5장 분류

이항 분류:
레드 와인과
화이트 와인 구분

화인 데이터 셋

- **캘리포니아 어바인 대학 제공**
- **특징 12 개, p108**
  - ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
  - 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
  - 'pH', 'sulphates', 'alcohol', 'quality']

```
# 5.1 와인 데이터셋 불러오기
import pandas as pd
red = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-
databases/wine-quality/winequality-red.csv', sep=';')
white = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-
databases/wine-quality/winequality-white.csv', sep=';')
print(red.head())
print(white.head())
```

와인 데이터셋 합치기

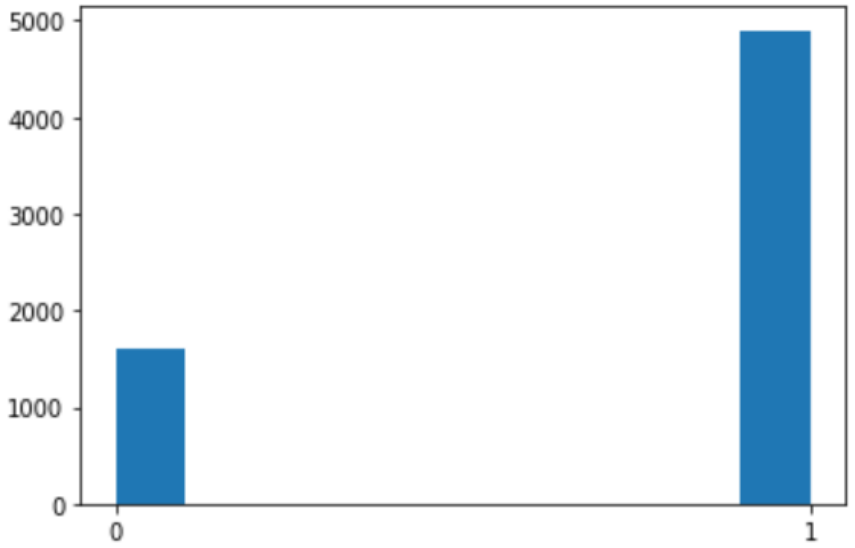
- **두 데이터를 합하여**
  - 메소드 pd.concat()
- **와인 구분, 열 type 추가**
  - 레드 와인: 0
  - 화이트 와인: 1

```
# 5.2 와인 데이터셋 합치기
red['type'] = 0
white['type'] = 1
print(red.head(2))
print(white.head(2))

wine = pd.concat([red, white])
print(wine.describe())
```

```
   fixed acidity  volatile acidity  citric acid  ...  alcohol  quality  type
0            7.4              0.70         0.0  ...      9.4        5     0
1            7.8              0.88         0.0  ...      9.8        5     0

[2 rows x 13 columns]
   fixed acidity  volatile acidity  citric acid  ...  alcohol  quality  type
0            7.0              0.27         0.36  ...      8.8        6     1
1            6.3              0.30         0.34  ...      9.5        6     1

[2 rows x 13 columns]
       fixed acidity  volatile acidity  ...      quality         type
count    6497.000000       6497.000000  ...  6497.000000  6497.000000
mean        7.215307          0.339666  ...     5.818378     0.753886
std         1.296434          0.164636  ...     0.873255     0.430779
min         3.800000          0.080000  ...     3.000000     0.000000
25%         6.400000          0.230000  ...     5.000000     1.000000
50%         7.000000          0.290000  ...     6.000000     1.000000
75%         7.700000          0.400000  ...     6.000000     1.000000
max        15.900000          1.580000  ...     9.000000     1.000000

[8 rows x 13 columns]
```

레드 와인 화이트 와인 수

```
# 5.3 레드 와인과 화이트 와인 type 히스토그램
import matplotlib.pyplot as plt
plt.hist(wine['type'])
plt.xticks([0, 1])
plt.show()

print(wine['type'].value_counts())
```




```
1    4898
0    1599
Name: type, dtype: int64
```

정규화

- **정규화 이후**
  - 최소 0, 최대 1

```
# 5.5 데이터 정규화
wine_norm = (wine - wine.min()) / (wine.max() - wine.min())
print(wine_norm.head())
print(wine_norm.describe())
```

```
   fixed acidity  volatile acidity  citric acid  ...   alcohol   quality  type
0       0.297521          0.413333     0.000000  ...  0.202899  0.333333   0.0
1       0.330579          0.533333     0.000000  ...  0.260870  0.333333   0.0
2       0.330579          0.453333     0.024096  ...  0.260870  0.333333   0.0
3       0.611570          0.133333     0.337349  ...  0.260870  0.500000   0.0
4       0.297521          0.413333     0.000000  ...  0.202899  0.333333   0.0

[5 rows x 13 columns]
       fixed acidity  volatile acidity  ...      quality         type
count    6497.000000       6497.000000  ...  6497.000000  6497.000000
mean        0.282257          0.173111  ...     0.469730     0.753886
std         0.107143          0.109758  ...     0.145543     0.430779
min         0.000000          0.000000  ...     0.000000     0.000000
25%         0.214876          0.100000  ...     0.333333     1.000000
50%         0.264463          0.140000  ...     0.500000     1.000000
75%         0.322314          0.213333  ...     0.500000     1.000000
max         1.000000          1.000000  ...     1.000000     1.000000

[8 rows x 13 columns]
```

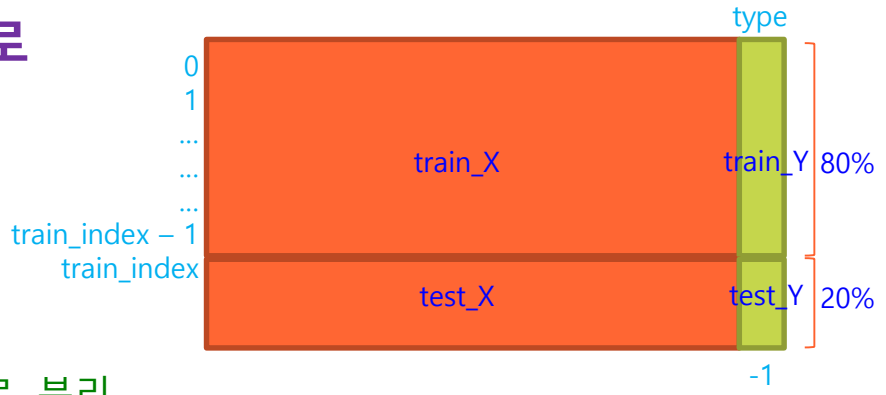
레드와인과 화이트 와인 행 섞기

```
# 5.6 데이터 섞은 후 numpy array로 변환
import numpy as np

wine_shuffle = wine_norm.sample(frac=1)
print(wine_shuffle.head())
wine_np = wine_shuffle.to_numpy()
print(wine_np[:5])
```

```
      fixed acidity  volatile acidity  citric acid  ...   alcohol   quality  type
464        0.636364          0.156667     0.325301  ...  0.173913  0.500000   0.0
4256       0.181818          0.166667     0.156627  ...  0.173913  0.333333   1.0
441        0.190083          0.200000     0.120482  ...  0.347826  0.500000   1.0
3676       0.123967          0.146667     0.180723  ...  0.811594  0.666667   1.0
165        0.330579          0.366667     0.289157  ...  0.217391  0.333333   0.0

[5 rows x 13 columns]
[[0.63636364 0.15666667 0.3253012  0.02300613 0.12458472 0.01388889
  0.02073733 0.22344322 0.20155039 0.26966292 0.17391304 0.5
  0.        ]
 [0.18181818 0.16666667 0.15662651 0.0690184  0.06976744 0.05208333
  0.26036866 0.13591671 0.33333333 0.10674157 0.17391304 0.33333333
  1.        ]
 [0.19008264 0.2        0.12048193 0.09202454 0.03986711 0.08333333
  0.30184332 0.12897629 0.4496124  0.26404494 0.34782609 0.5
  1.        ]
 [0.12396694 0.14666667 0.18072289 0.00920245 0.03322259 0.08333333
  0.20046083 0.00597648 0.45736434 0.1011236  0.8115942  0.66666667
  1.        ]
 [0.33057851 0.36666667 0.28915663 0.01687117 0.15116279 0.04513889
  0.20737327 0.17331791 0.36434109 0.2247191  0.2173913  0.33333333
  0.        ]]
```

학습 데이터와 테스트 데이터 분리

- 특징에서 마지막 값을 정답으로
- 정답을 원 핫 인코딩으로




```
# 5.7 train 데이터와 test 데이터로 분리
import tensorflow as tf
train_idx = int(len(wine_np) * 0.8)

train_X, train_Y = wine_np[:train_idx, :-1], wine_np[:train_idx, -1]
test_X, test_Y = wine_np[train_idx:, :-1], wine_np[train_idx:, -1]
print(train_X[0])
print(train_Y[0])
print(test_X[0])
print(test_Y[0])

train_Y = tf.keras.utils.to_categorical(train_Y, num_classes=2)
test_Y = tf.keras.utils.to_categorical(test_Y, num_classes=2)
print(train_Y[0])
print(test_Y[0])
```

정답 제외

정답만

원 핫 인코딩

```
y = [0, 1, 2, 3]
tf.keras.utils.to_categorical(y, num_classes=4)
```

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]], dtype=float32)
```

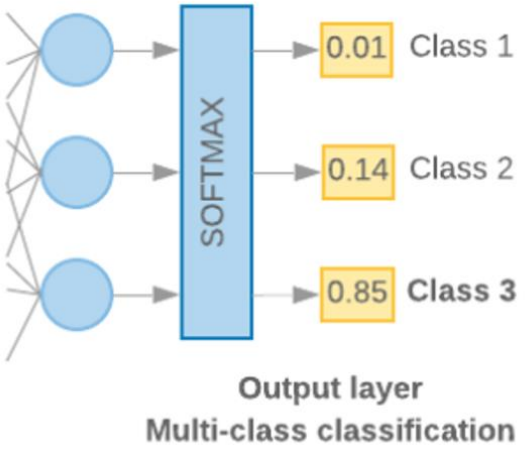
딥러닝 모델

```
# 5.8 와인 데이터셋 분류 모델 생성
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=48, activation='relu', input_shape=(12,)),
    tf.keras.layers.Dense(units=24, activation='relu'),
    tf.keras.layers.Dense(units=12, activation='relu'),
    tf.keras.layers.Dense(units=2, activation='softmax')
])

model.compile(optimizer = tf.keras.optimizers.Adam(lr=0.07),
              loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 48)                624
_________________________________________________________________
dense_1 (Dense)              (None, 24)                1176
_________________________________________________________________
dense_2 (Dense)              (None, 12)                300
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 26
=================================================================
Total params: 2,126
Trainable params: 2,126
Non-trainable params: 0
_________________________________________________________________
```

분류에서의 활성화 함수

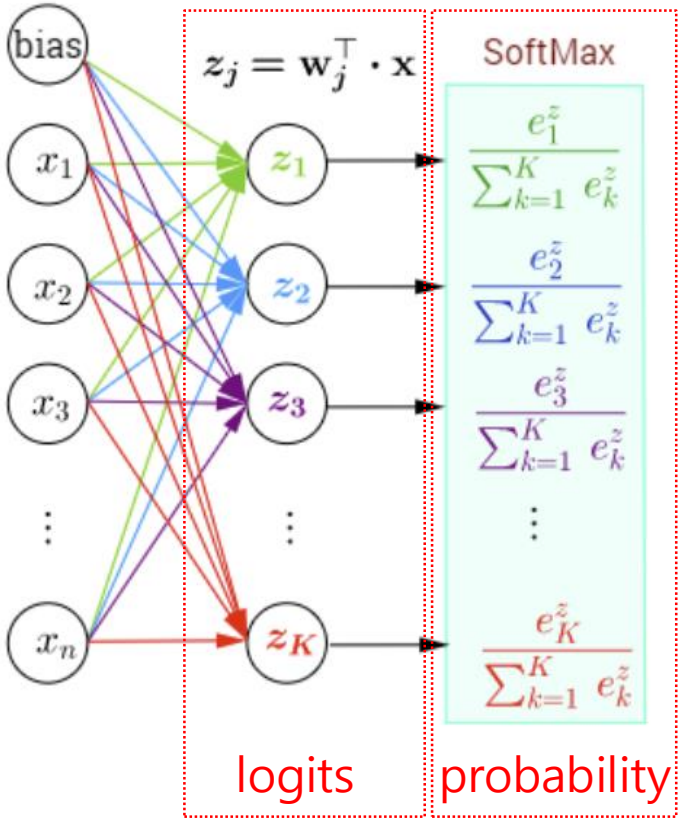
- **마지막 층은**
  - 소프트맥스 함수
    - **결과의 총합은 1**
    - **큰 값을 강조하고 작은 값은 약화시키는 효과**




```
# 5.8 와인 데이터셋 분류 모델 생성
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=48, activation='relu', input_shape=(12,)),
    tf.keras.layers.Dense(units=24, activation='relu'),
    tf.keras.layers.Dense(units=12, activation='relu'),
    tf.keras.layers.Dense(units=2, activation='softmax')
])

model.compile(optimizer = tf.keras.optimizers.Adam(lr=0.07),
              loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

소프트맥스 함수

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



The softmax as

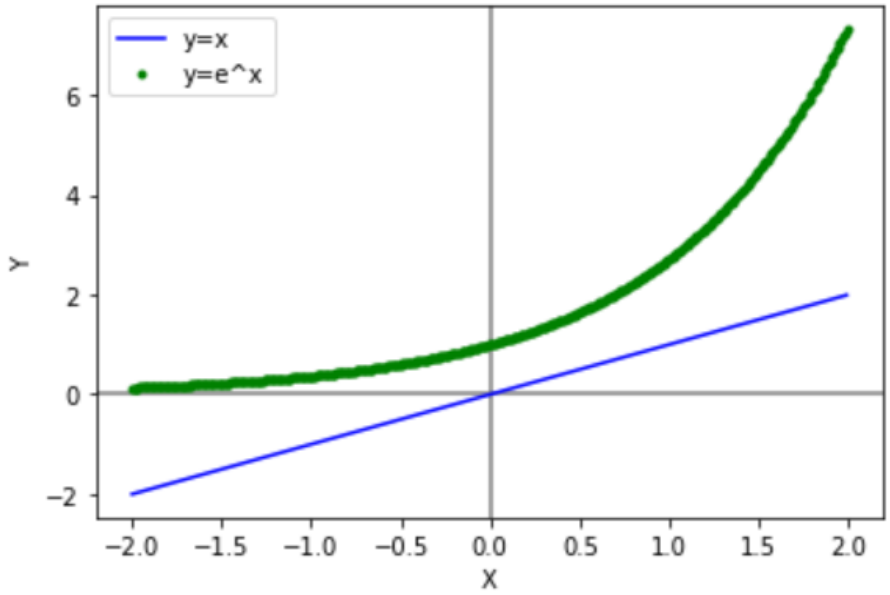
$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

This will result in a normalization of the output adding up to 1, interpretable as a probability mass functionn.

http://rinterested.github.io/statistics/softmax.html

지수승 e^x 효과

- **자연수 e를 밑으로 하는 지수 함수**
  - 음수는 양수로
  - 작은 수는 작게, 큰 수는 더욱 크게




```
# 그림 5.5 출력 코드
import matplotlib.pyplot as plt
import math
import numpy as np
x = np.arange(-2, 2, 0.01)
e_x = math.e ** x

plt.axhline(0, color='gray')
plt.axvline(0, color='gray')
plt.plot(x, x, 'b-', label='y=x')
plt.plot(x, e_x, 'g.', label='y=e^x')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```

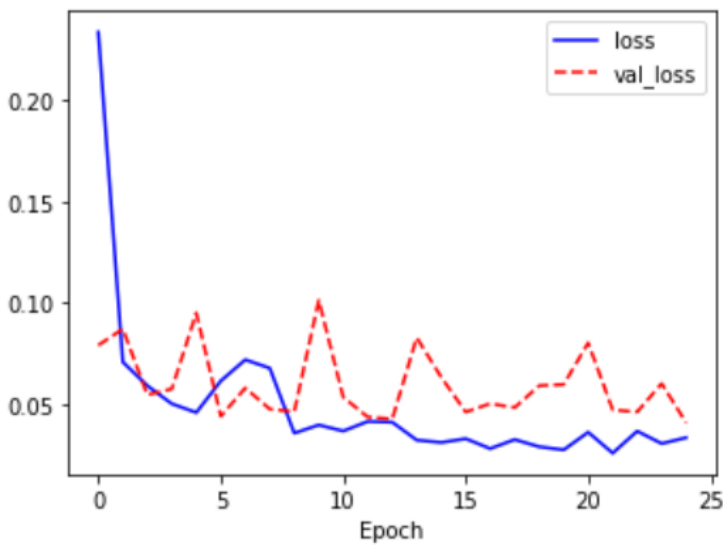
학습

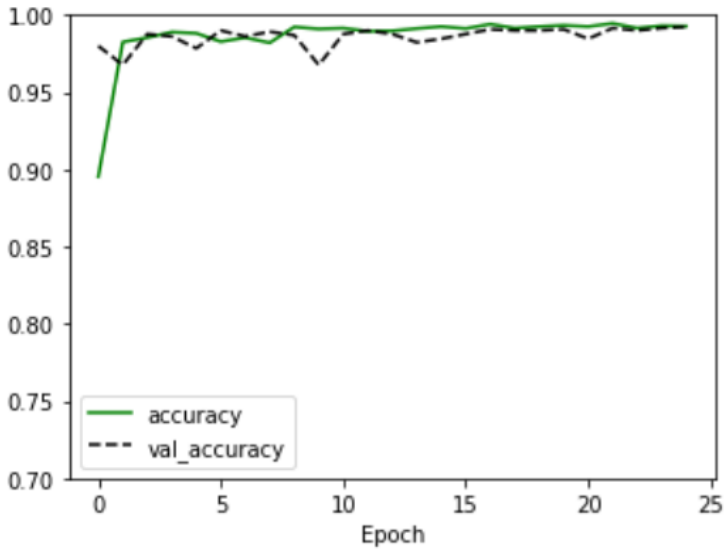
```
# 5.9 와인 데이터셋 분류 모델 학습
history = model.fit(train_X, train_Y, epochs=25, batch_size=32, validation_split=0.25)
```

```
Epoch 22/25
122/122 [==============================] - 0s 2ms/step - loss: 0.0320 - accuracy: 0.9933 - val_loss: 0.0272 - val_accuracy: 0.9946
Epoch 23/25
122/122 [==============================] - 0s 2ms/step - loss: 0.0334 - accuracy: 0.9913 - val_loss: 0.1146 - val_accuracy: 0.9638
Epoch 24/25
122/122 [==============================] - 0s 2ms/step - loss: 0.0428 - accuracy: 0.9900 - val_loss: 0.0526 - val_accuracy: 0.9931
Epoch 25/25
122/122 [==============================] - 0s 2ms/step - loss: 0.0394 - accuracy: 0.9879 - val_loss: 0.0325 - val_accuracy: 0.9946
```

학습 과정 시각화

```
# 5.10 분류 모델 학습 결과 시각화
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'k--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()

plt.show()
```





평가

```
1 # 5.11 분류 모델 평가
2 model.evaluate(test_X, test_Y)
```

```
41/41 [==============================] - 0s 1ms/step - loss: 0.0316 - accuracy: 0.9915
[0.0315738245844841, 0.9915384650230408]
```

다항 분류:
와인 품질 분류

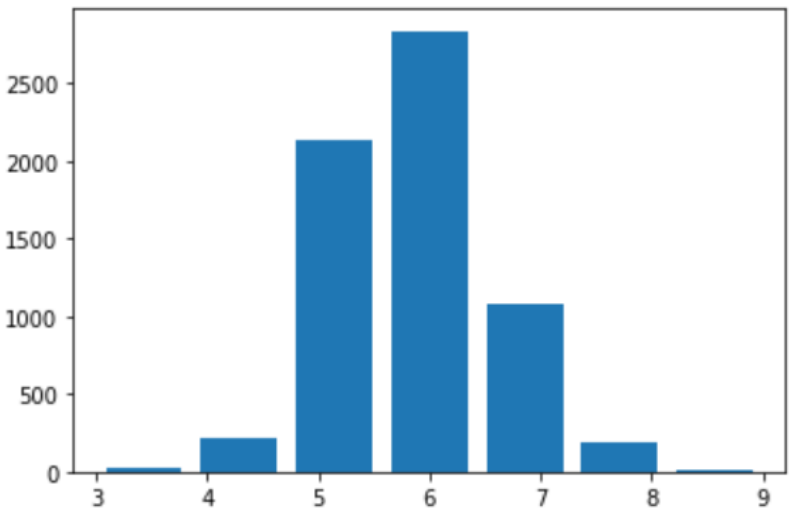
와인 데이터 셋의 'quality'

- **등급 3~9**
  - 이 모든 등급을 예측하기에는 등급에 따른 데이터 수 차이가 큼
    - **다시 등급을 3개 정도로 나누어 예측**

```
[20]  1 # 5.12 품질 데이터 확인
      2 print(wine['quality'].describe())
      3 print(wine['quality'].value_counts())
```

```
count    6497.000000
mean        5.818378
std         0.873255
min         3.000000
25%         5.000000
50%         6.000000
75%         6.000000
max         9.000000
Name: quality, dtype: float64
6    2836
5    2138
7    1079
4     216
8     193
3      30
9       5
Name: quality, dtype: int64
```

```
[21]  1 # 5.13 품질 히스토그램 시각화
      2 import matplotlib.pyplot as plt
      3 plt.hist(wine['quality'], bins=7, rwidth=0.8)
      4 plt.show()
```



새로운 등급인 new_quality를 생성

- **조건에 맞는 값을 새로운 열에 추가**
  - df.loc[data['컬럼'] 조건, '새로운 컬럼명'] = '값'

```
# 5.14 품질을 3개의 범주(좋음, 보통, 나쁨)로 재분류
wine.loc[wine['quality'] <= 5, 'new_quality'] = 0
wine.loc[wine['quality'] == 6, 'new_quality'] = 1
wine.loc[wine['quality'] >= 7, 'new_quality'] = 2

print(wine['new_quality'].describe())
print(wine['new_quality'].value_counts())
```

```
count    6497.000000
mean        0.829614
std         0.731124
min         0.000000
25%         0.000000
50%         1.000000
75%         1.000000
max         2.000000
Name: new_quality, dtype: float64
1.0    2836
0.0    2384
2.0    1277
Name: new_quality, dtype: int64
```

정규화와 원핫 인코딩

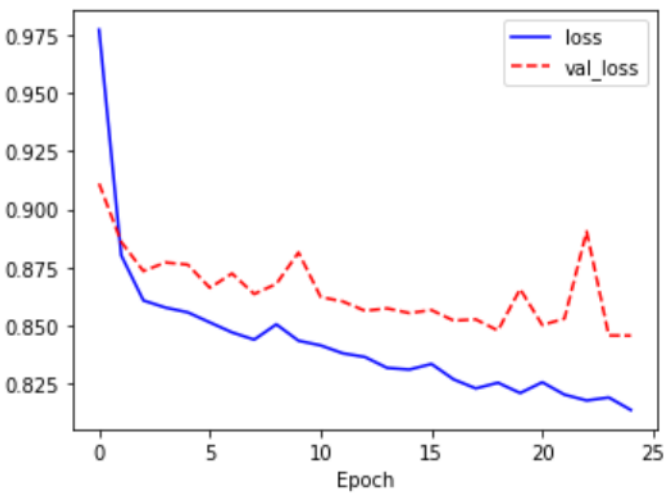
```
# 5.15 데이터 정규화 및 train, test 데이터 분리
del wine['quality']
wine_backup = wine.copy()
wine_norm = (wine - wine.min()) / (wine.max() - wine.min())
wine_norm['new_quality'] = wine_backup['new_quality']
wine_shuffle = wine_norm.sample(frac=1)
wine_np = wine_shuffle.to_numpy()

train_idx = int(len(wine_np) * 0.8)
train_X, train_Y = wine_np[:train_idx, :-1], wine_np[:train_idx, -1]
test_X, test_Y = wine_np[train_idx:, :-1], wine_np[train_idx:, -1]
train_Y = tf.keras.utils.to_categorical(train_Y, num_classes=3)
test_Y = tf.keras.utils.to_categorical(test_Y, num_classes=3)
```

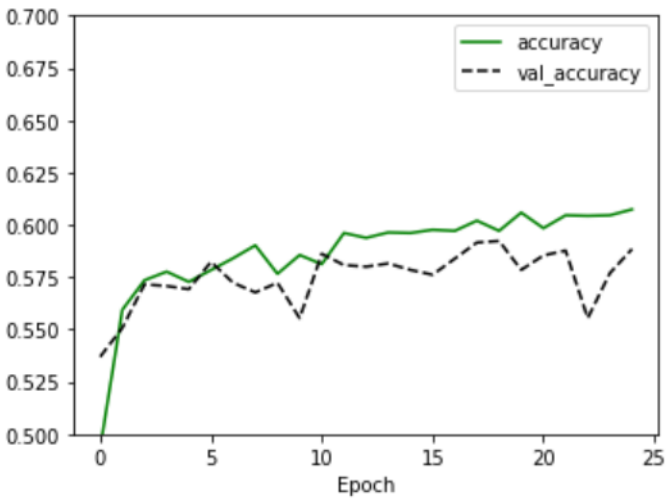
딥러닝 모델

```
# 5.16 와인 데이터셋 다항 분류 모델 생성 및 학습
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=48, activation='relu', input_shape=(12,)),
    tf.keras.layers.Dense(units=24, activation='relu'),
    tf.keras.layers.Dense(units=12, activation='relu'),
    tf.keras.layers.Dense(units=3, activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.003),
              loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_X, train_Y, epochs=25, batch_size=32,
                    validation_split=0.25)
```

학습 과정 시각화

```
# 5.17 다항 분류 모델 학습 결과 시각화
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'k--
', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.5, 0.7)
plt.legend()

plt.show()
```





평가

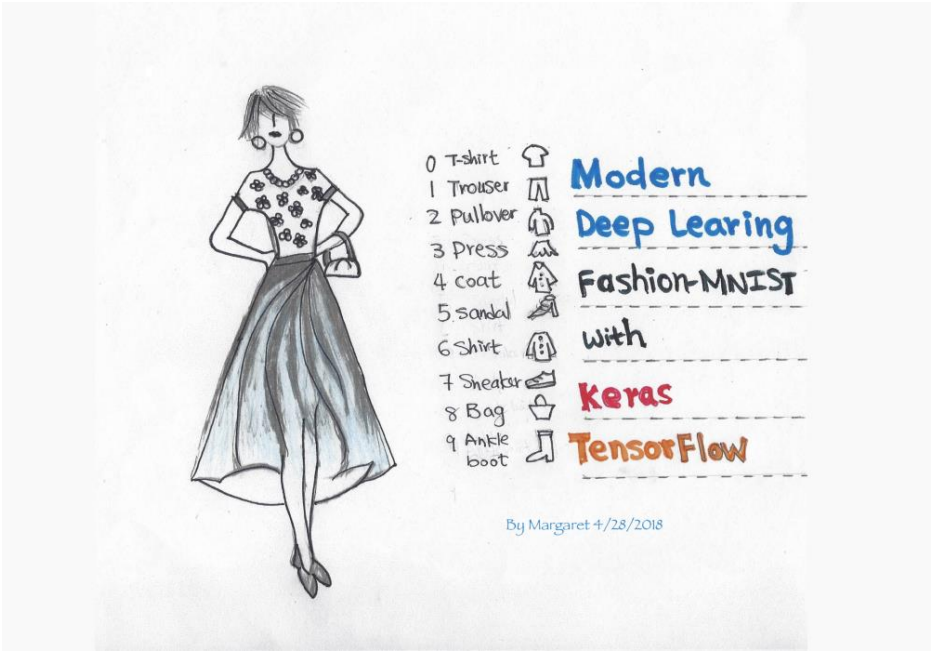
```
1 # 5.18 다항 분류 모델 평가
2 model.evaluate(test_X, test_Y)
```

```
41/41 [==============================] - 0s 1ms/step - loss: 0.8130 - accuracy: 0.6023
[0.812984824180603, 0.6023076772689819]
```

다항 분류: 패션 MNIST

Fashion-MNIST 데이터 저장

- 미리 섞여진 fashoin-mnist의 학습 데이터와 테스트 데이터 로드




```
# 필요 모듈 임포트
# tensorflow와 tf.keras를 임포트합니다
import tensorflow as tf
from tensorflow import keras

# ① 문제와 정답 데이터 지정
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()
# 10 개의 분류 이름 지정
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

# 데이터 전처리
# 샘플 값을 정수(0~255)에서 부동소수(0~1)로 변환
train_images, test_images = train_images / 255.0, test_images / 255.0
```

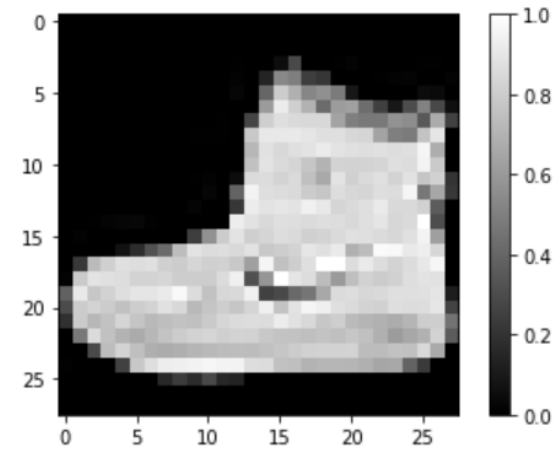
데이터셋 불러오기

```
[34]  1 # 5.19 Fashion MNIST 데이터셋 불러오기
      2 fashion_mnist = tf.keras.datasets.fashion_mnist
      3 (train_X, train_Y), (test_X, test_Y) = fashion_mnist.load_data()
      4
      5 print(len(train_X), len(test_X))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
60000 10000
```

데이터 확인

```
[41]  1 # 5.20 데이터 확인
      2 import matplotlib.pyplot as plt
      3 plt.imshow(train_X[0], cmap='gray')
      4 plt.colorbar()
      5 plt.show()
      6
      7 print(train_Y[0])
```




```
9
```

```
[43]  1 # 10 개의 분류 이름 지정
      2 class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
      3                'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
      4
      5 print(class_names[train_Y[0]])
```

```
Ankle boot
```

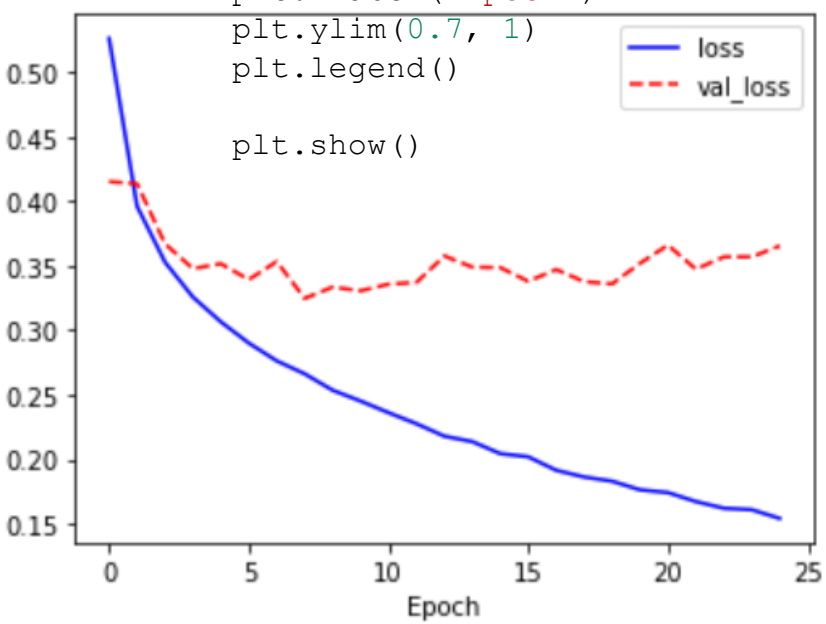
정규화, 모델 생성과 학습

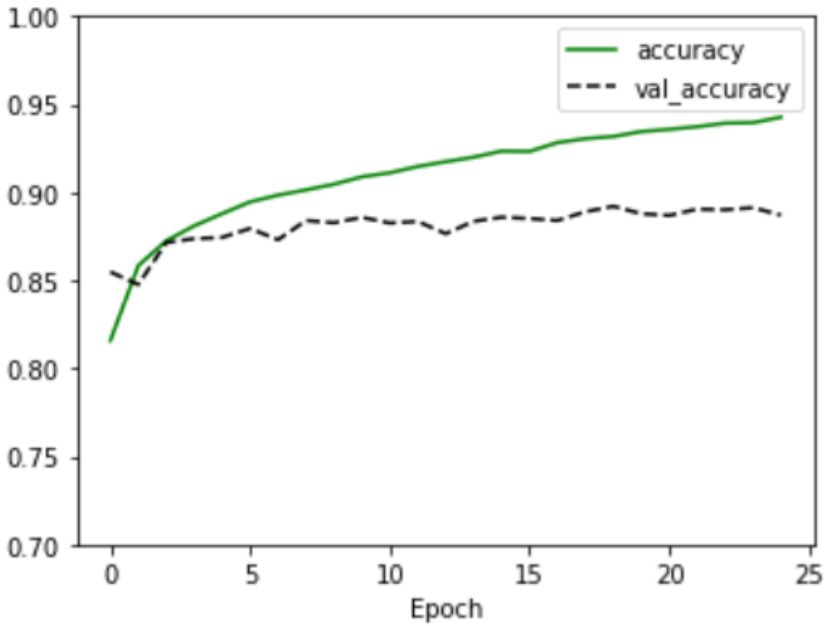
- **loss='sparse_categorical_crossentropy'**
  - 정답을 원핫인코딩 불필요

```
# 5.21 데이터 정규화
train_X = train_X / 255.0
test_X = test_X / 255.0

print(train_X[0])
# 5.22 Fashion MNIST 분류 모델
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(units=128, activation='relu'),
    tf.keras.layers.Dense(units=10, activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
# 5.23 Fashion MNIST 분류 모델 학습
history = model.fit(train_X, train_Y, epochs=25, validati
on_split=0.25)
```

시각화

```
# 5.24 Fashion MNIST 분류 모델 학습 결과 시각화
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'k--
', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.7, 1)
plt.legend()

plt.show()
```





평가

```
[40]  1 # 5.25 Fashion MNIST 분류 모델 평가
      2 model.evaluate(test_X, test_Y)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.4018 - accuracy: 0.8823
[0.40182268619537354, 0.8823000192642212]
```