Part A. Preliminaries

1. (10pts) Cross Validation

Summarize the k-fold cross-validation method for evaluating a classifier (less than one page)

* K- fold cross validation 사용 이유

data가 많지 않은 상황에서는 validation data의 수도 매우 작아진다. 이렇게 되면, validation set의 분할에 대한 validation score의 분산이 높아지게 된다. 이는 training data와 validation data로 어떤 데이터 포인트가 선택되었는지에 따라 validation score가 크게 달라질 수 있음을 의미한다. 또한, 선택된 validation data를 바탕으로 모델을 튜닝하다 보면, 해당 validation set에만 overfitting되는 결과가 생기게 된다. 따라서 신뢰 있는 모델 평가를 하기 어려워진다. 이러한 상황에서 가장 좋은 방법은 K-fold cross validation을 사용하는 것이다.

* 데이터가 적은 상황에서는 K-fold cross validation을 사용하는 것이 신뢰할 수 있는 모델 평가 방법이다.
* K-fold cross validation 이란?

K개의 모델을 각각 만들어 K-1 개의 분할(fold)에서 훈련하고 나머지 분할에서는 평가하는 방법이다. Iteration을 통해 모든 데이터 셋을 검증에 활용한다. 최종 validation score는 k개의 validation score의 평균이 된다.

* 장점

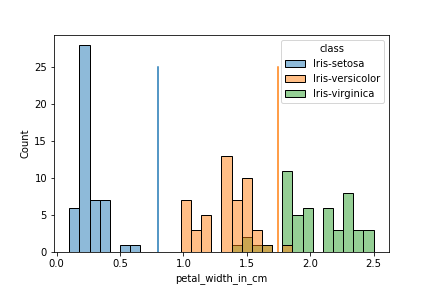
모든 데이터 셋을 validation에 활용할 수 있어서 특정 validation set에 overfitting 되는 것을 방지할 수 있다.

* 코드 실행을 위해 설치해야하는 라이브러리: seaborn-0.11.0, matplotlib, pandas, numpy

Part B. Iris classification using text data

1. (15pts) For four different features, show four separate histograms where each shows three classes in different colors. Find your optimal decision boundaries to minimize the costs. Report your analysis about the result.

Source code : histplot.py (해당 코드 실행 후, plots/ 폴더 안에 plot 이미지들이 저장됨)

* 1. 
* Decision boundary:

Let x be petal\_width\_in\_cm, be predicted class

if x < 0.8, = Iris-setosa

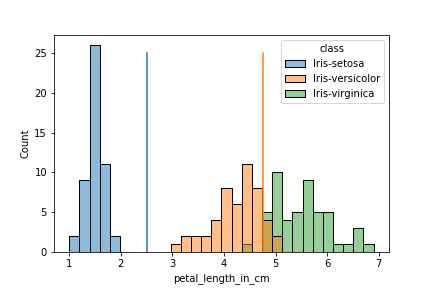
if 0.8 <= x <1.75, = Iris-versicolor

if 1.75 <= x, = Iris-virginica

* Analysis:

Petal width가 각 class를 분류함에 있어서 좋은 feature라고 판단된다.

Setosa에 관해서는 오류 없이 잘 분류하지만, versicolor 와 virginica를 구분함에 있어서는 약간의 오류가 존재한다.



* Decision boundary:

Let x be petal\_length\_in\_cm, be predicted class

if x < 2.5, = Iris-setosa

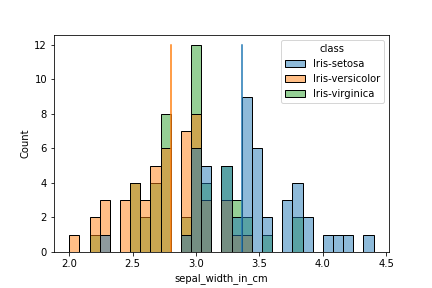
if 2.5 <= x <4.75, = Iris-versicolor

if 4.75 <= x, = Iris-virginica

* Analysis:

Petal length가 각 class를 분류함에 있어서 좋은 feature라고 판단된다.

Setosa에 관해서는 오류 없이 잘 분류하지만, versicolor 와 virginica를 구분함에 있어서는 약간의 오류가 존재한다.



* Decision boundary:

Let x be sepal\_width \_in\_cm, be predicted class

if x < 2.8, = Iris-versicolor

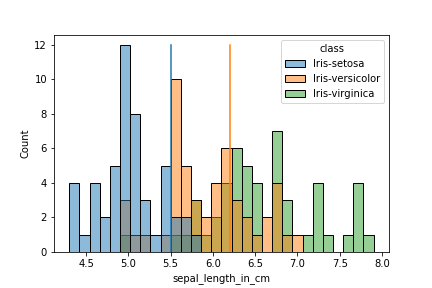
if 2.8 <= x < 3.36, = Iris-virginica

if 3.36 <= x, = Iris-setosa

* Analysis:

Sepal width feature 하나로는 각 class를 분류하기 어렵다고 판단된다.

각 class를 분류할 때 많은 오류가 존재한다.



* Decision boundary:

Let x be sepal\_width \_in\_cm, be predicted class

if x < 5.5, = Iris-setosa

if 5.5 <= x < 6.2, = Iris-versicolor

if 6.2 <= x, = Iris-virginica

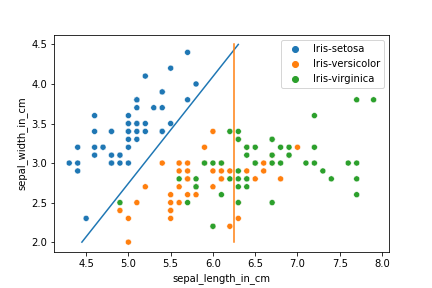
* Analysis:

Sepal length feature 하나로는 각 class를 분류하기 어렵다고 판단된다.

각 class를 분류할 때 많은 오류가 존재한다.

1. (15pts) Visualize the data in 2D space to show the distribution of the attributes. You should show all possible combinations of pairs of attributes, six cases in total. Find your optimal decision boundary in each case to minimize the costs. Report your analysis about the result.

Source code: scatterplot.py (해당 코드 실행 후, plots/ 폴더 안에 plot 이미지들이 저장됨)



* Decision boundary:

Let x be sepal\_length \_in\_cm, y be sepal\_width\_in\_cm, be predicted class

if y > 1.35x - 4, = Iris-setosa

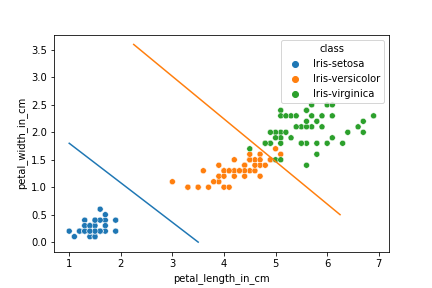
if 1.35x - 4 <= y and x < 6.25, = Iris-versicolor

if 6.25 <= x, = Iris-virginica

* Analysis:

1-C, 1-D에서 볼 수 있듯이, sepal length 와 sepal width는 각각 하나의 feature만을 이용하여 분류할 때는 좋은 성능이 나오지 못했다. 그러나 두 feature를 모두 이용하여 분류할 경우, 각각 따로 이용하여 분류할 때보다는 좀 더 나은 분류 성능을 보인다.

Setosa의 경우 잘 분류하지만, versicolor와 virginica를 구분함에 있어서 오류가 존재한다.



* Decision boundary:

Let x be petal\_length \_in\_cm, y be petal\_width\_in\_cm, be predicted class

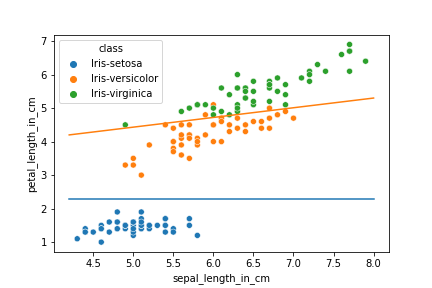
if y < -0.72x + 2.52, = Iris-setosa

if -0.72x + 2.52<= y < -0.72x + 5.86, = Iris-versicolor

if y >= -0.72x + 5.86, = Iris-virginica

* Analysis:

1-A, 1-B의 결과에서 볼 수 있듯이, Petal length와 petal width는 각각 class를 분류함에 있어서 좋은 feature였다. 두 feature를 모두 사용하여 분류할 경우 역시 좋은 분류 성능을 보이는 것을 볼 수 있다.



* Decision boundary:

Let x be sepal\_length\_in\_cm, y be petal\_length\_in\_cm, be predicted class

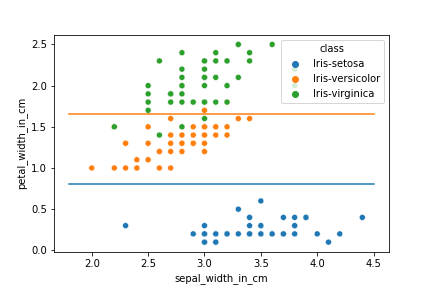
if y < 2.3, = Iris-setosa

if 2.3<= y < 0.29x + 2.98, = Iris-versicolor

if y >= 0.29x + 2.98, = Iris-virginica

* Analysis:

1-B, 1-D의 결과에서 볼 수 있듯이, sepal length보다는 petal length가 class를 분류함에 있어서 더 좋은 feature였다. Sepal length와 petal length 두 feature를 사용하였을 때, 대부분 petal length를 사용하여 분류하고, Versicolor와 virginica를 분류할 때 petal length의 역할이 조금만 들어가는 것을 볼 수 있다.



* Decision boundary:

Let x be sepal\_width\_in\_cm, y be petal\_width\_in\_cm, be predicted class

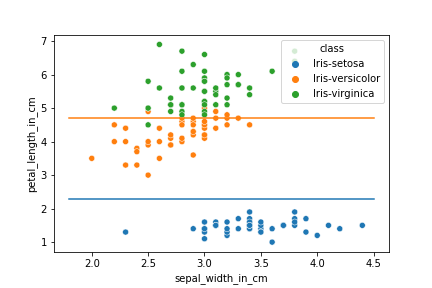
if y < 0.8, = Iris-setosa

if 0.8 <= y <1.65, = Iris-versicolor

if 1.65 <= y, = Iris-virginica

* Analysis:

1-A, 1-C의 결과에서 볼 수 있듯이, sepal width보다는 petal width가 class를 분류함에 있어서 더 좋은 feature였다. Sepal width와 petal width 두 feature를 사용하였을 때, sepal width는 사용하지 않고, petal width만을 사용하여 분류하는 것을 볼 수 있다.



* Decision boundary:

Let x be sepal\_width\_in\_cm, y be petal\_length\_in\_cm, be predicted class

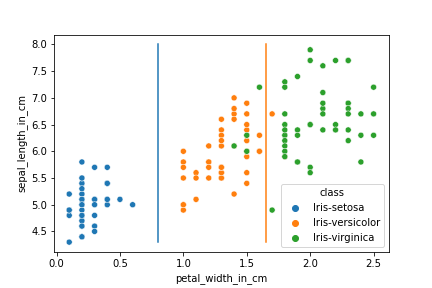
if y < 2.3, = Iris-setosa

if 2.3 <= y < 4.7, = Iris-versicolor

if 4.7 <= y, = Iris-virginica

* Analysis:

1-B, 1-C의 결과에서 볼 수 있듯이, sepal width보다는 petal length가 class를 분류함에 있어서 더 좋은 feature였다. Sepal width와 petal length 두 feature를 사용하였을 때, sepal width는 사용하지 않고, petal length만을 사용하여 분류하는 것을 볼 수 있다.



* Decision boundary:

Let x be petal\_width\_in\_cm, y be sepal\_length\_in\_cm, be predicted class

if x < 0.8, = Iris-setosa

if 0.8 <= x < 1.65, = Iris-versicolor

if 1.65 <= x, = Iris-virginica

* Analysis:

1-A, 1-D의 결과에서 볼 수 있듯이, sepal length보다는 petal width가 class를 분류함에 있어서 더 좋은 feature였다. Sepal length와 petal width 두 feature를 사용하였을 때, sepal length는 사용하지 않고, petal width만을 사용하여 분류하는 것을 볼 수 있다.

1. (20pts) Write a code to classify three classes of iris based on the Bayesian decision rule; by finding a pmf(probability mass function) of likelihood and defining a prior. Evaluate your classifier by computing precision and recall. Use 5-fold cross validation for evaluation. You may set or test your own priors.

Source code: Bayesian\_classifier\_by\_pmf.py (관련 함수들은 function.py에 정의되어 있음)

* 해당 python파일을 실행시키면, inference 과정과 모델 평가 결과를 볼 수 있다.

1. 5-fold cross validation:

각 fold의 train, validation data의 클래스 비율을 같게 총 5개의 fold로 나누었다.

총 데이터 개수: 150개

fold별 train data 개수: 120개 = (각 클래스별 40개) \* (클래스 개수 3개)

fold별 validation data 개수: 30개 = (각 클래스별 10개) \* (클래스 개수 3개)

1. Naïve bayesian classifier:

--- (식 1)

(Posterior) = (likelihood)\*(prior)/(evidence)

* Likelihood:
  + --- (식 2)

총 4개의 feature에 대해 각각의 pmf를 구하고, 각 feature는 독립이라고 가정하여 각 값들을 곱하였다.

* + Pmf: 각 fold의 train data를이용한다. Data를 class, feature별로 구분한 뒤, interval로 나누어 그 구간에 대한 likelihood값을 구함.

Input example:

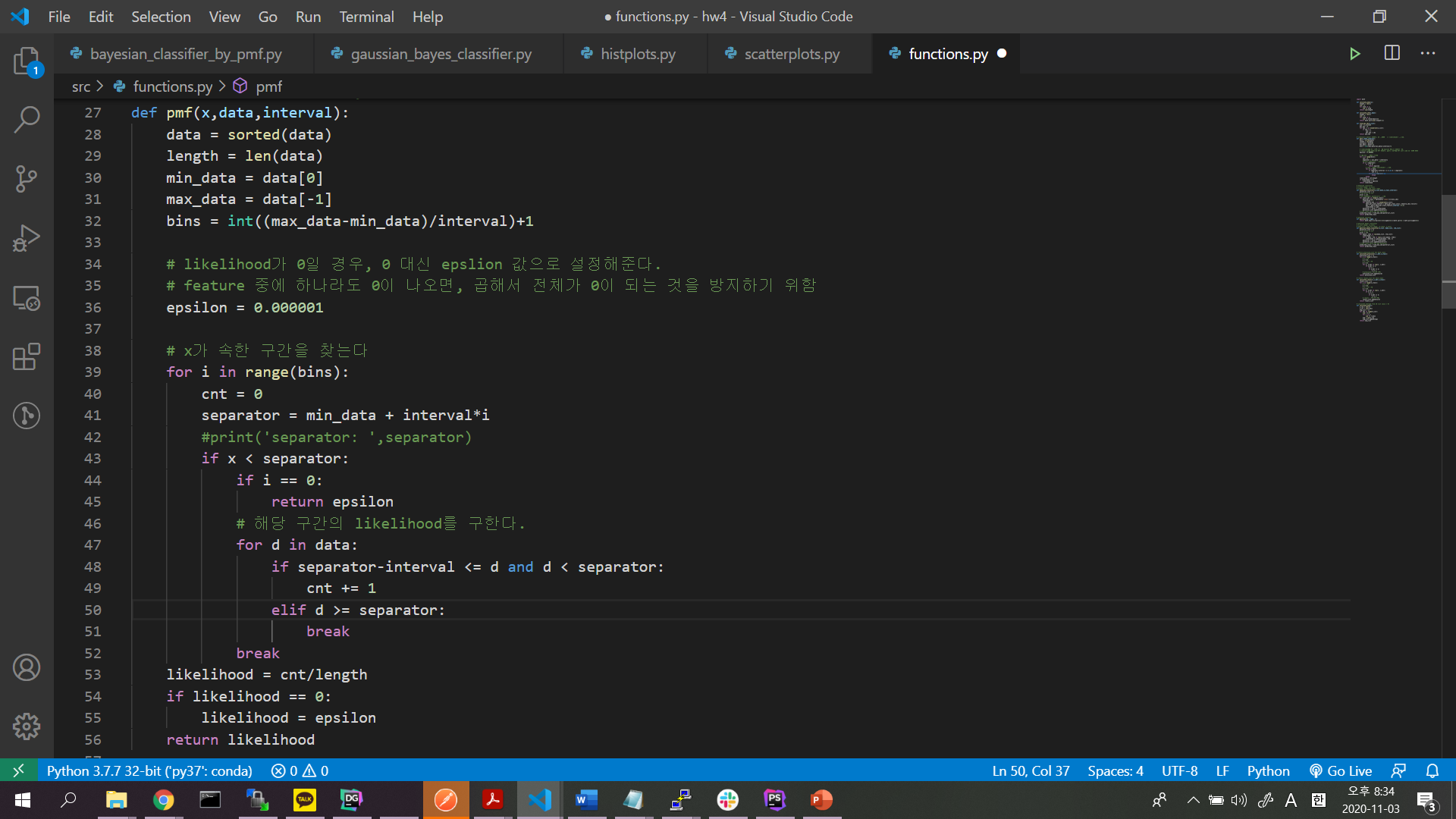
X = 6.9, data = [7.0, 6.4, 6.9, 5.5, 6.5, 5.7…], interval = 0.2

Output example:

0.1

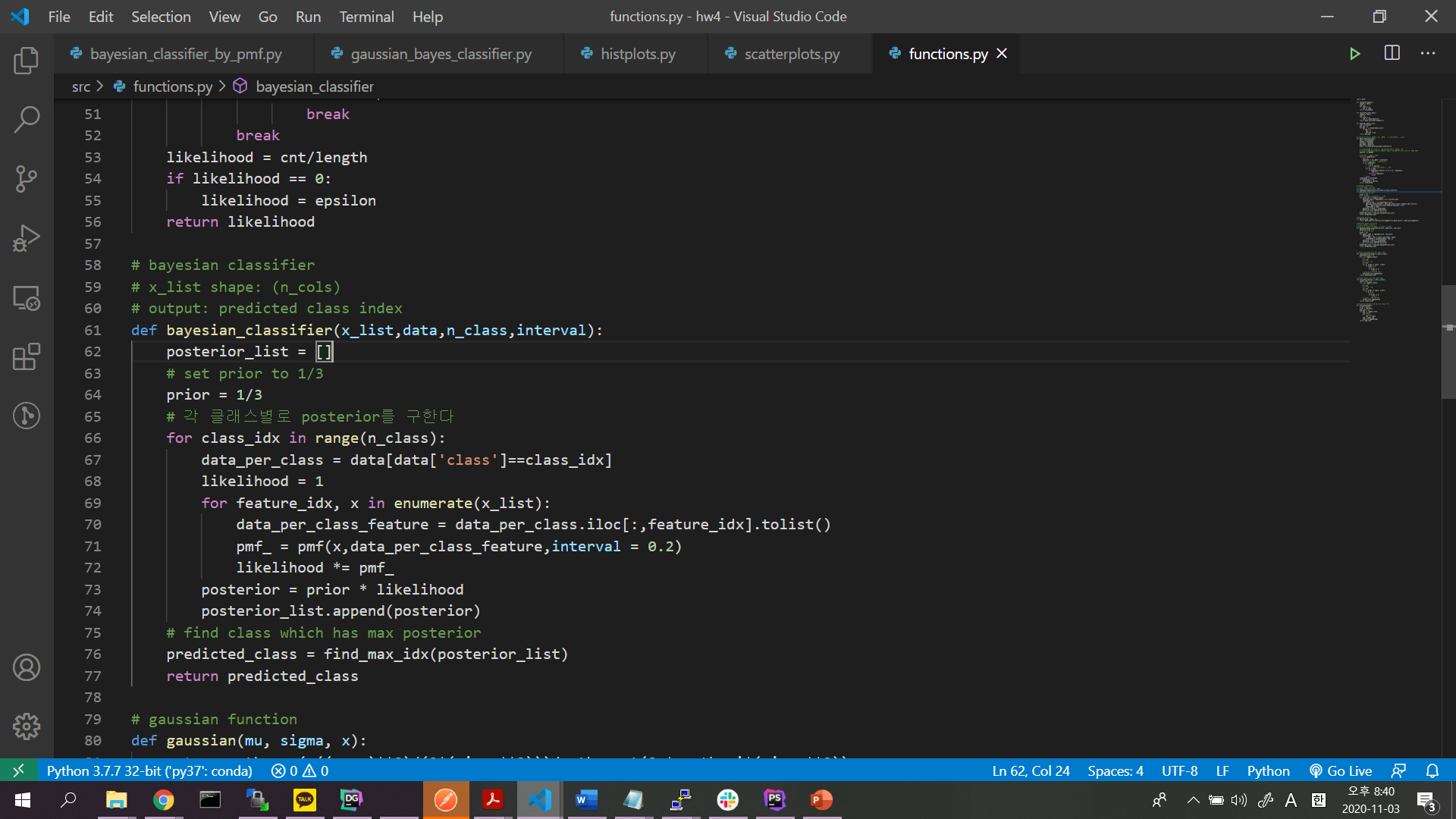
(식 2)에서 볼 수 있듯이, 가 하나라도 0이 되면, posterior가 0이 되어버리므로,

일 경우에는 아주 작은 값 (0.000001)으로 설정해 두었다.



<참고: function.py>

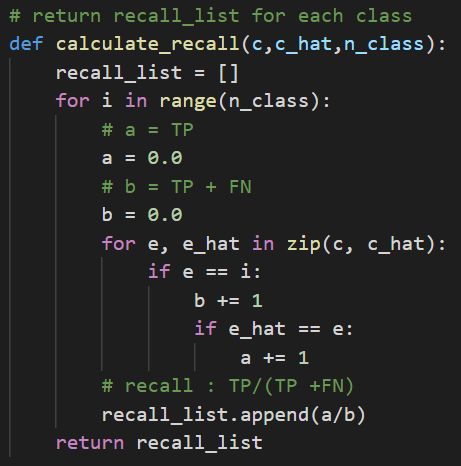
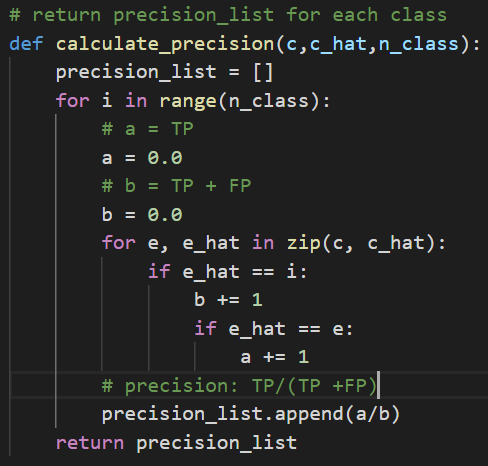
* Prior: 각 클래스의 개수가 동일한 것을 고려하여, 1/3로 같게 설정하였다.
* Evidence: 고려하지 않음
* Classifier: 각 클래스 별로 posterior를 구한 뒤, 그 값이 가장 큰 class의 index를 리턴한다.



<참고: function.py>

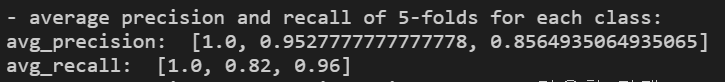
1. Evaluation: 각 fold에 대하여, validation data의 각 행마다 Bayesian\_classifier를 통해 inference를 진행 후, 실제 class값과 비교하여 precision 과 recall을 구했다.

최종적으로 각 fold의 precision과 recall의 평균값을 구해 모델을 평가하였다.



<참고: function.py>

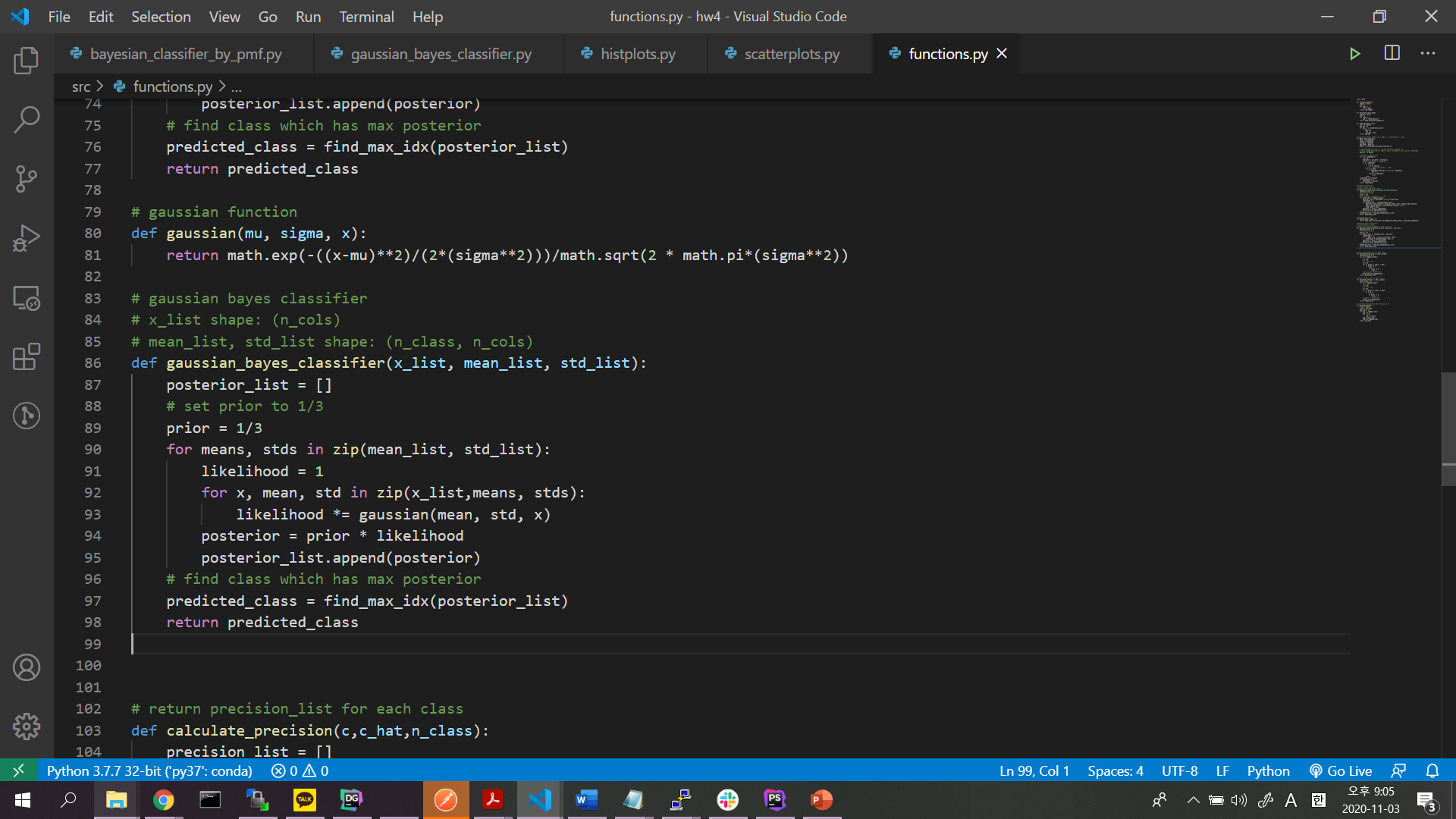
결과:



1. (20pts) Write a code to classify three classes of iris based on the Bayesian decision rule; by modeling with a Gaussian distribution. Evaluate your classifier by computing precision and recall. Use 5-fold cross validation for evaluation. You may set or test your own priors.

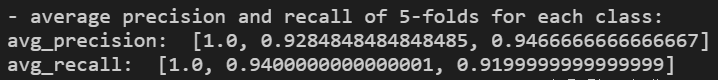
Source code: gaussian\_bayes\_classifier.py (관련 함수들은 function.py에 정의되어 있음)

* 해당 python파일을 실행시키면, inference 과정과 모델 평가 결과를 볼 수 있다.
* 5-fold cross validation, evaluation 방법은 위의 방법과 동일.
* Gaussian classifier:
  + 3번과 같이 각 feature별로 likelihood를 구해서 곱하는 방식은 같으나, likelihood를 구할 때 pmf를 이용하는 것이 아니라, gaussian으로 분포를 가정하여 구하였다.
  + 각 fold에 사용되는 trainig data를 이용하여 gaussian을 가정한다. Class, feature 별로 data의 평균과 표준편차를 구해서 각각의 gaussian 함수를 만들고, (식 2)와 같이 feature 별로 곱하여 각 class에 대한 likelihood를 계산하였다.



<참고 function.py>

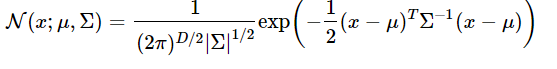
결과:

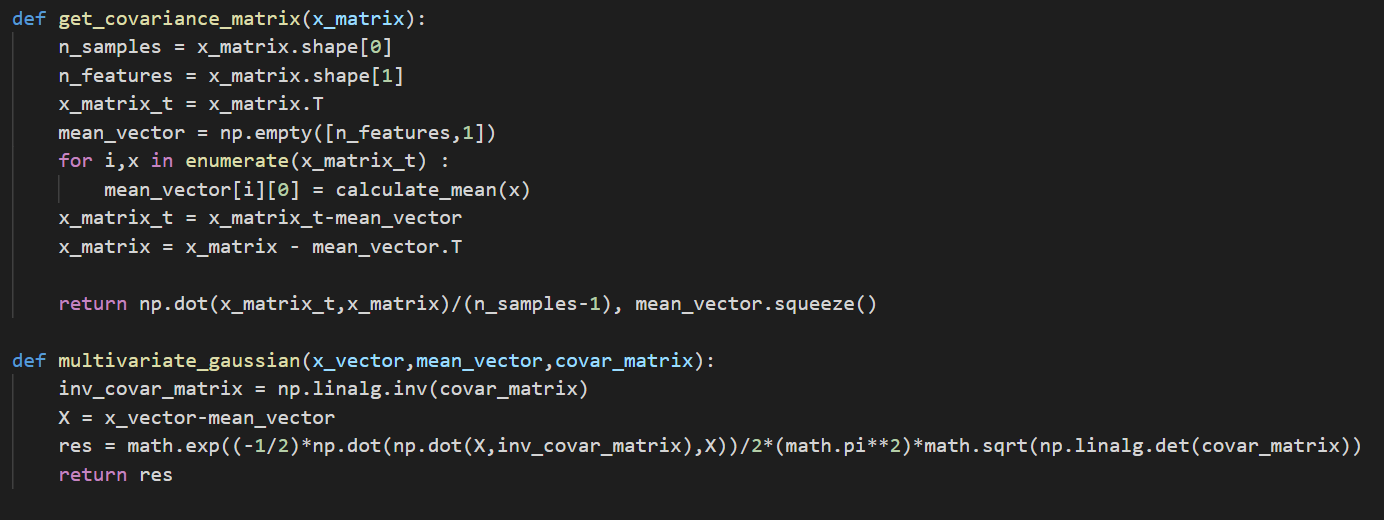


1. (30pts) Extend your work by using a multi-variate Gaussian model. Discuss about your results.

Source code: multivariate\_gaussian\_bayes\_classifier.py (관련 함수들은 function.py에 정의되어 있음)

* 해당 python파일을 실행시키면, inference 과정과 모델 평가 결과를 볼 수 있다.
* Multi-variate gaussion function





* Get\_covariance\_matrix

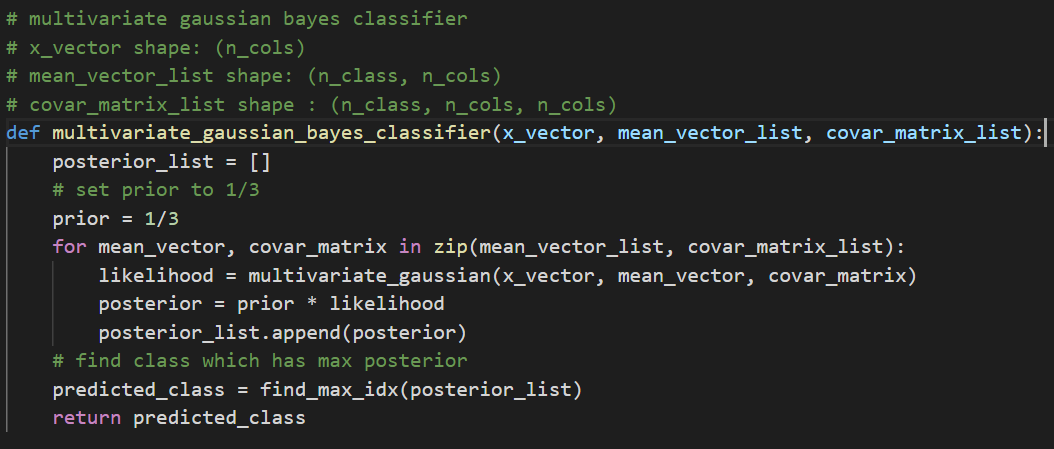
입력: x 행렬 (shape: (n\_sampls, n\_cols))

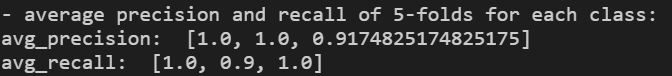
출력: 공분산 행렬, feature별 mean vector

* Multivariate\_gaussian :

입력: x\_vector, mean\_vector, 공분산 행렬

출력: mean\_vector, 공분산 행렬을 이용하여 multivariate gaussian을 가정하고, x\_vector를 입력으로하는 multivariate gaussian 함수 값 리턴



결과: 

* Analysis:

pmf, gaussian을 이용한 bayesian decision 방법은, 각 feature들이 독립이라고 가정하여 각 feature마다 각각의 likelihood를 구하여 곱하는 방식이었다.

Multi-variate gaussian model에서는 각 feature가 독립이라고 가정하지 않는 점에서 큰 차이점이 있다고 생각한다. Feature간의 독립을 가정하지 않으므로, 각 feature별로 likelihood 값을 곱하는 것이 아니라, 여러 feature를 입력으로 받는 하나의 multivariate gaussian을 가정하여 likelihood를 구한다.

feature간의 상관관계가 존재할 수 있음을 고려함으로써 precision, recall 결과 또한 향상됨을 볼 수 있다.

1. (Extra credit 10pts max) Develop a simple deep neural network to solve the problem. Compare the classification performance.

케라스 이용하였음.

참고: deepNeuralNetwork.html