01 - Naive Bayesian - Gaussian - Lab

August 25, 2021

1 01 - Naive Bayesian - Gaussian - Lab

===Task===

- Generate a 2 class data using sklearn.
- Put Gaussian Naive Classification into class
- Fit the model on the data then calculate accuracy accordingly.

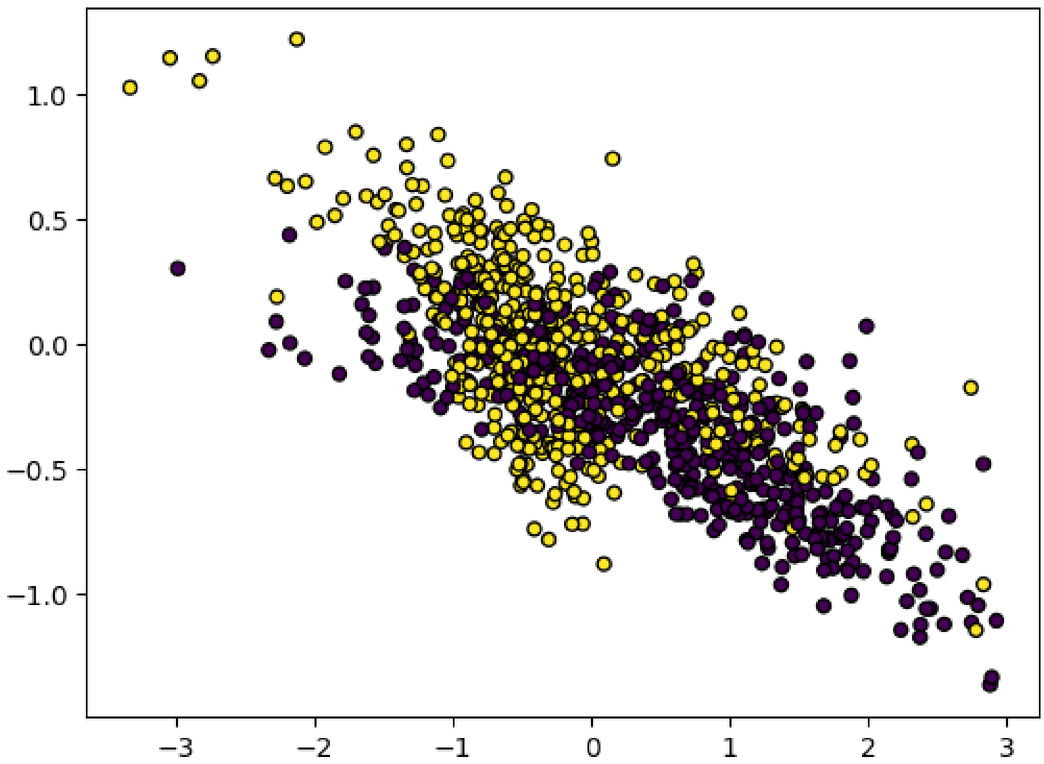
2 Generate a 2 class data using sklearn.

[31]:

```
# <your code here>
import matplotlib.pyplot as plt
import numpy as np

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X, y = make_classification(n_samples=1000, n_features=10,
                            n_informative=3, n_redundant=5,
                           n_classes=2, random_state=42)
plt.scatter(X[:,0], X[:, 1], marker='o', c=y, s=25, edgecolor='k')
plt.show()
```



[32]:
```
# scale the data to have 0 mean and std 1

scaler = StandardScaler()
X = scaler.fit_transform(X)

# split train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[33]:
```
# define a function to find mean and std of each class
def mean_std(X, y):
    n = X.shape[1]

    mean = np.zeros((2, n))
    std = np.zeros((2, n))

    for label in [0, 1]:
        mean[label, :] = X[y == label].mean(axis=0)
        std[label, :] = X[y == label].std(axis=0)
    return mean, std

# dedine a function to get gaussian pdf given X, mean and std
```

```
def gaussian_pdf(X, mean , std):
    pdf = 1/(np.sqrt(2 * np.pi * std**2)) * np.exp(-1/2 * ((X - mean)/std)**2)
    return pdf
```

3 Put Gaussian Naive Classification into class

[34]:

```
class GaussianNB:
    def __init__(self):
        self.mean = None
        self.std = None
        self.m = None
    def fit(self, X, y):

        self.m = X.shape[0]
        # get the mean and std
        self.mean, self.std = mean_std(X, y)

        # get the prior
        self.prior0 = len(y[y==0]) / self.m
        self.prior1 = len(y[y==1]) / self.m


    def predict(self, X, y):
        # get the likelihood for each feature of class 0 and 1
        px_y0 = gaussian_pdf(X, self.mean[0, :], self.std[0, :])
        px_y1 = gaussian_pdf(X, self.mean[1, :], self.std[1, :])

        # multiply all likelihood features for each class
        likelihood0 = px_y0.prod(axis=1)
        likelihood1 = px_y1.prod(axis=1)

        # multiply with prior to get posterior
        posterior0 = self.prior0 * likelihood0
        posterior1 = self.prior1 * likelihood1

        yhat = 1 * posterior1 > posterior0
        return yhat
```

4 Fit the model on the data then calculate accuracy accordingly.

[35]:

```
cls = GaussianNB()
cls.fit(X_train, y_train)
yhat = cls.predict(X_test, y_test)
```

```
[36]: from sklearn.metrics import average_precision_score, classification_report
      print("=========Average precision score=======")
      print(average_precision_score(y_test, yhat))
      print("=========Classification report=======")
      print("Report: ", classification_report(y_test, yhat))
```

```
=========Average precision score=======
0.8689678742310321
=========Classification report=======
Report:                precision    recall  f1-score   support

           0       0.84      0.98      0.90       105
           1       0.97      0.79      0.87        95

    accuracy                           0.89       200
   macro avg       0.91      0.89      0.89       200
weighted avg       0.90      0.89      0.89       200
```

```
[ ]:
```