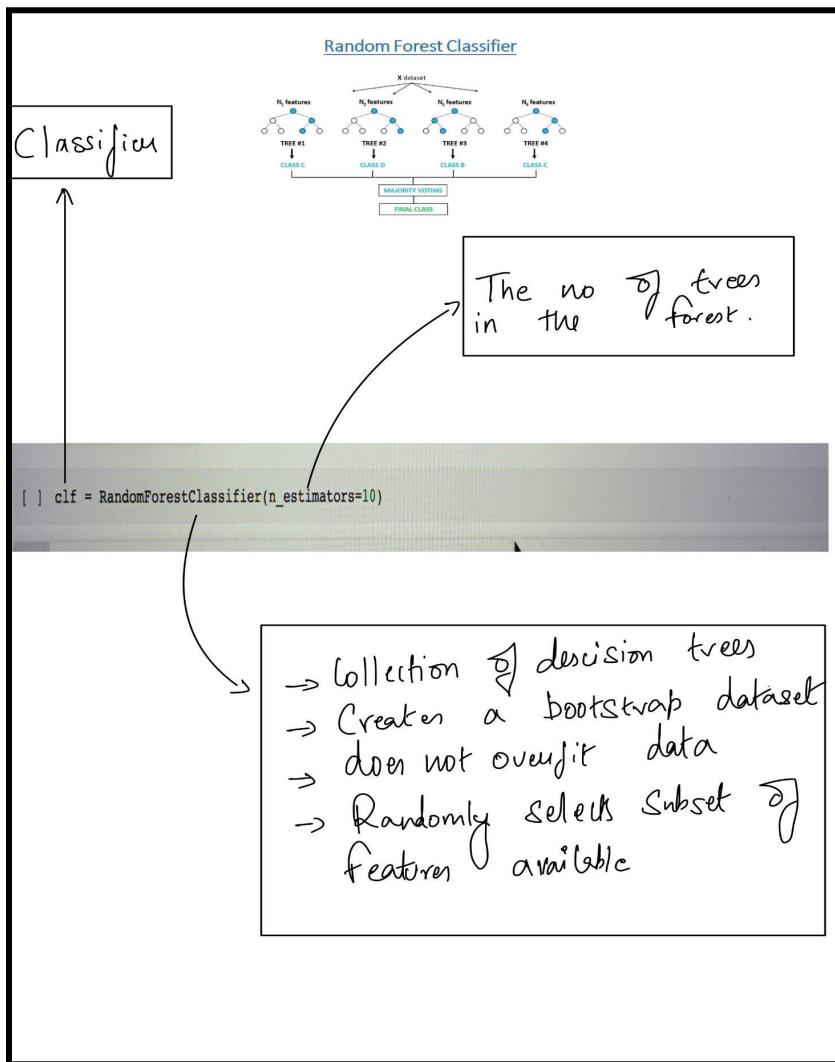


Building a Predictive Model with Decision Trees: An Implementation Guide

Understanding the random forest classifier :



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`



The screenshot shows a Jupyter Notebook cell with the following code and its execution results:

```
+ Code + Text
File Edit View Insert Runtime Tools Help
Comment Share
RAM Disk Editing
[124]: from sklearn.metrics import confusion_matrix
       from sklearn.metrics import classification_report
       from sklearn.metrics import accuracy_score
       clf = RandomForestClassifier(n_estimators=10, 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=10)

[125]: spam_test_predict=clf.predict(spam_test_data)
#confusion_matrix(spam_test_target,spam_test_predict)
#classification_report(spam_test_target,spam_test_predict)
accuracy_score(spam_test_target,spam_test_predict)

0.9186402897743104

[126]: confusion_matrix(spam_test_target,spam_test_predict)

array([[2054, 123],
       [169, 1243]])

[127]: classification_report(spam_test_target,spam_test_predict)

          precision    recall  f1-score   support
ham      0.92      0.94      0.93    2177
spam    0.91      0.92      0.90    3589
macro avg  0.92      0.93      0.91    5766
weighted avg 0.91      0.92      0.90    5766

[128]: accuracy_score(spam_test_target,spam_test_predict)

0.9186402897743104
```

2) n_estimators = 50

A screenshot of a Jupyter Notebook interface titled "Untitled0.ipynb". The code cell at [114] imports necessary libraries and defines a Random Forest classifier with n_estimators=50. The code at [115] performs predictions on test data and calculates accuracy. The output shows an accuracy score of 0.9267205349679577. The code cell at [116] prints a confusion matrix, and the code cell at [117] prints a classification report. The classification report table is as follows:

	precision	recall	f1-score	support	ham	spam	0.91	0.91	0.91	141
accuracy	0.93	0.93	0.93	3589	0.92	0.92	0.93	0.93	0.93	3
macro avg	0.92	0.92	0.92	3589	0.92	0.92	0.93	0.93	0.93	3
weighted avg	0.92	0.92	0.92	3589	0.92	0.92	0.93	0.93	0.93	3

The code cell at [118] calculates accuracy again, resulting in 0.9267205349679577.

2) n_estimators = 100

A screenshot of a Jupyter Notebook interface titled "Untitled0.ipynb". The code cell at [144] imports necessary libraries and defines a Random Forest classifier with n_estimators=100. The code at [145] performs predictions on test data and calculates accuracy. The output shows an accuracy score of 0.9286709389802174. The code cell at [146] prints a confusion matrix, and the code cell at [147] prints a classification report. The classification report table is as follows:

	precision	recall	f1-score	support	ham	spam	0.91	0.91	0.91	141
accuracy	0.93	0.93	0.93	3589	0.93	0.93	0.93	0.93	0.93	3
macro avg	0.93	0.93	0.93	3589	0.93	0.93	0.93	0.93	0.93	3
weighted avg	0.93	0.93	0.93	3589	0.93	0.93	0.93	0.93	0.93	3

The code cell at [148] calculates accuracy again, resulting in 0.9286709389802174.

3) n_estimators = 500

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
[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)

[150] 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
ham        0.94      0.94      0.94    2177
          macro avg    0.94      0.94      0.94    3589
weighted avg    0.91      0.91      0.91    141

[154] accuracy_score(spam_test_target,spam_test_target_predict)
0.93006408470326
```

4) n_estimators = 1000

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
[134] from sklearn.metrics import classification_report
      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
ham        0.94      0.95      0.94    2177
          macro avg    0.93      0.93      0.93    3589
weighted avg    0.92      0.91      0.91    141

[138] accuracy_score(spam_test_target,spam_test_target_predict)
0.9320144887155196
```

5) n_estimators = 5000

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Untitled0.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Code Cells:**
 - [139] Import statements:

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```
 - [139] Model definitions:

```
clf = RandomForestClassifier(n_estimators=5000, 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)
```
 - [140] Model prediction and accuracy:

```
RandomForestClassifier(max_features='sqrt', n_estimators=5000)

[140] 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)
```

Output: 0.9320144887155196
 - [141] Confusion matrix:

```
[141] confusion_matrix(spam_test_target,spam_test_target_predict)
```

Output: array([[2053, 124],
 [120, 1292]])
 - [142] Classification report:

```
[142] classification_report(spam_test_target,spam_test_target_predict)
```

Output:

	precision	recall	f1-score	support	ham	spam	0.91	0.92	0.91	141
accuracy	0.93	0.93	0.93	3589	0.93	0.93	0.93	0.93	0.93	3
	2	3589	2177	3589	nmacro avg	nweighted avg	0.91	0.92	0.91	141
 - [143] Accuracy score:

```
[143] accuracy_score(spam_test_target,spam_test_target_predict)
```

Output: 0.9320144887155196
- Runtime Metrics:** RAM, Disk, Editing.
- Bottom Bar:** Up, Down, Cell, Kernel, Help, etc.

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:

```

Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[74] from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
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", splitter = "best", max_depth = 10, min_samples_split=10, min_samples_leaf=1, random_state=20, class_weight = "balanced")
#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)

DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
max_depth=10, max_features='sqrt',
min_impurity_decrease=0, min_samples_split=10,
min_weight_fraction_leaf=0, random_state=20)

[76] spam_test_target_predict=clf.predict(spam_test_data)
accuracy_score(spam_test_target,spam_test_target_predict)
0.8754527723599889

[77] confusion_matrix(spam_test_target,spam_test_target_predict)
array([[1844, 333],
       [114, 1298]])

[78] classification_report(spam_test_target,spam_test_target_predict)
          precision    recall  f1-score   support
ham        0.94      0.85     0.89    2177
spam       0.80      0.92     0.85    141
macro avg    0.87      0.88     0.87    3589
weighted avg    0.88      0.88     0.88     3

accuracy_score(spam_test_target,spam_test_target_predict)
0.8754527723599889

```

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 screenshot shows a Jupyter Notebook interface with a red border. The title bar says "Untitled0.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and "All changes saved". The toolbar has icons for Comment, Share, and settings. The code cell contains Python code for importing libraries and fitting a Random Forest classifier to spam data. The output cell shows the accuracy score of 0.9281136806910003. The next cell shows a confusion matrix, and the final cell shows a classification report with various metrics.

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators=100, max_features="sqrt", random_state = 42, criterion ="entropy", max_depth=10, min_samples_split=10, min_samples_leaf = 1)
#clf = DecisionTreeClassifier(criterion = "entropy", splitter = "best", max_depth = 10, min_samples_split=10, min_samples_leaf=1, random_state=42, class_weight = "balanced")
#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(criterion='entropy', max_depth=10, max_features='sqrt',
                       max_leaf_nodes=50, min_samples_split=10,
                       min_weight_fraction_leaf=0, random_state=42)

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

0.9281136806910003

[148] confusion_matrix(spam_test_target,spam_test_target_predict)

array([[2069, 108],
       [150, 1262]])

[149] classification_report(spam_test_target,spam_test_target_predict)

              precision    recall  f1-score   support
          ham      0.93      0.95      0.94    2177
          spam      0.92      0.93      0.92    3589
   macro avg  0.93      0.94      0.93    5566
weighted avg  0.93      0.93      0.93    141

accuracy_score(spam_test_target,spam_test_target_predict)

0.9281136806910003
```

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.

- K Neighbors classifier

```

Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[211] from sklearn.ensemble import AdaBoostClassifier
      from sklearn.linear_model import LogisticRegression
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.metrics import confusion_matrix
      from sklearn.metrics import classification_report
      from sklearn.metrics import accuracy_score
      #clf = RandomForestClassifier(n_estimators=100, max_features="sqrt", random_state = 42, criterion = "entropy", max_depth=10, min_samples_split=10, min_samples_leaf = 1)
      #clf = DecisionTreeClassifier(criterion = "entropy", splitter = "best", max_depth = 10, min_samples_split=10, min_samples_leaf=1, random_state=42, class_weight = "balanced")
      clf = KNeighborsClassifier(n_neighbors=3, weights = "uniform", algorithm = "auto", leaf_size = 30, p = 2, n_jobs=1)
      #LRI = LogisticRegression()
      #clf = AdaBoostClassifier(n_estimators = 200,base_estimator=LRI)
      clf.fit(spam_training_data, spam_training_target)

KNeighborsClassifier(n_jobs=10, n_neighbors=3)

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

0.7447757035385901

[214] confusion_matrix(spam_test_target,spam_test_target_predict)

array([[1794,  383],
       [ 533,  879]])

[215] classification_report(spam_test_target,spam_test_target_predict)

          precision    recall  f1-score   support
ham        0.77      0.82      0.80      2177
          0.73      0.72      0.73      3589
macro avg  0.77      0.82      0.80      5899
accuracy   0.74      0.74      0.74      1413

accuracy_score(spam_test_target,spam_test_target_predict)

0.7447757035385901

```

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

```

LogisticRegression(class_weight='none', fit_intercept=1.0,
                   intercept_scaling=1.0, random_state=42)

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

0.910002786291446

[420] confusion_matrix(spam_test_target,spam_test_target_predict)

array([[1993,  184],
       [ 139, 1273]])

[421] classification_report(spam_test_target,spam_test_target_predict)

          precision    recall  f1-score   support
ham        0.93      0.92      0.93      2177
          0.90      0.91      0.91      3589
macro avg  0.93      0.92      0.93      5899
accuracy   0.91      0.91      0.91      1413

accuracy_score(spam_test_target,spam_test_target_predict)

0.910002786291446

```

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

```
✓ [os] from sklearn.metrics import confusion_matrix
✓ [os] from sklearn.metrics import classification_report
✓ [os] 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
  ham       0.90      0.91      0.91    1412
accuracy                           0.94
  0.92      0.92      0.93      0.93    3589
   weighted avg       0.92      0.93      0.93    3589

          precision    recall  f1-score   support
  ham       0.94      0.94      0.94    2177
accuracy                           0.94
  0.92      0.92      0.92      0.92    2177
   macro avg       0.94      0.94      0.94    2177
   spa       0.92

[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.