exam_2021_01_27_notebook_sol

June 28, 2022

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
from matplotlib import cm

from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder, MinMaxScaler,␣
 ↪StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪ParameterGrid
from scipy.optimize import linear_sum_assignment
from sklearn.model_selection import cross_val_score
from sklearn.metrics import silhouette_score, silhouette_samples,␣
 ↪accuracy_score, classification_report, confusion_matrix,␣
 ↪plot_confusion_matrix

from sklearn.ensemble import BaggingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN

from sklearn.tree import plot_tree
%matplotlib inline

rnd_state = 42
```

[2]:
```
from utils import plot_silhouette, plot_clusters, Normalization,␣
 ↪Standardization, remap
```

```
[9]: data_file = 'exam_2021_01_27.csv'
     delimiter = ','
     df = pd.read_csv(data_file, sep=delimiter)
```

```
[10]: df.describe()
```

```
[10]:                 0           1           3           4
      count  141.000000  140.000000  137.000000  150.000000
      mean     5.897872    3.036429    1.290511    1.000000
      std      0.820232    0.437654    0.733934    0.819232
      min      4.300000    2.000000    0.100000    0.000000
      25%      5.200000    2.800000    0.400000    0.000000
      50%      5.800000    3.000000    1.400000    1.000000
      75%      6.400000    3.300000    1.800000    2.000000
      max      7.900000    4.400000    2.500000    2.000000
```

```
[26]: df
```

```
[26]:        0    1  2    3  4
      4    5.0  3.6  a  0.2  0
      7    5.0  3.4  a  0.2  0
      10   5.4  3.7  a  0.2  0
      11   4.8  3.4  a  0.2  0
      13   4.3  3.0  a  0.1  0
      ..   …    …   ..  …   ..
      145  6.7  3.0  d  2.3  2
      146  6.3  2.5  d  1.9  2
      147  6.5  3.0  d  2.0  2
      148  6.2  3.4  d  2.3  2
      149  5.9  3.0  d  1.8  2

      [122 rows x 5 columns]
```

```
[27]: df=df.dropna()
      df
```

```
[27]:        0    1  2    3  4
      4    5.0  3.6  a  0.2  0
      7    5.0  3.4  a  0.2  0
      10   5.4  3.7  a  0.2  0
      11   4.8  3.4  a  0.2  0
      13   4.3  3.0  a  0.1  0
      ..   …    …   ..  …   ..
      145  6.7  3.0  d  2.3  2
      146  6.3  2.5  d  1.9  2
      147  6.5  3.0  d  2.0  2
      148  6.2  3.4  d  2.3  2
```

```
149  5.9  3.0  d  1.8  2

[122 rows x 5 columns]
```

[12]:
```
transf_dtype = np.int32
ordinal_transformer = OrdinalEncoder(dtype = transf_dtype)
ordinal_features =[2]
print("The ordinal features are:")
print(ordinal_features)
```

```
The ordinal features are:
[2]
```

[17]:
```
preprocessor = ColumnTransformer(
    transformers = [('ord', ordinal_transformer, ordinal_features)],remainder =␣
 ↪'passthrough'
    )

df1=preprocessor.fit_transform(df)
```

[19]:
```
df_transformed = pd.DataFrame(df1)
```

[47]:
```
df_transformed
```

[47]:
```
       0    1    2    3    4
0    0.0  5.0  3.6  0.2  0.0
1    0.0  5.0  3.4  0.2  0.0
2    0.0  5.4  3.7  0.2  0.0
3    0.0  4.8  3.4  0.2  0.0
4    0.0  4.3  3.0  0.1  0.0
..   …  …  …  …  …
117  3.0  6.7  3.0  2.3  2.0
118  3.0  6.3  2.5  1.9  2.0
119  3.0  6.5  3.0  2.0  2.0
120  3.0  6.2  3.4  2.3  2.0
121  3.0  5.9  3.0  1.8  2.0

[122 rows x 5 columns]
```

[58]:
```
X = df_transformed.drop(2, axis=1)
df_transformed[2] = (df_transformed[2]).astype(int)
y = df_transformed[2]
```

[59]:
```
ts = 0.3

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=ts, random_state=rnd_state)
```

```
print("Training on %d examples" % len(X_train))
```

```
Training on 85 examples
```

[60]:

```
y_train
```

[60]:

```
30     2
77     3
65     2
9      3
33     3
      ..
106    2
14     4
92     3
51     2
102    2
Name: 2, Length: 85, dtype: int32
```

[86]:

```
def print_results(model):
    print("-"*40)
    print("Best parameters set found on train set:")
    print()
    # if best is linear there is no gamma parameter
    print(model.best_params_)
    print()
    print("Grid scores on train set:")
    print()
    means = model.cv_results_['mean_test_score']
    stds = model.cv_results_['std_test_score']
    params = model.cv_results_['params']
    for mean, std, params_tuple in zip(means, stds, params):
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params_tuple))
    print()
    print("Detailed classification report for the best parameter set:")
    print()
    print("The model is trained on the full train set with cross validation")
    print("The scores are computed on the full test set.")
    print()
    y_true, y_pred = y_test, model.predict(X_test)
    print(classification_report(y_true, y_pred))
    print("Below is the normalized confusion_matrix")
    print()
    Xc=confusion_matrix(y_test, m.predict(X_test))
    Xn = (Xc-Xc.min())/(Xc.max()-Xc.min())
    print(Xn)
    print("below is the plot_confusion_matrix")
    print()
    print(plot_confusion_matrix(model,X_train,y_train))
    print()
```

[66]:
```
model_lbls = [
              'model1',
              'model2',
            ]

# Set the parameters to be explored by the grid for each classifier
tuned_param_dt = [{'max_depth': list(range(1,20))}]
tuned_param_nb = [{'var_smoothing': [10, 1, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6,␣
 ↪1e-07, 1e-8, 1e-9, 1e-10]}]
tuned_param_lp = [{'early_stopping': [True]}]
tuned_param_svc = [{'kernel': ['rbf'],
                    'gamma': [1e-3, 1e-4],
                    'C': [1, 10, 100, 1000],
                    },
                    {'kernel': ['linear'],
                     'C': [1, 10, 100, 1000],
                    },
                   ]
tuned_param_knn =[{'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}]


models = {
    'model1': {'name': 'Decision Tree',
           'estimator': DecisionTreeClassifier(),
           'param': tuned_param_dt,
          },
    'model2': {'name': 'Gaussian Naive Bayes',
           'estimator': GaussianNB(),
           'param': tuned_param_nb
          },
     #'lp': {'name': 'Linear Perceptron   ',
     #      'estimator': Perceptron(),
      #     'param': tuned_param_lp,
       #   },
    #'svc':{'name': 'Support Vector      ',
        #   'estimator': SVC(),
         #  'param': tuned_param_svc
          #},
    #'knn':{'name': 'K Nearest Neighbor ',
     #      'estimator': KNeighborsClassifier(),
     #      'param': tuned_param_knn
    # }
}

scores = [
          'accuracy',
#           'recall',
    #       'precision'
         ]
```

[67]:

```
results_short = {}
modelslist = []
for m in model_lbls:
    print('-'*40)
    print("Trying model {}".format(models[m]['name']))
    clf = GridSearchCV(models[m]['estimator'], models[m]['param'], cv=5,
                       scoring='%s_macro' % score,
                       return_train_score = False,
                       n_jobs = 2, # this allows using multi-cores
                       )
    clf.fit(X_train, y_train)
    print_results(clf)
    modelslist.append(clf)
    results_short[m] = clf.best_score_
```

```
----------------------------------------
Trying model Decision Tree

C:\Anaconda\lib\site-packages\sklearn\model_selection\_split.py:666:
UserWarning: The least populated class in y has only 3 members, which is less
than n_splits=5.
  warnings.warn(("The least populated class in y has only %d"

Best parameters set found on train set:

{'max_depth': 4}

Grid scores on train set:

0.420 (+/-0.394) for {'max_depth': 1}
0.648 (+/-0.409) for {'max_depth': 2}
0.702 (+/-0.362) for {'max_depth': 3}
0.750 (+/-0.403) for {'max_depth': 4}
0.729 (+/-0.391) for {'max_depth': 5}
0.723 (+/-0.344) for {'max_depth': 6}
0.717 (+/-0.383) for {'max_depth': 7}
0.743 (+/-0.331) for {'max_depth': 8}
0.741 (+/-0.359) for {'max_depth': 9}
0.718 (+/-0.345) for {'max_depth': 10}
```

```
0.733 (+/-0.323) for {'max_depth': 11}
0.742 (+/-0.366) for {'max_depth': 12}
0.744 (+/-0.396) for {'max_depth': 13}
0.734 (+/-0.388) for {'max_depth': 14}
0.718 (+/-0.345) for {'max_depth': 15}
0.718 (+/-0.345) for {'max_depth': 16}
0.716 (+/-0.353) for {'max_depth': 17}
0.718 (+/-0.345) for {'max_depth': 18}
0.716 (+/-0.353) for {'max_depth': 19}

Detailed classification report for the best parameter set:

The model is trained on the full train set.
The scores are computed on the full test set.

              precision    recall  f1-score   support

           2       1.00      0.33      0.50        18
           3       0.57      0.94      0.71        18
           4       0.00      0.00      0.00         1

    accuracy                           0.62        37
   macro avg       0.52      0.43      0.40        37
weighted avg       0.76      0.62      0.59        37


----------------------------------------
Trying model Gaussian Naive Bayes
Best parameters set found on train set:

{'var_smoothing': 0.1}

Grid scores on train set:

0.208 (+/-0.094) for {'var_smoothing': 10}
0.402 (+/-0.396) for {'var_smoothing': 1}
0.592 (+/-0.218) for {'var_smoothing': 0.1}
0.555 (+/-0.276) for {'var_smoothing': 0.01}
0.538 (+/-0.282) for {'var_smoothing': 0.001}
0.538 (+/-0.282) for {'var_smoothing': 0.0001}
0.538 (+/-0.282) for {'var_smoothing': 1e-05}
0.538 (+/-0.282) for {'var_smoothing': 1e-06}
0.538 (+/-0.282) for {'var_smoothing': 1e-07}
0.538 (+/-0.282) for {'var_smoothing': 1e-08}
0.538 (+/-0.282) for {'var_smoothing': 1e-09}
0.538 (+/-0.282) for {'var_smoothing': 1e-10}

Detailed classification report for the best parameter set:
```

```
The model is trained on the full train set.
The scores are computed on the full test set.

              precision    recall  f1-score   support

           2       0.70      0.89      0.78        18
           3       0.86      0.33      0.48        18
           4       0.14      1.00      0.25         1

    accuracy                           0.62        37
   macro avg       0.57      0.74      0.50        37
weighted avg       0.76      0.62      0.62        37


C:\Anaconda\lib\site-packages\sklearn\model_selection\_split.py:666:
UserWarning: The least populated class in y has only 3 members, which is less
than n_splits=5.
  warnings.warn(("The least populated class in y has only %d"
```

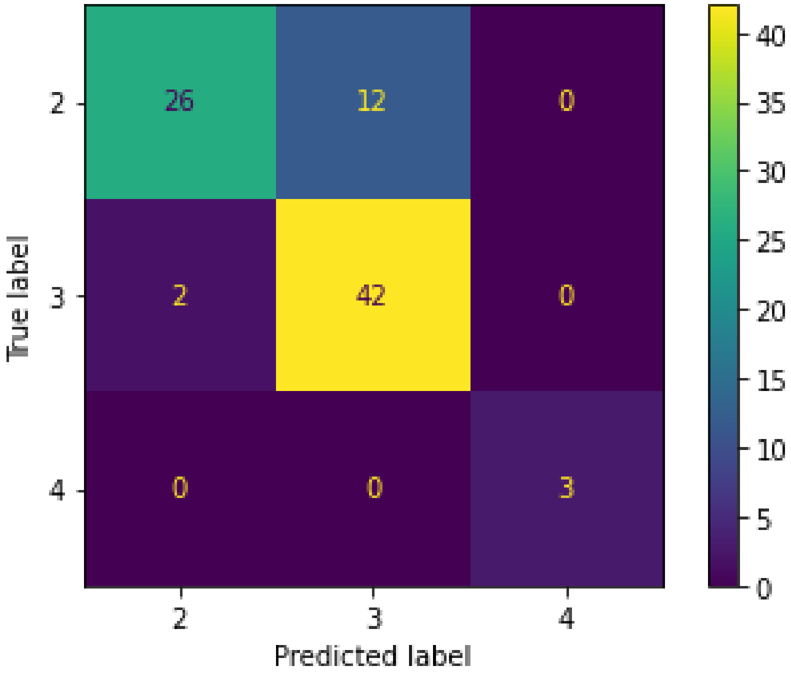
[87]:

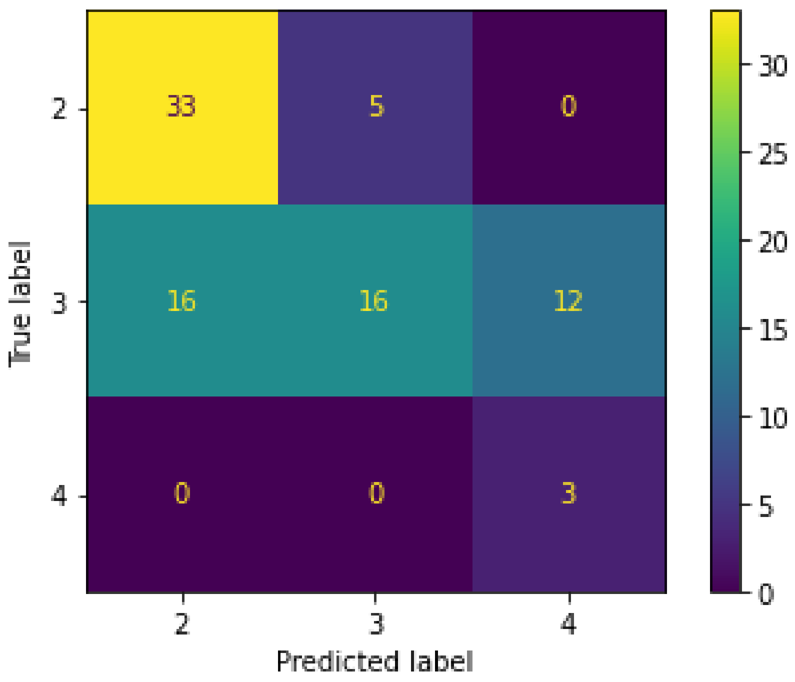
```python
for m in modelslist:
    print_results(m)
```

```
----------------------------------------
Best parameters set found on train set:

{'max_depth': 4}

Grid scores on train set:

0.420 (+/-0.394) for {'max_depth': 1}
0.648 (+/-0.409) for {'max_depth': 2}
0.702 (+/-0.362) for {'max_depth': 3}
0.750 (+/-0.403) for {'max_depth': 4}
0.729 (+/-0.391) for {'max_depth': 5}
0.723 (+/-0.344) for {'max_depth': 6}
0.717 (+/-0.383) for {'max_depth': 7}
0.743 (+/-0.331) for {'max_depth': 8}
0.741 (+/-0.359) for {'max_depth': 9}
0.718 (+/-0.345) for {'max_depth': 10}
0.733 (+/-0.323) for {'max_depth': 11}
0.742 (+/-0.366) for {'max_depth': 12}
0.744 (+/-0.396) for {'max_depth': 13}
0.734 (+/-0.388) for {'max_depth': 14}
0.718 (+/-0.345) for {'max_depth': 15}
0.718 (+/-0.345) for {'max_depth': 16}
0.716 (+/-0.353) for {'max_depth': 17}
```

```
0.718 (+/-0.345) for {'max_depth': 18}
0.716 (+/-0.353) for {'max_depth': 19}

Detailed classification report for the best parameter set:

The model is trained on the full train set with cross validation
The scores are computed on the full test set.

              precision    recall  f1-score   support

           2       1.00      0.33      0.50        18
           3       0.57      0.94      0.71        18
           4       0.00      0.00      0.00         1

    accuracy                           0.62        37
   macro avg       0.52      0.43      0.40        37
weighted avg       0.76      0.62      0.59        37

Below is the normalized confusion_matrix

[[0.35294118 0.70588235 0.        ]
 [0.         1.         0.05882353]
 [0.         0.05882353 0.        ]]
below is the plot_confusion_matrix

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x000001F23ECA6D00>

----------------------------------------
Best parameters set found on train set:

{'var_smoothing': 0.1}

Grid scores on train set:

0.208 (+/-0.094) for {'var_smoothing': 10}
0.402 (+/-0.396) for {'var_smoothing': 1}
0.592 (+/-0.218) for {'var_smoothing': 0.1}
0.555 (+/-0.276) for {'var_smoothing': 0.01}
0.538 (+/-0.282) for {'var_smoothing': 0.001}
0.538 (+/-0.282) for {'var_smoothing': 0.0001}
0.538 (+/-0.282) for {'var_smoothing': 1e-05}
0.538 (+/-0.282) for {'var_smoothing': 1e-06}
0.538 (+/-0.282) for {'var_smoothing': 1e-07}
0.538 (+/-0.282) for {'var_smoothing': 1e-08}
0.538 (+/-0.282) for {'var_smoothing': 1e-09}
0.538 (+/-0.282) for {'var_smoothing': 1e-10}
```

```
Detailed classification report for the best parameter set:

The model is trained on the full train set with cross validation
The scores are computed on the full test set.

              precision    recall  f1-score   support

           2       0.70      0.89      0.78        18
           3       0.86      0.33      0.48        18
           4       0.14      1.00      0.25         1

    accuracy                           0.62        37
   macro avg       0.57      0.74      0.50        37
weighted avg       0.76      0.62      0.62        37

Below is the normalized confusion_matrix

[[1.     0.0625 0.0625]
 [0.4375 0.375  0.3125]
 [0.     0.     0.0625]]
below is the plot_confusion_matrix

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x000001F23FE7BC10>
```




33
5
0
16
16
12
0
0
3
2
3
4
True label
2
3
4
Predicted label
30
25
20
15
10
5
0

[ ]: