# INDEX

2021년 2월 22일 월요일 오전 9:55

# #0222

2021년 2월 8일 월요일 오전 9:28

**■ python full code**
**#spyder editor 에 복사해서 실습**
**#중간 진한 부분으로 다른 Part로 구분**
**#끝으로 갈수록 완성된 ML 코드 (모든 모델을 사용할 수 있도록 setting ongoing)**
**#진한 글씨로 Python editor 장 단위를 구분**

```
import pandas as pd
import seaborn as sns

df = pd.read_csv("c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩wisc_bc_data.csv")
#R과는 달리 stringsAsFactors=True 를 지정하지 않아도 된다.



# DataFrame 확인
print(df.shape)
#%%
print(df.info())  #데이터 구조 확인  #R의 str() 함수와 유사
#%%
print(df.describe())  #df의 요약통계정보 출력  #R의 summay(df) 결과와 유사



#%% 행을 선택하는 방법

#Pandas
#emp[행][열] emp[조건 [컬럼명] ]
#emp[행][열] emp[조건 c("ename","sal")]

#df.iloc() ---> iloc : index loc.
print(df.iloc[0:5, ])  #0~5번째 행까지 출력하시오 ~ df.iloc[행번호, 열번호]
print(df.iloc[-5: ,])  #끝에서 5번째 행부터 끝까지 출력하시오



#%% 열을 선택하는 방법
print(df.iloc[ :, [0,1] ])  #0~1번째 열 선택
print(df.iloc[ :, : ])  #전체열 다 선택

#판다스 데이터 프레인은 어떻게 구성되었는가?
#게시글 19번  #https://cafe.daum.net/oracleoracle/SgNT/19
#numpy 리스트(일반 리스트)로 컬럼 하나를 구성 : 시리즈
#numpy 리스트(일반 리스트)로 컬럼 여러개를 구성 : 데이터 프레임



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

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


#%% 데이터 정규화 수행
#1. 스케일링 : 평균은 0이고 표준편차가 1인 데이터로 분포시키는 방법
#2. min/max 정규화 : 0~1 사이의 숫자로 변경

#스케일링
from sklearn import preprocessing
X=preprocessing.StandardScaler().fit(X).transform(X)
print(X)


#훈련데이터와 테스트데이터를 분리하는 작업
from sklearn.model_selection import train_test_split  #import module
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)
#test_size=0.3 : 훈련과 테스트가 7:3 비율로 나누어진다
#random_state=10 은 seed 값 설정부분 ---> 동일난수열을 지정하여 결과값 관찰 가능

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


#%% 스케일링 결과 확인

#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

#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 훈련을 시행

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


#%% 모델 평가
```

```
from sklearn.metrics import confusion_matrix  #이원교차표(confusion_matrix) module import

conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)  #작은 이원교차표 출력


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


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



#%% Find Hyper Parameter k within knn model
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)

#print errors with indexnum(=k)
for k, i in enumerate(errors):
    print(k, '--->', i)

#Visualization
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()



#%% 문제1.
"""
위 코드에서 적절한 k 값을 알아내는 for 문을 구현하시오 (위 코드의 enumerate 구문 참고)
"""

#%% 문제2.
```

"""위에서 알아낸 에러가 가장 에러가 낮은 k 값은 7,8,9,10 이었다.
k값에 7일 넣었을때의 정확도를 구하시오"""

```
import pandas as pd
import seaborn as sns
df = pd.read_csv("c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩wisc_bc_data.csv")

X = df.iloc[:, 2:].to_numpy()
y = df['diagnosis'].to_numpy()

#스케일링
from sklearn import preprocessing
X=preprocessing.StandardScaler().fit(X).transform(X)
print(X)

#훈련데이터와 테스트데이터를 분리
from sklearn.model_selection import train_test_split  #import module
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)

#z-score 표준화 수행 결과 확인
for col in range(29):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(29):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')


# 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 훈련을 시행

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


# 모델 평가
from sklearn.metrics import confusion_matrix  #이원교차표(confusion_matrix) module import

conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)  #작은 이원교차표 출력


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

```
print(report)


# 정확도 확인
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)
"""
0.97의 정확도가 출력된다. 의료 데이터이기 때문에 아주 높은 정확도가 요구된다.
하지만 정확도가 100%가 나오긴 힘들기 때문에 FN 을 0으로 만들어 줄 것을 요구하는 경우가 많다.

FN은 1종오류에 해당하는 것으로 관심범주는 positive(암), Negative(정상) 일때,
 암환자를 정상환자로 잘못 예측하는 것을 의미한다.

방금 수행했던 시각화는 k값이 변경하는 것에 따른 오류값을 2차원 그래프로 시각화 한 것이다.
아래 문제에서는 k값이 변경될 때마다 FN 과 Accuracy 값을 출력해 보자
"""



#%% 문제3
#Accuracy, Error, FN value 그래프를 같이 시각화
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()
    # .ravel() 을 사용하지 않으면 작은 이원교차표가 나온다
    # .ravel() 을 사용하면 이원교차표의 값들을 출력할 수 있다.

    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 조정부분
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)

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

#Error
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')

#FN value
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 knn 으로 분류하시오!
#%% 문제5. iris 데이터에 대해서 가장 정확도가 좋은 k 값을 지정하시오!
#%% Part1.
##1. data 준비
#Import modules
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

#Add colnames
col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']

#csv 파일에서 DataFrame을 생성
df = pd.read_csv("c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩iris2.csv", encoding='UTF-8', header=None, names=col_names)
```

```
#print(dataset)

X = df.iloc[:, 0:4].to_numpy()
y = df['Class'].to_numpy()


#스케일링
from sklearn import preprocessing
X=preprocessing.StandardScaler().fit(X).transform(X)


#훈련데이터와 테스트데이터를 분리
from sklearn.model_selection import train_test_split  #import module
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)

#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()}')



#%% Part2
##2. 학습/예측(Training/Pradiction)
from sklearn.neighbors import KNeighborsClassifier

#k-NN 분류기를 생성
classifier = KNeighborsClassifier(n_neighbors=11)  #knn 모델 생성


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


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



#%% Part3
##3. 모델 평가
from sklearn.metrics import confusion_matrix  #이원교차표(confusion_matrix) module import
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)  #작은 이원교차표 출력


# 정밀도, 재현율, f1 score 확인
from sklearn.metrics import classification_report
```

```
report = classification_report(y_test, y_pred)
print(report)


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



#%% Part4
##4. Visualization

#Accuracy, Error, FN value 그래프를 같이 시각화
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 = []
f1_list = []
for i in range(1,30):  #range --> parameter k range
    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()
    # .ravel() 을 사용하지 않으면 작은 이원교차표가 나온다
    # .ravel() 을 사용하면 이원교차표의 값들을 출력할 수 있다.

    #F1 score set
    report=classification_report(y_test, y_pred, digits=2, output_dict=True)
    f1 = report['macro avg']['f1-score']
    f1_list.append(f1)

    #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}')  #3 by 3 coufusion matrix
    print(f"""k : {i},
          acc : {round(accuracy_score(y_test, y_pred),5)},
          err : {round(1-accuracy_score(y_test, y_pred),5)},
          f1-score : {round(f1,5)}""")
df_s = pd.DataFrame(data=dict(k=range(1,30), acc=acclist, err=err_list, F1_score=f1_list))
```

```
#그래프 사이즈 조정부분
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)

#Accuracy
plt.subplot(131)  #subplot(행, 열, index)
plt.plot(acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
plt.xlabel("k value")
plt.ylabel('Accuracy')

#Error
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')


#F1_score
plt.subplot(133)
plt.plot(f1_list, color='black', marker='o', markerfacecolor='gray')
plt.title('F1_score', size=15)
plt.xlabel("k value")
plt.ylabel('F1_score')


#FN value  #not use now
# 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.tight_layout()
plt.show()


#Select
print(df_s[['k','acc','err','F1_score']] [df_s['acc']==1] )


print(df_s)
###EndLine
```

**#%% 문제6. wisc_bc_data.csv 를 knn으로 분류하기 (정확도 상승을 위해 scaling 하지 말고 min/max 정규화를 수행하시오!)**

```
#%% Part1.
##1. Prepare Data
#Import modules
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


#Create DataFrame with .csv file
df = pd.read_csv("c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩wisc_bc_data.csv")


#Set point n_dimension : array
X = df.iloc[:, 2:].to_numpy()
y = df['diagnosis'].to_numpy()


# #Scaling
# from sklearn import preprocessing
# X=preprocessing.StandardScaler().fit(X).transform(X)


#Min/Max Normalization **
from sklearn import preprocessing
X=preprocessing.MinMaxScaler().fit(X).transform(X)
#보통 scaling 보다 min/max 정규화가 예측력이 더 좋은 결과를 보인다.


#Devide ---> Trainset:Testset = 7:3
from sklearn.model_selection import train_test_split  #import module
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)


#Check out Normalization | Scaling
for col in range(30):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

for col in range(30):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')
```

```
#%% Part2
##2. 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 훈련을 시행


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



#%% Part3
##3. 모델 평가
from sklearn.metrics import confusion_matrix  #이원교차표(confusion_matrix) module import
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)  #작은 이원교차표 출력


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


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



#%% Part4
##4. Visualization

#Accuracy, Error, FN value 그래프를 같이 시각화
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 = []
f1_list = []
for i in range(1,30):  #range --> parameter k range
    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)

    #F1 score set
    report=classification_report(y_test, y_pred, digits=2, output_dict=True)
    f1 = report['macro avg']['f1-score']
    f1_list.append(f1)

    #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}')  #3 by 3 coufusion matrix
    print(f"""k : {i},
          acc : {round(accuracy_score(y_test, y_pred),5)}
          err : {round(1-accuracy_score(y_test, y_pred),5)}
          FN : {fn}
          f1-score : {round(f1,5)}""")
df_s = pd.DataFrame(data=dict(k=range(1,30), acc=acclist, err=err_list,FN=fn_list, F1_score=f1_list))


#Mediate Graph size
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)

#Accuracy
plt.subplot(131)  #subplot(행, 열, index)
plt.plot(acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
plt.xlabel("k value")
plt.ylabel('Accuracy')

# #Error
# 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')
```

```
#F1_score
plt.subplot(133)
plt.plot(f1_list, color='black', marker='o', markerfacecolor='gray')
plt.title('F1_score', size=15)
plt.xlabel("k value")
plt.ylabel('F1_score')


#FN value
plt.subplot(132)
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.tight_layout()
plt.show()


#Select
#print(df_s[['k','acc','err','F1_score']] [df_s['acc']==1] )


print(df_s)
###EndLine
```

"""

**■NaiveBayes**

**R이 좋은 함수와 패키지가 파이썬보다 더 많다**

**현업에서는 파이썬으로 머신러닝을 구현하는 경우가 더 많다**

**앞선 knn R과 파이썬의 차이점 : Python 에서는 factor 변환이 필요없다**

"""

**#%% 문제7. 위 나이브 베이즈 모델의 성능을 더 올리시오!**

```
#기존 정확도 : 0.7333(베루누이 모델) ---> 1(가우시안)
#%%
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:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩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  #GaussianNB가 더 분류를 잘한다
#model = GaussianNB() # Gaussian Naive Bayes 모델 선택 - 연속형 자료

#model = BernoulliNB(alpha=0.1)
#model = BernoulliNB()  #acc : 0.73

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


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)

#%%
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)

# 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. 유방암 데이터를 나이브 베이즈로 분류하기(겹침)**
**#그래프 출력 코드까지 활용해서 시각화도 연습하시오!(laplace 설정값에 따른 그래프 변화 추가예정)**

```
#%% Part1.
##1. Prepare Data
#Import modules
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


#Create DataFrame with .csv file
#col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']  #sample
#df = pd.read_csv('c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩iris2.csv', encoding='UTF-8', header=None, names=col_names)  #sample
df = pd.read_csv("c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩wisc_bc_data.csv")


#Set point n_dimension : array
X = df.iloc[:, 2:].to_numpy()
y = df['diagnosis'].to_numpy()


# #Scaling
# from sklearn import preprocessing
# X=preprocessing.StandardScaler().fit(X).transform(X)
```

```
#Min/Max Normalization **
from sklearn import preprocessing
X=preprocessing.MinMaxScaler().fit(X).transform(X)
#Accuracy ---> scaling < min/max normalization


#Devide ---> Trainset:Testset = 7:3
from sklearn.model_selection import train_test_split  #import module
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.3, random_state = 10)


#Check out Normalization | Scaling
for col in range(30):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

print('')  #for your eye

for col in range(30):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')




#%% Part2
##2. Training/Pradiction
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB  #GaussianNB is better

#BernoulliNB sample
#model = BernoulliNB(alpha=0.1)
#model = BernoulliNB()  #acc :

#GaussianNB
#model = GaussianNB(var_smoothing=1e-03)
model = GaussianNB()  #acc :

#Make model
model.fit( X_train, y_train )

#Prediction
y_pred= model.predict(X_test)
print(y_pred)  #check



#%% Part3
##3. Evaluation model
from sklearn.metrics import confusion_matrix  #module import

#Counfusion matrix(no column)
```

```
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)
#TP FP
#FN TN

#Precision, Recall, F1_score, Accuracy
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)


#Accuracy Checkout & Details
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)



#%% Part4
##4. Visualization

#Accuracy, Error, FN value, F1 score
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 = []
f1_list = []
for i in range(1,30):  #range --> parameter laplace range
    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)

    #F1 score set
    report=classification_report(y_test, y_pred, digits=2, output_dict=True)
    f1 = report['macro avg']['f1-score']
    f1_list.append(f1)

    #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}')  #3 by 3 coufusion matrix
```

```
    print(f"""k : {i},
          acc : {round(accuracy_score(y_test, y_pred),5)}
          err : {round(1-accuracy_score(y_test, y_pred),5)}
          FN : {fn}
          f1-score : {round(f1,5)}""")
df_s = pd.DataFrame(data=dict(k=range(1,30), acc=acclist, err=err_list,FN=fn_list, F1_score=f1_list))


#Mediate Graph size
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)

#Accuracy
plt.subplot(131)  #subplot(행, 열, index)
plt.plot(acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
plt.xlabel("k value")
plt.ylabel('Accuracy')

# #Error
# 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')


#F1_score
plt.subplot(133)
plt.plot(f1_list, color='black', marker='o', markerfacecolor='gray')
plt.title('F1_score', size=15)
plt.xlabel("k value")
plt.ylabel('F1_score')


#FN value
plt.subplot(132)
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.tight_layout()
plt.show()
```

```
#Select
#print(df_s[['k','acc','err','F1_score']] [df_s['acc']==1] )


print(df_s)
###EndLine
```

**#%%문제 10. NaiveBayes : wine.csv**

**#문제 11. KNN : wine.csv**

```
#%% Part1.1
##1. Prepare Data
#Import modules
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


#Create DataFrame with .csv file
#col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']  #sample
#df = pd.read_csv('c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩iris2.csv', encoding='UTF-8', header=None, names=col_names)  #sample
df = pd.read_csv("c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩wine.csv")


#Check out
print(df.shape)  #(178,14)
print(df.info)  #Label col. = Type
print(df.describe())  #summary
print(df)


#%% Part1.2
#Set point n_dimension : array
X = df.iloc[:, 1:].to_numpy()
y = df['Type'].to_numpy()
print(df.shape)  #(178,14)


# #Scaling
# from sklearn import preprocessing
# X=preprocessing.StandardScaler().fit(X).transform(X)
```

```
#Min/Max Normalization **
from sklearn import preprocessing
X=preprocessing.MinMaxScaler().fit(X).transform(X)
#Accuracy ---> scaling < min/max normalization


#Devide ---> Trainset:Testset = 9:1
from sklearn.model_selection import train_test_split  #import module
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.1, random_state = 10)


#Check out Normalization | Scaling
for col in range(13):
    print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')

print('')  #for your eye

for col in range(13):
    print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')




#%% Part2
##2.1 Training/Pradiction : NaiveBayes
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB  #GaussianNB is better

#BernoulliNB sample
#model = BernoulliNB(alpha=0.1)
#model = BernoulliNB()  #acc :

#GaussianNB
#model = GaussianNB(var_smoothing=1e-03)
model = GaussianNB()  #acc :

#Make model
model.fit( X_train, y_train )

#Prediction
y_pred= model.predict(X_test)
print(y_pred)  #check


#%%
##2.2 Training/Pradiction : KNN
from sklearn.neighbors import KNeighborsClassifier

#Making k-NN classifier
classifier = KNeighborsClassifier(n_neighbors=15)
```

```
#Modeling
classifier.fit(X_train, y_train)

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



#%% Part3
##3. Evaluation model
from sklearn.metrics import confusion_matrix  #module import

#Counfusion matrix(no column)
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)
#TP FP
#FN TN

#Precision, Recall, F1_score, Accuracy
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)


#Accuracy Checkout & Details
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)



#%% Part4
##4. Visualization

#Accuracy, Error, FN value, F1 score
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 = []
f1_list = []
for i in range(1,30):  #range --> parameter laplace range
    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)

    #F1 score set
    report=classification_report(y_test, y_pred, digits=2, output_dict=True)
    f1 = report['macro avg']['f1-score']
    f1_list.append(f1)

    #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}')  #3 by 3 coufusion matrix
    # print(f"""k : {i},
    #       acc : {round(accuracy_score(y_test, y_pred),5)}
    #       err : {round(1-accuracy_score(y_test, y_pred),5)}
    #       FN : {fn}
    #       f1-score : {round(f1,5)}""")
#df_s = pd.DataFrame(data=dict(k=range(1,30), acc=acclist, err=err_list,FN=fn_list, F1_score=f1_list))  #sample
df_s = pd.DataFrame(data=dict(k=range(1,30), acc=acclist, err=err_list, F1_score=f1_list))

#Mediate Graph size
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)

#Accuracy
plt.subplot(131)  #subplot(행, 열, index)
plt.plot(acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
plt.xlabel("k value")
plt.ylabel('Accuracy')

# #Error
# 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')


#F1_score
plt.subplot(133)
plt.plot(f1_list, color='black', marker='o', markerfacecolor='gray')
plt.title('F1_score', size=15)
```

```
plt.xlabel("k value")
plt.ylabel('F1_score')


# #FN value
# plt.subplot(132)
# 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.tight_layout()
# plt.show()


#Select
#print(df_s[['k','acc','err','F1_score']] [df_s['acc']==1] )


print(df_s)
###EndLine
```

**##파이썬 나이브 베이즈 사이킷런 함수 3가지**

**#1. BernoulliNB : 이산형 데이터를 분류할 때 적합**
**#2. GaussianNB : 연속형 데이터를 분류할 때 적합**
**#3. MultinomialNB**
**#url : https://m.blog.naver.com/PostView.nhn?blogId=dairum_enc&logNo=221409597367&proxyReferer=https:%2F%2Fwww.google.com%2F**

**#%% 문제12. Mushrooms.csv**
**#문제13. 정확도를 0.99로 만드는 laplace 값을 알아내시오! (graph도)**
**"""**
**#Python에서는 R과는 달리 위 csv 파일을 NaiveBayes 나 KNN 으로 분류하기 위해선**
**모든 컬럼이 명목형 데이터이기 때문에 0과 1로 치환하여 작업을 수행해야 한다.**
**#관련 url : https://github.com/MrDavidYu/Poisonous_Mushrooms_Classification/blob/master/Mushroom_ML_NB.ipynb**
**"""**

```
#%% Part1.1
```

```
##1. Prepare Data
#Import modules
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


#Create DataFrame with .csv file
#col_names = ['sepal-length', 'sepal-width','petal-length', 'petal-width','Class']  #sample
#df = pd.read_csv('c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩iris2.csv', encoding='UTF-8', header=None, names=col_names)  #sample
df = pd.read_csv("c:₩₩Users₩₩icecr₩₩OneDrive₩CSV_file₩₩mushrooms.csv")
df = pd.get_dummies(df, drop_first=True)  #drop_first help modeling better**


#Check out
print(df)
print(df.shape)  #(8124,96)
print(df.info)  #Label col. = Type
print(df.describe())  #summary


#%% Part1.2
#Set point n_dimension : array
X = df.iloc[:, 1:].to_numpy()  #Independent variable
y = df.iloc[:, 0].to_numpy()  #Dependent variable
print(df.shape)  #(8124,96)


# #Scaling
# from sklearn import preprocessing
# X=preprocessing.StandardScaler().fit(X).transform(X)


# #Min/Max Normalization  #Whole X variables are Nominal variables
# from sklearn import preprocessing
# X=preprocessing.MinMaxScaler().fit(X).transform(X)
# #Accuracy ---> scaling < min/max normalization


#Devide ---> Trainset:Testset = 75:25
from sklearn.model_selection import train_test_split  #import module
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.25, random_state = 10)


# #Check out Normalization | Scaling
# for col in range(22):
#     print(f'평균 = {X_train[:, col].mean()}, 표준편차= {X_train[:, col].std()}')
```

```
# print('')  #for your eye

# for col in range(22):
#     print(f'평균 = {X_test[:, col].mean()}, 표준편차= {X_test[:, col].std()}')



#%% Part2
##2.1 Training/Pradiction : NaiveBayes
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB  #GaussianNB is better
from sklearn.naive_bayes import MultinomialNB

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

#GaussianNB
#model = GaussianNB(var_smoothing=1e-04)
model = GaussianNB(var_smoothing=0.004)  #acc :

#Multinomial
#model = MultinomialNB()


#Make model
model.fit( X_train, y_train )

#Prediction
y_pred= model.predict(X_test)
print(y_pred)  #check


#%%
##2.2 Training/Pradiction : KNN
from sklearn.neighbors import KNeighborsClassifier

#Making k-NN classifier
classifier = KNeighborsClassifier(n_neighbors=11)

#Modeling
classifier.fit(X_train, y_train)

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



#%% Part3
```

```
##3. Evaluation model

#Counfusion matrix(no column)
from sklearn.metrics import confusion_matrix  #module import
conf_matrix= confusion_matrix(y_test, y_pred)
print(conf_matrix)
#TP FP
#FN TN

#Precision, Recall, F1_score, Accuracy
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)


#Accuracy Checkout & Details
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_pred)
print(accuracy)




#%% Part4
##4. Visualization

#Accuracy, Error, FN value, F1 score
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


#Bins
acclist = []
err_list = []
fn_list = []
f1_list = []


## NB
for i in np.arange(0.001, 0.031, 0.001):
    #F1 score set
    report=classification_report(y_test, y_pred, digits=2, output_dict=True)
    f1 = report['macro avg']['f1-score']
    f1_list.append(f1)

    acclist.append(accuracy_score(y_test, y_pred))
    err_list.append(np.mean(y_pred != y_test))
```

```
df_s = pd.DataFrame(data=dict(Laplace=np.arange(0.001, 0.031, 0.001), acc=acclist, err=err_list, F1_score=f1_list))


## KNN
# for i in range(1,30):  #range --> parameter laplace range
#     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)

#     #F1 score set
#     report=classification_report(y_test, y_pred, digits=2, output_dict=True)
#     f1 = report['macro avg']['f1-score']
#     f1_list.append(f1)

#     #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}')  #3 by 3 coufusion matrix
#     print(f"""k : {i},
#           acc : {round(accuracy_score(y_test, y_pred),5)}
#           err : {round(1-accuracy_score(y_test, y_pred),5)}
#           FN : {fn}
#           f1-score : {round(f1,5)}""")


#df_s = pd.DataFrame(data=dict(k=range(1,30), acc=acclist, err=err_list,FN=fn_list, F1_score=f1_list))  #sample
#df_s = pd.DataFrame(data=dict(k=range(1,30), acc=acclist, err=err_list, F1_score=f1_list))



#Mediate Graph size
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)

#Accuracy
plt.subplot(131)  #subplot(행, 열, index)
plt.plot(acclist,color='blue', marker='o', markerfacecolor='red')
plt.title('Accuracy', size=15)
plt.xlabel("k value")
plt.ylabel('Accuracy')

#Error
plt.subplot(132)
```

```
plt.plot(err_list, color='green', marker='o', markerfacecolor='yellow')
plt.title('Error', size=15)
plt.xlabel("k value")
plt.ylabel('error')

#F1_score
plt.subplot(133)
plt.plot(f1_list, color='black', marker='o', markerfacecolor='gray')
plt.title('F1_score', size=15)
plt.xlabel("k value")
plt.ylabel('F1_score')

# #FN value
# plt.subplot(132)
# 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.tight_layout()
plt.show()


#Select
#print(df_s[['k','acc','err','F1_score']] [df_s['acc']==1] )



###EndLine
```

#EndLine==================

# #0223

2021년 2월 8일 월요일 오전 9:28

## ■3장. 의사결정트리

의사결정트리 ---> 랜덤포레스트(앙상블+의사결정트리)

#0223 --> jupyter 에 필기

#EndLine==================

# #0224

2021년 2월 8일 월요일 오전 9:28

## ■Review

1. 파이썬으로 knn 구현
2. 파이썬으로 naivebayes 구현
3. 파이썬으로 decision tree 구현
4. 파이썬으로 regression구현 ( Simple, Multi)

# ■오늘 진도

Multi regression
SVM
#0223 --> jupyter notebook 에 필기

**머신러닝 데이터 분석 5가지 단계**

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

**[Index for ML database analizing]**
Step1. Load DB & Details
Step2. Analizing DB, Visualization
Step3. Training : Create model
Step4. Evaluation : Predict model
Step5. Performance Improvement

**성능개선 방법 : 단순회귀 --> 다항회귀로 변경해서 성능을 올린다.**

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

Simple_Regression_practice_weight.csv

| 단축키 | 기능 |
|---|---|
| ESC | 코드모드에서 명령모드로 전환 |
| enter | 명령모드에서 코드모드로 전환 |
| m | 코드셀에서 마크다운셀로 변경 |
| y | 마크다운셀에서 코드셀로 변경 |
| a | 현재 셀 위로 셀추가 |
| b | 현재 셀 아래로 셀추가 |
| dd | 셀삭제 |
| ctrl + shift + - | 셀나누기 |
| shift + m | 셀합치기 |
| shift + 엔터 | 셀 실행 후 다음 셀로 이동 |
| ctrl + 엔터 | 셀 실행 후 셀에 계속 머물기 |
| s | 저장 |

## ■6. Mulit Regression (다중회귀)

종속변수에 영향을 주는 독립변수가 여러개인 경우의 회귀분석 방법

**R ---> Python 으로 실행**

예제1. 미국 우주 왕복선 폭파원인

예제2. 미국 대학교 입학점수에 영향을 미치는 과목 분석

예제3. 미국 국민 의료비에 영향을 주는 요소 분석

#EndLine==================

# #0225

2021년 2월 8일 월요일 오전 9:28

# ■공지

**1. 3월 7일 24:00 까지 kaggle 순위 제출 (일정변경)**

**R을 활용한 머신러닝부터 지금까지 복습**
Q. 머신러닝을 이용해서 데이터에서 무엇을 발견하고 하는가?
A. 정확도가 가장 높고 오차가 극도로 낮은 기계학습 모델을 생성하려고만 하는것은 아니다.

리눅스와 하둡 포트폴리오처럼 data 에서 정보를 얻어내기 위해서 ML 을 활용하는 것이다.
데이터에서 유용한 정보를 얻어내고, 예측을 하기 위해서 ML 을 하는 것이다.

리눅스와 하둡 ---> 데이터 저장 공간 구성 및 관리소
SQL, python, R ---> 데이터를 읽고 정보를 추출하여 시각화 및 결론 도출

**데이터에서 유용한 정보를 발견한 예시**
1. 회귀분석 : insuarece.csv(미국 의료비 데이터)
- 비만인 사람이 흡연까지 하면 의료비가 더 증가 : bmi30_smokeryes 파생변수를 생성
- 위 파생변수를 추가하여 모델을 생성 및 예측하여 결정계수 및 정확도 향상

2. 분류모델 생성 : titanic.csv(타이타닉 데이터)
- 여자와 아이의 생존률이 더 높았음
- random forest 로 추가 정확도 향상을 보았음

즉, 모델의 파라미터를 조정하여 예측력을 상승시키는 것도 중요하지만 모델을 고객에게 납득시키는 것 또한 중요
위와 같은 논리적으로 이해할 수 있는 파생변수의 생성 및 시각화를 이용하면 된다.

# ■7. 로지스틱 회귀

종속변수가 범주형인 경우에 적용되는 회귀분석 모형(수제비 p3-8) ---> 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 회귀분석 : 종속변수가 범주형인 데이터

오늘 진도는 로지스틱 회귀분석이고, seaborn 내장된 타이타닉 데이터를 이용해서 분석
SQL 로는 상위 11% 에 들었음 ---> SQL 안써.. Python 으로 도전!

seaborn 내장 데이터보다 titanic 실제 데이터에는 결측치와 이상치가 더 많다.
오늘 실습의 주요 내용은 데이터 전처리에 대한 부분이다.

**질문**

1. 이 데이터에 맞는 가장 좋은 머신러닝 알고리즘과 코드는 무엇인지?
2. 이 데이터에 맞는 가장 좋은 파생변수는 무엇인지?

## 머신러닝 데이터 분석을 하기 위한 단계**

1. 데이터 불러오기
   <---- 파생변수 추가 (women_child)
2. 데이터 탐색 및 전처리 : 결측치(NAN) 처리 : age, embark_town, embark
3. 데이터 정규화 또는 표준화 : 훈련되는 모델이 이상치에 덜 민감해지고 단위를 일정하게 조정하기 위해 수행
4. 범주형 데이터에 대한 더미변수 생성 (Python ---> R에서는 수행 X)
5. 훈련데이터와 테스트 데이터로 분할
6. 머신러닝 모델 생성
7. 훈련데이터로 머신러닝 모델 훈련
8. 머신러닝 모델 평가
9. 머신러닝 모델 성능 개선

**결측치에 대한 설명**

머신러닝 데이터 분석의 정확도는 분석 데이터의 품질에 의해 좌우된다.
데이터 품질을 높이기 위해서는 누락 데이터 처리, 중복 데이터 처리 등 오류를 수정하고 분석 및 목적에 맞게 변형하는 과정이 필요하다.
데이터 프레임에는 원소 데이터 값이 종종 누락되는 경우가 있다.
일반적으로 데이터 값이 존재하지 않는 누락 데이터를 NaN 으로 표기한다.
머신러닝 분석 모형에 데이터를 입력하기 전에 반드시 누락 데이터를 제거하거나 다른 적절한 값으로 대체하는 과정이 필요하다. **
누락 데이터가 많아지면 데이터의 품질이 저하되고 머신러닝 분석 알고리즘을 왜곡하는 현상이 발생하기 때문이다.
따라서 아래의 3가지 함수를 잘 알아야 한다.

## NaN control function

1. 누락 데이터를 찾는 함수
   1.1) isnull() : 누락데이터이면 True 를, 아니면 False 를 반환하는 함수
   1.2) notnull() : 누락데이터이면 False 를, 아니면 True 를 반환하는 함수

ex)

```
import seaborn as sns
tat = sns.load_dataset('titanic')
print(tat.isnull(), end='₩n₩n')
print(tat.isnull().sum())
```

**문제29. emp 데이터 프레임의 누락 데이터는 몇개인가?**

```
import pandas as pd
emp = pd.read_csv("c:₩₩Users₩₩82103₩₩OneDrive₩CSV_file₩emp3.csv")
print(emp.isnull(), end='₩n₩n')
print(emp.isnull().sum(), end='₩n₩n')
```

```
print(sum(emp.isnull().sum()))  #11
```

2. 누락 데이터를 제거하는 함수

2.1) 열을 삭제

2.2) 행을 삭제

**문제30. emp 데이터 프레임에서 comm 의 결측치를 확인하고 comm 의 결측치 행들을 삭제하시오!**

```
import pandas as pd
emp = pd.read_csv("c:₩₩Users₩₩82103₩₩OneDrive₩CSV_file₩emp3.csv")
print(emp.isnull(), end='₩n₩n')
print(emp.isnull().sum(), end='₩n₩n')
print(sum(emp.isnull().sum()))  #11
```

3. 누락 데이터를 다른 데이터로 치환하는 함수**

3.1) 평균값으로 누락 데이터를 바꾸기

3.2) 최빈값으로 누락 데이터를 바꾸기

3.3) 이웃하고 있는 주변의 인접데이터로 누락 데이터를 바꾸기

3.4) 누락 데이터가 아닌 데이터에 대한 회귀 예측값으로 누락 데이터를 바꾸기 : 이 경우가 가장 좋은 예측력이 좋은 모델생성가능

실 코드 3.1)

**성능 개선을 위해 데이터를 시각화해서 데이터에서 스토리를 뽑아내는 방법**

**범주형 데이터를 처리하는 방법**

1. 더미변수로 처리 : 숫자 1과 0으로 표현

- 여자 또는 아이이면 1이고 else 0 으로 표현
- (파생변수 추가)

2. 구간 분할로 처리 : 일정한 구간으로 나누는 것(연속형 수치데이터)

- 미세먼지의 농도는 연속형 수치형 데이터여서 누군가가 나에게 오늘 미세먼지 농도가 어떠냐? 하고 물어보았을때
  2.345㎍/m³ 이라고 하면 알아들을 수 없다. 때문에 보통 '좋음', '나쁨', '매우좋음' 등으로 표현하곤 한다.
  이는 연속형 수치데이터인 미세먼지의 농도를 구간으로 분할되어 말하는 것인데 이를 ML 에도 적용한다.
- 타이타닉 데이터 또한 운임을 그냥 날렸지만 구간분할하여 파생변수로 처리해주면 예측력이 올라간다.

**문제 37. 여자와 아이데 대한 파생변수를 추가해서 0.85까지 올렸던 코드를**
**logistic regression 이 아니라 앙상블 기법이 추가된 rf 로 수행해서 정확도가 더 올라가는지  확인하시오!**

**문제 38. 로지스틱 회귀로 시본의 타이타닉 분류 모델을 만드는데 여자와 아이 파생변수 말고 아래의 그래프를 보고 더 좋은 파생변수를 생각해서 학습 시키고 정확도 출력되는 화면을 올리시오~**

#EndLine==================

# #0302

2021년 2월 8일 월요일 오전 9:28

## ■ 7장. 신경망

사람의 뇌 ----------> 컴퓨터를 이용한 지능처리

↓ ↓

생물학적 신경망내에서 인공신경망에서는 가중치라는 것으로

반복적인 시그널이 발생할 때 기억의 효과를 대체할 수 있음을 설명했다.

그 시그널을 기억하는 일종의

학습효과가 있다.

※ 파라미터와 하이퍼 파라미터의 차이 (수제비 3-12)

1. 파라미터?

모델 내부에서 확인이 가능한 변수로 데이터를 통해서 산출이 가능한 값 예측을 수행할 때, 모델에 의해 요구되어지는 값들. 사람에 의해 수작업으로 측정되지 않음

예: 신경망에서의 가중치, 회귀분석세어의 결정계수, 서포트 벡터머신에서의 서포트벡터

2. 하이퍼 파라미터?

모델에서 외적인 요소로 데이터 분석을 통해 얻어지는 값이 아니라 사용자가 직접 설정해주는 값. 모델의 파라미터값을 측정하기 위해 알고리즘 구현과정에서 사용. 하이퍼 파라미터는 주로 알고리즘 사용자에 의해 결정됨

예: 신경망에서의 학습률(러닝레이트), knn에서의 k값, 의사결정트리에서의 나무의 깊이. 서포트벡터머신에서의 C값과 gamma값

최적의 하이퍼파라미터를 알아내는 기법

r > caret 패키지

python > grid search 기법

* 퍼셉트론? 뇌의 신경세포 하나를 컴퓨터로 흉내낸 것

반복학습 (중요도)

↓

1. 뇌: 신호 ----------> 암기

가중치

↓ 1. 신호가 흐른다.

2. 인공신경망: 신호 ----------> 0. 신호가 흐르지 않는다.

컴퓨터 뇌

↓ ↓

정보를 메모리의 특정위치에 저장 뇌는 정보를 저장하는 공간이 따로 없다.

↓

신경세포(뉴런)의 연결관계를 변경하는 방식으로 정보를 저장

R: neuralnet 함수: 콘크리트의 강도를 예측

파이썬: 사이킷런의 neural_network 함수: 콘크리트의 강도를 예측

1. 예측: 콘크리트 데이터

2. 분류: 타이타닉 데이터

**문제 39. concrete.csv 를 신경망으로 정확도를 예측하시오!**
신경망 코드_concreate.ipynb

**문제 40. 위 신경망 성능을 더 올리시오**
하이퍼 파라미터를 자동으로 알아내게 하는 파이썬의 기능인 grid search 를 이용

**[hint code]**

```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV

param_grid = [
        {
            'activation' : ['identity', 'logistic', 'tanh', 'relu'],
            'solver' : ['lbfgs', 'sgd', 'adam'],
            'hidden_layer_sizes': [
             (1,),(2,),(3,),(4,),(5,),(6,),(7,),(8,),(9,),(10,),(11,), (12,),(13,),(14,),(15,),(16,),(17,),(18,),(19,),(20,),(21,)
             ],
            'random_state' : [0,1,2,3,4,5,6,7,8,9,10]
        }
       ]

clf = GridSearchCV(MLPRegressor(), param_grid, cv=3, n_jobs = -1, verbose = 2 )

clf.fit(X_train, y_train)


print("Best parameters set found on development set:")
print(clf.best_params_)
```

**41. random_state 도 GridSearchCV 에 포함시키려면?**
param_grid=[] 에 'random_state' : [0,1,2,3,4,5,6,7,8,9,10] 추가
이렇게 하면 computation cost 가 올라간다(소요시간 ↑)

**42. grid search 를 이용하지 말고 보스톤 하우징 데이터의 수치 예측을 하는 신경망 코드를 작성하시오! (boston.csv)**
ANN_boston_csv

**43. 문제 42번을 GridSearchCV 를 이용해서 cor 을 향상시키는 파라미터를 제시하고 cor 값을 명시하시오!**

**●model 결과 세분값 출력 설정!**

```
from sklearn import set_config
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(criterion="entropy", max_depth = 4)

set_config(print_changed_only=True)
clf
# DecisionTreeClassifier(criterion='entropy', max_depth=4)

set_config(print_changed_only=False)
clf
# DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
#                        max_depth=4, max_features=None, max_leaf_nodes=None,
#                        min_impurity_decrease=0.0, min_impurity_split=None,
#                        min_samples_leaf=1, min_samples_split=2,
#                        min_weight_fraction_leaf=0.0, presort='deprecated',
#                        random_state=None, splitter='best')
```

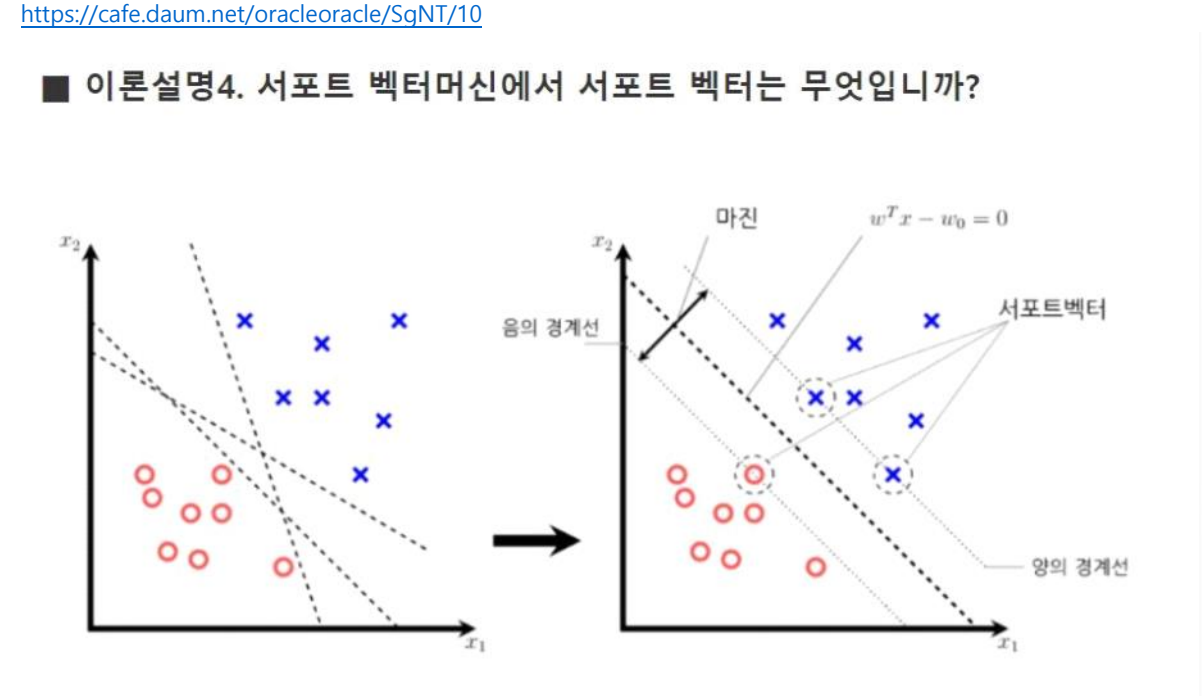
**## 분류기 세부사항 출력 옵션 설정(간략한 정리)**

```
from sklearn import set_config

#set_config(print_changed_only=True)  # 생략
set_config(print_changed_only=False)  # 다 출력
```

# ■SVM

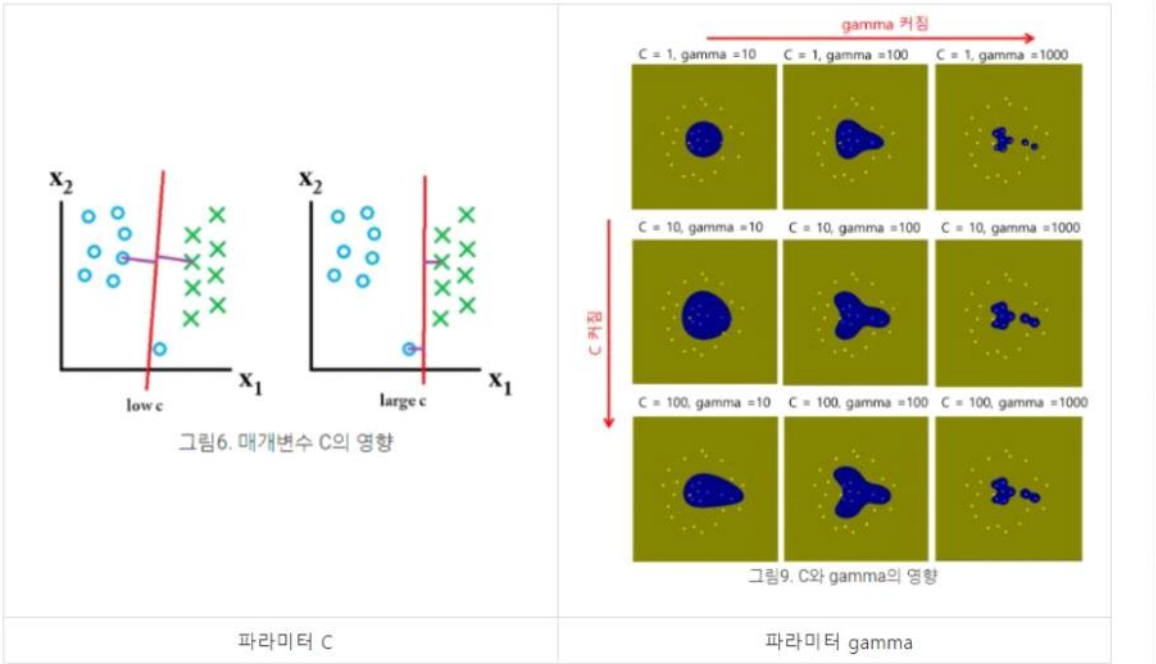
https://cafe.daum.net/oracleoracle/SqNT/10




```
# parameter : support vector
```

# hyper parameter : C, gamma

**C, gamma**




**SVM**

서포트 벡터 머신은 기계학습 분야의 하나로 패턴인식, 자료분석을 위한 지도학습 모델이며 주로 분류와 회귀분석을 위해 사용.
서포트 벡터 머신 알고리즘은 데이터를 분류하는 가장 큰 폭을 가진 경계를 찾는 알고리즘이다.
비선형 분류를 하기 위해서는 주어진 데이터를 고차원 특징 공간으로 사상하는 작업이 필요한데, 이를 효율적으로 하기 위해 커널 트릭을 사용하기도 한다.

**예제**

seborn 의 타이타닉 데이터를 분류하는 서포트 벡터 머신 모델 코드
file name : SVM_titanic_csv

**44) 지금 현재의 정확도는 0.81 인데, 아이와 여자먼저라는 파생변수를 추가하고 정확도를 확인하시오!**

file : SVM_titanic_csv_파생변수추가

**45) gridsearch 를 이용해서 최적의 하이퍼 파라미터를 알아내시오!**

hyper parameters : C, gamma + kenel

**GridSearch options**

estimator : classifier, regressor, pipeline이 사용될 수 있다.
param_grid : 파라미터 딕셔너리. (파라미터명과 사용될 여러 파라미터 값을 지정)
scoring : 예측 성능을 측정할 평가 방법. 보통은 사이킷런에서 제공하는 문자열
(예: 'accuracy')을 넣지만 별도의 함수도 직접 지정이 가능하다
cv : 교차 검증을 위해 분할되는 폴드 수.
refit : True면 가장 최적의 하이퍼 파라미터를 찾은 뒤 입력된 estimator 객체를 해당 하이퍼 파라미터로 재학습시킨다. (default:True)

**46) titanic.csv를 SVM 으로 최고로 올릴 수 있는 타이타닉 데이터의 정확도는 0.83 이었다. 오전에 배웠던 girdSearch 를 이용한 신경망 코드를 이용해서 수치예측이 아닌 분류로 머신러닝 함수를 변경해서 훈련시키고 정확도를 확인하시오!**

from sklearn.neural_network import MLPRegressor ---> 수치예측

from sklearn.neural_network import MLPClassifier---> 분류

file : ANN_titanic_csv_파생변수추가(선생님코드)

**정확도를 올리기 위해서 실험해야 할 내용****

1. 신경망을 2층 ---> 3층으로 변경
2. 나이의 결측치 row 177 개를 삭제하지 않고 치환( mean, mod, 이름의 호칭의 평균값(seaborn 에는 없음) )
3. 나이와 움임의 이상치를 제거하고 학습

# ■비지도학습1 : 연관규칙

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

**연관규칙**

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

**예제**

module install : mlxtend

conda install mlxtend  <--- anaconda prompt 창을 관리자 권한으로 실행한 뒤 module import

pip install mlxtend  #위 방법이 안되면 pip 로 실행

conda Vs. pip

pip 는 python에 직접 설치(더 세분화된 tool 에 설치하는 개념)

conda 는 anaconda 전반에 걸쳐 설치

**공식참고**

●●

**지지도 : support(X) = n(X) / N = p(X)**

**신뢰도 : confidence( X ---> Y ) = p(Y|X) = n(X∩Y) / n(X) = p(X∩Y) / p(X)**

**향상도 : lift( X ---> Y ) = confidence(X--->Y) / support(Y)** ---> R p.391 을 참고해서 식을 혼동하지 않도록 하자!

**[code] print 및 주석은 file : "연관규칙.pylnb" 를 참고**

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder  # 연관성 분석을 위한 데이터 전처리 module
from mlxtend.frequent_patterns import apriori  # 연관분석을 위한 apriori Algorithm 함수

dataset=[
```

```
['사과','치즈','생수'],  #transaction1
['생수','호두','치즈','고등어'],  #transaction2
['수박','사과','생수'],  #transaction3
['생수','호두','치즈','옥수수']]  #transaction4

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 )

from mlxtend.frequent_patterns import association_rules
print( association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3) )
```

**문제 47. 아래의 데이터에서 연관성을 도출하시오!**

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

file : 연관규칙-문제47

**48. R ML수업 때 실습했던 보습학원 데이터로 연관분석을 하시오!**
**(보습학원이 있는 건물에는 어떤 업종이 많이 있는가?)**

# ■비지도학습2 : k-means

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

#EndLine==================

# #0303

2021년 3월 1일 월요일 오후 4:52

# ■10장. kmeans 알고리즘

[수제비 3-60]
주어진 데이터를 k개의 군집으로 묶는 알고리즘으로 k 개 만큼 군집수를 초기값으로 지정하고, 각 개체를 가까운 초기값에 할당하여 군집을 형성하고 각 군집의 평균을 재계산하여 초기값을 갱신하는 과정을 반복하여 k개의 최종군집을 형성한다.

**군집 절차 순서**
1. k개 객체 선택
2. 할당(Assingment)
3. 중심갱신
4. 반복

**k-means 의 hyper parameter**
k값

**비지도학습**
비지도학습은 입력 데이터에 대한 정답인 레이블(label)이 없는 상태에서 데이터가 어떻게 구성되었는지 알아내는 기계학습 방법
1) k-means
2) som (자기 조직화 지도) : 인공신경망을 활용한 비지도학습
   (Self Organization Map)

## □k-means 를 파이썬으로 구현하는 예제

file : k-means 파이썬 예제

**문제 49. 아래의 데이터 프레임으로 k-means를 구성하고 분석하시오!**
file : k-means 문제49

**문제50. UCI 데이터 중 식품에 관련한 데이터를 k-means 알고리즘으로 분류하시오!**
(k=5 를 주어 5개의 군집으로 식품을 분류)

**문제51. wisc_bc_data.csv 를 k-means 알고리즘으로 분류하시오!**
file : k-means_wisc

**문제52. 아래의 사이트를 참고해서 iris 데이터를 사이킷런의som 패키지로 군집화하는 실습을 수행하시오!**
1. 아나콘다 프롬프트 창을 열고 pip install sklearn-som 실행
2. 아래의 사이트의 예제를 수행하시오
   url : https://pypi.org/project/sklearn-som/

## □Random Forest

**수제비 3-97**

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

**예제1. iris 데이터를 의사결정트리로 분류하는 실습 + gridsearch 기능 추가**

file : DescionTree_gridsearch_iris

**예제2. iris 데이터를 랜덤포레스트로 분류하는 실습 + gridsearch 기능 추가**

file : RandomForest_gridsearch_iris

**예제3. 시본의  데이터로 랜덤포레스트 모델 생성하기**

file :  RandomForest_gridsearch_titanic_seaborn

**문제53. 예제 3번 시본의 타이타닉 랜덤포레스트 모델의 oob 점수는 0.74, accuracy 는 0.80 이었다.**
**예제 2번에서 사용한 greedSearch 코드를 추가하여 해당 모델의 성능을 개선하시오!**

file :  RandomForest_gridsearch_titanic_seaborn  ---> #2 인덱스 참고

위 파일에는 kaggle 제출하는 csv 파일 생성코드도 포함

**Kaggle : titanic competition Recommend**

1. 신경망 모델 + grid search
2. 나이의 결특치를 SQL 때처럼 호칭의 평균값으로 치환
3. 랜덤포레스트 모델 + grid search
4. 앙상블 : xgboost 모델 + grid search

**문제 55. 신경망 + gridsearch 로 kaggle 순위 도전 ---> R 로도 도전 (0307 24:00 pm 까지)**

file :

#####endLine=====