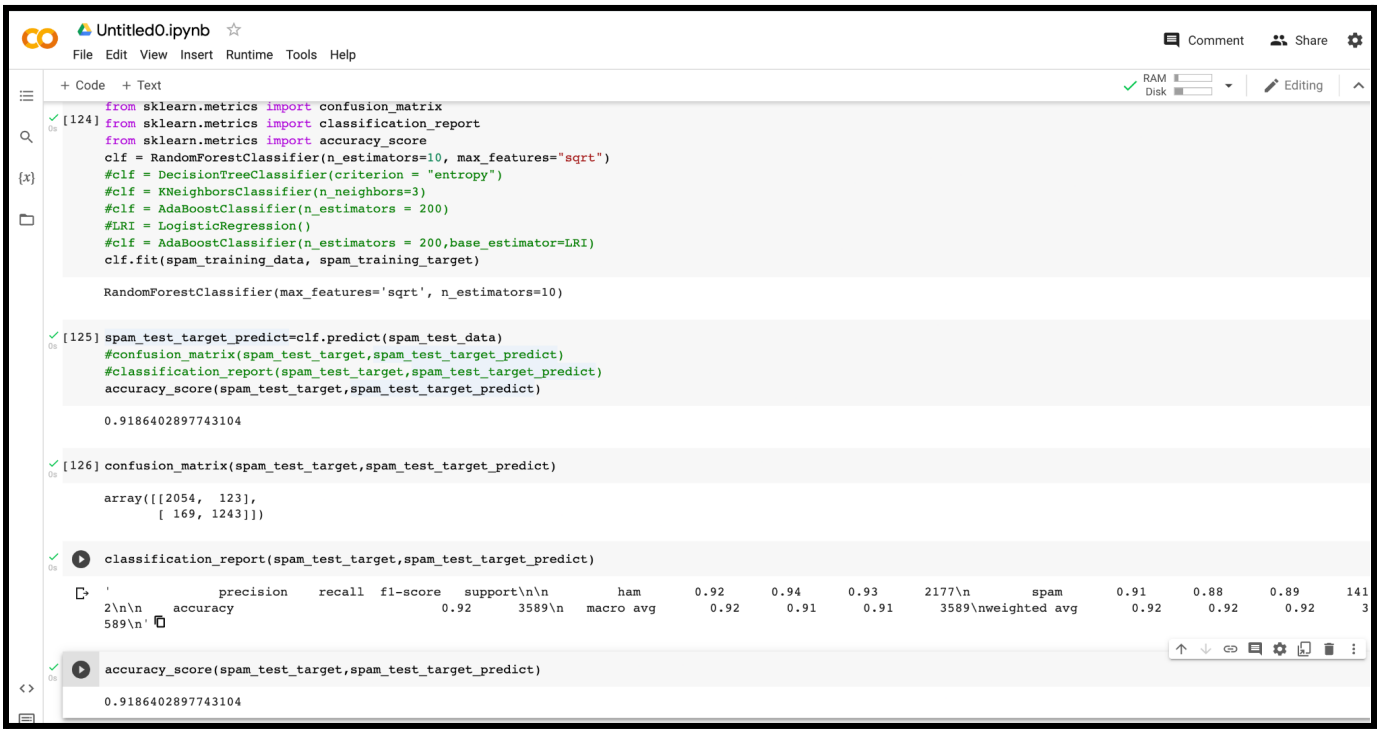
Building a Predictive Model with Decision Trees: An Implementation Guide

Understanding the random forest classifer :

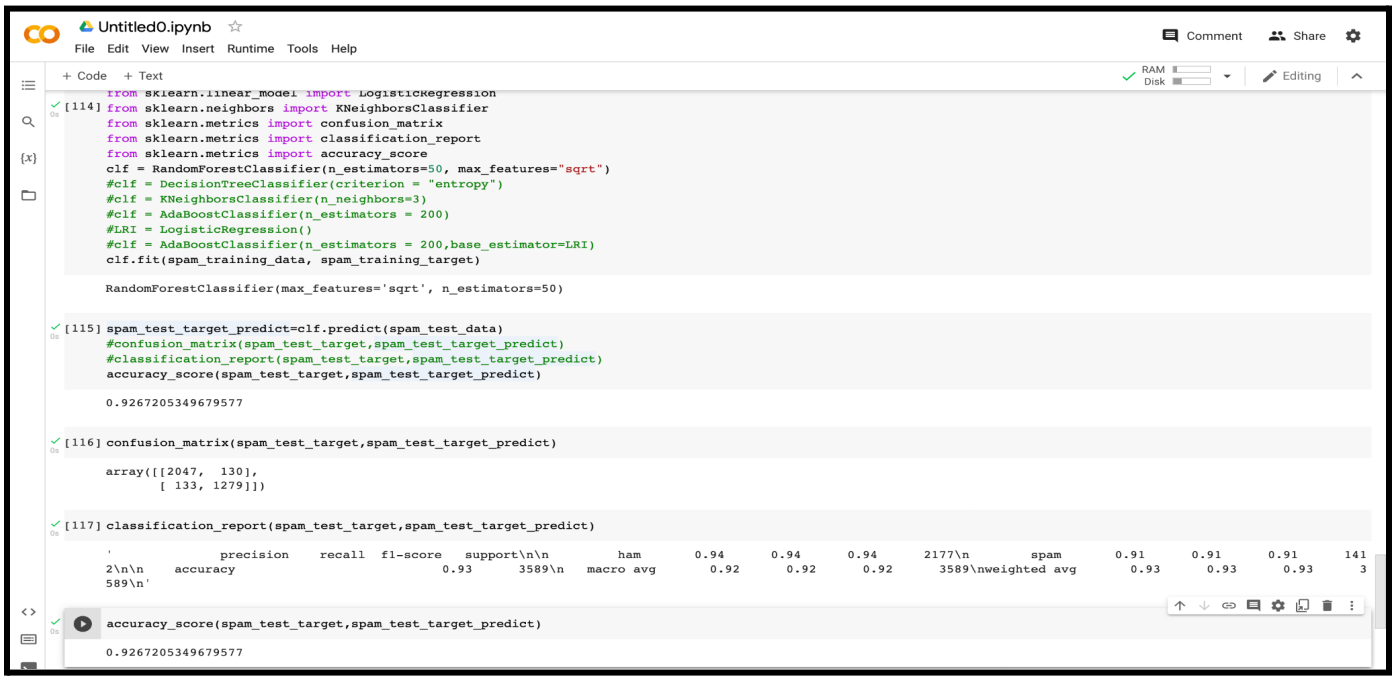


In order for us to consider the number of features at each split we must use the max_feature parameter. We are going to use the sqrt option, as it helps us take the square root of the total no of features. We can also get empirically good default values if we use the sqrt option especially for classification tasks. The default of Max features has changed from “auto” to “sqrt”.

1) n_estimators = 10

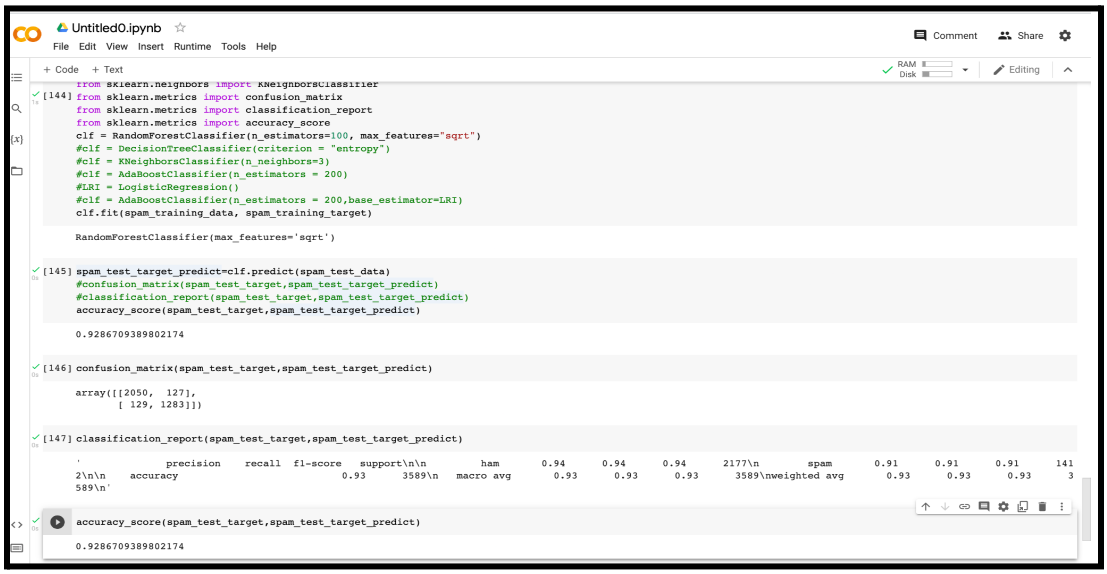


2) n_estimators = 50

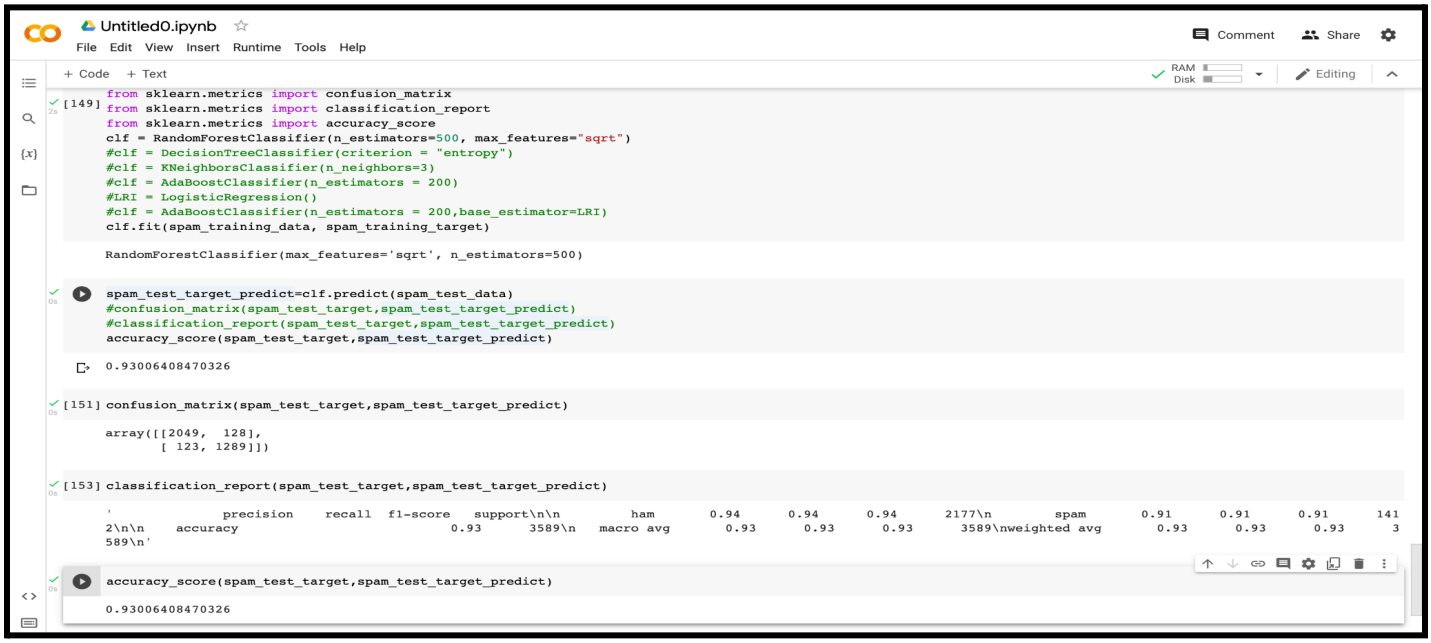


---

2) n_estimators = 100



3) n_estimators = 500



Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

```
[149] from sklearn.metrics import confusion_matrix
      from sklearn.metrics import classification_report
      from sklearn.metrics import accuracy_score
      clf = RandomForestClassifier(n_estimators=500, max_features="sqrt")
      #clf = DecisionTreeClassifier(criterion = "entropy")
      #clf = KNeighborsClassifier(n_neighbors=3)
      #clf = AdaBoostClassifier(n_estimators = 200)
      #LRI = LogisticRegression()
      #clf = AdaBoostClassifier(n_estimators = 200,base_estimator=LRI)
      clf.fit(spam_training_data, spam_training_target)

      RandomForestClassifier(max_features='sqrt', n_estimators=500)

      spam_test_target_predict=clf.predict(spam_test_data)
      #confusion_matrix(spam_test_target,spam_test_target_predict)
      #classification_report(spam_test_target,spam_test_target_predict)
      accuracy_score(spam_test_target,spam_test_target_predict)

      0.93006408470326

[151] confusion_matrix(spam_test_target,spam_test_target_predict)

      array([[2049,  128],
             [ 123, 1289]])

[153] classification_report(spam_test_target,spam_test_target_predict)

      '              precision    recall  f1-score   support\n\n         ham       0.94      0.94      0.94      2177\n        spam       0.91      0.91      0.91       141
      2\n\n    accuracy                           0.93      3589\n   macro avg       0.93      0.93      0.93      3589\nweighted avg       0.93      0.93      0.93      3
      589\n'

      accuracy_score(spam_test_target,spam_test_target_predict)

      0.93006408470326
```

4) n_estimators = 1000

Untitled0.ipynb

File Edit View Insert Runtime Tools Help All changes saved

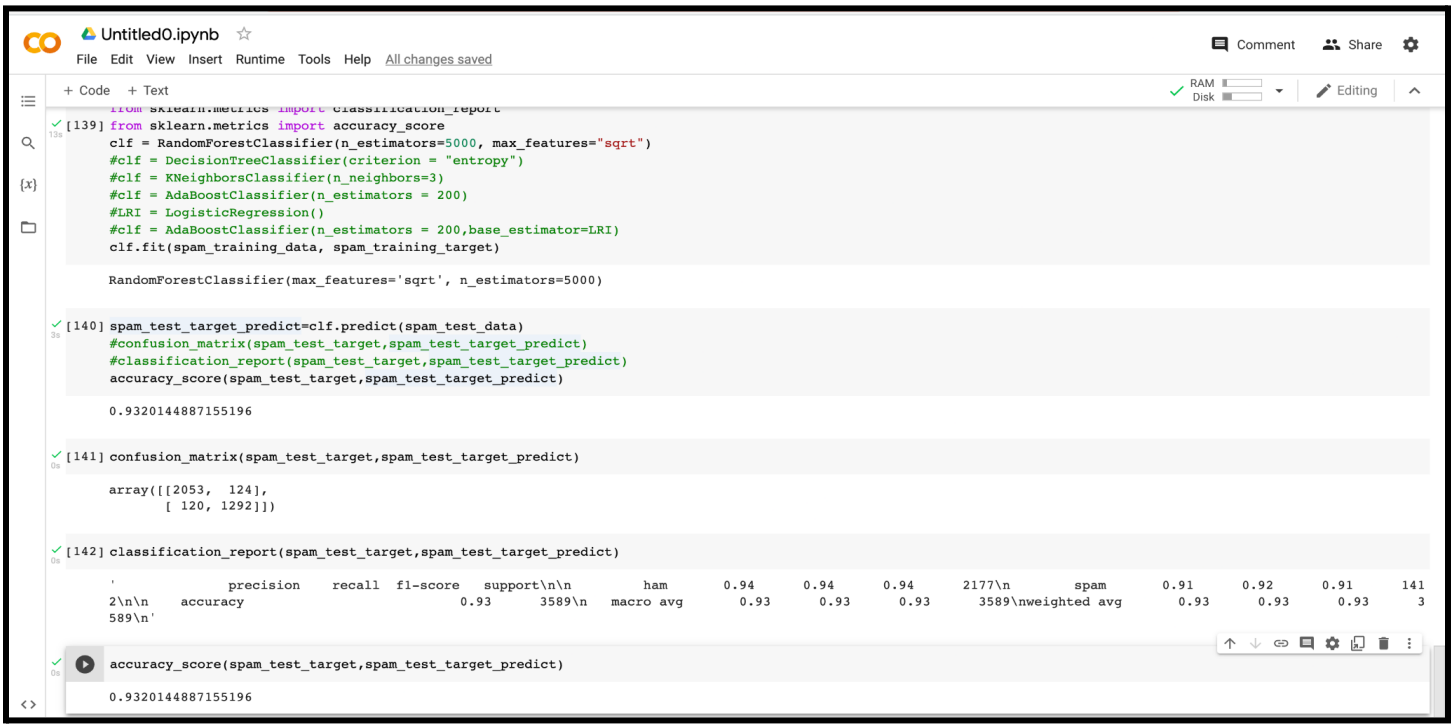
Comment Share

+ Code + Text

RAM Disk Editing

```
[134] from sklearn.metrics import accuracy_score
      clf = RandomForestClassifier(n_estimators=1000, max_features="sqrt")
      #clf = DecisionTreeClassifier(criterion = "entropy")
      #clf = KNeighborsClassifier(n_neighbors=3)
      #clf = AdaBoostClassifier(n_estimators = 200)
      #LRI = LogisticRegression()
      #clf = AdaBoostClassifier(n_estimators = 200,base_estimator=LRI)
      clf.fit(spam_training_data, spam_training_target)

      RandomForestClassifier(max_features='sqrt', n_estimators=1000)

[135] spam_test_target_predict=clf.predict(spam_test_data)
      #confusion_matrix(spam_test_target,spam_test_target_predict)
      #classification_report(spam_test_target,spam_test_target_predict)
      accuracy_score(spam_test_target,spam_test_target_predict)

      0.9320144887155196

[136] confusion_matrix(spam_test_target,spam_test_target_predict)

      array([[2060,  117],
             [ 127, 1285]])

[137] classification_report(spam_test_target,spam_test_target_predict)

      '              precision    recall  f1-score   support\n\n         ham       0.94      0.95      0.94      2177\n        spam       0.92      0.91      0.91       141
      2\n\n    accuracy                           0.93      3589\n   macro avg       0.93      0.93      0.93      3589\nweighted avg       0.93      0.93      0.93      3
      589\n'

      accuracy_score(spam_test_target,spam_test_target_predict)

      0.9320144887155196
```

5) n_estimators = 5000



Comparing the results of all the classifiers (with the best possible parameter setting for each classifier). Use classification accuracy (# of instances correctly classified/total # of instances presented for classification), per class classification accuracy , and confusion matrix to compare the classifiers. (Classifiers : Random forest, Decision Tree classifier)

- **Decision Tree classifier:**

The decision tree classifier got an accuracy of 87% after setting the parameters to the following values **criterion = "entropy", splitter = "best", max_depth = 10, min_samples_split=10, min_samples_leaf=1, random_state=42, class_weight = "balanced", min_weight_fraction_leaf=0, max_features="sqrt", min_impurity_decrease=0,** these parameters were tuned with the best possible parameter setting.

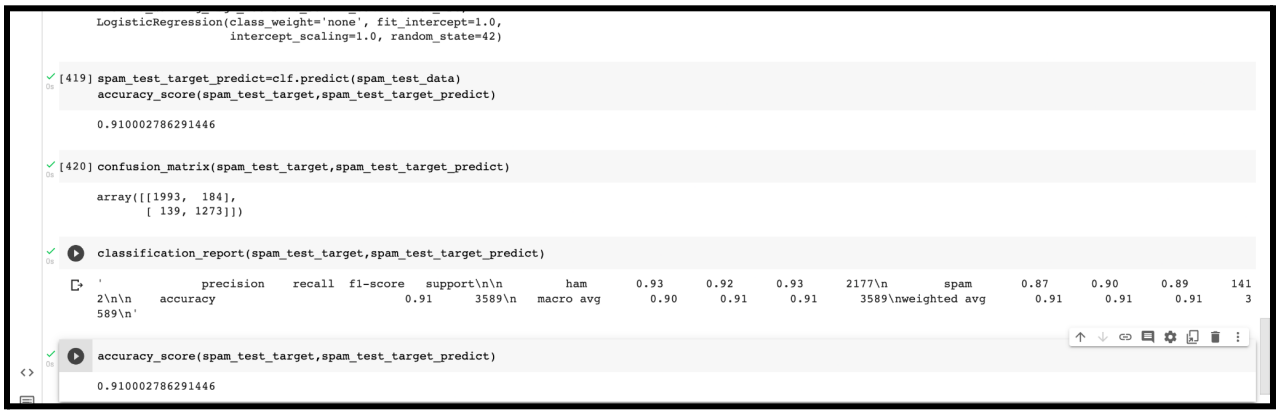
- **Random Forest Classifier:**

The Random Forest Classifier got an accuracy of 92% after setting the parameters to the following values ***n_estimators=200, max_features="sqrt", random_state = 42, criterion ="entropy", max_depth=10, min_samples_split=10, min_samples_leaf = 1,min_weight_fraction_leaf=0, max_leaf_nodes = 50,*** *these parameters were tuned with the best possible parameter setting.*

- **Kn Neighbors classifier**

The KNeighborsclassifier got an accuracy of 74% after setting the parameters to the following values ***n_neighbors=3, weights = "uniform", algorithm = "auto", leaf_size = 30, p = 2, n_jobs=1,*** *These parameters were tuned with the best possible parameter setting.*

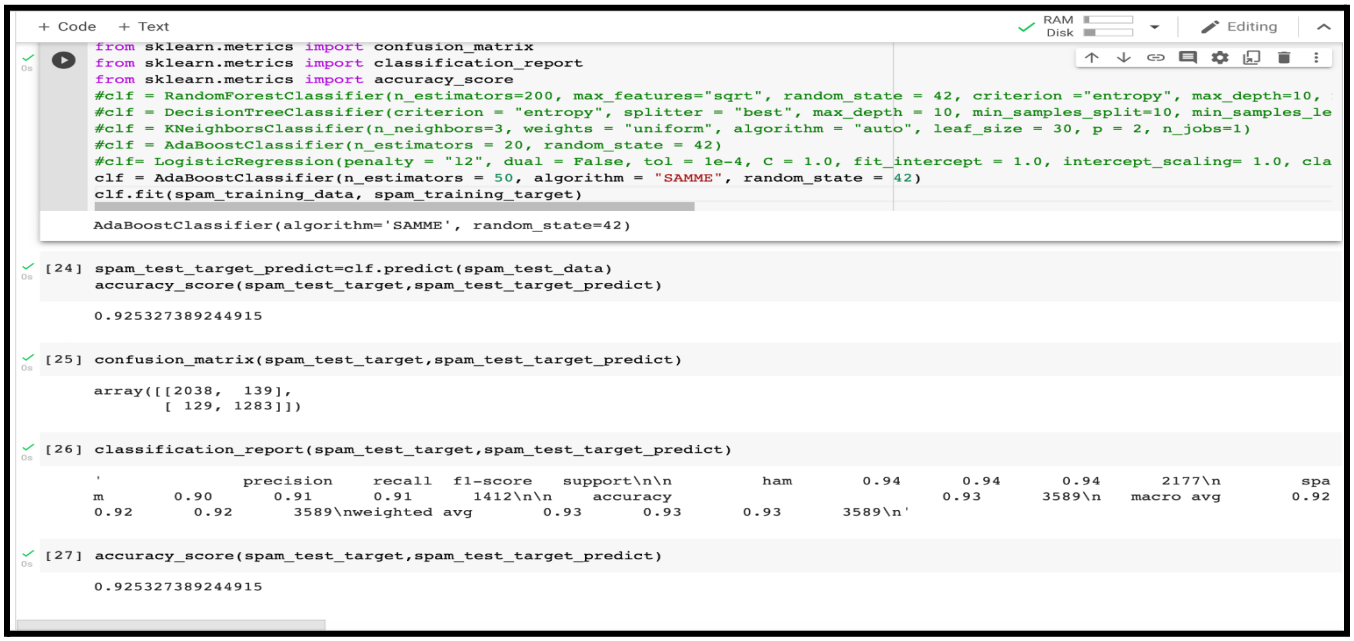
- **Logistic Regression Classifier**



The logistic regression classifier got an accuracy of 91% after setting the parameters to the following values ***penalty = "l2", dual = False, tol = 1e-4, C = 1.0, fit_intercept = 1.0, intercept_scaling= 1.0,***

class_weight="none", random_state = 42, solver = "lbfgs", max_iter=100, multi_class="auto", verbose = 0, warm_start = False, These parameters were tuned with the best possible parameter setting.

- **Adaboost classifier**



```
+ Code  + Text                                                    RAM Disk   Editing

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
#clf = RandomForestClassifier(n_estimators=200, max_features="sqrt", random_state = 42, criterion ="entropy", max_depth=10,
#clf = DecisionTreeClassifier(criterion = "entropy", splitter = "best", max_depth = 10, min_samples_split=10, min_samples_le
#clf = KNeighborsClassifier(n_neighbors=3, weights = "uniform", algorithm = "auto", leaf_size = 30, p = 2, n_jobs=1)
#clf = AdaBoostClassifier(n_estimators = 20, random_state = 42)
#clf= LogisticRegression(penalty = "l2", dual = False, tol = 1e-4, C = 1.0, fit_intercept = 1.0, intercept_scaling= 1.0, cla
clf = AdaBoostClassifier(n_estimators = 50, algorithm = "SAMME", random_state = 42)
clf.fit(spam_training_data, spam_training_target)

AdaBoostClassifier(algorithm='SAMME', random_state=42)

[24] spam_test_target_predict=clf.predict(spam_test_data)
     accuracy_score(spam_test_target,spam_test_target_predict)

     0.925327389244915

[25] confusion_matrix(spam_test_target,spam_test_target_predict)

     array([[2038,  139],
            [ 129, 1283]])

[26] classification_report(spam_test_target,spam_test_target_predict)

     '              precision    recall  f1-score   support\n\n         ham       0.94      0.94      0.94      2177\n         spa
     m       0.90      0.91      0.91      1412\n\n    accuracy                           0.93      3589\n   macro avg       0.92
     0.92      0.92      3589\nweighted avg       0.93      0.93      0.93      3589\n'

[27] accuracy_score(spam_test_target,spam_test_target_predict)

     0.925327389244915
```

The adaboost classifier got an accuracy of 91% after setting the parameters to the following values ***n_estimators = 50, algorithm = "SAMME", random_state = 42,*** *These parameters were tuned with the best possible parameter setting.*
