1. KNN의 작동원리 이해 및 구현하기

1-1. 데이터 만들기

- data_loader()를 통해서 데이터를 불러오세요
- 4번째 column까지만 사용할겁니다. 4번째 column까지의 값을 list로 바꾸어 knn_data를 만드세요
  (hint : tolist(), iloc)

```
def data_loader():
    from sklearn.datasets import load_iris
    data = pd.DataFrame(load_iris()['data'])
    data.columns = ["sepal_length","sepal_width","petal_length","petal_
width"]
    target_name = list(load_iris().target)
    data['species'] = target_name
    return data
```

1-2. knn 함수 만들기 (1)

- euclidian_distance를 계산하는 함수를 만드세요
- 인자값은 x,y 이며 x,y는 모두 list입니다.
  (hint : math라는 라이브러리에 sqrt를 사용하시면 루트 씌우실 수 있어요)
- 만든 함수를 사용해서 [1,1,1], [2,3,4]의 euclidian distance의 값을 보여주세요

1-3. knn 함수 만들기 (2)

- knn을 구하는 함수를 만드세요
- 인자값은 total, target, n 입니다.
  (total : 전체 데이터, target : knn을 구하고 싶은 데이터, n : 이웃 수)
- return 값으로 가장 가까운 순서대로 index 값들을 도출하시면 됩니다.
- 최종적으로 아래와 같은 결과가 도출되면 성공입니다.

```
def takeSecond(elem):
    return elem[1]
## 튜플을 값으로 갖는 리스트일 때, 튜플의 인자값을 sort의 key로 사용할 수 있는
함수
```

```
a = getting_knn(knn_data, knn_data[0],5)
print(a)
```

```
[41, 13, 8, 38, 42]
```

※Knn 전체를 구현한다고 생각하지 마시고, 일종의 filtering처럼 target 백터와 가장 가까운 백터 5개를 추출한다고 생각하시면 더 좋을 것 같습니다! 실제 KNN의 작동원리와는 다를 수 있지만 그 과정 중 하나의 단계를 구현하는 것이라 생각하면 됩니다.

2-1. MCAR 데이터를 위한 인덱스 값 만들기

- random_int 함수를 사용해서 4개의 각기 다른 random_index를 가진 리스트를 만드세요
- random 숫자는 20개씩 입니다.
- 각각의 seed값은 [2061,4220,407,1996] 입니다.

```
## random_int
def random_int(num, seed=407):
    import random
    from random import randint
    random.seed(seed)
    random_index = []
    i = 0
    while True:
        random_number = randint(0,149)
        if random_number not in random_index:
            random_index.append(random_number)
            i = i+1
            if i == num:
                break
            else:
                pass
        else:
            pass

    return random_index
```

2-2. MCAR 데이터 만들기

- data를 새로 불러오고 species column을 제거하세요
- 2-1번에서 만든 index값들을 각각의 column에 적용시키세요
- 총 80개의 NA가 생겨야 합니다.

2-3. 행별로 KNN 구하기

- 각 행별로 nearest neighbor를 찾으세요 (n=5)
- nan_euclidean_distances라는 함수를 사용해서 거리를 계산하세요
- 위에서 만들어 놓은 getting_knn 함수를 조금 커스터마이즈해서 사용하세요!
  (기존처럼 하면 안됩니다!)
- NaN이 포함되기 때문에 list로 접근하면 안되고 이제는 array로 접근해야합니다.

2-4. KNN IMPUTATION 시작하기

- 2-4의 목표는 첫 번째 column에 존재하는 모든 nan를 imputation하는 것입니다.
- 각 행 별 첫 번째 값이 nan일 경우 해당 군집의 평균값을 이용하여 nan을 채우도록 하세요!
- [nan,1,1], [2,2,2],[2,3,4] 이면 [nan,1,1]이 [2,1,1]이 되는거라고 생각하시면 됩니다.
- array의 indexing을 사용하시면 아주 쉽게 구하실 수 있습니다..?
- np.nan 활용해보세요

※마지막 np.nan은 np.nanmean과 np.isna를 말하는겁니다!

2-5. KNN IMPUTATION 전체 채우기

- 2-5의 목표는 모든 column에 대해서 knn imputation을 하는 것입니다.
- 2-4를 모든 column에 대해서 실행해주세요!

---

3. 그만 구현하자. 있는걸 써보자

- 여기까지 온 여러분께 박수..

3-1. knn_imputer을 이용해서 na imputation을 해보세요!

- sklearn 라이브러리 안에 있습니다.
- 원래는 스케일링 다 해줘야하는데 서로 귀찮으니깐 했다고 생각합시다 :)
- k=5로 두시면 됩니다

```
## 초기값
data = data_loader()
data = data.drop(['species'],axis=1)
for col in enumerate(data.columns[:4]):
    name = col[1]
    indexing = col[0]
    target_list = rand_index[indexing]
    data[name].loc[target_list] = None
```

3-2 이번에는 fancyimpute라는 패키지를 이용해서 knn imputation을 해보세요

- 원래는 스케일링 다 해줘야하는데 서로 귀찮으니깐 했다고 생각합시다 :)
- https://www.geeksforgeeks.org/missing-data-imputation-with-fancyimpute/

https://www.geeksforgeeks.org/missing-data-imputation-with-fancyimpute/

3-3 이번에는 fancyimpute라는 패키지를 이용해서 mice imputation을 해보세요

- 원래는 스케일링 다 해줘야하는데 서로 귀찮으니깐 했다고 생각합시다 :)
- https://www.geeksforgeeks.org/missing-data-imputation-with-fancyimpute/

4 숨겨진 셀 1개

---

4. imputation 결과로 모델링 진행하기

4-1. 전체 데이터셋 구성하기

- 아래의 코드를 실행해서 데이터를 만들어주세요
- 해당 데이터를 이용해서 train /test를 만들겁니다.
- X : species 를 제외한 변수 / Y : speices
- train_test_split를 이용해서 층화추출로, 40%를 test로 분리하세요
- 이때 random_state는 407로 합니다.

```
## 기본 코드
data = data_loader()
for col in enumerate(data.columns[:4]):
    name = col[1]
    indexing = col[0]
    target_list = rand_index[indexing]
    data[name].loc[target_list] = None
```

4-2. 상황별 데이터셋 구성하기

- 3가지의 na imputation을 적용한 데이터셋을 만드세요
(mean imputation, knn ,mice)
- library를 사용하셔도 됩니다.
- 모든 연산은 train을 기준입니다. train의 연산결과를 test에 적용하시면 됩니다
- python에서는 fit과 transform을 통해서 train에 적용한 것을 test에 동일하게 적용할 수 있습니다

4-3. 로지스틱 회귀 모형을 활용해서 평가하기

- 3가지의 데이터셋을 모두 학습한 뒤, test 예측 결과에 대한 accuracy 각각 구하세요
- scikit-learn에 있는 acc구하는 함수 사용하시면 편합니다.

다들 한 학기 동안 과제 열심히 푸시느라 고생 많으셨습니다!

이제 남은 주분 2주차 동안 조금만 더 힘내셔서 멋진 결과물 낼 수 있기를 기원합니다!

뒤풀이 때 뵙겠습니다아아아아아