PS15

January 24, 2023

0.1 Random forests

0.2 Iris plants classification

[1]:

```python
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
```

[2]:

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
#from sklearn.metrics import accuracy_score
```

[7]:

```python
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
# n_jobs=-1 : use all available CPU's
rnd_clf.fit(X_train, y_train)

y_pred = rnd_clf.predict(X_test)
print(rnd_clf.score(X_test,y_test))

#print(accuracy_score(y_pred, y_test))
```

```
1.0
1.0
```

[8]:

```python
for name, score in zip(iris.feature_names, rnd_clf.feature_importances_):
    print(name, score)
```

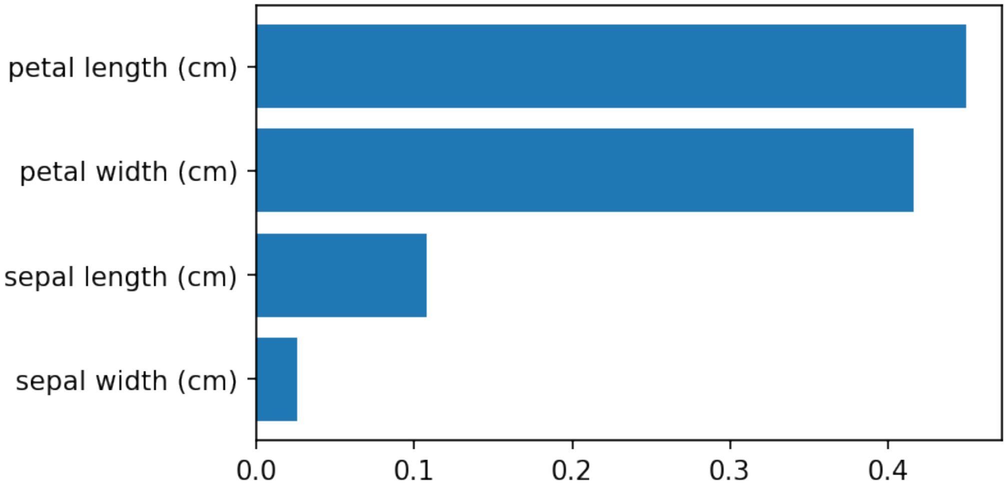
```
sepal length (cm) 0.10810853310073656
sepal width (cm) 0.025670524096247496
petal length (cm) 0.44972571693418767
petal width (cm) 0.4164952258688282
```

[12]:

```python
import matplotlib.pyplot as plt
import numpy as np
```

```
sorted_idx = rnd_clf.feature_importances_.argsort()
plt.figure(figsize=(5,3), dpi=150)
plt.barh(np.asarray(iris.feature_names)[sorted_idx],
         rnd_clf.feature_importances_[sorted_idx])


#sorted_idx = rnd_clf.feature_importances_.argsort()[::-1]
#plt.barh(np.asarray(iris.feature_names)[sorted_idx][::-1],
#          rnd_clf.feature_importances_[sorted_idx][::-1])
```

```
[12]: <BarContainer object of 4 artists>
```



0.3 California Housing Price prediction

```
[13]: from sklearn.datasets import fetch_california_housing
      cali_prices = fetch_california_housing()
      X_reg = cali_prices.data
      y_reg = cali_prices.target
```

```
[14]: from sklearn.model_selection import train_test_split
      from sklearn.ensemble import RandomForestRegressor
      from sklearn.metrics import mean_squared_error
```

```
[15]: X_train,X_test,y_train,y_test = train_test_split(X_reg,y_reg,test_size = 0.01)
      for_reg = RandomForestRegressor()
      for_reg.fit(X_train, y_train)
      y_pred_train = for_reg.predict(X_train)
      y_pred_test = for_reg.predict(X_test)
      print(mean_squared_error(y_pred_train, y_train))
print(mean_squared_error(y_pred_test, y_test))
```

```
0.034267374420638049
0.21248308755938008
```

0.4 MNIST classification

[18]:

```
from keras.datasets import mnist
from sklearn.model_selection import train_test_split

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train / 255., X_test / 255.

X_train = X_train.reshape(-1, 28*28)
X_test =  X_test.reshape(-1, 28*28)

#X_train = X_train.reshape(X_train.shape[0], -1)
#X_test =  X_test.reshape(X_test.shape[0], -1)
```

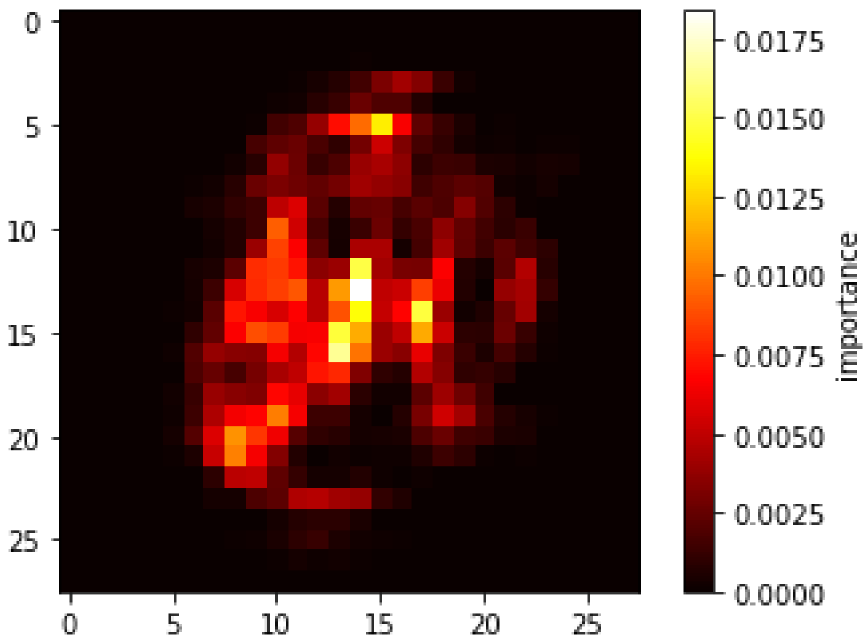
[20]:

```
from sklearn.ensemble import RandomForestClassifier
#from sklearn.metrics import accuracy_score

rnd_clf = RandomForestClassifier(n_estimators=500,
                                 max_leaf_nodes=16,
                                 n_jobs=-1)
rnd_clf.fit(X_train, y_train)
y_pred = rnd_clf.predict(X_test)
print(rnd_clf.score(X_test,y_test))
#print(accuracy_score(y_pred,y_test))
```

```
0.8309
```

[16]:

```
import matplotlib.pyplot as plt

feature_importances = rnd_clf.feature_importances_
feature_importances = feature_importances.reshape(28, 28)
plt.imshow(feature_importances, cmap='hot')
plt.colorbar(label='importance')
plt.show()
```



0.5 Hyperparameter search: GridSearchCV

[17]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

Iris = load_iris()
X = Iris.data
y = Iris.target
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

[18]:
```
from sklearn.model_selection import GridSearchCV
```

[19]:
```
param_grid = {'n_estimators': [3,10,100,500],
              'max_features': [0.25, 0.5, 0.75, 1]}

forest_clf = RandomForestClassifier(max_depth=2)
grid_search = GridSearchCV(forest_clf,param_grid,cv=5,scoring='accuracy')
grid_search.fit(X_train, y_train)
```

[19]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(max_depth=2),
             param_grid={'max_features': [0.25, 0.5, 0.75, 1],
                         'n_estimators': [3, 10, 100, 500]},
             scoring='accuracy')
```

[20]:
```
print(grid_search.best_params_)
```

```
{'max_features': 0.25, 'n_estimators': 100}
```

[21]:

```
print(grid_search.best_estimator_)
```

```
RandomForestClassifier(max_depth=2, max_features=0.25)
```

[22]:

```
print(grid_search.best_score_)
```

```
0.9416666666666667
```

[23]:

```
feature_importances = grid_search.best_estimator_.feature_importances_
print(feature_importances)
```

```
[0.21406567 0.15593039 0.32120427 0.30879968]
```

[24]:

```
y_pred = grid_search.best_estimator_.predict(X_test)
print(accuracy_score(y_pred, y_test))
```

```
0.9666666666666667
```

0.6 Hyperparameter search: RandomizedSearchCV

[25]:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
Iris = load_iris()
X = Iris.data
y = Iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[26]:

```
from sklearn.model_selection import RandomizedSearchCV
```

[27]:

```
param_distributions = {'n_estimators': range(500),
                       'max_features': range(1, 5)}

forest_clf = RandomForestClassifier(max_depth=2)

randomized_search = RandomizedSearchCV(forest_clf,
                                       param_distributions,
                                       cv=4, n_iter = 50,
                                       scoring='accuracy')

randomized_search.fit(X_train, y_train)
```

[27]:

```
RandomizedSearchCV(cv=4, estimator=RandomForestClassifier(max_depth=2),
                   n_iter=50,
                   param_distributions={'max_features': range(1, 5),
                                        'n_estimators': range(0, 500)},
                   scoring='accuracy')
```

[28]:

```
print(randomized_search.best_params_)
```

```
{'n_estimators': 212, 'max_features': 2}
```

[29]:

```
print(randomized_search.best_estimator_)
```

```
RandomForestClassifier(max_depth=2, max_features=2, n_estimators=212)
```

[30]:

```
print(randomized_search.best_score_)
```

```
0.9750000000000001
```

[31]:

```
feature_importances = randomized_search.best_estimator_.feature_importances_
print(feature_importances)
```

```
[0.1202956  0.00995066 0.43334277 0.43641097]
```

[32]:

```
y_pred = randomized_search.best_estimator_.predict(X_test)

print(accuracy_score(y_pred, y_test))
```

```
0.9333333333333333
```