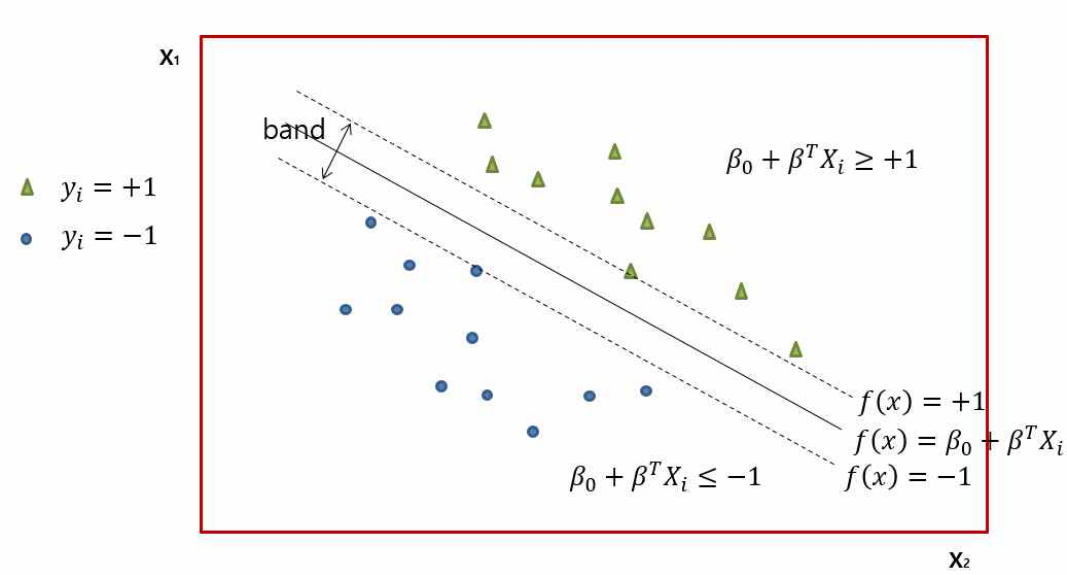
12. SVM

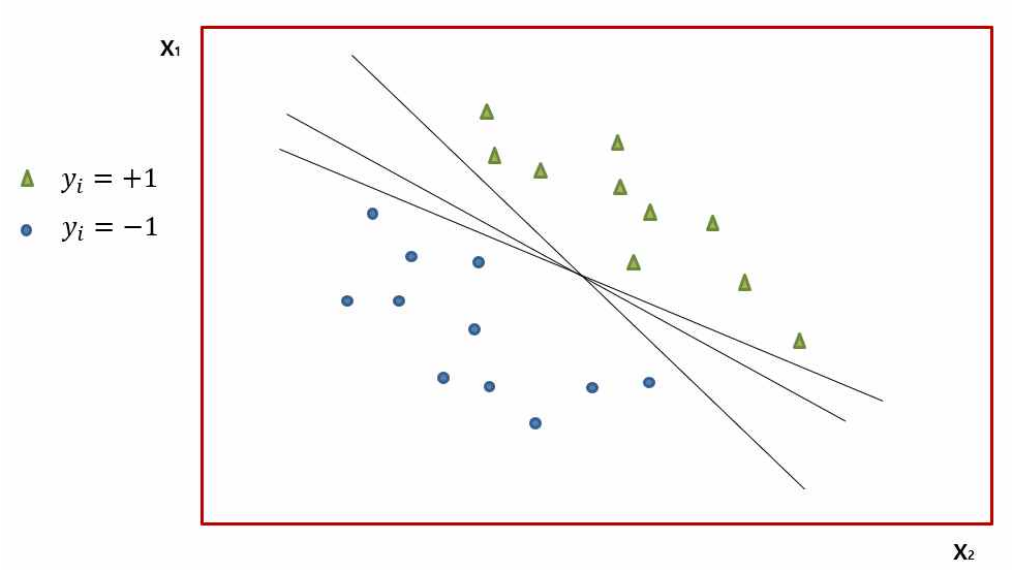
1. SVM의 이해

○ SVM

- 아래의 그림은 특성변수 X_1과 X_2에 따라 2개의 그룹으로 나누어진 것을 볼 수 있으며 두 개의 직선이 두 그룹을 나누고 있음.



- 이 직선은 두 그룹의 관측치 중에서 가장 가까운 거리를 정확하게 반으로 나누고 있고 그 결과로 상하로 대칭되는 밴드(band)를 보여주고 있음.
- 2개의 그룹을 나누는 수 많은 직선이 있을 것이며, 그림과 같이 동일하게 밴드를 그렸다고 상상해 보겠습니다.



- 그리고 이 밴드의 넓이가 두 개의 그룹을 나누는 벽이 두께라고 할 때 많은 직선 중 벽의 두께가 가장 두꺼운 직선이 가장 합리적인 분류선이 될 것임.
- 이것이 써포트벡터머신의 핵심적인 아이디어이며 밴드를 구성하는 2개의 점선 위의 자료를 써포트벡터라고 함.

- 두 그룹을 $y=\{-1,1\}$로 표기하고 그림의 직선이 $f(\boldsymbol{X})=\beta_0+\boldsymbol{\beta}'\boldsymbol{X}=0$이라고 할 때, $f(\boldsymbol{X})$가 분류선이므로 SVM은 모든 관측치에 대해

$$\beta_0+\boldsymbol{\beta}'\boldsymbol{X}_i \geq 1 \text{ 만약 } y_i=1$$

$$\beta_0+\boldsymbol{\beta}'\boldsymbol{X}_i \leq -1 \text{ 만약 } y_i=-1$$

를 만족하는 β_0와 $\boldsymbol{\beta}$를 구하는 문제가 됨.

- 즉, 그림에서 밴드의 경계 상위값이 1, 하위값이 -1이 되도록 하면 위의 식을 만족하게 됨. 경계상위 특성변수의 값을 $\boldsymbol{X}_+$, 하위 특성변수값을 $\boldsymbol{X}_-$로 표현하면

$$\beta_0+\boldsymbol{\beta}'\boldsymbol{X}_+=1,\ \beta_0+\boldsymbol{\beta}'\boldsymbol{X}_-=-1$$

이 되고 이 두 직선간의 거리는

$$\boldsymbol{\beta}'(\boldsymbol{X}_+-\boldsymbol{X}_-)=2$$

가 됨.

- 두 그룹을 완전하게 구분하는 모든 직선은 위의 식을 만족하므로 이는

표준화가 필요함. $\|\beta\| = \sqrt{\sum_{j=1}^{d}\beta_j^2}$ 이라고 정의한다면

$$\frac{\beta'}{\|\beta\|}(X_+ - X_-) = \frac{2}{\|\beta\|}$$

와 같이 표준화 할 수 있음.

- 우의 식은 밴드의 넓이를 나타내므로 이 넓이를 최대화 하는 것이 SVM의 목적임.
- 그러므로 SVM의 문제는 다음과 같이 $\|\beta\|$를 최소화하는 문제로 귀결됨.

  $y_i(\beta_0 + \beta' X_i) \geq 1$인 조건하에서 $\|\beta\|$를 최소화하는 β

- 그러나 실제 문제에서는 두 그룹이 완전하게 구분되는 학습데이터는 존재하지 않음.
- 이를 위해 슬랙변수라고 하는 완화변수 $\zeta \geq 0,\ i = 1,...,n$을 도입하여 $y_i(\beta_0 + \beta' X_i) \geq 1$ 조건을 완화하기 위해 $y_i(\beta_0 + \beta' X_i) \geq 1 - \zeta_i$로 바꾸어서 $\|\beta\|$를 최소화 함. 즉,

  $y_i(\beta_0 + \beta' X_i) \geq 1 - \zeta_i$인 조건하에서 $\|\beta\|$를 최소화하는 β

  로 SVM의 목적이 변하게 됨.

- 여기서 슬랙변수는 일종의 오차라고 해석할 수 있고, 이 오차를 어느 정도까지 허용한 상태에서 $\|\beta\|$를 최소로 만드는 β의 추정 문제로 연결됨.
- 즉, $y_i(\beta_0 + \beta' X_i) \geq 1 - \zeta_i$인 조건하에서 $\|\beta\| + c\sum_{i=1}^{n}\zeta_i$를 최소화하는 β를 구하는 문제로 생각할 수 있음. 여기서 c는 초모수임.
- 여기서 초모수 c를 크게주면 $\sum_{i=1}^{n}\zeta_i$가 약간만 커져도 목적함수 $\|\beta\| + c\sum_{i=1}^{n}\zeta_i$가 커지게 되고 오차인 ζ_i는 허용하지 않으므로 밴드의 넓이가 좁아지게 됨. 이는 overfitting 문제가 발생할 가능성이 높아지게 됨을 의미함.
- 반대로 초모수 c를 아주 작게주면 $\sum_{i=1}^{n}\zeta_i$가 커도 목적함수 $\sum_{i=1}^{n}\zeta_i$가 커도

되므로 밴드의 넓이가 커지게 되어 편의가 발생할 수 있음.

- 만약 데이터가 도넛과 같은 형태로 안과 밖에 두 집단 데이터가 구성이 되어 있다고 한다면 선형 SVM을 통해서는 성공적으로 구별하지 못할 것임.
- 따라서 다음과 같은 커널함수를 이용한 비선형 SVM을 고려할 수 있음.

p차 다항식(polynomial): $K(\boldsymbol{x}_i,\boldsymbol{x}_j)=(1+\gamma\boldsymbol{x}_i'\boldsymbol{x}_j)^p$

방사형기저함수(radial basis function, rbf):

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(-\gamma(\boldsymbol{x}_i-\boldsymbol{x}_j)'(\boldsymbol{x}_i-\boldsymbol{x}_j))$$

시그모이드 함수(sigmoid): $K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\tanh(\gamma\boldsymbol{x}_i'\boldsymbol{x}_j+k)^p$

- 비선형 SVM에서 커널함수가 rbf일 때 γ가 클수록 분산이 작은 정규분포 형태의 비선형모형이 되고, 반대로 γ가 작아질수록 분산이 큰 정규분포 형태의 비선형모형이 됨.
- 분산이 매우 큰 정규분포는 거의 선형이므로 실제적으로 rbf는 선형모형과 같아지게 됨.
- 또한 커널함수가 p차 다항식이나 시그모이드 함수일 때, γ가 클수록 비선형성이 강한 SVM이 됨.
- 결론적으로 비선형 SVM에서 γ가 클수록 분류선의 비선형이 강해지므로 Train 데이터에 매우 유연하게 적합되어 과대적합이 발생하게 됨.

2. sklearn을 활용한 SVM

- SVM을 실시하기 위해 먼저 sklearn 모듈을 활용한 SVM 방법에 대해서 살펴보도록 하겠음.

- SVM은 앞서 설명한데로 분류를 위한 분석으로, SVM을 통해 출력변수의 값을 분류하여 예측함.
- 따라서 지난 시간 사용하였던 분석방법과 마찬가지로 주어진 데이터를 100% 활용하여 SVM을 실시한 후 새롭게 데이터를 모형에 적용하였을 때 예측 성능이 떨어지는 경우가 많음.
- 이러한 현상을 해소하기 위하여 SVM 역시 모형을 만드는 Train 데이터와 검정을 하는 Test 데이터를 일정 비율로 나누어 Test 데이터로 모

형을 평가하는 데 이를 교차 검증을 이용함.

- 주어진 데이터를 Train 데이터와 Test 데이터로 분할은 지난 로지스틱 회귀분석, KNN과 마찬가지로 sklearn.model_selection 모듈의 train_test_split() 함수를 이용하여 실시할 수 있음.

```
from sklearn.model_selection import train_test_split
train 객체명, test 객체명 = train_test_split(데이터셋, test_size=비율,
random_state=번호, stratify = 출력변수)
```

- SVM은 로지스틱회귀분석, KNN과 마찬가지로 분류 방법이기 때문에 stratify 옵션을 이용하여 예측의 대상이 되는 출력변수에 대하여 층화추출을 통해 Train 데이터와 Test 데이터를 분류함.
- 그 외 train_test_split() 함수의 사용방법은 지난 로지스틱회귀분석을 참고 바람.

- 그럼 분석에 사용되는 Train 데이터를 활용하여 SVM을 실시하는 방법을 알아보도록 하겠음.
- 분석에 사용되는 Train 데이터를 활용하여 SVM 실시는 sklearn.svm 모듈의 SVC() 함수를 이용하여 실시할 수 있음.

```
from sklearn.svm import SVC
SVC(kernel='linear'/'rbf'/'poly'/'sigmoid', C=슬랙변수 초모수값, gamma = 비선형
SVM 비선형성 값, probability=True/False).fit(입력변수, 출력변수)
```

- SVC().fit() 함수에 분석에 사용되는 Train 데이터의 입력변수와 출력변수를 입력함.
- 그리고 kernel 옵션을 통해 선형 SVM 또는 비선형 SVM을 결정함. 여기서 'linear'가 기본값으로 이는 선형 SVM을 실시하기 위한 커널함수임. 그리고 'rbf', 'poly', 'sigmoid'는 비선형 SVM을 실시하기 위해 각각 방사형 기저함수, p차 다항식, 시그모이드함수를 커널함수로 사용함.
- C 옵션은 슬랙변수가 주어졌을 때의 초모수값을 지정함. 실수의 값을 입력할 수 있고, 기본값은 1로 지정되어 있음.
- gamma 옵션은 비선형 SVM을 실시할 때 비선형성 값을 지정함. 이 옵

션은 커널함수를 rbf, poly, sigmoid를 지정하였을 때 사용할 수 있음. gamma 옵션에는 실수값을 지정할 수 있음.

- probability 옵션은 확률추정을 활성화 할지 여부를 지정함. 기본적으로 False로 지정되어 있어 확률값을 계산하지 않음. 만약 predict_proba() 함수를 이용하여 출력변수의 예측확률값을 확인하려면 True를 지정하여 확률값을 계산하도록 해야 함.

- 위 명령어를 사용하면 다른 sklearn 모듈을 사용한 분석들과 마찬가지로 아무런 결과가 출력되지 않고, 분석이 실시되었다는 의미로 SVC()이라는 메시지만 출력됨.
- 이는 R과 같은 다른 프로그램들과 마찬가지로 분석이 수행된 결과를 다른 객체에 할당한 후 이를 다양한 함수에 활용하여 결과를 확인할 수 있음.

- 따라서 SCV().fit() 함수를 실행시킨 결과를 객체에 할당하여 다음과 같은 함수를 활용하여 SVM의 예측값을 확인할 수 있음.

예측값: SVM 객체**.predict**(입력변수) 확률값: svm 객체**.predict_proba**(입력변수)

- SVM에 의해 예측된 출력변수의 값을 확인하기 위해서는 SVM의 결과를 할당한 객체를 활용하여 predict() 함수를 사용하여 확인할 수 있음. 이 때 predict() 함수에 출력변수의 예측값을 계산하기 위한 입력변수를 입력하면 출력변수의 예측값을 확인할 수 있음. 만약 Test 데이터의 입력변수를 입력하면 생성된 SVM에 Test 데이터를 적용하였을 때의 출력변수의 예측값을 확인할 수 있음.

- 그리고 SVM 모형 평가에 사용되는 분류표, 정분류율, 평가지표를 확인하기 위해 sklearn.metrics 모듈의 confusion_matrix() 함수, accuracy_score() 함수, classification_report() 함수를 이용하여 실시할 수 있음.

from **sklearn.metrics** import **confusion_matrix, accuracy_score, classification_report**
분류표: **confusion_matrix**(출력변수 예측값, 출력변수)
정분류율: **accuracy_score**(출력변수, 출력변수 예측값)
분류 예측력 평가 지표: **classification_report**(출력변수, 출력변수 예측값)

- SVM을 평가하기 위해 실제 출력변수의 값과 예측값을 이원분할표의 형태로 나타내기 위해 confusion_matrix() 함수를 사용할 수 있음. 이 때, 이원분할표의 행 위치와 열 위치를 차례로 입력하여 분류표의 행과 열에 위치할 값을 지정할 수 있음.
- confusion_matrix() 함수에 의해 나타난 분류표 상의 정분류율을 계산하기 위해 accuracy_score() 함수를 사용할 수 있음.이 때, 출력변수와 출력변수 예측값을 차례로 입력함.
- SVM을 평가하기 위한 지표인 presicion, recall, f1_score를 계산하기 위해 classification_report() 함수를 사용할 수 있음. 이 때, 출력변수와 출력변수 예측값을 차례로 입력함.

- seaborn 모듈의 iris 데이터에서, species를 예측하기 위해 먼저 sepal_length, sepal_width, petal_length, petal_width를 사용한 선형 SVM을 실시하고자 함.
- info() 함수를 활용하여 자료를 확인한 결과 분석에 사용되는 모든 변수에서 결측자료가 없는 것을 확인할 수 있음.

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

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

In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

- 다음으로 모형을 생성하기 위한 Train 데이터와 검증을 위한 Test 데이터를 구분하기 위해 train_test_split() 함수를 사용하였음.

```
In [4]: from sklearn.model_selection import train_test_split

In [5]: train, test = train_test_split(df, test_size=0.3, random_state=1, stratify=df['species'])

In [6]: y_train=train['species']
   ...: X_train=train[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
   ...: y_test=test['species']
   ...: X_test=test[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
```

- 이 때 Train 데이터는 train 객체에 할당하고, Test 데이터는 test 객체에 할당하였음. 그런 후 X_train과 X_test 객체에 train 객체와 test 객체의 입력변수를 선택하여 각각 할당하였고, y_train과 y_test 객체에 train 객체와 test 객체의 출력변수를 선택하여 각각 할당하였음.

- Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 SVC().fit() 함수에 대입하여 선형 SVM을 실시하였음.

```
In [7]: from sklearn.svm import SVC

In [8]: svm=SVC(kernel='linear', C=1.0, probability=True, random_state=1).fit(X_train, y_train)

In [9]: svm.predict_proba(X_train)[:5]
Out[9]:
array([[0.97679222, 0.01152322, 0.01168456],
       [0.93280957, 0.04645169, 0.02073874],
       [0.01531616, 0.03733592, 0.94734792],
       [0.00964874, 0.02514918, 0.96520208],
       [0.96911164, 0.02044402, 0.01044434]])
```

- 이 때 선형 SVM을 실시하므로 kernel 옵션에 'linear'를 지정하였고, 슬랙변수의 초모수의 값 C는 기본값인 1을 지정하였음. 그리고 출력변수 값의 확률을 확인하기 위해 probability 옵션에 True를 지정하였음.
- SVC().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 svm이라는 객체에 할당하였고, predict_proba() 함수에 입력변수를 입력하여 그 결과를 확인 하였음.
- 그리고 svm 객체를 활용하여 출력변수의 예측값을 계산하였음.

```
In [10]: y_train_pred = svm.predict(X_train)
```

- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 선형 SVM을 평가하는 지표인 정분류율, Precision, Recall, F1-score 등을 계산하였음.

```
In [10]: y_train_pred = svm.predict(X_train)

In [11]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [12]: confusion_matrix(y_train_pred, y_train)
Out[12]:
array([[35,  0,  0],
       [ 0, 33,  1],
       [ 0,  2, 34]], dtype=int64)

In [13]: accuracy_score(y_train_pred, y_train)
Out[13]: 0.9714285714285714

In [14]: svm_report = classification_report(y_train, y_train_pred)

In [15]: print(svm_report)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        35
  versicolor       0.97      0.94      0.96        35
   virginica       0.94      0.97      0.96        35

    accuracy                           0.97       105
   macro avg       0.97      0.97      0.97       105
weighted avg       0.97      0.97      0.97       105
```

- 분류표, 정분류율, precision, recall, f1 score 등을 계산하기 위하여 sklearn.metrics 모듈의 confusion_matrix, accuracy_score, classification_report() 함수를 이용하였음. 그 결과 모형 생성에 사용된 Train 데이터에서는 species가 setosa, versicolor, virginica를 각각 35개, 33개, 34개 데이터에 대해서 제대로 분류하여 정분류율는 0.97142857로 나타났음. 그리고 species가 setosa인 경우 precision, recall, f1-score는 각각 1.00, 1.00, 1.00으로 나타났고, species가 versicolor인 경우 precision, recall, f1-score는 각각 0.97, 0.94, 0.96으로 나타났고, species가 virginica인 경우 precision, recall, f1-score는 각각 0.94, 0.97, 0.96으로 나타났음.

- 다음으로 predict() 함수에 Test 데이터의 독립변수 데이터인 X_test를 대입하여 Test 데이터의 종속변수 예측값을 y_test_pred 객체에 할당하여 선형 SVM을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [16]: y_test_pred = svm.predict(X_test)

In [17]: from sklearn.metrics import accuracy_score, confusion_matrix

In [18]: confusion_matrix(y_test_pred, y_test)
Out[18]:
array([[15,  0,  0],
       [ 0, 15,  1],
       [ 0,  0, 14]], dtype=int64)

In [19]: accuracy_score(y_test_pred, y_test)
Out[19]: 0.9777777777777777

In [20]: svm_report1 = classification_report(y_test, y_test_pred)

In [21]: print(svm_report1)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.94      1.00      0.97        15
   virginica       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

- 그 결과 모형 생성에 사용된 Test 데이터에서는 species가 setosa, versicolor, virginica를 각각 15개, 15개, 14개 데이터에 대해서 제대로 분류하여 정분류율는 0.97777로 나타났음. 그리고 species가 setosa인 경우 precision, recall, f1-score는 각각 1.00, 1.00, 1.00으로 나타났고, species가 versicolor인 경우 precision, recall, f1-score는 각각 0.94, 1.00, 0.97로 나타났고, species가 virginica인 경우 precision, recall, f1-score는 각각 1.00, 0.93, 0.97로 나타났음.
- 이는 Train 데이터를 활용하여 나타난 결과와 큰 차이가 존재하지 않는다는 것을 확인할 수 있음.

- 다음으로 Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 SVC().fit() 함수에 대입하여 비선형 SVM을 실시하였음.

```
In [22]: from sklearn.svm import SVC

In [23]: svm=SVC(kernel='rbf', C=1.0, gamma=0.2, probability=True, random_state=1).fit(X_train, y_train)

In [24]: svm.predict_proba(X_train)[:5]
Out[24]:
array([[0.94025384, 0.03167803, 0.02806813],
       [0.94990912, 0.03161942, 0.01847146],
       [0.01358309, 0.0313825 , 0.95503441],
       [0.0115695 , 0.01544497, 0.97298553],
       [0.96377102, 0.01889853, 0.01733045]])
```

- 이 때 비선형 SVM을 실시하므로 kernel 옵션에 기본 커널 옵션인 'rbf'를 지정하였고, 슬랙변수의 초모수의 값 C는 기본값인 1을 지정하였음. 또한, 커널함수의 감마값을 지정하기 위해 gamma 옵션에 0.2를 지정

하였음. 그리고 출력변수 값의 확률을 확인하기 위해 probability 옵션에 True를 지정하였음.

- SVC().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 svm이라는 객체에 할당하였고, predict_proba() 함수에 입력변수를 입력하여 그 결과를 확인 하였음.
- 그리고 svm 객체를 활용하여 출력변수의 예측값을 계산하였음.

```
In [25]: y_train_pred = svm.predict(X_train)
```

- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 비선형 SVM을 평가하는 지표인 정분류율, Precision, Recall, F1-score 등을 계산하였음.

```
In [25]: y_train_pred = svm.predict(X_train)

In [26]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [27]: confusion_matrix(y_train_pred, y_train)
Out[27]:
array([[35,  0,  0],
       [ 0, 33,  2],
       [ 0,  2, 33]], dtype=int64)

In [28]: accuracy_score(y_train_pred, y_train)
Out[28]: 0.9619047619047619

In [29]: svm_report = classification_report(y_train, y_train_pred)

In [30]: print(svm_report)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        35
  versicolor       0.94      0.94      0.94        35
   virginica       0.94      0.94      0.94        35

    accuracy                           0.96       105
   macro avg       0.96      0.96      0.96       105
weighted avg       0.96      0.96      0.96       105
```

- 분류표, 정분류율, precision, recall, f1 score 등을 계산하기 위하여 sklearn.metrics 모듈의 confusion_matrix, accuracy_score, classification_report() 함수를 이용하였음. 그 결과 모형 생성에 사용된 Train 데이터에서는 species가 setosa, versicolor, virginica를 각각 35개, 33개, 33개 데이터에 대해서 제대로 분류하여 정분류율는 0.9619047로 나타났음. 그리고 species가 setosa인 경우 precision, recall, f1-score는 각각 1.00, 1.00, 1.00으로 나타났고, species가 versicolor인 경우 precision, recall, f1-score는 각각 0.94, 0.94, 0.94로 나타났고, species가 virginica인 경우 precision, recall, f1-score는 각각 0.94, 0.94, 0.94로 나타났음.

- 다음으로 predict() 함수에 Test 데이터의 독립변수 데이터인 X_test를 대입하여 Test 데이터의 종속변수 예측값을 y_test_pred 객체에 할당하여 비선형 SVM을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [31]: y_test_pred = svm.predict(X_test)

In [32]: from sklearn.metrics import accuracy_score, confusion_matrix

In [33]: confusion_matrix(y_test_pred, y_test)
Out[33]:
array([[15,  0,  0],
       [ 0, 15,  1],
       [ 0,  0, 14]], dtype=int64)

In [34]: accuracy_score(y_test_pred, y_test)
Out[34]: 0.9777777777777777

In [35]: svm_report1 = classification_report(y_test, y_test_pred)

In [36]: print(svm_report1)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.94      1.00      0.97        15
   virginica       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

- 그 결과 모형 생성에 사용된 Test 데이터에서는 species가 setosa, versicolor, virginica를 각각 15개, 15개, 14개 데이터에 대해서 제대로 분류하여 정분류율는 0.97777로 나타났음. 그리고 species가 setosa인 경우 precision, recall, f1-score는 각각 1.00, 1.00, 1.00으로 나타났고, species가 versicolor인 경우 precision, recall, f1-score는 각각 0.94, 1.00, 0.97로 나타났고, species가 virginica인 경우 precision, recall, f1-score는 각각 1.00, 0.93, 0.97로 나타났음.
- 이는 Train 데이터를 활용하여 나타난 결과와 큰 차이가 존재하지 않는다는 것을 확인할 수 있음.

3. SVM의 실습

※ seaborn 모듈의 penguins 데이터에서, species를 예측하기 위해 bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g을 사용한 SVM을 실시하세요.

- info() 함수를 활용하여 자료를 확인한 결과 bill_length_mm,

bill_depth_mm, flipper_length_mm, body_mass_g에서 결측자료가 존재하는 것을 확인할 수 있음. 따라서 dropna() 함수를 이용하여 결측값을 제거할 필요가 있음.

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

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

In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB

In [4]: df = df.dropna(subset=['species', 'bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g'],
   ...:                how='any', axis=0)
```

- 다음으로 모형을 생성하기 위한 Train 데이터와 검증을 위한 Test 데이터를 구분하기 위해 train_test_split() 함수를 사용하였음.

```
In [5]: from sklearn.model_selection import train_test_split

In [6]: train, test = train_test_split(df, test_size=0.3, random_state=1, stratify=df['species'])

In [7]: y_train=train['species']
   ...: X_train=train[['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']]
   ...: y_test=test['species']
   ...: X_test=test[['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']]
```

- 이 때 Train 데이터는 train 객체에 할당하고, Test 데이터는 test 객체에 할당하였음. 그런 후 X_train과 X_test 객체에 train 객체와 test 객체의 입력변수를 선택하여 각각 할당하였고, y_train과 y_test 객체에 train 객체와 test 객체의 출력변수를 선택하여 각각 할당하였음.

- Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 SVC().fit() 함수에 대입하여 선형 SVM을 실시하였음.

```
In [8]: from sklearn.svm import SVC

In [9]: svm=SVC(kernel='linear', C=1.0, random_state=1, probability=True).fit(X_train, y_train)

In [10]: svm.predict_proba(X_train)[:5]
Out[10]:
array([[8.59491134e-01, 1.27284211e-01, 1.32246545e-02],
       [4.28171598e-02, 9.42189171e-01, 1.49936697e-02],
       [8.19316882e-01, 5.17330837e-02, 1.28950035e-01],
       [9.49835038e-01, 4.08496570e-03, 4.60799966e-02],
       [9.94659455e-01, 1.55697587e-04, 5.18484738e-03]])
```

- 이 때 선형 SVM을 실시하므로 kernel 옵션에 'linear'를 지정하였고, 슬랙변수의 초모수의 값 C는 기본값인 1을 지정하였음. 그리고 출력변수 값의 확률을 확인하기 위해 probability 옵션에 True를 지정하였음.
- SVC().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 svm이라는 객체에 할당하였고, predict_proba() 함수에 입력변수를 입력하여 그 결과를 확인 하였음.
- 그리고 svm 객체를 활용하여 출력변수의 예측값을 계산하였음.

```
In [10]: y_train_pred = svm.predict(X_train)
```

- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 선형 SVM을 평가하는 지표인 정분류율, Precision, Recall, F1-score 등을 계산하였음.

```
In [11]: y_train_pred = svm.predict(X_train)

In [12]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [13]: confusion_matrix(y_train_pred, y_train)
Out[13]:
array([[106,   2,   0],
       [  0,  45,   0],
       [  0,   0,  86]], dtype=int64)

In [14]: accuracy_score(y_train_pred, y_train)
Out[14]: 0.9916317991631799

In [15]: svm_report = classification_report(y_train, y_train_pred)

In [16]: print(svm_report)
              precision    recall  f1-score   support

      Adelie       0.98      1.00      0.99       106
   Chinstrap       1.00      0.96      0.98        47
      Gentoo       1.00      1.00      1.00        86

    accuracy                           0.99       239
   macro avg       0.99      0.99      0.99       239
weighted avg       0.99      0.99      0.99       239
```

- 분류표, 정분류율, precision, recall, f1 score 등을 계산하기 위하여 sklearn.metrics 모듈의 confusion_matrix, accuracy_score, classification_report() 함수를 이용하였음. 그 결과 모형 생성에 사용된 Train 데이터에서는 species가 Adelie, Chinstrap, Gentoo를 각각 106개, 45개, 86개 데이터에 대해서 제대로 분류하여 정분류율는 0.991631799로 나타났음. 그리고 species가 Adelie인 경우 precision, recall, f1-score는 각각 0.98, 1.00, 0.99로 나타났고, species가

Chinstrap인 경우 precision, recall, f1-score는 각각 1.00, 0.96, 0.98로 나타났고, species가 Gentoo인 경우 precision, recall, f1-score는 각각 1.00, 1.00, 1.00으로 나타났음.

- 다음으로 predict() 함수에 Test 데이터의 독립변수 데이터인 X_test를 대입하여 Test 데이터의 종속변수 예측값을 y_test_pred 객체에 할당하여 선형 SVM을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [17]: y_test_pred = svm.predict(X_test)

In [18]: from sklearn.metrics import accuracy_score, confusion_matrix

In [19]: confusion_matrix(y_test_pred, y_test)
Out[19]:
array([[44,  1,  0],
       [ 1, 20,  0],
       [ 0,  0, 37]], dtype=int64)

In [20]: accuracy_score(y_test_pred, y_test)
Out[20]: 0.9805825242718447

In [21]: svm_report1 = classification_report(y_test, y_test_pred)

In [22]: print(svm_report1)
              precision    recall  f1-score   support

      Adelie       0.98      0.98      0.98        45
   Chinstrap       0.95      0.95      0.95        21
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.98       103
   macro avg       0.98      0.98      0.98       103
weighted avg       0.98      0.98      0.98       103
```

- 그 결과 모형 생성에 사용된 Test 데이터에서는 species가 Adelie, Chinstrap, Gentoo를 각각 44개, 20개, 37개 데이터에 대해서 제대로 분류하여 정분류율는 0.9805825로 나타났음. 그리고 species가 Adelie인 경우 precision, recall, f1-score는 각각 0.98, 0.98, 0.98로 나타났고, species가 Chinstrap인 경우 precision, recall, f1-score는 각각 0.95, 0.95, 0.95로 나타났고, species가 Gentoo인 경우 precision, recall, f1-score는 각각 1.00, 1.00, 1.00으로 나타났음.
- 이는 Train 데이터를 활용하여 나타난 결과와 큰 차이가 존재하지 않는다는 것을 확인할 수 있음.

- 다음으로 Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 SVC().fit() 함수에 대입하여 비선형 SVM을 실시하였음.

```
In [23]: from sklearn.svm import SVC

In [24]: svm=SVC(kernel='rbf', C=1.0, gamma=0.2, random_state=1, probability=True)

In [25]: svm.fit(X_train, y_train)
Out[25]: SVC(gamma=0.2, probability=True, random_state=1)

In [26]: svm.predict_proba(X_train)[:5]
Out[26]:
array([[0.92032603, 0.06498635, 0.01468763],
       [0.05629265, 0.93025971, 0.01344764],
       [0.92029166, 0.06501384, 0.01469451],
       [0.92023656, 0.06505811, 0.01470533],
       [0.92033132, 0.06498187, 0.0146868 ]])
```

- 이 때 비선형 SVM을 실시하므로 kernel 옵션에 기본 커널 옵션인 'rbf'를 지정하였고, 슬랙변수의 초모수의 값 C는 기본값인 1을 지정하였음. 또한, 커널함수의 감마값을 지정하기 위해 gamma 옵션에 0.2를 지정하였음. 그리고 출력변수 값의 확률을 확인하기 위해 probability 옵션에 True를 지정하였음.
- SVC().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 svm이라는 객체에 할당하였고, predict_proba() 함수에 입력변수를 입력하여 그 결과를 확인 하였음.
- 그리고 svm 객체를 활용하여 출력변수의 예측값을 계산하였음.

```
In [27]: y_train_pred = svm.predict(X_train)
```

- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 비선형 SVM을 평가하는 지표인 정분류율, Precision, Recall, F1-score 등을 계산하였음.

```
In [27]: y_train_pred = svm.predict(X_train)

In [28]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [29]: confusion_matrix(y_train_pred, y_train)
Out[29]:
array([[106,   1,   0],
       [  0,  46,   0],
       [  0,   0,  86]], dtype=int64)

In [30]: accuracy_score(y_train_pred, y_train)
Out[30]: 0.99581589958159

In [31]: svm_report = classification_report(y_train, y_train_pred)

In [32]: print(svm_report)
              precision    recall  f1-score   support

      Adelie       0.99      1.00      1.00       106
   Chinstrap       1.00      0.98      0.99        47
      Gentoo       1.00      1.00      1.00        86

    accuracy                           1.00       239
   macro avg       1.00      0.99      0.99       239
weighted avg       1.00      1.00      1.00       239
```

- 분류표, 정분류율, precision, recall, f1 score 등을 계산하기 위하여 sklearn.metrics 모듈의 confusion_matrix, accuracy_score, classification_report() 함수를 이용하였음. 그 결과 모형 생성에 사용된 Train 데이터에서는 species가 Adelie, Chinstrap, Gentoo를 각각 106개, 46개, 86개 데이터에 대해서 제대로 분류하여 정분류율는 0.995815899로 나타났음. 그리고 species가 Adelie인 경우 precision, recall, f1-score는 각각 0.99, 1.00, 1.00으로 나타났고, species가 Chinstrap인 경우 precision, recall, f1-score는 각각 1.00, 0.98, 0.99로 나타났고, species가 Gentoo인 경우 precision, recall, f1-score는 각각 1.00, 1.00, 1.00으로 나타났음.

- 다음으로 predict() 함수에 Test 데이터의 독립변수 데이터인 X_test를 대입하여 Test 데이터의 종속변수 예측값을 y_test_pred 객체에 할당하여 비선형 SVM을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [33]: y_test_pred = svm.predict(X_test)

In [34]: from sklearn.metrics import accuracy_score, confusion_matrix

In [35]: confusion_matrix(y_test_pred, y_test)
Out[35]:
array([[45, 20, 28],
       [ 0,  1,  0],
       [ 0,  0,  9]], dtype=int64)

In [36]: accuracy_score(y_test_pred, y_test)
Out[36]: 0.5339805825242718

In [37]: svm_report1 = classification_report(y_test, y_test_pred)

In [38]: print(svm_report1)
              precision    recall  f1-score   support

      Adelie       0.48      1.00      0.65        45
   Chinstrap       1.00      0.05      0.09        21
      Gentoo       1.00      0.24      0.39        37

    accuracy                           0.53       103
   macro avg       0.83      0.43      0.38       103
weighted avg       0.77      0.53      0.44       103
```

- 그 결과 모형 생성에 사용된 Test 데이터에서는 species가 Adelie, Chinstrap, Gentoo를 각각 45개, 1개, 9개 데이터에 대해서 제대로 분류하여 정분류율는 0.53398058로 나타났음. 그리고 species가 Adelie인 경우 precision, recall, f1-score는 각각 0.48, 1.00, 0.65로 나타났고, species가 Chinstrap인 경우 precision, recall, f1-score는 각각 1.00, 0.05, 0.09로 나타났고, species가 Gentoo인 경우 precision, recall, f1-score는 각각 1.00, 0.24, 0.39로 나타났음.
- 이는 species가 Chinstrap와 Gentoo인 경우 Train 데이터를 활용하여 나타난 결과와 차이가 존재하는 것을 확인할 수 있음.
- 이러한 결과는 Train 데이터를 활용한 비선형 SVM이 overfitting되었다는 것을 알 수 있음.
- 이러한 경우 옵션값을 조정할 필요가 있음. 최적의 결과를 얻기 위한 옵션값 조정과 관련된 내용은 교차검정 부분의 수업 때 알아보도록 하겠음.