
```
1 import warnings
2 # 경고메세지 끄기
3 warnings.filterwarnings(action='ignore')
4
5 # 참고) 다시 출력하게 하기
6 # warnings.filterwarnings(action='default')
```

실습 4 - Titanic

```
[ ]  1 import pandas as pd
     2 import seaborn as sns
     3 titanic = sns.load_dataset('titanic')
```

데이터확인

```
[ ]  1 titanic.head()
```

```
# survived : 1-생존, 0-사망
# pclass : 객실등급
# sex : 성별
# age : 나이
# sibsp : 함께 탑승한 형제자매, 아내, 남편 수
# parch : 함께 탐승한 부모, 자식 수
# fare : 티켓요금
# embarked : 배에 탑승한 항구 이름(C = Cherbourn, Q = Queenstown, S = Southampton)
# class : 객실등급 숫자
# who : 성별
# adult_male : 성인남성
# deck : 배에 탑승한 항구 이름(C = Cherbourn, Q = Queenstown, S = Southampton)
# embark_town : 배에 탑승한 항구 이름(C = Cherbourn, Q = Queenstown, S = Southampton)
# alive : yes-생존, no-사망
# alone : 동반 탑승 유무

titanic.info()
```

```
# missing data가 있는지 다시한번 그래프를 그려서 확인
import missingno as mino
mino.matrix(titanic)

# nan값이 있는 데이터 : age, embarked, deck 는 확인 필요
```

boxplot

```
[ ]  1 data = titanic.iloc[:,1:]
     2 data.boxplot()
```

```
[ ]  1 # fare에 특이한 아웃라이어가 한개 확인됨.
     2 # 처리해주는 것이 좋을것으로 예상됨
```

데이터 전처리

```
[ ]  1 titanic = sns.load_dataset('titanic')
     2
     3 # 중복돈 컬럼은 우선 제외하고 진행 -> 추후에는 중복된 컬럼을 사용하여 nan 값을 보완해보는 것도 방법
     4 titanic = titanic[['survived','pclass','sex','age','sibsp','parch','fare','embarked','adult_male','alone']]
     5 print('초기데이터:',len(data))
```

```
초기데이터: 709
```

```
     1 # boxplot에서 500이상인 요금은 이상치로 생각하여 제외
     2 titanic[titanic['fare']>500]
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	adult_male	alone
258	1	1	female	35.0	0	0	512.3292	C	False	True
679	1	1	male	36.0	0	1	512.3292	C	True	False
737	1	1	male	35.0	0	0	512.3292	C	True	True

```
[ ]  1 data = titanic.drop([258,679,737])
```

```
[ ]  1 # 잘 삭제 되었는지 확인
     2 data.iloc[:,1:].boxplot()
```

```
     1 # nan 값 확인
     2 # data[data.isna().any(axis=1)]
     3 data[data.age.isna()]
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	adult_male	alone
5	0	3	male	NaN	0	0	8.4583	Q	True	True
17	1	2	male	NaN	0	0	13.0000	S	True	True
19	1	3	female	NaN	0	0	7.2250	C	False	True
26	0	3	male	NaN	0	0	7.2250	C	True	True
28	1	3	female	NaN	0	0	7.8792	Q	False	True
...	...	...	...	...	...	...	...	...	...	...
859	0	3	male	NaN	0	0	7.2292	C	True	True
863	0	3	female	NaN	8	2	69.5500	S	False	False
868	0	3	male	NaN	0	0	9.5000	S	True	True
878	0	3	male	NaN	0	0	7.8958	S	True	True
888	0	3	female	NaN	1	2	23.4500	S	False	False

177 rows × 10 columns

```
[ ]  1 data[data.embarked.isna()]
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	adult_male	alone
61	1	1	female	38.0	0	0	80.0	NaN	False	True
829	1	1	female	62.0	0	0	80.0	NaN	False	True

```
[ ]  1 # 우선 nan 데이터 삭제후 진행할게요. 매우는 방법도 개별로 한번 생각해서 진행해보세요.
     2 data.dropna(inplace = True)
     3 print('nan 삭제 후 데이터:',len(data))
```

```
nan 삭제 후 데이터: 709
```

```
[ ]  1 # missing data가 있는지 다시한번 그래프를 그려서 확인
     2 import missingno as mino
     3 mino.matrix(data)
```

```
[ ]
```

```python
# 범주형변수(str -> float)
from sklearn.preprocessing import LabelEncoder

# sex, adult_male, alone
# LabelEncoder 사용
encoding = LabelEncoder()
data['sex'] = encoding.fit_transform(data['sex']) # male = 1, female = 0
data['adult_male'] = encoding.fit_transform(data['adult_male']) # True = 1, False = 0
data['alone'] = encoding.fit_transform(data['alone']) # True = 1, False = 0

# embarked
# apply 사용, replace 사용
data['embarked'] = data['embarked'].apply(lambda x : x.replace('C','0').replace('Q','1').replace('S','2')) # C = 0, Q = 1, S = 2

data = data.astype('float') # Tensor의 입력은 int형태 X, Float로 변환 필요
y_encoding = encoding.fit_transform(data['survived'])
```

모델링

```python
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

X = data.iloc[:,1:]
y = data.iloc[:,0]  # survived

X_train, X_test, y_train, y_test = train_test_split(X, y_encoding, random_state=42) # test_size : default = 0.25

model = Sequential( )

model.add(Dense(8, input_dim = 9, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

model.fit(X_train, y_train, epochs = 10, batch_size = 10)
```

```python
test_loss, test_acc = model.evaluate(X_test,y_test,verbose=2)

print('test_loss',test_loss)
print('test_acc',test_acc)
```

```
6/6 - 0s - loss: 0.7985 - accuracy: 0.6404 - 143ms/epoch - 24ms/step
test_loss 0.7984855771064758
test_acc 0.6404494643211365
```

모델 성능개선

```python
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout

titanic = sns.load_dataset('titanic')
titanic = titanic[['survived','pclass','sex','age','sibsp','parch','fare','embarked','adult_male','alone']]
data = titanic.drop([258,679,737])

data.dropna(inplace = True)

encoding = LabelEncoder()
data['sex'] = encoding.fit_transform(data['sex']) # male = 1, female = 0
data['adult_male'] = encoding.fit_transform(data['adult_male']) # True = 1, False = 0
data['alone'] = encoding.fit_transform(data['alone']) # True = 1, False = 0
data['embarked'] = data['embarked'].apply(lambda x : x.replace('C','0').replace('Q','1').replace('S','2')) # C = 0, Q = 1, S = 2

data = data.astype('float')
y_encoding = encoding.fit_transform(data['survived'])

X = data.iloc[:,1:]
y = data.iloc[:,0]

X_train, X_test, y_train, y_test = train_test_split(X, y_encoding,random_state=42)

model = Sequential( )

```

```
[ ]
model.add(Dense(512, input_dim = 9, activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

model.fit(X_train, y_train, epochs = 100, batch_size = 12)
```

```
[ ]
train_loss, train_acc = model.evaluate(X_train,y_train,verbose=2)

print('train_loss',train_loss)
print('train_acc',train_acc)
```

```
17/17 - 0s - loss: 0.2814 - accuracy: 0.8814 - 209ms/epoch - 12ms/step
train_loss 0.28138166666030884
train_acc 0.8813559412956238
```

```
test_loss, test_acc = model.evaluate(X_test,y_test,verbose=2)

print('test_loss',test_loss)
print('test_acc',test_acc)
```

```
6/6 - 0s - loss: 0.6688 - accuracy: 0.7528 - 48ms/epoch - 8ms/step
test_loss 0.6688404083251953
test_acc 0.7528089880943298
```

더블클릭 또는 Enter 키를 눌러 수정