목차

2021년 3월 4일 목요일 오전 9:48

21.02.22

2021년 2월 22일 월요일 오전 10:16

유방암 데이터의 악성종양을 knn으로 분류하기

https://cafe.daum.net/oracleoracle/SgNT/2

```
import pandas as pd
import seaborn as sns

df = pd.read_csv("c:\\data\\wisc_bc_data.csv")

# DataFrame 확인
print(df.shape) # (569, 32) 569행 32열
print(df.info()) # R의 str(df)의 결과와 유사, 데이터 구조를 확인
print(df.describe()) # R에서의 summary(df)의 결과와 유사, 요약 통계정보

# 행을 선택하는 방법
print(df.iloc[0:5, ]) # 0~4번째 행 출력 df.iloc[행번호, 열번호]
print(df.iloc[-5: ,]) # -5번째 행부터 마지막 행까지 출력

# 열을 선택하는 방법
print(df.iloc[ :, [0,1] ]) # 0번째 열과 1번째 열을 출력
print(df.iloc[ :, : ]) # 전체 열 출력

'''
판다스 데이터 프레임 구성

numpy리스트(일반 리스트)로 컬럼 하나를 구성 : 시리즈
numpy리스트(일반 리스트)로 컬럼 여러개를 구성 : 데이터 프레임
```

https://cafe.daum.net/oracleoracle/SgNT/19

```
'''
#%%

# X = 전체 행, 마지막 열 제외한 모든 열 데이터 -> n차원 공간의 포인트
X = df.iloc[:, 2:].to_numpy() # 데이터 프레임의 2번째 열부터 끝까지 넘파이 array로 변환
y = df['diagnosis'].to_numpy()

print(df.shape) # (569, 32)
#print (len(X)) # 569
#print (len(y)) # 569
```

```
# 데이터 정규화
# 1. 스케일(scale) : 평균은 0이고 표준편차가 1인 데이터로 분포
# 2. min/max 정규화 : 0~1 사이의 숫자로 변경
from sklearn import preprocessing
X=preprocessing.StandardScaler().fit(X).transform(X)

# 훈련 데이터 70, 테스트 데이터 30으로 나눈다.
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)
# test_size =0.3 은 훈련과 테스트를 7대3 비율로 나누고 random_state = 10 은 seed 값을 설정하는 부분이다.

print(X_train.shape) # (398, 30)
print(y_train.shape) # (398,)

#%%

# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
# 평균은 0에 가깝고 표준편차는 1에 가까운 결과가 출력 된다.

# 학습/예측(Training/Pradiction)
from sklearn.neighbors import KNeighborsClassifier

# k-NN 분류기를 생성
classifier = KNeighborsClassifier(n_neighbors=5) # knn 모델 생성

# 분류기 학습
classifier.fit(X_train, y_train) # 훈련 데이터와 훈련 데이터의 라벨로 훈련

# 예측
y_pred= classifier.predict(X_test) # 테스트 데이터를 예측
print(y_pred)

# 모델 평가
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)

# 민감도, 재현율, 정확도, F1 스코어 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)
```

```
# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)

# k값이 5일때 정확도 0.9649
```

문제1. 위의 코드에서 적절한 k값을 알아내는 for문을 구현하시오.

```
# 위의 코드에서 적절한 k값을 알아내는 for문
import  numpy  as np

errors = []
for i in range(1, 31):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    errors.append(np.mean(pred_i != y_test))
print(errors)

for k, i in enumerate(errors):
    print(k, i)

# 위의 결과를 시각화
import matplotlib.pyplot as plt

plt.plot(range(1, 31), errors, marker='o')
plt.title('Mean error with K-Value')
plt.xlabel('k-value')
plt.ylabel('mean error')
plt.show()
```

문제2. 위에서 알아낸 에러가 낮은 k값은 7, 8, 9, 10 이였는데 k값에 7을 넣어서 정확도를 출력하시오.

```
# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
# 평균은 0에 가깝고 표준편차는 1에 가까운 결과가 출력 된다.

# 학습/예측(Training/Pradiction)
from sklearn.neighbors import KNeighborsClassifier

# k-NN 분류기를 생성
classifier = KNeighborsClassifier(n_neighbors=7) # knn 모델 생성
```

```
# 분류기 학습
classifier.fit(X_train, y_train) # 훈련 데이터와 훈련 데이터의 라벨로 훈련

# 예측
y_pred= classifier.predict(X_test) # 테스트 데이터를 예측
print(y_pred)

# 모델 평가
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)

# 민감도, 재현율, 정확도, F1 스코어 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)

# k값이 5일 때 정확도 0.9649
# k값이 7일 때 정확도 0.9707
```

위의 결과에서처럼 정확도가 100%가 나오기는 어렵기 때문에 FN(False Nagative)을 0으로 만들면 정확도가 100%가 아니더라도 사용가능하다.

위에서 한 시각화는 k값이 변경될 때마다 오류가 어떻게 되는지 2차원 그래프로 시각화 한것이고 이번에는 k값이 변경될 때마다 FN값과 정확도가 어떻게 되는지 확인해야 한다.

문제3. 위의 코드에서 적절한 k값을 알아내는 for 문을 구현하시오.

```
# 위의 코드에서 적절한 k값을 알아내는 for문2
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np


acclist = []
err_list = []
fn_list = []

for i in range(1,30):
    knn = KNeighborsClassifier(n_neighbors=i)
```

```
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    tn, fp,fn, tp = metrics.confusion_matrix(y_test, y_pred).ravel()
    fn_list.append(fn)
    acclist.append(accuracy_score(y_test, y_pred))
    err_list.append(np.mean(y_pred != y_test))

    print(f'k : {i} , acc : {accuracy_score(y_test, y_pred)} , FN : {fn}')

# 그래프 사이즈 조정
plt.figure(figsize=(12,6))
plt.subplots_adjust(left=0.125,
                    bottom=0.1,
                    right=1,
                    top=0.9,
                    wspace=0.2,
                    hspace=0.35)

# k값이 변경될 때마다 정확도에 대한 시각화
plt.subplot(131)
plt.plot(acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
plt.xlabel("k value")
plt.ylabel('Accuracy')

# k값이 변경될 때마다 에러에 대한 시각화
plt.subplot(132)
plt.plot(err_list, color='red', marker='o', markerfacecolor='blue')
plt.title('Error', size=15)
plt.xlabel("k value")
plt.ylabel('error')

# k값이 변경될 때마다 FN값에 대한 시각화
plt.subplot(133)
plt.plot(fn_list, color='green', marker='o', markerfacecolor='yellow')
plt.title('FN Value', size=15)
plt.xlabel("k value")
plt.ylabel('fn value')

plt.show()
```

문제4. iris 데이터를 knn으로 분류하시오.

```
# iris 데이터를 knn으로 분류
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
import pandas  as pd


# 1. 데이터 준비
```

```
col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']

# csv 파일에서 DataFrame을 생성
dataset = pd.read_csv('c:\\data\\iris2.csv', encoding='UTF-8', header=None, names=col_names)
#print(dataset)

X = dataset.iloc[: , :4].to_numpy()
y = dataset['Class'].to_numpy()

# 데이터 정규화
# 1. 스케일(scale) : 평균은 0이고 표준편차가 1인 데이터로 분포
# 2. min/max 정규화 : 0~1 사이의 숫자로 변경
from sklearn import preprocessing
X=preprocessing.StandardScaler().fit(X).transform(X)

# 훈련 데이터 70, 테스트 데이터 30으로 나눈다.
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)
# test_size =0.3 은 훈련과 테스트를 7대3 비율로 나누고 random_state = 10 은 seed 값을 설정하는 부분이다.

print(X_train.shape) # (105, 4)
print(y_train.shape) # (105,)

#%%

# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
# 평균은 0에 가깝고 표준편차는 1에 가까운 결과가 출력 된다.

# 학습/예측(Training/Pradiction)
from sklearn.neighbors import KNeighborsClassifier

# k-NN 분류기를 생성
classifier = KNeighborsClassifier(n_neighbors=5) # knn 모델 생성

# 분류기 학습
classifier.fit(X_train, y_train) # 훈련 데이터와 훈련 데이터의 라벨로 훈련

# 예측
y_pred= classifier.predict(X_test) # 테스트 데이터를 예측
print(y_pred)

# 모델 평가
```

```
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)

# 민감도, 재현율, 정확도, F1 스코어 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)

# k값이 5일 때 정확도 0.9555
```

문제5. iris 데이터에서 가장 정확도가 좋은 k 값을 지정해서 iris 데이터를 분류하는 knn 모델을 생성하는 전체 코드를 작성하시오.

```
# iris 데이터를 knn으로 분류
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
import pandas  as pd


# 1. 데이터 준비
col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']

# csv 파일에서 DataFrame을 생성
dataset = pd.read_csv('c:\\data\\iris2.csv', encoding='UTF-8', header=None, names=col_names)
#print(dataset)

X = dataset.iloc[: , :4].to_numpy()
y = dataset['Class'].to_numpy()

# 데이터 정규화
# 1. 스케일(scale) : 평균은 0이고 표준편차가 1인 데이터로 분포
# 2. min/max 정규화 : 0~1 사이의 숫자로 변경
from sklearn import preprocessing
X=preprocessing.StandardScaler().fit(X).transform(X)

# 훈련 데이터 70, 테스트 데이터 30으로 나눈다.
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)
# test_size =0.3 은 훈련과 테스트를 7대3 비율로 나누고 random_state = 10 은 seed 값을 설정하는 부
```

분이다.

```
print(X_train.shape) # (105, 4)
print(y_train.shape) # (105,)

#%%

# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
# 평균은 0에 가깝고 표준편차는 1에 가까운 결과가 출력 된다.

# 학습/예측(Training/Pradiction)
from sklearn.neighbors import KNeighborsClassifier

# k-NN 분류기를 생성
classifier = KNeighborsClassifier(n_neighbors=12) # knn 모델 생성

# 분류기 학습
classifier.fit(X_train, y_train) # 훈련 데이터와 훈련 데이터의 라벨로 훈련

# 예측
y_pred= classifier.predict(X_test) # 테스트 데이터를 예측
print(y_pred)

# 모델 평가
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)

# 민감도, 재현율, 정확도, F1 스코어 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)

# k값이 12일 때 정확도 1

#%%

# 위의 코드에서 적절한 k값을 알아내는 for문
import  numpy  as np
```

```
errors = []
for i in range(1, 31):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    errors.append(np.mean(pred_i != y_test))
print(errors)

for k, i in enumerate(errors):
    print(k+1, i)
```

위의 결과를 시각화

```
import matplotlib.pyplot as plt

plt.plot(range(1, 31), errors, marker='o')
plt.title('Mean error with K-Value')
plt.xlabel('k-value')
plt.ylabel('mean error')
plt.show()

#%%
```

위의 코드에서 적절한 k값을 알아내는 for문2

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np


acclist = []
err_list = []

for i in range(1,30):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acclist.append(accuracy_score(y_test, y_pred))
    err_list.append(np.mean(y_pred != y_test))

    print(f'k : {i} , acc : {accuracy_score(y_test, y_pred)} ')
```

그래프 사이즈 조정

```
plt.figure(figsize=(12,6))
plt.subplots_adjust(left=0.125,
                    bottom=0.1,
                    right=1,
                    top=0.9,
                    wspace=0.2,
                    hspace=0.35)
```

k값이 변경될 때마다 정확도에 대한 시각화

```
plt.subplot(121)
plt.plot(acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
```

```
plt.xlabel("k value")
plt.ylabel('Accuracy')

# k값이 변경될 때마다 에러에 대한 시각화
plt.subplot(122)
plt.plot(err_list, color='red', marker='o', markerfacecolor='blue')
plt.title('Error', size=15)
plt.xlabel("k value")
plt.ylabel('error')
```

문제6. 유방암 데이터의 정확도를 높이기 위해서 min/max 정규화로 변경하시오.

```
# X = 전체 행, 마지막 열 제외한 모든 열 데이터 -> n차원 공간의 포인트
X = df.iloc[:, 2:].to_numpy() # 데이터 프레임의 2번째 열부터 끝까지 넘파이 array로 변환
y = df['diagnosis'].to_numpy()

#print(df.shape) # (569, 32)
#print (len(X)) # 569
#print (len(y)) # 569

# 데이터 정규화
# 1. 스케일(scale) : 평균은 0이고 표준편차가 1인 데이터로 분포
# 2. min/max 정규화 : 0~1 사이의 숫자로 변경
from sklearn import preprocessing
#X=preprocessing.StandardScaler().fit(X).transform(X) # scale
X = preprocessing.MinMaxScaler().fit(X).transform(X) # min/max

# 훈련 데이터 70, 테스트 데이터 30으로 나눈다.
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)
# test_size =0.3 은 훈련과 테스트를 7대3 비율로 나누고 random_state = 10 은 seed 값을 설정하는 부분이다.

#print(X_train.shape) # (398, 30)
#print(y_train.shape) # (398,)

#%%

# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
# 평균은 0에 가깝고 표준편차는 1에 가까운 결과가 출력 된다.

# 학습/예측(Training/Pradiction)
```

```
from sklearn.neighbors import KNeighborsClassifier

# k-NN 분류기를 생성
classifier = KNeighborsClassifier(n_neighbors=12) # knn 모델 생성

# 분류기 학습
classifier.fit(X_train, y_train) # 훈련 데이터와 훈련 데이터의 라벨로 훈련

# 예측
y_pred= classifier.predict(X_test) # 테스트 데이터를 예측
#print(y_pred)

# 모델 평가
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)

# 민감도, 재현율, 정확도, F1 스코어 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)
```

knn_유방암
데이터...

knn_iris2
데이터...

나이브베이즈를 파이썬으로 구현하기

https://cafe.daum.net/oracleoracle/SgNT/21

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
import pandas  as pd


# 1. 데이터 준비
```

```
col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']

# csv 파일에서 DataFrame을 생성
dataset = pd.read_csv('c:\\data\\iris2.csv', encoding='UTF-8', header=None, names=col_names)
#print(dataset)

# DataFrame 확인
#print(dataset.shape) # (row개수, column개수)
#print(dataset.info()) # 데이터 타입, row 개수, column 개수, 컬럼 데이터 타입
#print(dataset.describe()) # 요약 통계 정보

#print(dataset.iloc[0:5]) # dataset.head()
#print(dataset.iloc[-5:]) # dataset.tail()


# X = 전체 행, 마지막 열 제외한 모든 열 데이터 -> n차원 공간의 포인트
X = dataset.iloc[:,:-1].to_numpy() # DataFrame을 np.ndarray로 변환
#print(X)

# 전체 데이터 세트를 학습 세트(training set)와 검증 세트(test set)로 나눔
# y = 전체 행, 마지막 열 데이터
y = dataset.iloc[:, 4].to_numpy()
#print(y)


# 데이터 분리
from sklearn.model_selection import train_test_split

# 전체 데이터 세트를 학습 세트(training set)와 검증 세트(test set)로 나눔
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 10)
print(len(X_train), len(X_test))

print(X_train[:3])
print(y_train[:3])


# 3. 거리 계산을 위해서 각 특성들을 스케일링(표준화)
# Z-score 표준화: 평균을 0, 표준편차 1로 변환

from sklearn.preprocessing import StandardScaler

# 3. 거리 계산을 위해서 각 특성들을 스케일링(표준화)
# Z-score 표준화: 평균을 0, 표준편차 1로 변환
scaler = StandardScaler() # Scaler 객체 생성
scaler.fit(X_train) # 스케일링(표준화)를 위한 평균과 표준 편차 계산
X_train = scaler.transform(X_train) # 스케일링(표준화 수행)
X_test = scaler.transform(X_test)

# 스케일링(z-score 표준화 수행 결과 확인)
```

```
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')


for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')

# 4. 학습/예측(Training/Pradiction)
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB


#model = GaussianNB(var_smoothing=1e-09) # Gaussian Naive Bayes 모델 선택 - 연속형 자료
#model = GaussianNB() # Gaussian Naive Bayes 모델 선택 - 연속형 자료

model = BernoulliNB(alpha=0.1)
#model = BernoulliNB()

model.fit( X_train, y_train )

# 예측
y_pred= model.predict(X_test)
print(y_pred)


#5. 모델 평가
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)

# 대각선에 있는 숫자가 정답을 맞춘 것, 그 외가 틀린 것

from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 이원 교차표 보는 코드
from sklearn import metrics
naive_matrix = metrics.confusion_matrix(y_test,y_pred)
print(naive_matrix)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)
```

문제7. 위의 나이브 베이즈 모델의 성능을 올리시오.

```
import numpy as np

# 6. 모델 개선 - laplace 값을 변화시킬 때, 에러가 줄어드는 지
```

```
errors = []


for i in  np.arange(0.0, 1.0, 0.001):
   model = GaussianNB( var_smoothing= i)
   model.fit(X_train, y_train)
   pred_i = model.predict(X_test)
   errors.append(np.mean(pred_i != y_test))
print(errors)


'''
for i in  np.arange(0.0, 1.0, 0.001):
   model = BernoulliNB(alpha = i)
   model.fit(X_train, y_train)
   pred_i = model.predict(X_test)
   errors.append(np.mean(pred_i != y_test))
print(errors)
'''

# 여기서 에러가 가장 적은 것을 선택

import matplotlib.pyplot as plt

plt.plot( np.arange(0.0, 1.0, 0.001), errors, marker='o')
plt.title('Mean error with laplace-Value')
plt.xlabel('laplace')
plt.ylabel('mean error')
plt.show()
```

문제8. 유방암 데이터의 나이브베이즈 모델을 파이썬으로 생성하고 정확도를 확인하시오.

문제9. 위의 나이브베이즈 모델을 생성할 때 min/max 정규화를 사용하였다면 이번에는 scale 함수를 적용해서 수행하고 정확도를 확인하시오.

나이브베이
즈_유방...

나이브베이
즈_iris2...

문제10. wine 데이터를 나이브베이즈 모델로 분류하시오.

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
import pandas  as pd

# 데이터셋 준비
df = pd.read_csv("c:\\data\\wine.csv")
#print (df)

# DataFrame 확인
#print(df.shape) # (178, 14)
#print(df.info()) # 데이터 타입
#print(df.describe()) # 요약 통계정보

# X = 전체 행, 마지막 열 제외한 모든 열 데이터 -> n차원 공간의 포인트
X = df.iloc[:, 1:].to_numpy()
y = df['Type'].to_numpy() # 라벨 컬럼
#print(y)

#print(df.shape)
#print(len(X))
#print(len(y))

# 정규화
from sklearn import preprocessing

#X=preprocessing.StandardScaler().fit(X).transform(X)
X=preprocessing.MinMaxScaler().fit(X).transform(X)

from sklearn.model_selection import train_test_split


# 훈련 데이터 90, 테스트 데이터 10으로 나눈다.
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1, random_state = 10)

#print(X_train.shape) # (160, 13)
#print(y_train.shape) # (160,)


# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')
```

```
for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
```

```
# 학습/예측(Training/Pradiction)
#from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

# k-NN 분류기를 생성
#classifier = KNeighborsClassifier(n_neighbors=12)

# 나이브베이즈 분류기를 생성
classifier = GaussianNB() #


# 분류기 학습
classifier.fit(X_train, y_train)

# 예측
y_pred= classifier.predict(X_test)
print(y_pred)

# 작은 이원교차표
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)


# 정밀도 , 재현율, f1 score 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)
```

문제11. 위의 와인 데이터를 knn으로 정확도를 확인하시오.

```
# iris 데이터를 knn으로 분류
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
import pandas  as pd
```

```
# 1. 데이터 준비
col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']

# csv 파일에서 DataFrame을 생성
dataset = pd.read_csv('c:\\data\\iris2.csv', encoding='UTF-8', header=None, names=col_names)
#print(dataset)

X = dataset.iloc[: , :4].to_numpy()
y = dataset['Class'].to_numpy()

# 데이터 정규화
# 1. 스케일(scale) : 평균은 0이고 표준편차가 1인 데이터로 분포
# 2. min/max 정규화 : 0~1 사이의 숫자로 변경
from sklearn import preprocessing
X=preprocessing.StandardScaler().fit(X).transform(X)

# 훈련 데이터 70, 테스트 데이터 30으로 나눈다.
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)
# test_size =0.3 은 훈련과 테스트를 7대3 비율로 나누고 random_state = 10 은 seed 값을 설정하는 부분이다.

print(X_train.shape) # (105, 4)
print(y_train.shape) # (105,)

#%%

# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(4):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
# 평균은 0에 가깝고 표준편차는 1에 가까운 결과가 출력 된다.

# 학습/예측(Training/Pradiction)
from sklearn.neighbors import KNeighborsClassifier

# k-NN 분류기를 생성
classifier = KNeighborsClassifier(n_neighbors=22) # knn 모델 생성

# 분류기 학습
classifier.fit(X_train, y_train) # 훈련 데이터와 훈련 데이터의 라벨로 훈련

# 예측
y_pred= classifier.predict(X_test) # 테스트 데이터를 예측
print(y_pred)
```

```
# 모델 평가
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)

# 민감도, 재현율, 정확도, F1 스코어 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)

# k값이 5일 때 정확도 0.9555

#%%

# 위의 코드에서 적절한 k값을 알아내는 for문
import  numpy  as np

errors = []
for i in range(1, 31):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    errors.append(np.mean(pred_i != y_test))
print(errors)

for k, i in enumerate(errors):
    print(k+1, i)

# 위의 결과를 시각화
import matplotlib.pyplot as plt

plt.plot(range(1, 31), errors, marker='o')
plt.title('Mean error with K-Value')
plt.xlabel('k-value')
plt.ylabel('mean error')
plt.show()

#%%

# 위의 코드에서 적절한 k값을 알아내는 for문2
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np


acclist = []
err_list = []
```

```
for i in range(1,31):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acclist.append(accuracy_score(y_test, y_pred))
    err_list.append(np.mean(y_pred != y_test))

    print(f'k : {i} , acc : {accuracy_score(y_test, y_pred)} ')
```

그래프 사이즈 조정

```
plt.figure(figsize=(12,6))
plt.subplots_adjust(left=0.125,
                    bottom=0.1,
                    right=1,
                    top=0.9,
                    wspace=0.2,
                    hspace=0.35)
```

k값이 변경될 때마다 정확도에 대한 시각화

```
plt.subplot(121)
plt.plot(range(1,31), acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
plt.xlabel("k value")
plt.ylabel('Accuracy')
```

k값이 변경될 때마다 에러에 대한 시각화

```
plt.subplot(122)
plt.plot(range(1,31), err_list, color='red', marker='o', markerfacecolor='blue')
plt.title('Error', size=15)
plt.xlabel("k value")
plt.ylabel('error')
```

knn_wine 데이터...

나이브베이즈_wine...

파이썬 나이브베이즈 사이킷런 3가지

1. BernoulliNB : 이산형 데이터를 분류할 때 적합
2. GaussianNB : 연속형 데이터를 분류할 때 적합
3. MultinomialNB

독버섯 데이터를 나이브베이즈 모델로 분류하기

문제12. 독버섯 데이터를 나이브베이즈 모델로 분류하시오.

```
import pandas as pd  # 데이터 전처리를 위해서
import seaborn as sns # 시각화를 위해서

df = pd.read_csv("c:\\data\\mushrooms.csv")

# get_dummies 함수를 이용해서 값의 종류에 따라 0 아니면 1로 변환한다.
df = pd.get_dummies(df)
#print (df)

# DataFrame 확인

#print(df.shape) # (8124, 119)
#print(df.info())
#print(df.describe())
#print(df)

# X = 전체 행, 마지막 열 제외한 모든 열 데이터 -> n차원 공간의 포인트
X = df.iloc[:, 2:].to_numpy()
y = df.iloc[:, 1].to_numpy()

#print(df.shape)
#print(len(X))
#print(len(y))

#%%

'''
# 정규화
from sklearn import preprocessing

X=preprocessing.StandardScaler().fit(X).transform(X)
#X=preprocessing.MinMaxScaler().fit(X).transform(X)
'''

from sklearn.model_selection import train_test_split


# 훈련 데이터 75, 테스트 데이터 25으로 나눈다.
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25, random_state = 10)

#print(X_train.shape)
#print(y_train.shape)

'''
# 스케일링(z-score 표준화 수행 결과 확인)
for col in range(4):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(4):
```

```
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
'''

# 학습/예측(Training/Pradiction)
#from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
#from sklearn.naive_bayes import MultinomialNB
#from sklearn.naive_bayes import BernoulliNB

# k-NN 분류기를 생성
#classifier = KNeighborsClassifier(n_neighbors=12)

# 나이브베이즈 분류기를 생성
classifier = GaussianNB() # 0.9615
#classifier = MultinomialNB() # 0.9497
#classifier = BernoulliNB() # 0.9350

# 분류기 학습
classifier.fit(X_train, y_train)

# 예측
y_pred= classifier.predict(X_test)
print(y_pred)

# 작은 이원교차표
from sklearn.metrics import confusion_matrix
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)


# 정밀도 , 재현율, f1 score 확인
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)

# 정확도 확인하는 코드
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)  #  0.941
```

문제13. 위의 독버섯 분류 나이브 베이즈 모델의 정확도를 높이는 laplace 값을 알아내시오.

나이브베이즈_독버...

21.02.23

2021년 2월 23일 화요일 오전 9:44

파이썬을 활용한 머신러닝

1. knn을 파이썬으로 구현하는 방법
2. naivebayes를 파이썬으로 구현하는 방법

df = pd.get_dummies(df, drop_first=True)
drop_first=True를 사용하게 되면 생성된 더미변수중 하나의 컬럼(첫번째)을 삭제한다.

의사결정트리
랜덤포레스트(앙상블 + 의사결트리)
회귀분석

독버섯 데이터를 의사결정 트리 알고리즘으로 분류하기

https://cafe.daum.net/oracleoracle/SgNT/5

의사결정트리_독버...

문제14. iris2 데이터를 의사결정트리로 분류하고 시각화 하시오.

의사결정트리_iris2...

문제15. 위의 코드에서 max_depth를 4를 주고 다시 모델을 만들고 시각화 하시오.

```
classifier = tree.DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

문제16. 화장품 데이터(skin.csv)를 이용해서 의사결정트리 모델을 생성하시오.

의사결정트
리_skin...

문제17. 독일은행 데이터의 채무 불이행자를 예측하고 의사결정트리 나무를 시각화 하시오.

의사결정트
리_독일...

문제18. 위의 의사결정트리의 모델을 의사결정트리 + 앙상블 기법을 적용한 랜덤포레스트로 구현하시오.

랜덤포레스
트_독일...

단순 회귀 분석

https://cafe.daum.net/oracleoracle/SgNT/7

mpg는 mile per gallon의 약자로 영국과 미국에서는 한국과 달리 갤런당 마일 단위로 연비를 표시한다. 한국은 리터당 킬로미터(km/L) 단위로 표시하는데 mpg열을 한국에서 사용하는 km/L로 변환해줘야 한다. 1갤런이 3.78541이고 1마일이 1.60934km 이므로 1mpg(mile per gallon)은 (1.60934 / 3.78541) = 0.425km/L 이 된다.

단순회귀분
석_auto...

문제. 체중과 키 데이터를 이용해서 단순 선형 회귀ㅜㄴ석을 하고 seaborn 그래프로 시각화 하시오.

https://cafe.daum.net/oracleoracle/SgNT/65

단순회귀분석_체중...

21.02.24

2021년 2월 24일 수요일 오전 9:39

복습

1. 파이썬으로 knn 구현하기
2. 파이썬으로 naivebayes 구현하기
3. 파이썬으로 decision tree 구현하기
4. 파이썬으로 regression 구현하기 (단순회귀분석, 다중회귀분석)

머신러닝 데이터 분석 5가지 단계

1. 데이터 수집과 설명 : pandas 사용
2. 데이터 탐색 및 시각화 : pandas, matplotlib, seaborn 사용
3. 머신러닝 모델 훈련 : sklearn 사용
4. 머신러닝 모델 평가 : pandas 사용
5. 머신러닝 모델 성능 개선 : pandas 사용 (파생변수 생성)

02/23 마지막 문제의 결정계수를 확인하고 성능을 높이시오.

https://cafe.daum.net/oracleoracle/SgNT/7

단순회귀를 다항회귀로 변경해서 성능을 올린다.

1. 단순회귀 : 독립변수 한개에 종속변수 한 개 (선형 회귀선)
2. 다항회귀 : 독립변수 한개에 종속변수 한 개 (비선형 회귀선)
3. 다중회귀 : 종속변수에 영향을 주는 독립변수가 여러개인 경우

첫번째 예제 단순회귀분석의 결과 그래프(무게와 연비간의 예측값과 실제값의 비교)를 보면 실제값은 왼쪽으로 편향되어있고 예측값을 반대로 오른쪽으로 편중되는 경향을 보인다. 따라서 독립변수(weight) 과 종속변수(mpg) 사이의 선형관계가 있지만, 모형의 오차를 더 줄일 필요가 있어 보인다. 앞에서 본 산포도를 보면 직선보다는 곡선이 더 적합해 보인다. 비선형 회귀분석을 통해 모형의 정확도를 더 높이시오.

다항회귀분석_auto...

문제19. 02/23 마지막 문제를 다항회귀로 비선형 회귀선을 만들어 성능을 올리시오.

다항회귀분
석_체중...

단순회귀의 결정계수 : 0.8657609246754262
다항회귀의 결정계수 : 0.9263965046109321

다중 회귀 분석

종속변수에 영향을 주는 독립변수가 여러개인 경우

ex)

1. 미국 우주 왕복선 폭파원인
2. 미국 대학교 입학점수에 영향을 미치는 과목 분석
3. 미국 국민 의료비에 영향을 주는 요소 분석

미국 우주 왕복선 폭파원인

다중회귀분
석_우주...

문제20. statsmodels 패키지를 이용해서 다중회귀 분석을 해보았는데 중요한 독립변수인 temperature 만 이용해서 단순회귀 분석을 진행하고 분석된 결과를 출력하시오.

```
# 단순 회귀 분석 코드
model = smf.ols(formula = 'distress_ct ~ temperature', data = df)

result = model.fit()
print (result.summary())
```

분석결과 :

Intercept 3.6984
temperature -0.0475

회귀식 :

y(파손수) = 3.6984 - 0.0475 * x1(온도)

미국 대학교 입학점수에 영향을 미치는 과목 분석

종속변수 : acceptance
독립변수 : academic, sports, music

위와 같이 독립변수의 단위가 서로 다를 때 표준화를 하고 회귀분석해야 한다.

다중회귀분
석_입학...

표준화를 하지 않았을 때는 체육점수가 학과점수보다 더 영향력이 크다.
표준화를 하면 학과점수가 체육점수보다 더 영향력이 크다.

미국 국민 의료비에 영향을 주는 요소 분석

문제21. 미국 의료비 데이터 (insurance.csv)를 다중회귀분석 하시오.

다중회귀분
석_의료...

sex[T.male] -131.3520
- 남성은 여성에 비해 매년 의료비가 131달러 비용이 적게 든다.

smoker[T.yes] 2.385e+04
- 흡연자는 비흡연자보다 매년 의료비가 23,860달러 비용이 더 든다.

age 256.8392
- 나이가 1살 많아질 때마다 평균적으로 의료비가 256달러 더 든다.

bmi 339.2899
- 비만지수가 증가할 때마다 339달러 더 든다.

children 475.6889
- 분양가족이 한명 더 늘어날 때마다 의료비가 475달러 더 든다.

region[T.northwest] -352.7901
region[T.southeast] -1035.5957
region[T.southwest] -959.3058
- 북동지역이 북서, 남동, 남서에 비해 의료비가 더 든다.

문제22. 비만인 사람은 의료비가 더 지출 되는지 파생변수를 추가해서

확인하시오.

(bmi30 이라는 파생변수를 추가하는데 bmi가 30 이상이면 1, 아니면 0)

```
def func_1(x):
   if x >= 30:
      return 1
   else:
      return 0

df['bmi30'] = df['bmi'].apply(func_1)

print (df)
```

파생변수 추가 방법

R : df$bmi <- ifelse(bmi >= 30, 1, 0)

파이썬 : df['bmi30'] = df['bmi'].apply(함수)

문제23. 비만인 사람을 분류하는 파생변수를 추가했으면 결정계수가 더 올라가는지 확인하시오.

```
# 다중 회귀 분석 코드
model = smf.ols(formula = 'expenses ~ age + sex + bmi + children + smoker + region + bmi30',
            data = df)

result = model.fit() # 모델 훈련
print (result.summary()) # 결과
```

문제24. 비만이면서 흡연까지 하면 의료비가 더 올라가는지 확인하시오.

다중회귀분
석_의료...

비만이면서 흡연까지 하게 되면 연간 의료비가 19,790 달러 더 들거라 예상

파이썬으로 다중공선성 확인하기

회귀 분석에서 사용된 모형의 일부 독립변수가 다른 독립변수와의 상관정도가 아주 높아서 회귀분석 결과에 부정적 영향을 미치는 현상

두 독리변수들끼리 서로에게 영향을 주고 있다면 둘 중 하나의 영향력을 검증할 때 다른 하나의 영향력이 약해진다.

ex)
학업성취도, 일 평균 음주량, 혈중 알코올 농도

팽창계수가 보통은 10보다 큰 것을 골라내고 까다롭게 하려면 5보다 큰 것을 골라낸다.

예를들어 일 평균 음주량, 혈중 알코올 농도 둘다 팽창계수가 높게 나온다면 둘중에 하나를 배고 아래와 같이 회귀분석한다.

학업성취도, 일 평균 음주량 -> 회귀분석
학업성취도, 혈중 알코올 농도 -> 회귀분석

다중공선성 테스트

https://cafe.daum.net/oracleoracle/SgNT/97

crab.csv : 게의 크기, 무게 등에 대한 데이터로 종속변수가 y컬럼인데 0과 1로 분류하는 데이터

다중공선성
_crab 데...

문제25. test_vif1.csv의 팽창계수를 확인하여 vif 지수가 높은 독립변수들이 무엇이 있는지 확인하시오.

test_vif1.csv : 아이큐, 공부시간, 시험점수로 구성되어 있는 데이터

다중공선선
_test_vif1...

step 함수 파이썬으로 구현하기
내가 추가한 파생변수가 유의미한 파생변수인지 확인하고 여러 독립변수들 중에 불필요한 독립변수를 제거하고 필요한 독립변수들만 골라내서 회귀분석 할 때 사용

문제26. 새로 추가한 파생변수가 유의미한 파생변수인지 확인하는 작업을 수행하시오.
(bmi30)

```
# R의 step 함수의 기능을 가지고 있는 패키지 import
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
import pandas as pd

#%%
# 데이터 불러오기
df = pd.read_csv('c:\\data\\insurance.csv', engine='python', encoding='CP949')

print (df.head())

#%%
# 파생변수 추가
def func_1(x):
    if x >= 30:
        return 1
    else:
        return 0

df['bmi30'] = df['bmi'].apply(func_1)


#%%
#  컬럼 인코딩
df = pd.get_dummies(df, drop_first=True)

print (df.head())
print (df.columns.values) # 컬럼명을 numpy array 형태로 출력

#%%
# 독립변수와 종속변수를 numpy array 로 변환
X = df.iloc[:, [0,1,2,4,5,6,7,8,9]].to_numpy() # numpy array 형태
y = df.iloc[:, 3].to_numpy() # numpy array 형태

#print (X)
#print (y)

#%%
# 회귀 모델 생성
#from sklearn.linear_model import LinearRegression
model = LinearRegression()

#%%
# step 함수를 이용해서 필요한 독립변수들을 선별
selector = RFE(model, n_features_to_select = 6, step = 1)
#  n_features_to_select = 선별할 독립변수의 갯수, step = 스텝횟수
selector = selector.fit(X, y)

print (df.iloc[:, [0,1,2,4,5,6,7,8,9]].head())
print (selector.support_) # 선별된 독립변수가 True로 출력
print (selector.ranking_) # 순위 출력
```

```
#%%
# 학습을 마친 모형에 test data를 적용하여 결정계수(R-제곱) 계산
model.fit(X, y)
r_square = model.score(X, y)

print(r_square)
print('\n')

# 회귀식의 기울기
print('기울기 a: ', model.coef_)
print('\n')

# 회귀식의 y절편
print('y절편 b', model.intercept_)
print('\n')
```

문제27. 비만이면서 흡연을 하는 사람에 대한 파생변수를 bmi30_smoker라는 이름으로 추가하시오.

step함수_
의료비...

문제28. 위 문제에서 선별된 독립변수들만 가지고 회귀모델을 생성하고 훈련시켜 나온 결정계수가 어떻게 되는지 확인하시오.

21.02.25

2021년 2월 25일 목요일 오전 9:46

복습
리눅스와 하둡 : 데이터 저장공간 구성
SQL, 파이썬의 시각화 : 데이터를 읽고 정보를 추출해서 결론 도출

로지스틱 회귀
종속변수가 범주형인 경우에 적용되는 회귀분석 모형 (반응변수)

회귀분석 종류
1. 단순회귀 분석 : 종속변수가 연속형 수치 데이터
2. 다항회귀 분석 : 종속변수가 연속형 수치 데이터
3. 다중회귀 분석 : 종속변수가 연속형 수치 데이터
4. 로지스틱 회귀 분석 : 종속변수가 범주형인 데이터

타이타닉 데이터

https://cafe.daum.net/oracleoracle/SgNT/112

머신러닝 데이터 분석을 하기 위한 단계
1. 데이터 불러오기
   ------ 파생변수 추가 ------
2. 데이터 탐색 및 전처리 - 결측치 처리
3. 범주형 데이터에 대해 더미변수 생성
4. 데이터 정규화 또는 표준화 - 단위를 일정하게 조정
5. 훈련 데이터와 테스트 데이터 분할
6. 머신러닝 모델 생성
7. 훈련데이터로 머신러닝 모델 훈련
8. 머신러닝 모델 평가
9. 머신러닝 모델 성능 개선

결측치 처리
머신러닝 데이터 분석의 정확도는 분석 데이터의 품질에 의해 좌우된다.
데이터 품질을 높이기 위해서는 누락 데이터 처리, 중복 데이터 처리 등 오류를 수정하고 분석 목적에 맞게 변형하는 과정이 필요하다.
데이터 프레임에는 원소 데이터 값이 종종 누락되는 경우가 있다.
일반적으로 데이터 값이 존재하지 않는 누락 데이터를 NaN 으로 표시한다.
머신러닝 분석 모형에 데이터를 입력하기 전에 반드시 누락 데이터를 제거하거나 다른 적절한

값으로 대체하는 과정이 필요하다.
누락 데이터가 많아지면 데이터의 품질이 떨어지고 머신러닝 분석 알고리즘을 왜곡하는 현상이 발생하기 때문이다.

데이터 표준화
실무에서 접하는 데이터셋은 다양한 사람들의 손에 거쳐서 만들어진다.
여러곳에서 수집한 자료들은 대소문자 구분, 약칭 활용 등 여러가지 원인에 의해서 다양한 형태로 표현된다.
잘 정리된것으로 보이는 자료를 자세히 들여다보면 서로 다른 단위가 섞여있거나 같은 대상을 다른 형식으로 표현하는 경우가 의외로 많다.
그래서 단위 환산을 통해서 같은 단위 형식으로 변환을 해줄 필요가 있다.

1. 누락 데이터를 찾는 함수
    a. isnull() : 누락 데이터는 True로 반환하고, 누락 데이터가 아니면 False를 반환
    b. notnull() : 누락 데이터는 False를 반환하고, 누락 데이터가 아니면 True를 반환

- ex)
```
import seaborn as sns
tat = sns.load_dataset('titanic')
tat.head().isnull()
tat.head().isnull().sum()
```

문제29. emp 데이터 프레임의 누락 데이터는 몇개인지 확인하시오.

```
emp.isnull().sum()
```

2. 누락 데이터를 제거하는 함수
    a. 열을 삭제
    -
```
import seaborn as sns
tat = sns.load_dataset('titanic')
tat.isnull().sum()

# 누락데이터(NaN)이 500개 이상이면 열을 삭제
tat.dropna(axis=1, thresh=500, inplace=True)

tat.columns
```

    -
```
import seaborn as sns
tat = sns.load_dataset('titanic')
tat2 = tat.dropna(axis=1, thresh=500)

tat2.columns
```

    b. 행을 삭제
    -
```
import seaborn as sns
```

```
tat = sns.load_dataset('titanic')
tat.deck.isnull().sum()

tat.dropna(axis=0, inplace=True)
tat.deck.isnull().sum()
```

문제30. emp 데이터 프레임에서 comm의 결측치를 확인하고 comm의 결측치 행들을 삭제하시오.

```
import pandas as pd

emp = pd. read_csv('c:\\data\\emp.csv')

emp.dropna(axis=0, inplace=True)

emp.isnull().sum()
```

3. 누락 데이터를 다른 데이터로 치환하는 함수
   a. 평균값으로 누락 데이터를 치환

```
- import seaborn as sns
  tat = sns.load_dataset('titanic')

  mean_age = tat['age'].mean(axis=0)

  tat['age'].fillna(mean_age, inplace=True)
```

문제31. 위의 예제 코드에서 나이 데이터의 누락 테이터를 평균값으로 치환했을 때와 안했을 때의 차이를 확인 하시오.

```
import seaborn as sns
tat = sns.load_dataset('titanic')

mean_age = tat['age'].mean(axis=0)

print (tat['age'].isnull().sum())

tat['age'].fillna(mean_age, inplace=True)

print (tat['age'].isnull().sum())
```

   b. 최빈값으로 누락 데이터를 치환

```
- import seaborn as sns
  tat = sns.load_dataset('titanic')

  most_age = tat['age'].value_counts(dropna=True).idxmax()

  print (most_age)

  tat['age'].fillna(most_age, inplace=True)
```

```
print (tat['age'].isnull().sum())
```

문제32. 위의 코드에서 최빈값에 해당하는 나이를 출력하시오.

```
import seaborn as sns
tat = sns.load_dataset('titanic')

most_age = tat['age'].value_counts(dropna=True).idxmax()

print (most_age)
```

c. 이웃하고 있는 주변의 데이터로 누락 데이터를 치환

- 문법 : df['age'].fillna(method='ffill', inplace=True)

```
import seaborn as sns
tat = sns.load_dataset('titanic')

tat['deck'].fillna(method='ffill', inplace=True) # 결측치 아래의 데이터로 채워넣는다.
tat['deck'].fillna(method='bfill', inplace=True) # 결측치 위의 데이터로 채워넣는다.

print (tat.isnull().sum())
```

d. 누락 데이터가 아닌 데이터에 대한 회귀 예측값으로 누락 데이터를 치환

- 결측치가 너무 많으면 컬럼을 지운다.
- 결측치가 다른 값으로 치환하고자 한다면 아래의 3가지 방법으로 치환
  - 평균값, 최빈값, 주변의 행들로 치환, 회귀분석의 예측값

문제33. emp 데이터 프레임에서 월급과 커미션 컬럼을 삭제하시오.

```
import pandas as pd

emp = pd. read_csv('c:\\data\\emp.csv')

emp.drop(['ename', 'comm'], axis=1, inplace=True)

print (emp)
```

문제34. emp 데이터 프레임에서 NA가 있는 행은 무조건 지우시오.

```
import pandas as pd

emp = pd. read_csv('c:\\data\\emp.csv')

emp.dropna(axis=0, how='any', inplace=True)
```

```
print (emp)
```

문제35. women_child 파생변수를 추가하고 다시 학습시켜서 모델의 성능을 확인하시오.

로지스틱회
귀_타이...

범주형 데이터를 처리하는 방법

1. 더미변수로 처리 : 숫자 0과 1로 표현
   - 여자 또는 아이는 1이고 아니면 0으로 표현

2. 구간 분할로 처리 : 일정한 구간으로 나눈다.
   - 미세먼지의 농도 (좋음, 보통, 나쁨, 매우나쁨)

구간분할 설명

https://cafe.daum.net/oracleoracle/SgNT/134

구간분할

문제36. emp 데이터 프레임의 월급을 3개로 구간 분할 하는데 고소득, 중간소득, 저소득으로 나눠서 출력하는 sal_bin 파생변수를 추가하시오.

```
import pandas as pd
import numpy as np

emp = pd.read_csv('c:\\data\\emp.csv')

count, bin_dividers = np.histogram(emp.sal, bins=3)

print (count) # 각 구간에 속하는 값의 갯수
print (bin_dividers) # 경계값 리스트

bin_names = ['저소득', '중간소득', '고소득']

emp['sal_bin'] = pd.cut(x = emp.sal, bins=bin_dividers, labels=bin_names, include_lowest=True)
```

```
print (emp)
```

seaborn의 타이타닉 데이터를 로지스틱 회귀로 분류

https://cafe.daum.net/oracleoracle/SgNT

문제37. 문제35번 코드를 로지스틱 회귀가 아니라 앙상블 기법이 추가된 랜덤포레스트로 수행해서 정확도가 더 올라가는지 확인하시오.

```
### 기본 라이브러리 불러오기
import pandas as pd
import seaborn as sns

'''
[Step 1] 데이터 준비/ 기본 설정
'''

# load_dataset 함수를 사용하여 데이터프레임으로 변환
df = sns.load_dataset('titanic')

#  IPython 디스플레이 설정 - 출력할 열의 개수 한도 늘리기
pd.set_option('display.max_columns', 15)

print (df.head())

#%%
# 결측치 확인
print (df.isnull().sum())

#%%
'''
[Step 2] 데이터 탐색/ 전처리
'''

# NaN값이 많은 deck 열을 삭제, embarked와 내용이 겹치는 embark_town 열을 삭제
rdf = df.drop(['deck', 'embark_town'], axis=1)
print (rdf.head())

#%%
# age 열에 나이 데이터가 없는 모든 행을 삭제 - age 열(891개 중 177개의 NaN 값)
# how='all' # 행 또는 열의 모든 값이 NaN 일때 사용 가능
rdf = rdf.dropna(subset=['age'], how='any', axis=0)
print (rdf.head())

#%%
# embarked 열의 NaN값을 승선도시 중에서 가장 많이 출현한 값으로 치환하기
most_freq = rdf['embarked'].value_counts(dropna=True).idxmax()
rdf['embarked'].fillna(most_freq, inplace=True)
print (rdf.head())
```

```
#%%
'''
[Step 3] 분석에 사용할 속성을 선택 및 원핫 인코딩
'''

#print (rdf.info())

# 분석에 활용할 열(속성)을 선택
ndf = rdf[['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'embarked']]
print (ndf.head())

#%%
# 원핫인코딩 - 범주형 데이터를 모형이 인식할 수 있도록 숫자형으로 변환
onehot_sex = pd.get_dummies(ndf['sex'])
ndf = pd.concat([ndf, onehot_sex], axis=1)

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

ndf.drop(['sex', 'embarked'], axis=1, inplace=True)
print (ndf.head())

#%%
'''
[Step 4] 데이터셋 구분 - 훈련용(train data)/ 검증용(test data)
'''

# 속성(변수) 선택
X=ndf[['pclass', 'age', 'sibsp', 'parch', 'female', 'male',
    'town_C', 'town_Q', 'town_S']] #독립 변수 X
y=ndf['survived']                  #종속 변수 Y

#%%
print (X.head())

#%%
# 설명 변수 데이터를 정규화(normalization)
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)
print(X)

#%%
# train data 와 test data로 구분(7:3 비율)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)

print('train data 개수: ', X_train.shape)
print('test data 개수: ', X_test.shape)
print('\n')

#%%
```

```
'''
[Step 5] 로지스틱 분류 모형 - sklearn 사용
'''

from  sklearn.ensemble   import  RandomForestClassifier

tree_model = RandomForestClassifier( n_estimators=100,
                                     oob_score=True,
                                      random_state= 9 )

#%%
'''
[Step 6] train data를 가지고 모형 학습
'''

# train data를 가지고 모형 학습
tree_model.fit(X_train, y_train)

#%%
'''
[Step 7] test data를 가지고 y_hat을 예측 (분류)
'''
# test data를 가지고 y_hat을 예측 (분류)
y_hat = tree_model.predict(X_test)

print(y_hat[0:10])
print(y_test.values[0:10])
print('\n')

#%%
'''
[Step 8] 모형 성능 평가 - Confusion Matrix 계산
'''

# 모형 성능 평가 - Confusion Matrix 계산
from sklearn import metrics
svm_matrix = metrics.confusion_matrix(y_test, y_hat)
print(svm_matrix)
print('\n')

#%%
# 모형 성능 평가 - 평가지표 계산
svm_report = metrics.classification_report(y_test, y_hat)
print(svm_report)
```

문제38. 로지스틱 회귀로 시본의 타이타닉 분류 모델을 만드는데 여자와 아이 파생변수 말고 아래의 그래프를 보고 더 좋은 파생변수를 생각해서 학습 시키고 정확도를 출력하시오.

21.03.02

2021년 3월 2일 화요일 오전 9:49

신경망

인공신경망에서는 가중치라는 것으로 기억의 효과를 대체할 수 있음을 설명

파라미터와 하이퍼 파라미터의 차이

1. 파라미터 : 모델 내부에서 확인이 가능한 변수로 데이터를 통해서 산출이 가능한 값
   예측을 수행할 때, 모델에 의해 요구되어지는 값
   사람에 의해 수작업으로 측정되지 않는다.

   - 신경망에서의 가중치, 회귀분석에서의 결정계수, 서포트 벡터머신의 서포트 벡터

2. 하이퍼 파라미터 : 모델에서 외적인 요소로 데이터 분석을 통해 얻어지는 값이 아니라 사용자가 직접 설정해주는 값
   모델의 파라미터 값을 측정하기 위해 알고리즘 구현 과정에서 사용
   주로 알고리즘 사용자에 의해 결정된다.

   - 신경망에서의 학습률(러닝레이트), knn에서의 k값, 의사결정트리에서의 나무의 깊이, 서포트 벡터 머신에서의 C값과 gamma값

퍼셉트론 : 뇌의 신경세포 하나를 컴퓨터로 흉내 낸 것

문제39. R을 활용한 머신러닝에서 사용한 콘크리트 데이터를 파이썬의 신경망으로 강도를 예측하시오.

https://cafe.daum.net/oracleoracle/SgNT/142

문제40. 위의 신경망에 성능을 올리시오.

(뉴런의 갯수를 늘리거나 층을 더 늘리는 신경망에 연관된 하이퍼 파라미터를 알아내야 한다.)

신경망_콘크리트...

하이퍼 파라미터를 자동으로 알아내는 grid search를 이용

문제41. 모델을 생성할 때 설정하는 random_stats도 gridsearch해서 알아내게 하시오.

신경망_콘크리트...

문제42. grid search를 이용하지 않고 보스톤 하우징 데이터의 수치 예측을 하는 신경망코드를 작성하시오.

문제42

문제43. 보스톤 하우징 데이터의 가격 수치예측하는 신경망 코드를 grid Search를 이용해서 작성하고 상관계수값을 올리시오.

신경망_boston...

서포트 벡터 머신

문제46. gridsearch을 이용한 신경망 코드를 이용해서 수치예측이 아닌 분류로 머신러닝 함수를 변경해서 훈련시키고 정확도를 확인하시오.

정확도를 올리기 위한 실험 내용

1. 신경망을 2층 에서 3층으로 변경

2. 나이의 결측치행 177개를 삭제하는게 아니라 다른 값으로 치환한다.
   - 나이의 평균값
   - 나이의 최빈값
   - 실제 케글애서는 이름의 호칭의 평균값 (시본 타이타닉에는 이름이 없다.)
3. 나이와 운임의 이상치를 제거하고 학습

연관규칙

https://cafe.daum.net/oracleoracle/SgNT/12

간단한 성능 측정치(지지도, 신뢰도, 향상도)를 이용해서 거대한 데이터에서 데이터간의 연관성을 찾는 알고리즘

ex)

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder # 연관성 분석을 위한 데이터 전처리
from mlxtend.frequent_patterns import apriori # 연관성 분석을 위한 아프리오리 알고리즘 함수

dataset=[['사과','치즈','생수'],
['생수','호두','치즈','고등어'],
['수박','사과','생수'],
['생수','호두','치즈','옥수수']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_) #위에서 나온걸 보기 좋게 데이터프레임으로 변경

frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)
print(frequent_itemsets ) # 사람들이 많이 산 물건들을 출력

from mlxtend.frequent_patterns import association_rules # 연관성을 찾는 함수 import
print( association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3) )
# 신뢰도를 기준으로 물품들간의 연관성을 출력하는데 임계치를 0.3 이상인 것만 출력

# 생수와 사과는 같은곳에 진열한다.
```

문제47. 아래의 데이터에서 연관성을 도출하시오.

```
dataset=[['빵','우유'],
['맥주','빵','기저귀','우유'],
['맥주','콜라','기저귀','우유'],
['콜라','빵','기저귀','우유']]
```

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder # 연관성 분석을 위한 데이터 전처리
from mlxtend.frequent_patterns import apriori # 연관성 분석을 위한 아프리오리 알고리즘 함수

dataset=[['빵','우유'],
['맥주','빵','기저귀','우유'],
['맥주','콜라','기저귀','우유'],
['콜라','빵','기저귀','우유']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_) #위에서 나온걸 보기 좋게 데이터프레임으로 변경

frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)
print(frequent_itemsets ) # 사람들이 많이 산 물건들을 출력

from mlxtend.frequent_patterns import association_rules # 연관성을 찾는 함수 import
print( association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3) )
# 신뢰도를 기준으로 물품들간의 연관성을 출력하는데 임계치를 0.3 이상인 것만 출력

print( association_rules(frequent_itemsets, metric="lift", min_threshold=0.3) )
# 향상도를 기준으로 물품들간의 연관성을 출력
```

문제48. R을 활용한 머신러닝에서 사용한 보습학원 데이터로 연관분석 하시오. 보습학원이 있는 건물에는 어떤 업종이 많이 있는지 확인하시오.

아프리오리
알고리즘...

21.03.03

2021년 3월 3일 수요일 오전 9:45

아프리오리 알고리즘 (장바구니 분석) 파이썬 코드 정리

https://cafe.daum.net/oracleoracle/SgNT/162

아프리오리
알고리즘...

k-means 알고리즘
주어진 데이터를 k개의 군집으로 묶는 알고리즘으로 k개 만큼 군집수를 초깃값으로 지정하고, 각 객체를 가까운 초깃값에 할당하여 군집을 형성하고 각 군집의 평균을 재계산하여 초깃값을 갱신하는 과정을 반복하여 k개의 최종군집을 형성한다.

군집 절차 순서

1. k개 객체 선택
2. 할당 (Assingment)
3. 중심갱신
4. 반복

k-means의 하이퍼 파라미터 : k값

비지도 학습 : 비지도학습은 입력 데이터에 대한 정답인 레이블이 없는 상태에서 데이터가 어떻게 구성되었는지 알아내는 기계학습 기법이다.

1. k-means
2. som (자기 조직화 지도) : 인공신경망을 활용한 비지도 학습

https://cafe.daum.net/oracleoracle/SgNT/11

k-means 예
제

문제49. 아래의 예제로 데이터 프레임을 만들고 토마토가 어느 군집에 속하는지 출력하시오.

예제:

```
from pandas import Series, DataFrame
import numpy as np

data = DataFrame({ "x": [10,1,10,7,3,1,6],   "y": [9,4,1,10,10,1,7] }, index=['APPLE','BACON','BANANA','CARR
OT','CELERY','CHEESE','TOMATO'])
data

### 기본 라이브러리 불러오기
'''
[Step 1] 데이터 준비
'''
import pandas as pd
import matplotlib.pyplot as plt
from pandas import Series, DataFrame
import numpy as np

df = DataFrame({ "x": [10,1,10,7,3,1,6],   "y": [9,4,1,10,10,1,7] },
            index=['APPLE','BACON','BANANA','CARROT','CELERY','CHEESE','TOMATO'])

print (df)
#%%
'''
[Step 2] 데이터 탐색
'''
# 데이터 살펴보기
print(df.head())
print('\n')

# 데이터 자료형 확인
print(df.info())
print('\n')

# 데이터 통계 요약정보 확인
print(df.describe())
print('\n')

#%%
'''
[Step 3] 데이터 전처리
'''
# 비지도 학습이기 때문에 y값을 생성하지 않는다.

# 분석에 사용할 속성을 선택
X = df.iloc[:, :]
print(X[:5])
print('\n')
```

```
# k-means가 이상치에 민감하므로 이상치의 영향력을 감쇠시키기 위해 표준화한다.

# 설명 변수 데이터를 정규화
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)

print(X[:5])
print('\n')

#%%
'''
[Step 4] k-means 군집 모형 - sklearn 사용
'''
# sklearn 라이브러리에서 cluster 군집 모형 가져오기
from sklearn import cluster

# 모형 객체 생성
kmeans = cluster.KMeans(init='k-means++', n_clusters=2, n_init=10)

# 모형 학습
kmeans.fit(X)

# 예측 (군집)
cluster_label = kmeans.labels_
print(cluster_label)
print('\n')

# 예측 결과를 데이터프레임에 추가
df['Cluster'] = cluster_label
print(df)
print('\n')

#%%
# 그래프로 표현 - 시각화
df.plot(kind='scatter', x='x', y='y', c='Cluster', cmap='Set1',
        colorbar=False, figsize=(10, 10))

#%%
df.plot(kind='scatter', x='x', y='y', c='Cluster', cmap='Set1',
        colorbar=True, figsize=(10, 10))
plt.show()
plt.close()
```

문제50. UCI 데이터 중 식품에 관련한 데이터를 k-means 알고리즘으로 분류하시오.

k-means_
식품 데...

문제51. 유방암 데이터를 파이썬의 k-means로 분류하시오.

k-means_
유방암...

문제52. 아래의 사이트를 참고해서 iris 데이터를 사이킷런의 som 패키지로 군집화하는 실습을 수행하시오.

1. 아나콘다 프롬프트 창 : pip install sklearn-som
2. 아래의 사이트의 예제를 수행
   https://pypi.org/project/sklearn-som/

동영상 확인

랜덤 포레스트

랜덤 포레스트는 의사결정나무의 특징인 분산이 크다는 점을 고려하여 배깅과 부스팅보다 더 많은 무작위성을 주어 약한 학습기들을 생성한 후 이를 선형결합하여 최종 학습기를 만드는 방법이다.

ex1)
iris 데이터를 의사결정트리로 분류하는 실습 + gridsearch 기능 추가
https://cafe.daum.net/oracleoracle/SgNT/77

의사결정트
리

ex2)
iris 데이터를 랜덤포레스트로 분류하는 실습 + gridsearch 기능 추가
https://cafe.daum.net/oracleoracle/SgNT/79

랜덤포레스
트

ex3)
시본의 타이타닉 데이터를 랜덤포레스트 모델 생성하기
https://cafe.daum.net/oracleoracle/SgNT/6

랜덤포레스트_타이...

문제53. 예제3. 시본의 타이타닉 랜덤포레스트 모델의 oob 점수는 0.74였고 정확도는 0.80인데 gridsearch 코드를 추가하고 모델의 성능을 올리시오.

캐글 도전

https://cafe.daum.net/oracleoracle/SgNT/9

현재 ongoing 중인 캐글 타이타닉 로지스틱 회귀 모델 만들기

타이타닉_kaggle_v1

1. 신경망 모델 + grid search
2. 나이의 결측치를 SQL때 처럼 호칭의 평균값으로 치환
3. 랜덤포레스트 모델 + grid search
4. 앙상블 : xgboost 모델 + grid search

문제54. 로지스틱 회귀 모델 + gridsearch를 구현해서 kaggle에서 점수를 확인하시오.

문제55. 신경망 + gridsearch로 다시 한번 kaggle에 도전하시오.

신경망(gridsearc...

kaggle

2021년 3월 5일 금요일 오후 4:07

https://cafe.daum.net/oracleoracle/SgNT/176

신경망 + gird search 기능

신경망
(gridsearc...

개선방법

1. 알고리즘 변경 : Xgboost, Adboost
2. 결측치 수정 : SQL때 처럼 이름의 호칭의 평균값으로 결측치를 채운다
   - https://cafe.daum.net/oracleoracle/Sg0x/969
3. 운임의 이상치를 제거 x
4. 결측치 치환
   a. 평균값
   b. 최빈값
   c. 회귀예측값 (나이를 종속변수, 나머지를 독립변수)