SVM

Support Vector Machine의 약자

이름대로 Vector 개념을 사용

SVM은 서포트 벡터(support vectors)를 사용해서 결정 경계(Decision Boundary)를 정의하고, 분류되지 않은 점을 해당 결정 경계와 비교해서 분류

여기서 서포트 벡터(support vectors)는 결정 경계에 가장 가까운 각 클래스의 점들을 말함

그래서 분류되지 않은 새로운 점이 나타나면 경계의 어느 쪽에 속하는지 확인해서 분류 과제를 수행할 수 있음

In [1]:
```
import pandas as pd
import seaborn as sns

from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
```

In [2]:
```
df = sns.load_dataset('titanic')
```

In [3]:
```
rdf = df.drop(['deck', 'embark_town'], axis=1)
rdf = rdf.dropna(subset=['age'], how='any', axis=0)
most_freq = rdf['embarked'].value_counts(dropna=True).idxmax()
rdf['embarked'].fillna(most_freq, inplace=True)
```

In [4]:
```
features = rdf[['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'embarked']]
```

In [5]:
```
onehot_sex = pd.get_dummies(features['sex'])
features = pd.concat([features, onehot_sex], axis=1)

onehot_embarked = pd.get_dummies(features['embarked'], prefix='town')
features = pd.concat([features, onehot_embarked], axis=1)

features.drop(['sex', 'embarked'], axis=1, inplace=True)
```

Modeling

In [6]:
```
X=features[['pclass', 'age', 'sibsp', 'parch', 'female', 'male', 'town_C', 'town_Q'
y=features['survived']                    #종속 변수 Y
```

In [7]:
```
X = preprocessing.StandardScaler().fit(X).transform(X)
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
print('train data 개수: ', X_train.shape)
print('test data 개수: ', X_test.shape)
```

```
train data 개수:  (499, 9)
test data 개수:  (215, 9)
```

SVM Model

모델 객체 생성 (kernel='rbf' 적용)

sklearn에서 가져온 svm 모듈의 svc() 함수를 사용하여 모델 객체를 생성

이때 데이터를 벡터 공간으로 매핑하는 함수를 커널이라 하는데

일단은 'rbf' 옵션으로 함수를 적용한다

rbf : radical basis function (방사형 기저 함수)

이외에도 Linear, Polynimial, Sigmoid 등의 커널 옵션을 줄 수 있다

In [9]:
```
svm_model = svm.SVC(kernel='linear', C = 0.1)
```

In [10]:
```
svm_model.fit(X_train, y_train)
```

Out[10]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [11]:
```
y_hat = svm_model.predict(X_test)
```

In [12]:
```
svm_report = metrics.classification_report(y_test, y_hat)

print(svm_report)
```

```
              precision    recall  f1-score   support

           0       0.79      0.85      0.82       125
           1       0.77      0.69      0.73        90

    accuracy                           0.78       215
   macro avg       0.78      0.77      0.77       215
weighted avg       0.78      0.78      0.78       215
```

In [13]:
```
r_square = svm_model.score(X_test, y_test)
print(r_square)
```

```
0.7813953488372093
```

In [ ]:
```

```