ch02 지도학습 - knn - 실습

- Machine Learning with sklearn @ DJ,Lim
- 최종 update : 22/05

학습 내용

- titanic 데이터 셋을 활용하여 knn 모델을 구현한다.
- 가장 높은 일반화 성능을 갖는 k의 값은 무엇인지 찾아보자.

목차

- 01 데이터 준비
- 02 머신러닝 모델 만들고 예측하기

01 데이터 준비

In [ ]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
```

In [ ]:
```python
dat = pd.read_csv("train.csv")
dat
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050
...	...	...	...	...	...	...	...	...	...	.
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.000

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.450
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.750

891 rows × 12 columns

In [ ]:
```
# 컬럼(or feature)명 확인
dat.columns
```

Out[ ]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [ ]:
```
dat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

데이터 선택 및 나누기

- test_size : 테스트 데이터 셋 비율 선택
- random_state : 데이터을 뽑을 때, 지정된 패턴으로 선택

In [ ]:
```
X = dat[ ['Pclass' , 'SibSp'] ]
y = dat['Survived']

# 90% : 학습용, 10% : 테스트용
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.1,
                                                    random_state=0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[ ]:
```
((801, 2), (90, 2), (801,), (90,))
```

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=2)
model.fit(X_train, y_train)

### 예측시키기
pred = model.predict(X_test)
```

In [ ]:
```
(pred == y_test).sum() / len(pred)
```

Out[ ]:
```
0.6
```

In [ ]:
```
print("테스트 세트의 정확도 : {:.2f}".format(np.mean(pred == y_test)))
```

```
테스트 세트의 정확도 : 0.60
```

결측치 처리 및 레이블 인코딩

In [ ]:
```
dat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
 12  Sex_num      891 non-null    int64  
dtypes: float64(2), int64(6), object(5)
memory usage: 90.6+ KB
```

In [ ]:
```
dat.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

map함수

- [Series].map(함수 또는 변경값) : Series를 대상으로 원하는 함수 적용 또는 값을 대체
- 값으로 dict, Series를 대상으로 한다.
- https://pandas.pydata.org/docs/reference/api/pandas.Series.map.html

In [ ]:
```
mapping = { "male":1, 'female':2 }
dat['Sex_num'] = dat['Sex'].map(mapping)
dat.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [ ]:
```
# [].fillna( ) : 결측값을 채운다.
val_mean = dat['Age'].mean()
dat['Age'] = dat['Age'].fillna( val_mean )
dat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
```

```
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
 12  Sex_num      891 non-null    int64
dtypes: float64(2), int64(6), object(5)
memory usage: 90.6+ KB
```

학습, 테스트 데이터 셋 나누기

In [ ]:
```python
X = dat[ ['Pclass' , 'SibSp', 'Sex_num', 'Age'] ]
y = dat['Survived']

# 90% : 학습용, 10% : 테스트용
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.1,
                                                    random_state=0)


X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[ ]:
```
((801, 4), (90, 4), (801,), (90,))
```

In [ ]:
```python
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=2)
model.fit(X_train, y_train)
### 예측시키기
pred = model.predict(X_test)
print("테스트 세트의 정확도 : {:.2f}".format(np.mean(pred == y_test)))
```

```
테스트 세트의 정확도 : 0.76
```

k값에 따른 정확도 확인

In [ ]:
```python
tr_acc = []
test_acc = []
k_nums = range(1, 22, 2)# 1,3,5~21

for n in k_nums:
    # 모델 선택 및 학습
    model = KNeighborsClassifier(n_neighbors=n)
    model.fit(X_train, y_train)

    # 정확도 구하기
    acc_tr = model.score(X_train, y_train)
    acc_test = model.score(X_test, y_test)

    # 정확도 값 저장.
    tr_acc.append(acc_tr)
    test_acc.append(acc_test)

    print("k : ", n)
    print("학습용셋 정확도 {:.3f}".format(acc_tr) )
    print("테스트용셋 정확도 {:.3f}".format(acc_test) )
```

```
k :  1
학습용셋 정확도 0.885
```

```
테스트용셋 정확도 0.756
k :  3
학습용셋 정확도 0.863
테스트용셋 정확도 0.800
k :  5
학습용셋 정확도 0.850
테스트용셋 정확도 0.800
k :  7
학습용셋 정확도 0.839
테스트용셋 정확도 0.733
k :  9
학습용셋 정확도 0.830
테스트용셋 정확도 0.722
k :  11
학습용셋 정확도 0.826
테스트용셋 정확도 0.744
k :  13
학습용셋 정확도 0.824
테스트용셋 정확도 0.756
k :  15
학습용셋 정확도 0.815
테스트용셋 정확도 0.744
k :  17
학습용셋 정확도 0.805
테스트용셋 정확도 0.756
k :  19
학습용셋 정확도 0.792
테스트용셋 정확도 0.711
k :  21
학습용셋 정확도 0.790
테스트용셋 정확도 0.722
```

In [ ]:

```python
import seaborn as sns
print(sns.__version__)
```

```
0.11.0
```

In [ ]:

```python
# tr_acc = []
# test_acc = []
dat = { "tr_acc":tr_acc, "test_acc":test_acc }
data_df = pd.DataFrame(dat, index=range(1, 22, 2))
data_df
```

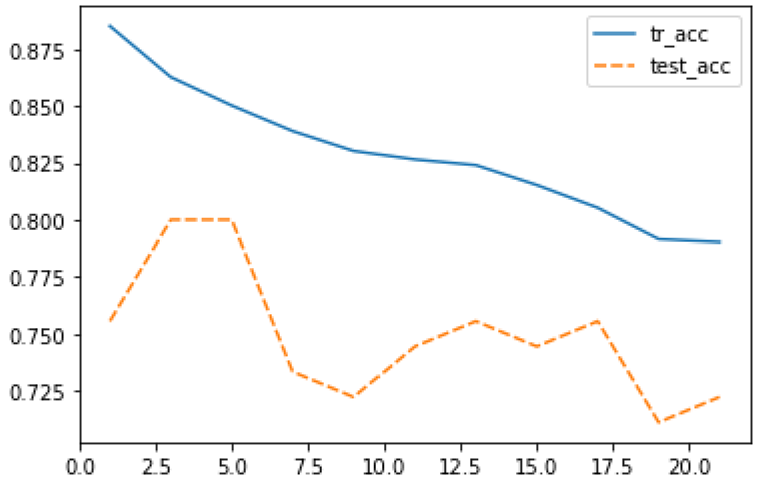
Out[ ]:

	tr_acc	test_acc
1	0.885144	0.755556
3	0.862672	0.800000
5	0.850187	0.800000
7	0.838951	0.733333
9	0.830212	0.722222
11	0.826467	0.744444
13	0.823970	0.755556
15	0.815231	0.744444
17	0.805243	0.755556
19	0.791511	0.711111
21	0.790262	0.722222

In [ ]:

```python
import matplotlib.pyplot as plt
```

In [ ]:
```
sns.lineplot(data=data_df, palette="tab10")
plt.show()
```



REF

- map 메소드 : https://pandas.pydata.org/docs/reference/api/pandas.Series.map.html