گزارش‌کار

شناسایی آماری الگو

تمرین ۳

آرمین خیاطی

9931153

سبحان نامی

9831767

1399/10/13

بخش اول

در این بخش قصد داریم با استفاده از تابع توزیع گوسی ۲ دیتاست مجزا با استفاده از پارامترهای زیر تولید و با مدل Bayesian آنها را classify کنیم.

Dataset#1:

$$\text{Class1:} \quad \mu = \begin{pmatrix} 3 \\ 6 \end{pmatrix} \quad \sum = \begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix}$$

$$\text{Class2:} \quad \mu = \begin{pmatrix} 5 \\ 4 \end{pmatrix} \quad \sum = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

$$\text{Class3:} \quad \mu = \begin{pmatrix} 6 \\ 6 \end{pmatrix} \quad \sum = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Dataset#2:

$$\text{Class1:} \quad \mu = \begin{pmatrix} 3 \\ 6 \end{pmatrix} \quad \sum = \begin{pmatrix} 1.5 & 0.1 \\ 0.1 & 0.5 \end{pmatrix}$$

$$\text{Class2:} \quad \mu = \begin{pmatrix} 5 \\ 4 \end{pmatrix} \quad \sum = \begin{pmatrix} 1 & -0.20 \\ -0.20 & 2 \end{pmatrix}$$

$$\text{Class3:} \quad \mu = \begin{pmatrix} 6 \\ 6 \end{pmatrix} \quad \sum = \begin{pmatrix} 2 & -0.25 \\ -0.25 & 1.5 \end{pmatrix}$$

پیاده سازی

در این تمرین از زبان پایتون و کتابخانه‌های Sklearn، Pandas، Numpy, `scipy` و Matplotlib برای خواندن فایل ها، انجام پیش پردازش ها و محاسبات و رسم نمودار ها استفاده شده است. کد این بخش از تمرین در فایل Part1.ipynb موجود است. در ادامه به تشریح توابع نوشته شده می پردازیم.

تولید دیتاست

generate_dataset

```
################################ DATA Generator SECTION ##################################
################################ EXECUTED ONLY ONCE     ##################################
################################ DATA ARE IN CSV FILES  ##################################

mu_s_1 = [[3, 6],
      [5, 4],
      [6, 6]]
cov_s_1 = [[[1.5, 0], [0, 1.5]],
          [[2, 0], [0, 2]],
          [[1, 0], [0, 1]]]


mu_s_2 =[[3, 6],
        [5, 4],
        [6, 6]]
cov_s_2 = [[[1.5, 0.1], [0.1, 1.5]],
          [[1, -0.20], [-0.20, 2]],
          [[2, -0.25], [-0.25, 1.5]]]

SIZE = 500

def generate_dataset(mu_s, cov_s, label_sampels_size):
   dataset = pd.DataFrame(data={'X1': [], 'X2': [], 'Y':[]})
   for i, mu_cov in enumerate(zip(mu_s, cov_s)):
       mu, cov = mu_cov
       x1, x2 = np.random.multivariate_normal(mu, cov, label_sampels_size).T
       temp = pd.DataFrame(data={'X1': x1, 'X2': x2, 'Y': [i]*label_sampels_size})
       dataset = pd.concat([dataset, temp],axis=0)
   return dataset
```

این تابع با دریافت مقادیر مختلف برای توزیع گوسی دیتاست مورد نظر را تولید می‌کند.

مدل کردن دیتاست‌ها

load_data

```
def load_data(path):
 data = pd.read_csv(path)
 X_train, X_test, y_train, y_test =
train_test_split(data[['X1','X2']],data['Y'], test_size=0.2,
stratify=data['Y'] ,random_state=42)
 return X_train, X_test, y_train, y_test
```

data_info

```
def data_info(X, y):
   n_features = X.shape[1]
   classes = np.unique(y)
   classes.sort()
   means = []#np.asmatrix(np.zeros((classes.size, n_features)))
   priors = np.zeros(classes.size)
   return (n_features, classes, means, priors)
```

این تابع وظیفه جمع آوری اطلاعات دیتاست را دارد.

class_covariance

```
def class_covariance(X, mean):
   return (X - mean).T @ (X - mean) / X.shape[0]
```

محاسبه covariance هر کلاس.

fit

```
def fit(X_train, y_train):
   n_features, classes, means, priors = data_info(X_train, y_train)
   cov_matrices = []
   for i, y in enumerate(classes):
       priors[i] = y_train[y_train == y].size / y_train.size
       mean = np.mean(X_train[y_train == y], axis=0)
       means.append(np.asmatrix(mean))
       cov_matrices.append(class_covariance(np.asmatrix(X_train[y_train ==
y].values), means[i]))
   return means, priors, cov_matrices, classes
```

مدل کردن دیتاست.

probability

```
def probability(X, mean, prior, covariance_matrix):
      X = np.asmatrix(X.values)
   cov_matrix_det = np.linalg.det(covariance_matrix)
   cov_matrix_inv = np.linalg.pinv(covariance_matrix)
   Xm = X - mean
   Xm_covariance = (Xm @ cov_matrix_inv) @ Xm.T
   Xm_covariance_sum = Xm_covariance.sum(axis=1)
   return -0.5*Xm_covariance_sum - 0.5*np.log(cov_matrix_det) +
```

```
np.log(prior)
```

تابع محاسبه احتمال برای پیشبینی.

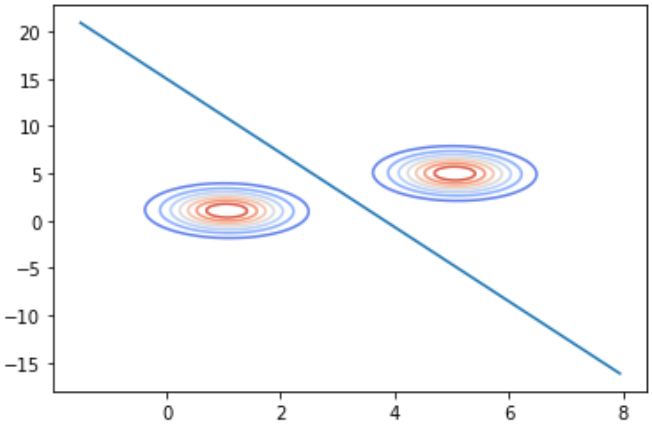
predict

```
def predict(X, means, priors, cov_matrices, classes):
    probs = np.asmatrix(np.zeros((X.shape[0], priors.size)))
    for i, _ in enumerate(classes):
        probs[:, i] = probability(X, means[i], priors[i], cov_matrices[i])
    probs_arg_max = np.argmax(probs, axis=1)
    return probs_arg_max
```

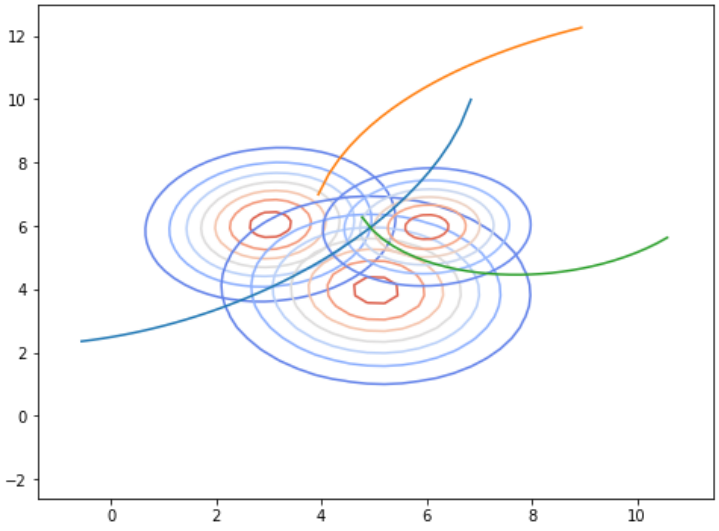
پاسخ سوالات

1- What is the main difference between two datasets? Explain your answer using your results and plots. compare this part with partb of homework#2?

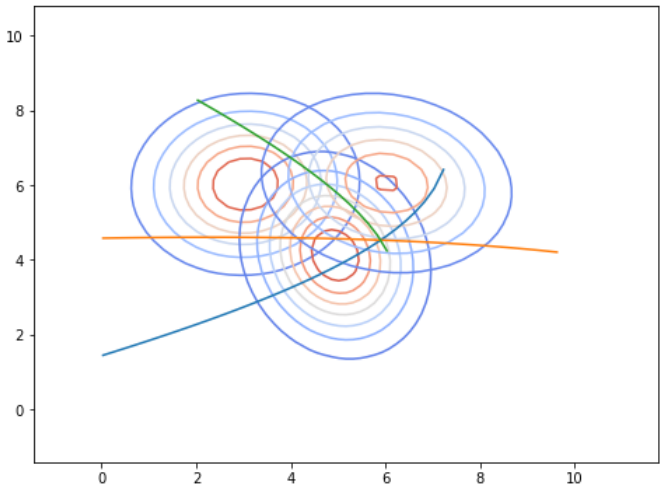
تفاوت دیتاست که در این تمرین ایجاد شده با دیتاست تمرین قبلی در ماتریس های کوواریانس آن می باشد. در تمرین قبلی ماتریس کوواریانس برای همه کلاس ها یکسان بود و این باعث میشد که داده ها در کلاستر هایی به سایز و شکل یکسان حول میانگینشان قرار بگیرند. به همین دلیل رفتار داده ها یکسان و مرز تصمیم خطی است.



اما در این تمرین، داده های هر کلاس ماتریس کوواریانس خود را دارند و مرز تصمیم دیگر خطی نیست. قطر فرعی نیز در دیتاست دوم صفر نبود و این یعنی فیچر ها با هم وابستگی دارند و مستقل نیستند. کلاس هایی هم که مقادیر قطر اصلی ماتریس کوواریانس آن ها متفاوت هستند باعث می‌شود شکل Countor ما بیضوی و به سمت ویژگی ای که مقدار بزرگتری دارد کشیده شود. به دلیل متفاوت بودن ماتریس کوواریانس برای هر کلاس، سطح مقطع ما یک سطح مقطع Hyperquadratic از هر فرمی مانند hyperplanes, جفتی از hyperplane ها ، Hypepararaboloids، hyperellipsoid، hypersphere و hyperhyperboloid میشود. به دلیل صفر نبودن قطر فرعی کووریانس های دیتاست دوم مشاهده می‌کنید که Contour های بیضوی شکل کاملا در جهت افقی یا عمودی کشیده نشده‌اند و کمی چرخش داشته اند. اما در دیتاست اول یا در تمرین قبلی این چرخش را به دلیل صفر بودن قطر فرعی، نداریم و Contour ها فقط در جهت افقی یا عمودی کشیده شده اند.



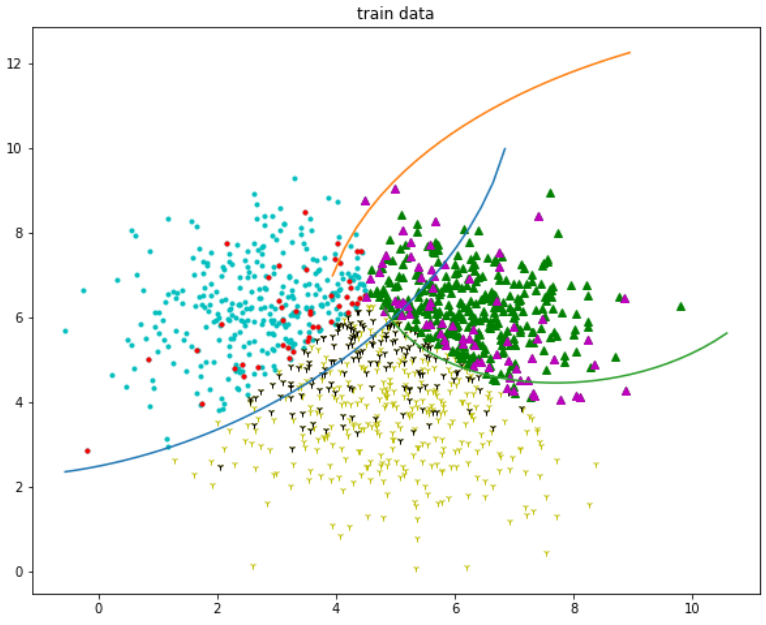
دیتاست اول



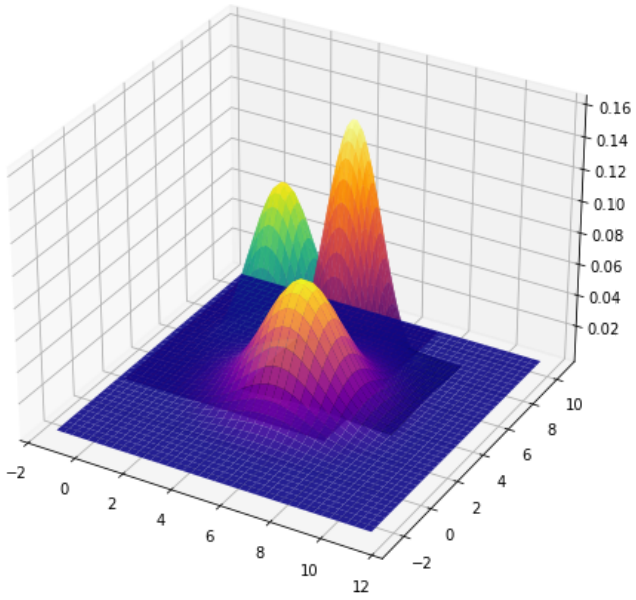
دیتاست دوم

نتیجه

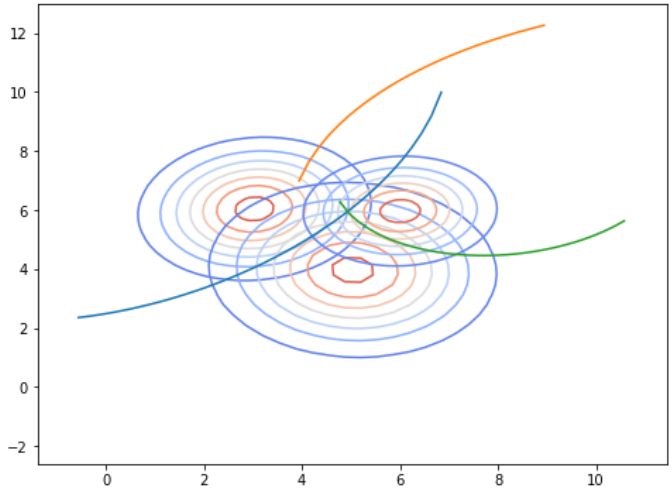
Test Accuracy	Train Accuracy	
0.7966666666666666	0.7991666666666667	dataset1
0.7333333333333333	0.7666666666666667	dataset2



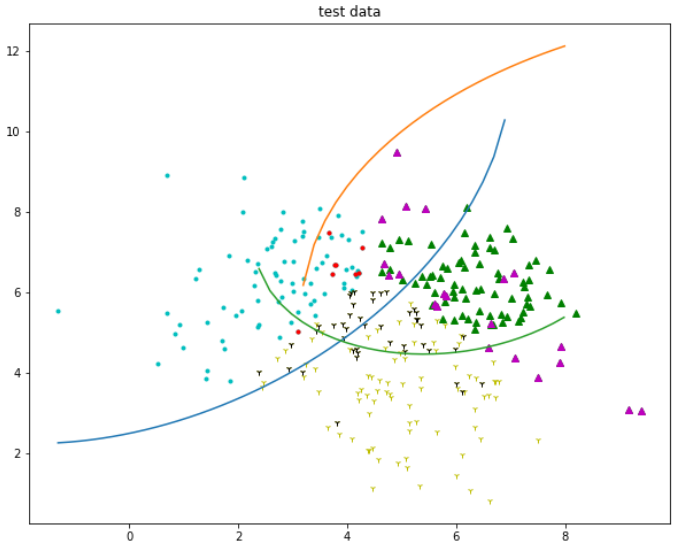
نموداری decision boundary دیتاست ۱ train



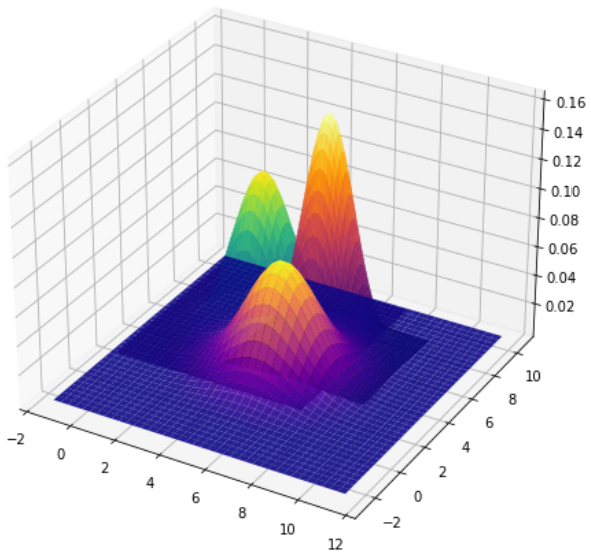
نمودار PDF سه بعدی دیتاست ۱ train



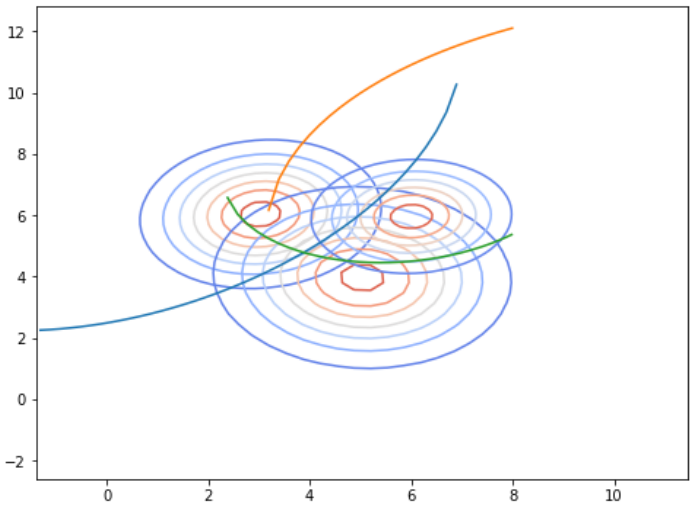
نمودار PDF دوبعدی دیتاست ۱ train



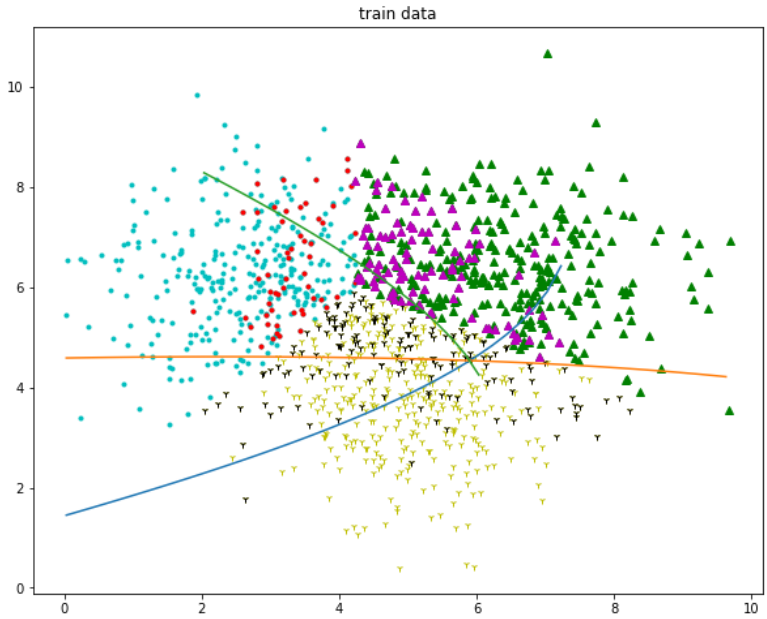
نمودار Decision Boundary دیتاست ۱ test



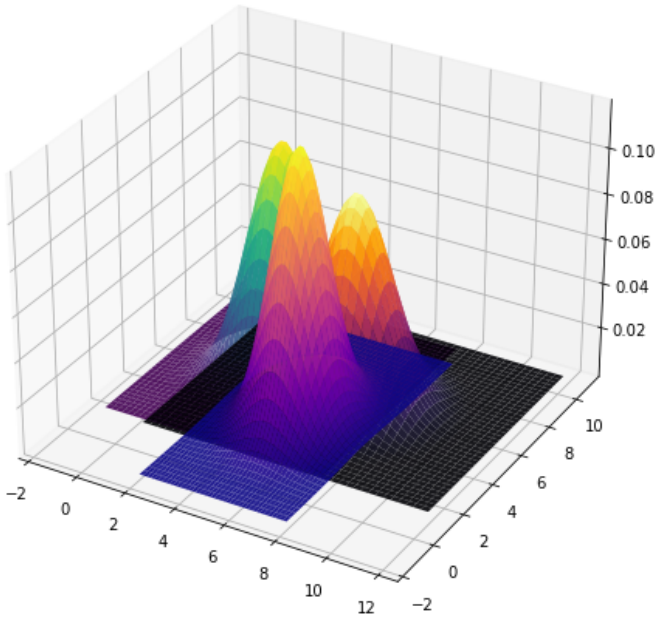
نمودار PDF سه بعدی دیتاست ۱ test



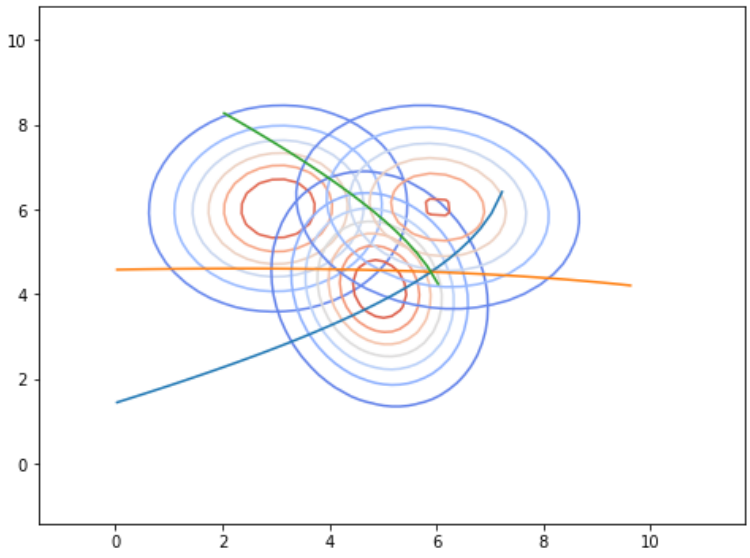
نمودار PDF دو بعدی دیتاست ۱ test



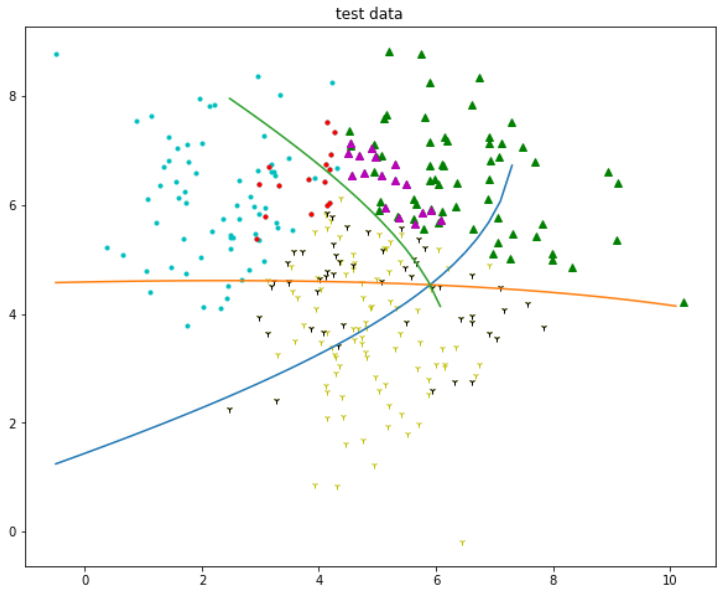
نمودار Decision Boundary دیتاست ۲ train



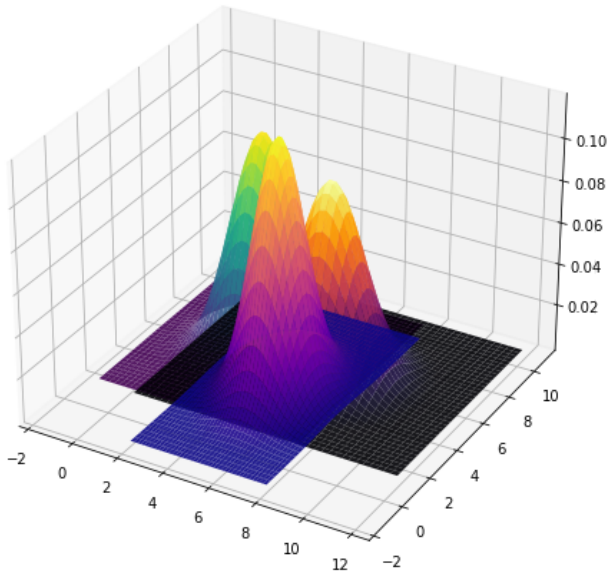
نمودار PDF سه بعدی دیتاست ۲ train



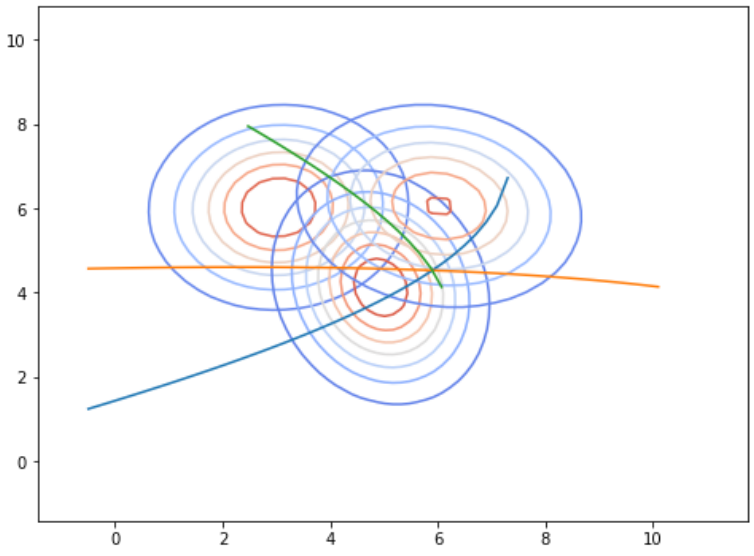
نمودار PDF دو بعدی دیتاست ۲ train



نمودار Decision Boundary دیتاست ۲ test



نمودار PDF سه بعدی دیتاست ۲ test



نمودار PDF دو بعدی دیتاست test ۲

بخش دوم

در این بخش شما باید یک طبقه بندی Naive Bayes را برای تشخیص احساسات مثبت و منفی از بررسی‌ها پیاده سازی کنید. دیتاست مورد نظر "Sentiment Labelled Sentences" می‌باشد. این مجموعه داده حاوی جملاتی با برچسب احساسات مثبت یا منفی است که از بررسی محصولات ، فیلم ها و رستوران ها استخراج شده است.

پیاده سازی

در این تمرین از زبان پایتون و کتابخانه‌های Sklearn ،Pandas ،Numpy, و nltk برای خواندن فایل ها، انجام پیش پردازش ها و محاسبات و رسم نمودار ها استفاده شده است. کد این بخش از تمرین در فایل Part2.ipynb موجود است. در ادامه به تشریح توابع نوشته شده می پردازیم.

load_data

```
def load_data(path):
    with open(path,"r") as text_file:
        lines = text_file.read().split('\n')
    lines = [line.split("\t") for line in lines if len(line.split("\t"))==2
and line.split("\t")[1]!='']
    train_sentences = [preprocess(line[0]) for line in lines]
    train_labels = [int(line[1]) for line in lines]
    return pd.DataFrame({'line': train_sentences, 'label': train_labels})
```

خواندن داده‌های دیتاست.

cleanpunc

```
def cleanpunc(line):
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',line)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    return  cleaned
```

حذف حروف اضافه از دیتا.

cleanstop

```
def cleanstop(line):
    stop = set(stopwords.words('english'))
    filtered_sentence = []
```

```
    for w in line.split():
        if w.isalpha() and len(w)>2:
            w = w.lower()
            if(w not in stop):
                filtered_sentence.append(w)
    return filtered_sentence
```

حذف stop word های زبان.

stemming

```
def stemming(word):
    snowball  = nltk.stem.SnowballStemmer('english')
    return (snowball.stem(word.lower())).encode('utf8')
```

تبدیل کلمات به ریشه اصلی.

word_count

```
def word_count(data, label):
    for line, label in zip(data.values, label.values):
        word_counts = collections.Counter(line)
        for word, count in sorted(word_counts.items()):
            if word not in WORD_COUNTS.keys():
                WORD_COUNTS[word] = count
            else:
                WORD_COUNTS[word] += count
            if word not in POS_NEG_WORDS[label].keys():
                POS_NEG_WORDS[label][word] = count
            else:
                POS_NEG_WORDS[label][word] += count
```

محاسبه تعداد تکرار (فرکانس) کلمات در دیتاست.

prior

```
def prior(y_data, classes):
    priors = np.zeros(len(classes))
    for i, c in enumerate(classes):
        priors[i] = len(y_data[y_data==c]) / len(y_data)
    return priors
```

محاسبه احتمال خلفی هر کلاس.

p_x_y

```
def p_x_y(X,y):
```

```
    result = []
    label_total = 0
    for w in POS_NEG_WORDS[y].keys():
        label_total += POS_NEG_WORDS[y][w]
    for j, line in enumerate(X.values):
        result.append(_p_x_Y(line, y, label_total))
    return np.array(result)

def _p_x_Y(line, label, label_total):
    prob = 0
    for w in line:
        p = 0
        if w in POS_NEG_WORDS[label].keys():
            p = (POS_NEG_WORDS[label][w] / label_total)+0.1
        else:
            p = 0.1
        prob += np.log(p)
    return prob
```

محاسبه احتمال x به شرط لیبل y

fit

```
def fit(X_data, y_data):
    classes = np.unique(y_data)
    classes.sort()
    priors = prior(y_data, classes)
    result = np.zeros((X_data.shape[0], len(classes)))
    for i, y in enumerate(classes):
        X = X_data#[y_data == y]
        pxy_py = p_x_y(X, y) + priors[i]
        result[:,i] =  pxy_py
    return np.argmax(result, axis=1)
```

مدل کردن دیتاست.

پاسخ سوالات

1. if you multiply many small probabilities you may run into problems with numeric precision, what is the problem? To handle it, we recommend that you compute the logarithms of the probabilities instead of the probabilities. Explain why this approach can help?

در ضرب تعدادی زیادی احتمال که اعدادی بین صفر و یک هستند، ممکن است جواب نهایی خیلی کوچک شود و به سمت صفر میل کند. در این صورت در محاسبات دچار مشکل خواهیم شد. به همین دلیل برای حل این مشکل از تکنیک لگاریتم گیری استفاده میکنیم تا اعمال ضرب به جمع تبدیل شده و حاصل به صفر میل نکند.

2. What is the base assumption of Naïve Bayes classifier? Why is it important?

فرض اصلی کلاسیفایر naïve bayes مستقل بودن ویژگی ها از هم دیگر و عدم تاثیر یک ویژگی بر ویژگی دیگر است. مزیت این فرض این است که این کلاسیفایر از دیگر مدل ها مانند Logistic Regression بهتر عمل کرده، دیتای کمتری نیاز خواهد داشت و سریع تر همگرا خواهد شد.

نتیجه

amazon	imdb	yelp	
0.6725	0.8575	0.835	**Train class 0**
0.9575	0.79	0.8775	**Train class 1**
0.815	0.82375	0.85625	**Train Total**
0.49	0.73	0.67	**Test class 0**
0.92	0.7	0.84	**Test class 1**
0.705	0.715	0.755	**Test Total**