과목명 : 인공지능

학번: 201601989 이름: 김진섭

과제 설명 AI_HW1 Bayesian Classifier

Dataset: Random하게 2개의 Gaussian을 이용해서 2-Class Dataset을 만든다.

(요구사항)

1. Data를 train:test로 분할한다.
2. Sklearn의 GaussianNB 함수를 이용해서 Classifier A를 만든다.
3. Prior를 변경해서 또 다른 Classifier B를 만든다.
4. A와 B의 test data에 대한 실제 label 및 모델의 예측 결과를 그림으로 표현한다.
5. A와 B의 test data에 대한 accuracy를 계산해서 비교한다.

데이터 생성 Dataset : Random하게 2개의 Gaussian을 이용해서 2-class Dataset 생성

```
means = [(2,2), (-2,-2)]
covs = [[[2,0],[0,2]],[[2,0],[0,2]]]
```

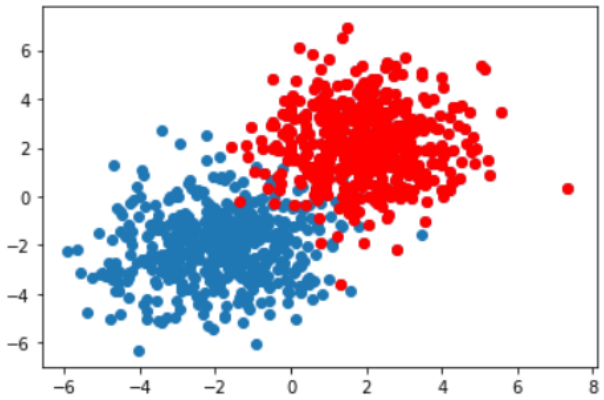
```
class_1 = np.random.multivariate_normal(means[0], covs[0], 500)
class_2 = np.random.multivariate_normal(means[1], covs[1], 500)
```

```
label_1 = np.ones(500)
label_2 = -1*np.ones(500)
```

```
datax = np.concatenate((class_1,class_2), axis=0)
datay = np.concatenate((label_1,label_2), axis=0)
```

```
plt.scatter(datax[:,0], datax[:,1])
plt.scatter(class_1[:,0], class_1[:,1], c="red")
```

```
<matplotlib.collections.PathCollection at 0x25b3d459dd8>
```



설명

데이터를 랜덤하게 2개의 정규분포를 이용하여 2-class Dataset을 생성하였다.
각각 평균과 분산을 위 means와 cons에 넣어주었고, class_1과 class_2로 500개의 데이터를 생성하였다.
그 뒤 각 클래스에 label을 붙여주는데 class_1은 1을, class_2는 –1로 구분하도록 붙여주었다.
두 데이터를 concatenate로 합쳐서 총 1000개의 데이터를 위처럼 그림으로 보여주었다.

요구사항 1 Data를 train:test로 분할한다.

```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(datax, datay, test_size=0.3, random_state=500)
```

```
print(train_x.shape)
print(test_x.shape)
print(train_y.shape)
print(test_y.shape)
```

```
(700, 2)
(300, 2)
(700,)
(300,)
```

설명

데이터를 train 데이터와 test데이터로 분할하는 과정이다.
sklearn의 train_test_split을 활용하였고, test의 size를 0.3으로 하여 총 300개의 데이터를 테스트 데이터로 사용하기로 분류하였다.
아래에 shape를 출력하여 데이터가 분리된 개수를 확인해 주었다.

요구사항 2 Sklearn의 GaussianNB 함수를 이용해서 Classifier A를 만든다.

```
from sklearn.naive_bayes import GaussianNB
bayes = GaussianNB()
bayes.fit(train_x,train_y)
predict_1 = bayes.predict(test_x)
```

설명

GaussianNB 함수를 활용하여 분류기를 만들고, test 데이터를 예측한 결과를 predict_1에 넣어주었다.

요구사항 3 Prior를 변경해서 또 다른 Classifier B를 만든다.

```
bayes_prior = GaussianNB(priors=[0.2, 0.8])
bayes_prior.fit(train_x, train_y)
predict_2 = bayes_prior.predict(test_x)
```

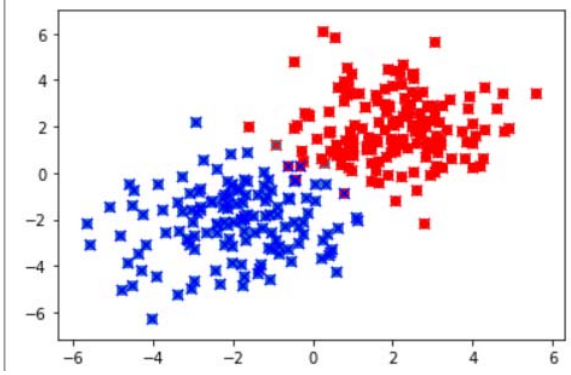
설명

Prior를 주어 분류기를 생성하고, test 데이터를 예측한 결과를 predict_2에 넣어주었다.

요구사항 4-1 A의 test data에 대한 실제 label 및 모델의 예측 결과를 그림으로 표현한다.

```
#prior가 동일한 경우
plt.scatter(test_x[:,0], test_x[:,1])
plt.scatter(test_x[np.where(test_y==1),0], test_x[np.where(test_y==1),1], c="red")
plt.scatter(test_x[np.where(predict_1==1),0], test_x[np.where(predict_1==1),1], c="red", marker='x', s=50)
plt.scatter(test_x[np.where(predict_1==-1),0], test_x[np.where(predict_1==-1),1], c="blue", marker='x',s=50)
```

```
<matplotlib.collections.PathCollection at 0x25b3f8e46a0>
```



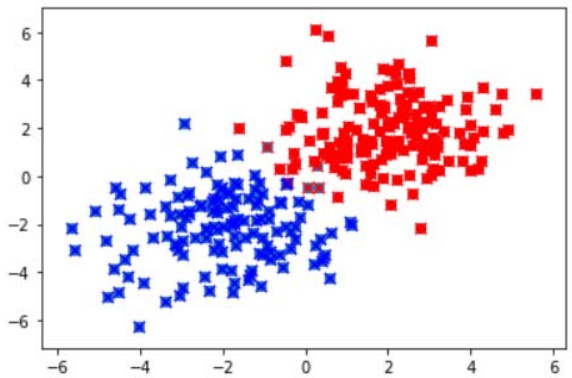
코드 설명

Prior가 동일한 경우의 예측 결과이다. 위에서 marker로 x표시되어있는 것이 오분류 된 데이터이다.

요구사항 4-2 B의 test data에 대한 실제 label 및 모델의 예측 결과를 그림으로 표현한다.

```
#prior가 0.2, 0.8인 경우
plt.scatter(test_x[:,0], test_x[:,1])
plt.scatter(test_x[np.where(test_y==1),0], test_x[np.where(test_y==1),1], c="red")
plt.scatter(test_x[np.where(predict_2==1),0], test_x[np.where(predict_2==1),1], c="red", marker='x', s=50)
plt.scatter(test_x[np.where(predict_2==-1),0], test_x[np.where(predict_2==-1),1], c="blue", marker='x',s=50)
```

```
<matplotlib.collections.PathCollection at 0x25b3f966128>
```



코드 설명

prior가 다른 경우에 대한 결과를 그림으로 표현한 것이다. marker로 x표시되어있는 것이 오분류 된 데이터이다.

요구사항 5 A와 B의 test data에 대한 accuracy를 계산해서 비교한다.

```
# prior가 동일한 경우
dif = test_y - predict_1
accuracy_1 = 1 - (np.size(np.where(dif != 0))/np.size(test_y))
print(accuracy_1)
```

```
0.98
```

```
# prior가 0.2, 0.8인 경우
dif = test_y - predict_2
accuracy_2 = 1 - (np.size(np.where(dif != 0))/np.size(test_y))
print(accuracy_2)
```

```
0.9833333333333333
```

설명

Prior가 동일한 경우와 그러지 않은 경우의 분류기의 예측결과를 계산하여보았다.
prior가 동일한 경우는 0.98, prior가 동일하지 않은 경우 0.98333... 이라는 결과가 나왔다.
결과를 보면 prior를 준 결과가 더 좋게 나타났는데, 이는 prior를 어떠한 값을 주느냐에 따라 달라지게 된다.
수행결과 prior가 영향을 끼치는 것을 알 수 있다.