Homework5_RF_python

April 11, 2021

1 Homework 5: Random Forest

Use both R (randomForest package) and Python (sklearn module) to answer the following questions:

1. Use the provided iris data set with the four features:
   a. Sepal.Length
   b. Sepal.Width
   c. Petal.Length
   d. Petal.Width and only two target Species setosa and versicolor.
2. Create train and test data sets using a fixed (stratified) split of 80-20.
3. Find an appropriate number of trees by testing from 100 to 1000 in steps of 100 using all provided features and (stratified) 10-fold cross validation.
4. Plot the resulting errors for the different numbers of trees, interpret the results, and choose an appropriate number of trees for your random forest (this might not be the model with the lowest error, e. g. when the error reaches a plateau). Train the final random forest on the whole training set.
5. Use the final model to evaluate test loss. Plot the confusion matrix and ROC curve and interpret them.
6. Discuss which actions you could take and why if your model does not perform well enough.

1.1 Imports

[83]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV


from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns
```

2 1 Loading data

[5]:
```
data = pd.read_csv("iris.csv")
data = data[((data.Species=="setosa") | (data.Species=="versicolor"))]
data
```

[5]:
```
    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width     Species
0            5.1          3.5           1.4          0.2      setosa
1            4.9          3.0           1.4          0.2      setosa
2            4.7          3.2           1.3          0.2      setosa
3            4.6          3.1           1.5          0.2      setosa
4            5.0          3.6           1.4          0.2      setosa
..           …            …             …            …          …
95           5.7          3.0           4.2          1.2  versicolor
96           5.7          2.9           4.2          1.3  versicolor
97           6.2          2.9           4.3          1.3  versicolor
98           5.1          2.5           3.0          1.1  versicolor
99           5.7          2.8           4.1          1.3  versicolor

[100 rows x 5 columns]
```

3 2 Split

[8]:
```
data.loc[data.Species=="setosa","Species"]= 0
data.loc[data.Species=="versicolor","Species"]= 1
data.Species=data.Species.astype(int)
train,test = train_test_split(data ,test_size = 0.2, random_state = 0)
train.head()
```

[8]:
```
    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width  Species
43           5.0          3.5           1.6          0.6        0
62           6.0          2.2           4.0          1.0        1
3            4.6          3.1           1.5          0.2        0
71           6.1          2.8           4.0          1.3        1
45           4.8          3.0           1.4          0.3        0
```

4 3 Find perfect number of trees

[91]:
```
rf = RandomForestClassifier(n_jobs=5)
parameters = {"n_estimators":[int(x) for x in np.linspace(start = 100, stop =␣
 ↪1000, num = 10)]}
gsearch= GridSearchCV(rf, parameters, cv=5,n_jobs=5,return_train_score=True)
```

[92]:
```
gsearch.fit(train.iloc[:,0:4], train.iloc[:,-1])
```

```
[92]: GridSearchCV(cv=5, estimator=RandomForestClassifier(n_jobs=5), n_jobs=5,
                   param_grid={'n_estimators': [100, 200, 300, 400, 500, 600, 700,
                                                800, 900, 1000]},
                   return_train_score=True)
```

```
[115]: gsearch.best_params_
```

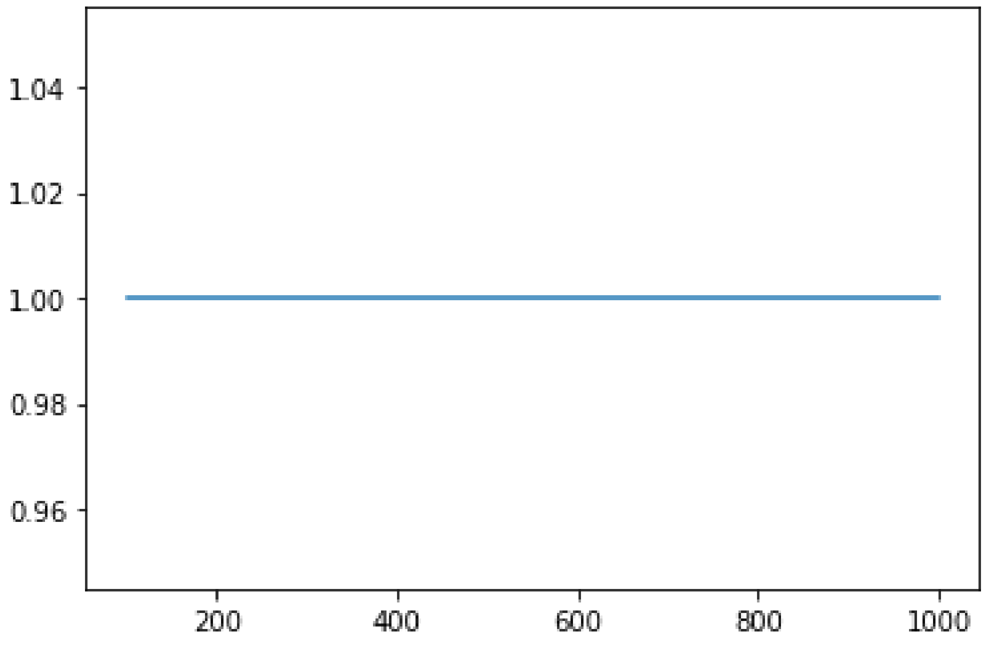
```
[115]: {'n_estimators': 100}
```

5 4 plotting results

```
[73]: gsearch.cv_results_["param_n_estimators"]

      plt.plot(list(gsearch.cv_results_["param_n_estimators"]),gsearch.
       ↪cv_results_["mean_test_score"])
```

```
[73]: [<matplotlib.lines.Line2D at 0x233f9009460>]
```



```
[78]: model = RandomForestClassifier(n_jobs=5, n_estimators=100)
      model.fit(train.iloc[:,0:4], train.iloc[:,-1])
```

```
[78]: RandomForestClassifier(n_jobs=5)
```

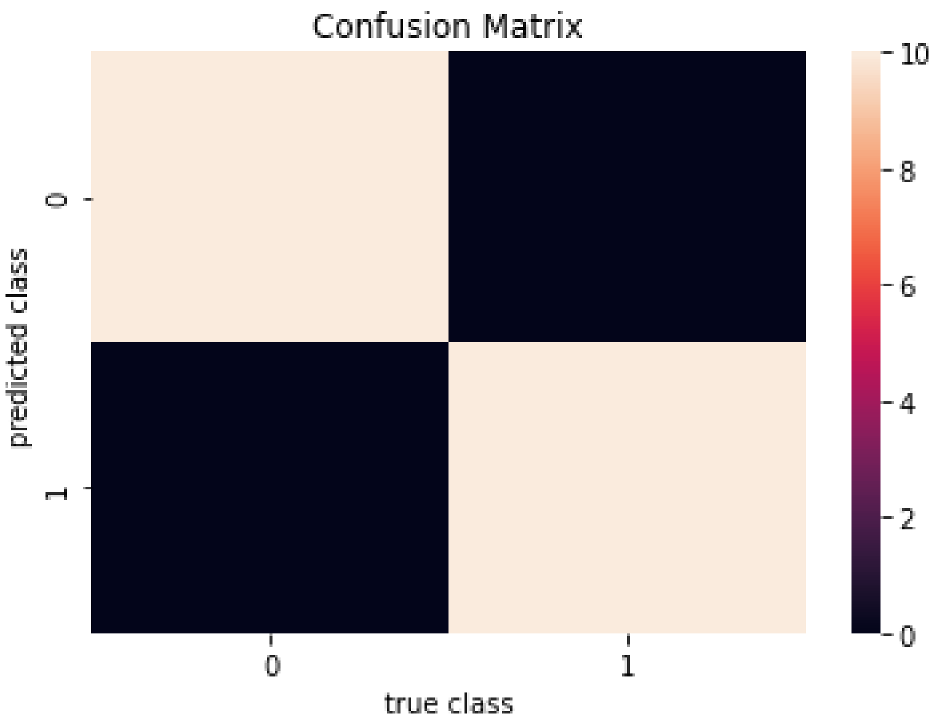
```
[84]: prediction= model.predict(test.iloc[:,0:4])
```

```
conf = confusion_matrix(prediction, test.Species)
conf

sns.heatmap(conf)

plt.title('Confusion Matrix')
plt.xlabel('true class')
plt.ylabel('predicted class')
```

[84]:
```
Text(33.0, 0.5, 'predicted class')
```



[117]:
```
conf
```

[117]:
```
array([[10,  0],
       [ 0, 10]], dtype=int64)
```

6 5 ROC

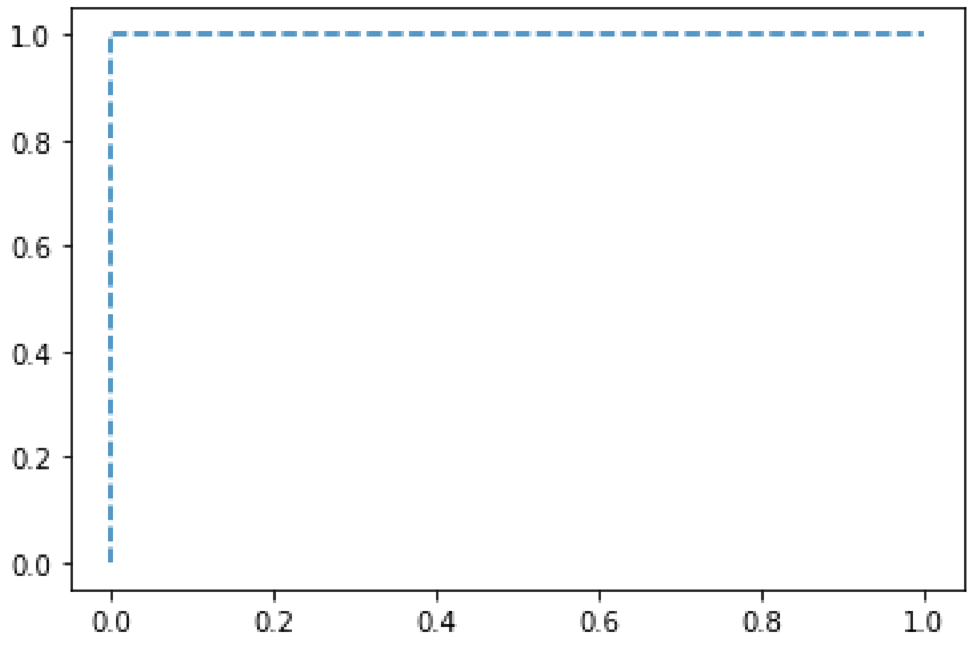
[134]:
```
from sklearn.metrics import roc_auc_score
import sklearn.metrics as metrics

probability = model.predict_proba(test.iloc[:,0:4])[:, 1]
```

```
fpr, tpr,_ = metrics.roc_curve(test.Species, probability)
roc_auc = metrics.auc(fpr, tpr)
```

```
[139]: plt.plot(fpr, tpr, linestyle='--', label='Random prediction (AUROC = %0.3f)' %␣
         ↪roc_auc)
```

```
[139]: [<matplotlib.lines.Line2D at 0x233f96708e0>]
```



7 5 Discussion

The model performs well enough. Since the performance reaches 100% accuracy indepent of number of the Trees, the lowest number of trees is taken. Higher number of trees increases complexity and evaluation time. Hence, it is a tradeof between complexity, time and performance. As letter stays the same there is no need for an increased number of trees.