**컴퓨터 응용확률 과제2**

12171651 컴퓨터공학과 오윤석

사용한 라이브러리

|  |  |  |  |
| --- | --- | --- | --- |
| Seaborn (hist 출력) | Numpy (데이터 변환) | Matplot (그래프 출력) | Math (계산) |

사용 언어: python 3.5.8

구현 환경: window 10, pycharm

**Problem**

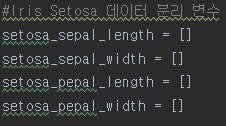
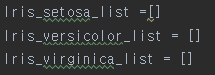
Part A. Preliminaries

1. Cross Validation

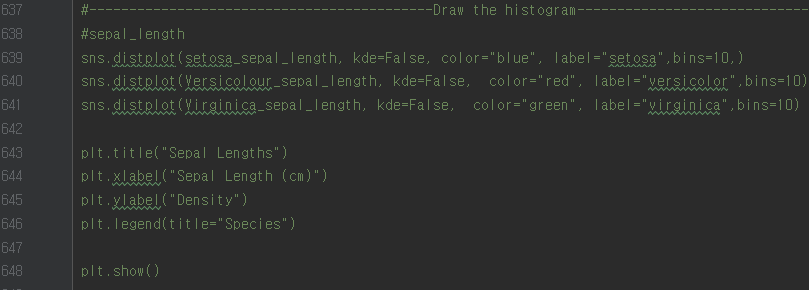
K – fold cross-validation이란 데이터를 k개의 fold로 나누어 진행하는 교차 검증이다. 이는 주로전체 데이터의 개수가 적을 때 사용을 하고, 적은 데이터로도 더 정확히 모델을 학습하고자 할 때 사용한다. 우리 과제에서는 총 150개의 데이터를 가지고 5-fold-cross-validation을 진행하였다. 이는 150개에 데이터를 5개의 fold로 나누고 (1 fold당 50개의 데이터) 여기서 4개의 데이터로 모델을 학습시키고 나머지 1개의 fold로 validation 즉 test를 진행한다. 이렇게 1번이 끝나면 validation을 바꾸고 또 다시 4개의 training fold를 지정하여 총 5번 반복한다. 이렇게 되면 모델은 다른 5개의 validation으로 test를 진행한 결과를 갖는다. 이 결과 즉 정확도는 주로 5개의 평균으로 성능을 평가하며, 때에 따라 다른 방법을 가지고 성능을 평가한다. 하지만 k fold cross validation의 단점은 회전 횟수가 많기 때문에 모델을 학습시키고 테스트하는 시간이 많이 소요되는 점이 있다.

Part B. Iris classification using text data

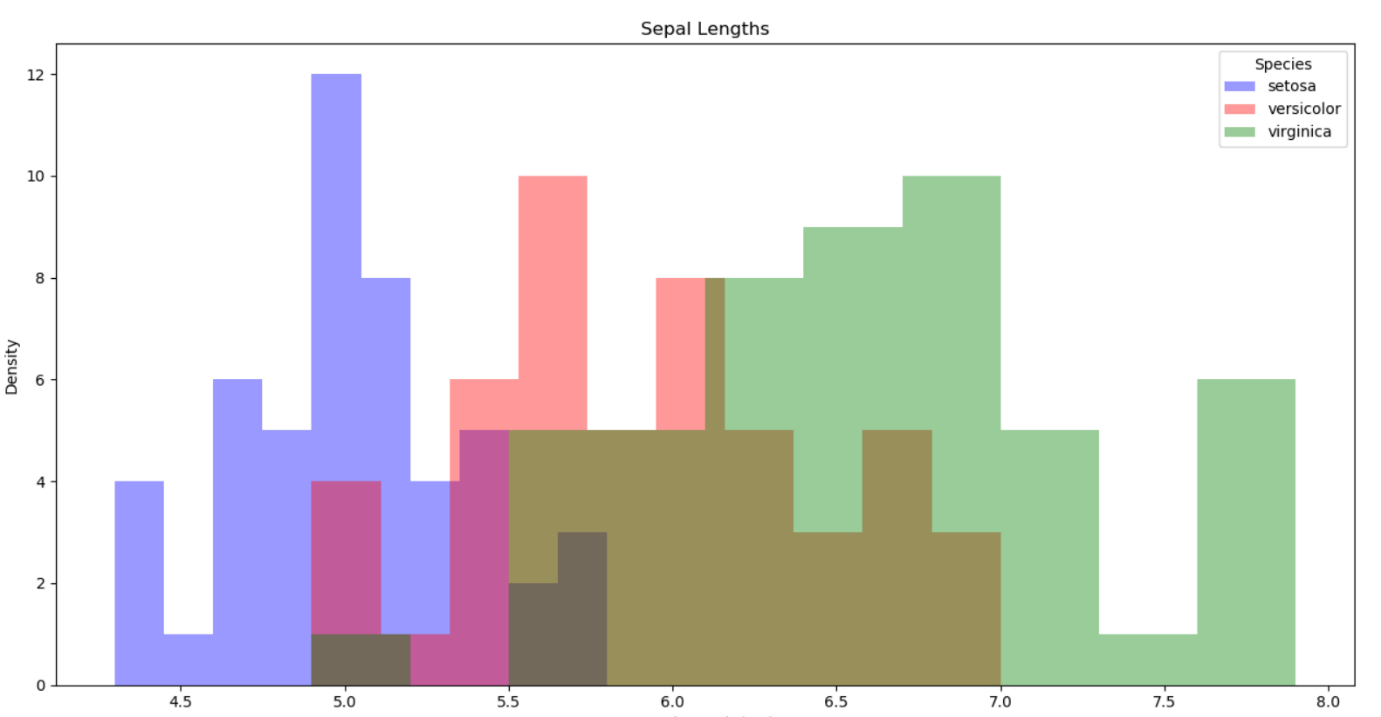
1. 우선 이 문제를 해결하기 위해서는 전체 데이터 셋 (150개)를 각 label과 특징별로 나누었다. 이 코드를 실행시키려면 65번째 줄에 경로를 iris.data가 있는 경로로 변경해야한다.

이런식으로 데이터를 우선 나누고 진행하였다.

히스토그램은 sns 라이브러리에 있는 distplot을 사용하였다. 코드 637줄부터 시작이다.

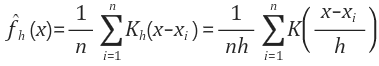


이 코드를 각 히스토그램을 구별하기 위해 색을 변화시켰고, bins는 10으로 잡았다.



그 결과 이렇게 히스토그램으로 나타낼 수 있었고 이를 각 label로 구별하기 위해 생각을 하였다. 원래는 각 히스토그램의 중심점을 기준으로 point를 줘서 선분으로 이으고 이를 바탕으로 교점을 경계선으로 두려고 했지만 아무런 너무 비효율적인 것 같아 검색 끝에 커널밀도함수라는 것을 알게 되었다. 밀도 함수란 얻어진 데이터들의 분포로부터 원래 변수의 확률적 분포 특성을 추정하는 것이다. 이 밀도함수의 확률은 구하고자 하는 x의 적분 값으로 확률을 얻을 수 있고 이를 통해 우리의 목표에 도달할 수 있을거라 생각하였다.

여기서 커널 밀도 함수는 히스토그램과는 달리 연속적인 함수로 나타낼 수 있는 큰 장점을 가진다.

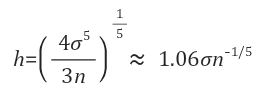


커널 밀도 함수의 전체적인 식 (출처: 위키피디아)

여기서 n은 데이터의 개수 (우리는 50), h는 우리가 변화시킬 수 있는 파라미터, k()는 커널 함수이다. 커널 밀도 함수는 어떤 커널 함수를 사용하는지에 따라 그 그래프가 천차 난별이다. 나는 수업 시간에 배운 가우시안이 생각나서 가우시안 커널 함수를 사용하였다. 지금까지 나온 커널 함수 중에는 Epanechnikov 커널이 가장 최적의 커널 함수라고 한다. 하지만 계산의 편의상 가우시안이 더 편하다고 한다.



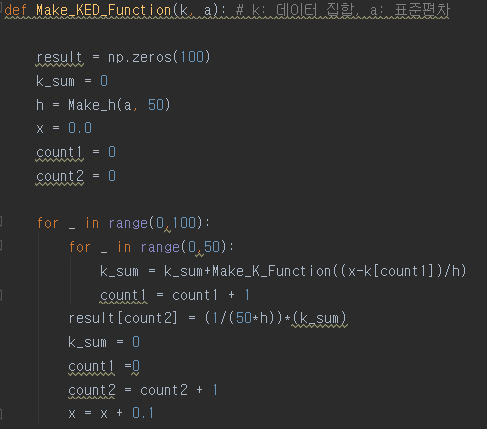
가우시안 커널 함수 (출처: 위키피디아)

또한 위에서 언급한 h값 즉 파라미터 값을 결정해야 하는데 가우시안의 가장 최적의 h값은 

(출처: 위키피디아)

라고 한다. 여기서 n은 데이터의 개수, 시그마는 표준편차이다.

이 이론들을 가지고 가우시안 커널 밀도 함수를 직접 구현하였다.



이 코드는 전체적인 커널 밀도 함수이다. 우선 for문에 들어가면 총 50번만큼 반복하여 가우시안 커널 함수를 만든다.

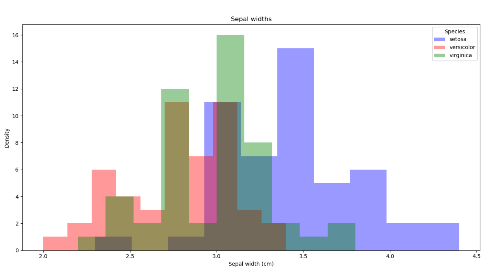
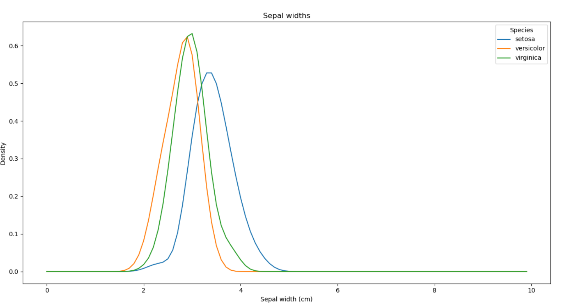


여기서 먼저 변수가 Make\_K\_Function에 들어가게 되는데 이는 위에서 설명한 가우시안 커널 함수와 같다.

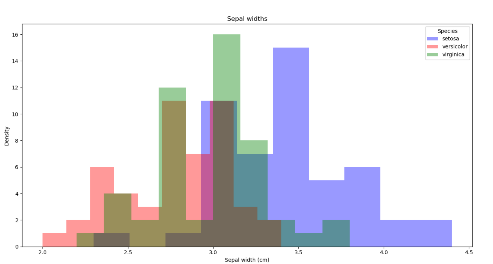
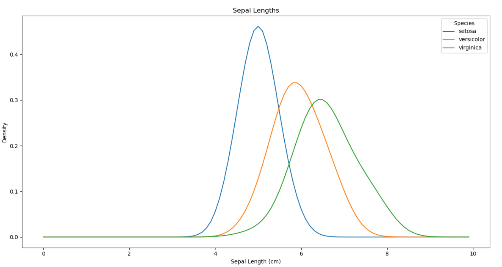


h또한 위에서 설명한 것과 똑같이 만들었다.

그 결과 모든 histogram들이 각자 커널 함수를 통해 연속적인 함수로 만들어졌다.



Sepal width에 대한 커널 밀도 함수와 히스토그램



Sepal length에 대한 커널 밀도 함수와 히스토그램

커널 밀도 함수에 관한 그래프는 580번째 줄부터 존재한다.

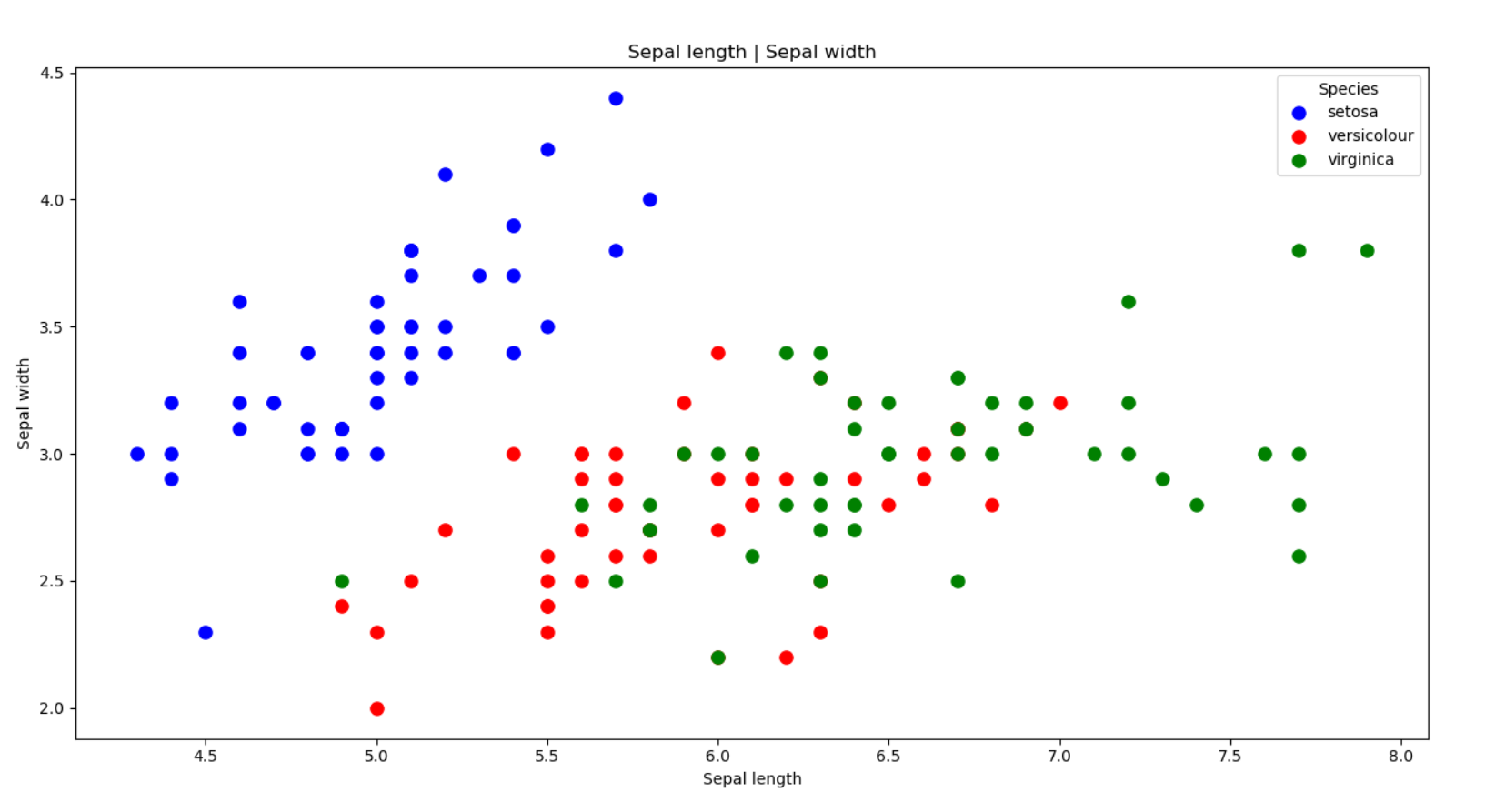
이제 여기서 경계선을 나누어야 하는데 한눈에 봐도 각 그래프의 교점이 가장 적절한 경계선이라 생각해 교점을 가지고 경계를 나누었다.

|  |
| --- |
| **sepal\_length, likelihood(5.5, 6.3)** |
| **sepal\_width, likelihood(3.0, 3.2)** |
| **pepal\_length, likelihood(2.2,4.9)** |
| **pepal\_width, likelihood(0.7,1.7)** |

그 결과 이렇게 각 데이터마다 경계점을 구할 수 있었다.

(경계점은 그래프의 눈금을 가지고 2째자리에서 반올림하여 구했다.)

2. 이 문제를 해결하기 위해 우선은 데이터를 산점도로 표현을 해야한다. 산점도로 표현하기 위해 plt.plot() 함수를 사용하였다. (726번째 줄)

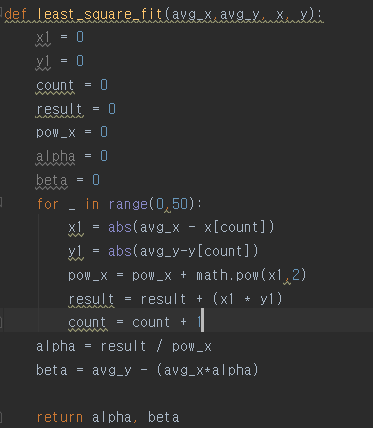


이런식으로 총 나올 수 있는 경우의 수 총 6개의 산점도 그래프를 구하였다. 하지만 이 산점도들을 각 분류를 해야 하는데 우선은 과제1과 마찬가지로 최소 제곱법을 이용하여 분류를 시도할 수 있었다. 최소 제곱법이란 선형 회귀에서 데이터를 통해 선을 그어서 예측하는 방법이다. 그리고 말그대로 각 데이터들의 오차를 최소한으로 하여 선을 긋는 것이다.

이 선은 y = ax + b꼴로 되어 있다. 이 함수를 구하는 법은 우선 x와 y의 편차가 필요하다 그리고 나서 각 데이터의 잔차를 구한다. 이 잔차를 가지고 잔차의 제곱의 합을 구한다. 그리고 나서 이 잔차를 a와 b에 대해 미분을 한 후 0으로 둔다. 이를 최종적으로 정리를 하게되면

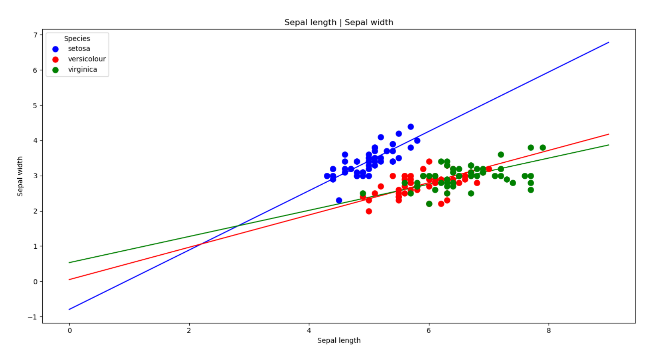
a = x와 y 편차의 제곱 합 / x 편차의 제곱 합

b = y 평균 – x 평균 \* a 가 된다.

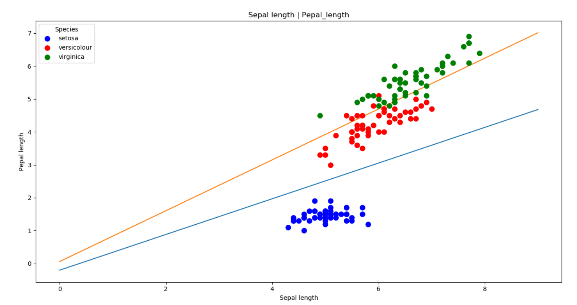
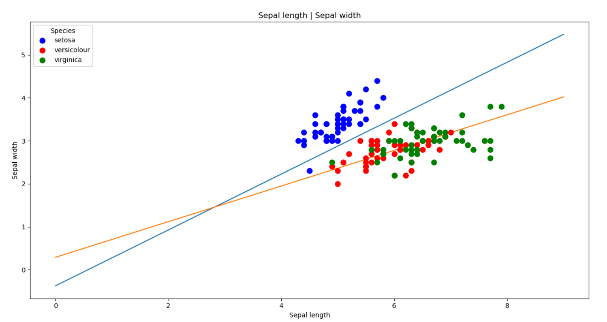


최소 제곱법을 구현한 함수이다.

이 최소 제곱법으로 구한 함수를 그래프에 두면



이런식으로 선형 회귀를 할 수 있게 된다. 하지만 우리가 원하는 건 예측이 아닌 분류 문제이다. 따서 각 선의 중앙에 선을 그어주게 되면 우리가 원하는 분류를 할 수 있게 된다.



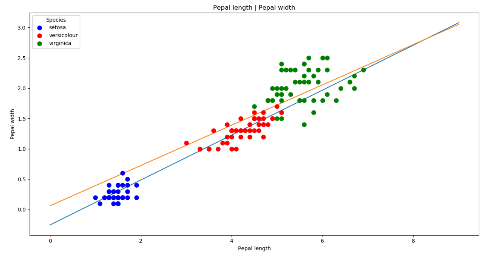
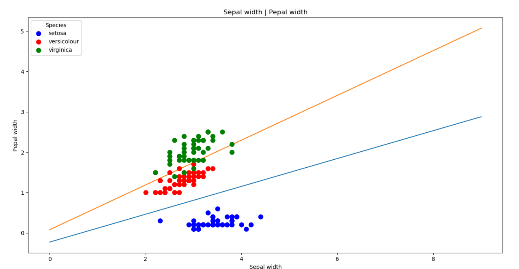
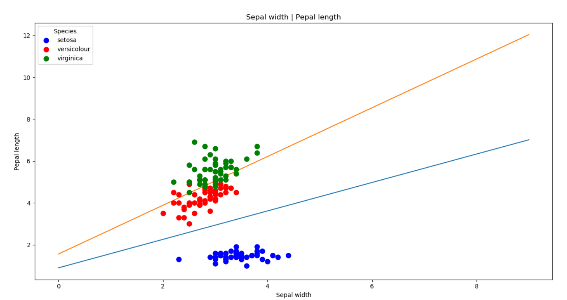
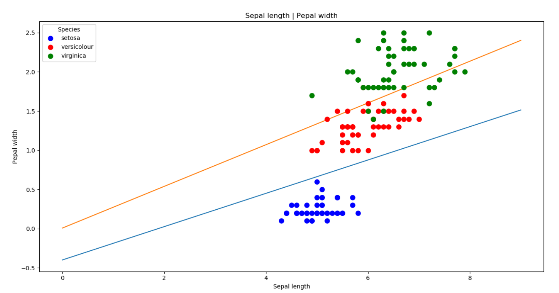
이 방법을 통해 최소의 오차로 boundary를 구할 수 있었다.

이 그래프 또한 726번째 줄에서 확인가능하다.

그리고 boundary는 이 함수를 통하여 분류 가능하다.

|  |  |  |
| --- | --- | --- |
| **data** | **a** | **b** |
| Sepal length, sepal width | y = 0.65x - 0.37 | y = 0.41x + 0.30 |
| Sepal length, papal length | y = 0.54x - 0.20 | y = 0.77x +0.06 |
| Sepal length, pepal width | y = 0.21x - 0.40 | y = 0.27x +0.01 |
| Sepal width, pepal length | y = 0.68x + 0.98 | y = 1.17x +1.55 |
| Sepal width, pepal width | y = 0.35x - 0.23 | y = 0.55x +0.08 |
| Pepal length, pepal width | y = 0.37x - 0.25 | y = 0.33x + 0.61 |

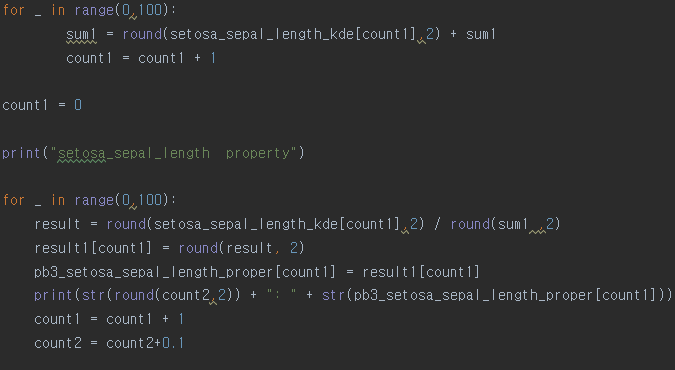
이 표에서 a는 파랑색 직선, b는 주황색 직선에 해당한다. 이 2 직선을 기점으로 label이 나뉘게 된다.



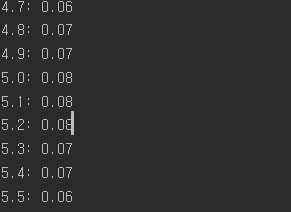
위에 2개까지 합쳐서 총 6개이다.

3. 이 문제는 지금까지 구한 boundary를 활용하여 직접 k-fold를 통하여 테스트를 하는 문제이다. 우선 지금까지의 데이터와 분류 자체가 학습을 시키는 모델보다는 boundary에 집중하고 프로그래밍을 하여 k-fold로는 training을 진행하지 못하였다.

코드는 324번째 줄부터 시작된다. 우선 나는 p(a|x)를 구하였다. 여기서 x는 입력 값이고, a는 라벨 값이다.



p(a|x)는 보다 간단하게 구할 수 있었다. 우선 part2 1번째 문제를 기반으로 구하였다. 우선 1번째 문제에서는 밀도 함수를 구하였는데 여기서 해당하는 x값(소수 첫째자리까지)에 해당하는 모든 밀도값을 더한 후 해당 x에 해당하는 값을 모든 밀도 값으로 나누면 합이 1인 확률이 된다. (반올림 때문에 정확히 1은 아니다) 이게 바로 p(a|x)이다.



예를 들어 sepal length가 5.1이면 이것이 setosa일 확률은 0.08이다.

이것을 각 label과 data 유형에 따라서 총 12종류를 구하였다. 코드를 실행시키면 output으로 나온다.

하지만 이를 pmf로 구현을 해야되는데 pmf에 해당하는 x값은 무조건 정수여야 한다. 여기서부터 과제가 막히게 되었고 결국 여기까지 구현할 수 밖에 없었다.