K-Fold Cross Validation & Grid Search & Feature Selection

Importing the libraries

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [2]:

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
```

Splitting the dataset into the Training set and Test set

In [3]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Feature Scaling

In [4]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the Kernel SVM model on the Training set

In [5]:

```
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state=0)
# classifier = SVC(C=0.25, gamma=0.8, kernel='rbf', random_state=0)
classifier.fit(X_train, y_train)
```

Out[5]:

```
SVC(random_state=0)
```

Making the Confusion Matrix

In [6]:

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

y_pred = classifier.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

```
[[55  3]
 [ 1 21]]
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80

0.95
```

Applying k-Fold Cross Validation

In [7]:

```
from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator=classifier, X=X_train, y=y_train, cv=10, verbose=1) # cv: Cross Validation=k=10
print(accuracies)
print(f"Accuracy: {np.round(accuracies.mean()*100, 2)} %") # average accuracy score
print(f"Std: {accuracies.std()*100} %")
```

```
[0.84375 0.875   0.90625 0.84375 0.9375  0.84375 0.90625 0.90625 1.
 0.9375 ]
Accuracy: 90.0 %
Std: 4.80071609241788 %
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.0s finished
```

Applying Grid Search to find the best model and the best parameters

In [8]:

```python
from sklearn.model_selection import GridSearchCV

parameters = [{'C': [0.25, 0.5, 0.75, 1], 'kernel': ['linear']},
              {'C': [0.25, 0.5, 0.75, 1], 'kernel': ['rbf'], 'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]}]
grid_search = GridSearchCV(estimator = classifier,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 10, # cv: Cross Validation, k=10
                           n_jobs = -1, # all cpu
                           verbose=1 # print
                          )
grid_search.fit(X_train, y_train)
print(grid_search.best_estimator_)
print(grid_search.best_params_)
print(grid_search.best_score_) # precision
print(grid_search.best_index_)
```

```
Fitting 10 folds for each of 40 candidates, totalling 400 fits
SVC(C=0.25, gamma=0.8, random_state=0)
{'C': 0.25, 'gamma': 0.8, 'kernel': 'rbf'}
0.90625
11
```

n_jobs设定工作的core数量

等于-1的时候，表示cpu里的所有core进行工作。

grid_search里的最好score

In [9]:

```python
grid_predictions = grid_search.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

print(classification_report(y_test, grid_predictions))
print(accuracy_score(y_test, grid_predictions))
print(confusion_matrix(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80

0.95
[[55  3]
 [ 1 21]]
```

手动调最好score

In [10]:

```python
classifier = SVC(C=0.25, gamma=0.8, kernel='rbf', random_state=0)
classifier.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = classifier.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80

0.95
[[55  3]
 [ 1 21]]
```

Feature Selection 递归特征消除（Recursive feature elimination） and Re-do Model Training

In [11]:

```python
from sklearn.feature_selection import RFE # 递归特征消除 (Recursive feature elimination)

classifier = SVC(C=0.25, gamma=0.8, kernel='rbf', random_state=0)
rfe = RFE(estimator=classifier, n_features_to_select=10) # find 10 columns that are most valuable
rfe.fit(X_train, y_train)
```

Out[11]: `RFE(estimator=SVC(C=0.25, gamma=0.8, random_state=0), n_features_to_select=10)`

In [12]:

```python
# Two columns shoud be included
rfe.support_
```

Out[12]: `array([ True,  True])`

In [13]:

```python
rfe.ranking_
```

Out[13]: `array([1, 1])`

In [14]:

```python
X = dataset.iloc[:, :-1]
X = X[X.columns[rfe.support_]]
y = dataset.iloc[:, -1]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.svm import SVC
classifier = SVC(C=0.25, gamma=0.8, kernel='rbf', random_state=0)
classifier.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = classifier.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80

0.95
[[55  3]
 [ 1 21]]
```