2주차 실습 과제

2017313107 이승태

1. 4장_Titanic dataset 탐색 및 전처리

(1) import 하기

```
[1]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import sklearn
     from sklearn.metrics import *
     from pandas import DataFrame, Series

     plt.style.use('seaborn')
     sns.set(font_scale=2.5)
```

(2) 데이터를 불러오고 df_train, df_test에 저장한 후, 데이터의 shape과 feature들을 출력

```
[2]  #캐글사이트(https://www.kaggle.com/c/titanic/data)에서 데이터 파일을 다운로드 받은 경로에 맞게 설정하여 데이터 로드.
     df_train = pd.read_csv('train.csv')
     df_test = pd.read_csv('test.csv')
```

```
[3]  #학습데이터, 테스트데이터의 차원 조회 (테스트데이터에는 'survived' 칼럼이 없음)
     df_train.shape, df_test.shape
```

```
((891, 12), (418, 11))
```

```
[4]  columns=df_train.columns
     columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

(3) 0~4번의 index를 갖는 데이터 보기

```
[5]  df_train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
[6]  df_test.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

(4) 각 feature들의 data type보기

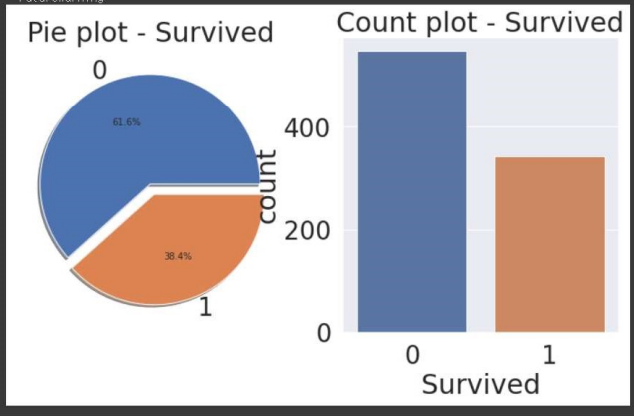
```
[7]  df_train.dtypes

     PassengerId      int64
     Survived         int64
     Pclass           int64
     Name            object
     Sex             object
     Age            float64
     SibSp            int64
     Parch            int64
     Ticket          object
     Fare           float64
     Cabin           object
     Embarked        object
     dtype: object
```

(5) servived의 개수 보기

```
[8]  df_train['Survived'].value_counts() #target value로 사용할 컬럼인 'Survived'의 클래스 별 데이터 수

     0    549
     1    342
     Name: Survived, dtype: int64
```

(6) survived feature에 대한 데이터 비율 가시화 하기

```
[9]  #'Survived' 컬럼 시각화
     f, ax = plt.subplots(1, 2, figsize =(12, 6))
     df_train["Survived"].value_counts().plot.pie(explode=[0,0.1], autopct="%1.1f%%", ax=ax[0], shadow=True)
     ax[0].set_title("Pie plot - Survived")
     ax[0].set_ylabel('')
     sns.countplot("Survived", data=df_train, ax=ax[1])
     ax[1].set_title("Count plot - Survived")

     plt.show()

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as
  FutureWarning
```



(7) 결측지가 얼마나 있는지 확인

```
[10] df_train.isnull().sum()

PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64

[11] df_test.isnull().sum()

PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

(8) 칼럼들을 제거하고 결측지를 평균값으로 대체하고 위와 마찬가지로 결측지의 개수를 확인

```
[12] #이번 실습에서 학습에 사용하지 않을 컬럼 제거
train = df_train.drop(['Cabin', 'Embarked', 'Name', 'Ticket', 'PassengerId'],axis=1)
test = df_test.drop(['Cabin', 'Embarked', 'Name', 'Ticket'],axis=1)

#결측치를 평균값으로 대체
train["Age"].fillna(train.groupby("Sex")["Age"].transform("mean"), inplace=True)
test["Age"].fillna(test.groupby("Sex")["Age"].transform("mean"), inplace=True)
test["Fare"].fillna(test.groupby("Sex")["Fare"].transform("median"), inplace=True)

[13] train.isnull().sum()

Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
dtype: int64

[14] test.isnull().sum()

PassengerId    0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
dtype: int64
```

(9) male과 female처럼 숫자로 나타내지지 않는 것들을 숫자로 바꾸고, 너무 벗어나는 값들은 모두 제거한다.

```
[15] #문자열로 이루어진 'sex' 컬럼은 숫자로 매핑
     sex_mapping = {"male": 0, "female": 1}
     train['Sex'] = train['Sex'].map(sex_mapping)
     test['Sex'] = test['Sex'].map(sex_mapping)

     #'Age', 'Fare' 컬럼에 대하여 평균으로부터 3시그마 안에 포함되지 않는 값들은 이상치로 판단하여 모두 제거
     age_mean = train['Age'].mean()
     age_std = train['Age'].std()
     indexNames = train[train['Age'] < age_mean - 3*age_std].index
     train.drop(indexNames , inplace=True)
     indexNames = train[train['Age'] > age_mean + 3*age_std].index
     train.drop(indexNames , inplace=True)
     fare_mean = train['Fare'].mean()
     fare_std = train['Fare'].std()
     indexNames = train[train['Fare'] < fare_mean - 3*fare_std].index
     train.drop(indexNames , inplace=True)
     indexNames = train[train['Fare'] > fare_mean + 3*fare_std].index
     train.drop(indexNames , inplace=True)
```

2. 4장_로지스틱회귀모형 학습 및 성능 평가

(1) train, test.csv전처리

```
[2]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import sklearn
     from sklearn.metrics import *
     from pandas import DataFrame, Series

     plt.style.use('seaborn')
     sns.set(font_scale=2.5)
     df_train = pd.read_csv('train.csv')
     df_test = pd.read_csv('test.csv')
     train = df_train.drop(['Cabin', 'Embarked', 'Name', 'Ticket', 'PassengerId'],axis=1)
     test = df_test.drop(['Cabin', 'Embarked', 'Name', 'Ticket'],axis=1)
     train["Age"].fillna(train.groupby("Sex")["Age"].transform("mean"), inplace=True)
     test["Age"].fillna(test.groupby("Sex")["Age"].transform("mean"), inplace=True)
     test["Fare"].fillna(test.groupby("Sex")["Fare"].transform("median"), inplace=True)
     sex_mapping = {"male": 0, "female": 1}
     train['Sex'] = train['Sex'].map(sex_mapping)
     test['Sex'] = test['Sex'].map(sex_mapping)
     age_mean = train['Age'].mean()
     age_std = train['Age'].std()
     indexNames = train[train['Age'] < age_mean - 3*age_std].index
     train.drop(indexNames , inplace=True)
     indexNames = train[train['Age'] > age_mean + 3*age_std].index
     train.drop(indexNames , inplace=True)
     fare_mean = train['Fare'].mean()
     fare_std = train['Fare'].std()
     indexNames = train[train['Fare'] < fare_mean - 3*fare_std].index
     train.drop(indexNames , inplace=True)
     indexNames = train[train['Fare'] > fare_mean + 3*fare_std].index
     train.drop(indexNames , inplace=True)
     #'4장_Titanic dataset 탐색 및 전처리.ipynb'에서 전처리 코드만 실행하는 부분
```

(2) survived했는지 안했는지를 label로 보기 위해 그 열을 빼낸후 label로 쓴다.

```
[3]  from sklearn.linear_model import LogisticRegression
     from sklearn import metrics
     from sklearn.model_selection import train_test_split

     X_train = train.drop('Survived', axis=1).values
     target_label = train['Survived'].values
     X_test = test.values
```

(3) 그 후 train set과 test set을 분리했다.

```
[4]  X_train.shape, X_test.shape

     ((864, 6), (418, 7))

[5]  X_tr, X_vld, y_tr, y_vld = train_test_split(X_train, target_label, test_size=0.2, random_state=2020)
     y_tr.shape, y_vld.shape

     ((691,), (173,))
```

(4) logistic regression으로 model을 생성하고, 모델을 학습시킨다(model.fit) 그후, model에 test set의 x를 넣어 값을 예측한다.

```
[6]  model = LogisticRegression()
     model.fit(X_tr, y_tr)
     prediction = model.predict(X_vld)

[7]  prediction

     array([1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
            1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
            0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
            0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
            0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
            0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
            1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1,
            1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0])
```

(5) test의 label과 비교해 accuracy를 측정한다. 그 후 confisuion matrix를 생성한 다음, 이를 이용하여 Precision, Recall, F1-score를 측정한다.

```
[8]  print('Number of people: {} \naccuracy: {:.2f}% '.format(y_vld.shape[0], 100 * accuracy_score(y_vld,prediction)))

     Number of people: 173
     accuracy: 78.03%

[9]  confusion_matrix(y_vld,prediction)

     array([[94, 24],
            [14, 41]])

[10] print('Precision: {:.2f}% \nRecall: {:.2f}% \nF1-score: {:.2f}% '.format(100*precision_score(y_vld,prediction),
                                                                             100*recall_score(y_vld,prediction),100*f1_score(y_vld,prediction)))

     Precision: 63.08%
     Recall: 74.55%
     F1-score: 68.33%
```

(6) cut off를 조절하여 모델의 성능을 비교해보았다.

```
#cut off 조절에 따른 모델의 성능을 평가해 보기 위하여 cut off
#cut off 값은 다양하게 선택 가능.
list = []
for i in np.linspace(0,1,100):
    pred = model.predict_proba(X_vld)[:,1] > i
    cf_mtx = confusion_matrix(y_vld, pred)
    acc = accuracy_score(y_vld, pred)
    tpr = cf_mtx[0,0] / cf_mtx[0].sum()
    fpr = cf_mtx[1,0] / cf_mtx[1].sum()
    f1 = f1_score(y_vld, pred)
    list.append([i, acc, f1, tpr, fpr])

cut_off = DataFrame(list)
cut_off.columns = ["CUTOFF", "ACC", "F1", "TPR", "FPR"]
cut_off
```

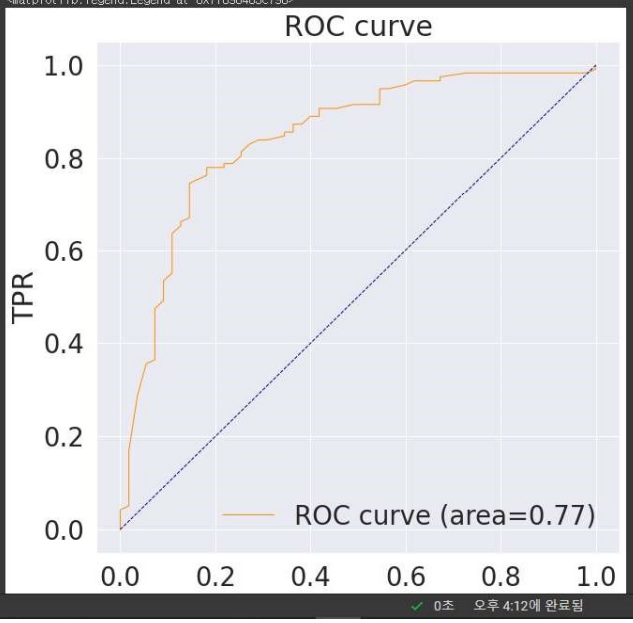
	CUTOFF	ACC	F1	TPR	FPR
0	0.000000	0.317919	0.482456	0.000000	0.000000
1	0.010101	0.323699	0.484581	0.008475	0.000000
2	0.020202	0.323699	0.484581	0.008475	0.000000
3	0.030303	0.329480	0.486726	0.016949	0.000000
4	0.040404	0.329480	0.486726	0.016949	0.000000
...	...	...	...	...	...
95	0.959596	0.687861	0.100000	0.983051	0.945455
96	0.969697	0.687861	0.100000	0.983051	0.945455
97	0.979798	0.676301	0.034483	0.983051	0.981818
98	0.989899	0.676301	0.000000	0.991525	1.000000
99	1.000000	0.682081	0.000000	1.000000	1.000000

100 rows × 5 columns

(7) 위에서 생성한 모델들을 이용하여 ROC그래프를 그려봤다.

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_vld, prediction)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(10,10))
plt.plot(cut_off["FPR"],cut_off["TPR"], color="darkorange", lw=1, label="ROC curve (area=%.2f)" %roc_auc)
plt.plot([0,1], [0,1], color='navy', lw=1, linestyle='--')
plt.title("ROC curve")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.legend(loc="lower right")
```

```
<matplotlib.legend.Legend at 0x7f896405c790>
```



0초 오후 4:12에 완료됨

(8) 위의 모델들 중 accuracy가 가장 큰 값의 confusion matrix를 만들었다.

```
[12] cut_off[cut_off["ACC"] == cut_off["ACC"].max()] #accuracy가 최대인 값
```

	CUTOFF	ACC	F1	TPR	FPR
70	0.707071	0.803468	0.653061	0.90678	0.418182

```
[13] cut_off_ACC_MAX = cut_off[cut_off["ACC"] == cut_off["ACC"].max()]["CUTOFF"][70]
     cut_off_ACC_MAX

     0.7070707070707072

[14] pred_ACC_MAX = model.predict_proba(X_vld)[:,1] > cut_off_ACC_MAX

[15] confusion_matrix(y_vld,pred_ACC_MAX)

     array([[107,  11],
            [ 23,  32]])
```

(9) F1스코어가 가장 큰 값을 가지는 모델의 confusion matrix를 그려 ㅓ보았다.

```
[16] cut_off[cut_off["F1"] == cut_off["F1"].max()] #F1-score가 최대인 값
```

	CUTOFF	ACC	F1	TPR	FPR
45	0.454545	0.791908	0.714286	0.779661	0.181818

```
[17] cut_off_F1_MAX = cut_off[cut_off["F1"] == cut_off["F1"].max()]["CUTOFF"][45]
     cut_off_F1_MAX

     0.4545454545454546

[18] pred_F1_MAX = model.predict_proba(X_vld)[:,1] > cut_off_F1_MAX

[19] confusion_matrix(y_vld,pred_F1_MAX)

     array([[92, 26],
            [10, 45]])
```

3. 5장_나이브베이즈

(1) import 하기

```
[1]  from sklearn.datasets import load_iris
     import pandas as pd
     import numpy as np
     from matplotlib import pyplot as plt
```

(2) 꽃의 생김새에 따른 분류 데이터를 load한 후 data를 확인해 보았다.

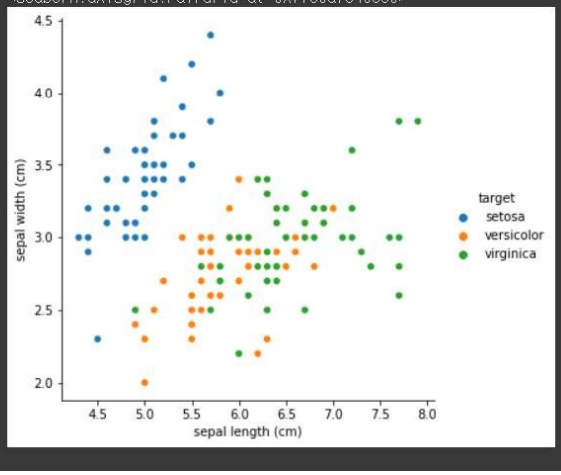
```
[2]  iris = load_iris()
     iris_frame = pd.DataFrame(data=np.c_[iris['data'], iris['target']],columns = iris['feature_names'] + ['target'])
     iris_frame['target'] = iris_frame['target'].map({1:"versicolor",0:"setosa",2:"virginica"})
     X = iris_frame.iloc[:,:-1]
     Y = iris_frame.iloc[:,[-1]]
     iris_frame
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

(3) sepal length와 sepal width에 대해 어떤 분포를 나타내고 있는지 그래프를 그려보았다.

```
[3]  import seaborn as sn

[4]  sn.pairplot(iris_frame,
                 x_vars=["sepal length (cm)"],
                 y_vars=["sepal width (cm)"],
                 hue = "target",
                 size=5)
```

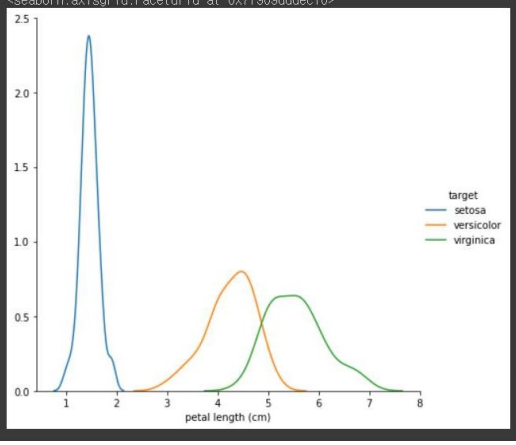
```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:1969:
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.PairGrid at 0x7f90a7940890>
```



(4) petal length에 대한 꽃의 분류를 그래프로 그려보았다.

```
[5]  sn.FacetGrid(iris_frame,
                  hue="target",
                  size=6).map(sn.kdeplot, "petal length (cm)").add_legend()
```

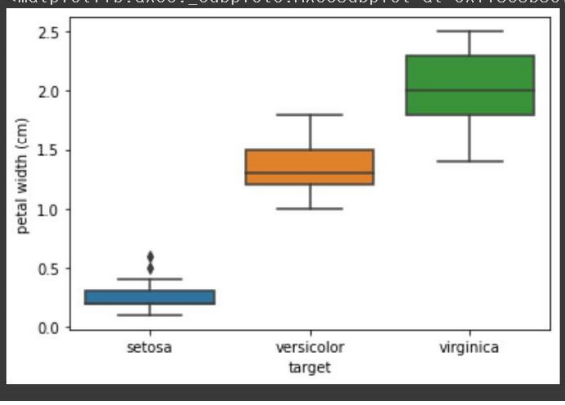
```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:316: UserWarning:
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7f909dddec10>
```



(5) 마찬가지로 petal width에 대한 꽃의 분류를 boxplot을 그려 나타내보았다.

```
[6]  sn.boxplot(x="target",y="petal width (cm)",
                data=iris_frame)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f909b8cfdd0>
```



(6) Naive Bayes classifier를 통해 model을 생성하고, fit해보았다.

```
[7]  import matplotlib.colors as colors
     from sklearn.naive_bayes import GaussianNB
     df1 = iris_frame[["sepal length (cm)" , "sepal width (cm)", "target" ]]
     X = df1.iloc[:,0:2]
     Y = df1.iloc[:,2].replace({'setosa':0,'versicolor':1,'virginica':2}).copy()
     NB=GaussianNB()
     NB.fit(X,Y)
     N=100
```

(7) 가우시안 나이브 모델을 가시적으로 표현해 보았다.

```
[8]  X_= np.linspace(4,8,N)
     Y_= np.linspace(1.5,5,N)
     X_,Y_= np.meshgrid(X_,Y_)

     color_list= ['Blues','Greens','Reds']
     my_norm = colors.Normalize(vmin=-1,vmax=1)
     g= sn.FacetGrid(iris_frame, hue = "target", size =10,
                     palette = 'colorblind').map(plt.scatter, "sepal length (cm)","sepal width (cm)",).add_legend()

     my_ax = g.ax
     zz = np.array( [NB.predict ([[xx,yy]]) [0] for xx,yy in zip(np.ravel(X_),np.ravel(Y_))])
     Z=zz.reshape(X_.shape)

     my_ax.contourf(X_,Y_,Z,2,alpha=.1, colors = ('blue','green','red'))
     my_ax.contour(X_,Y_,Z,2,alpha=1, colors = ('blue','green','red'))

     my_ax.set_xlabel('Sepal length')
     my_ax.set_ylabel('Sepal width')
     my_ax.set_title('Gaussian Naive Bayes boundaries')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:316: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
Text(0.5, 1.0, 'Gaussian Naive Bayes boundaries')
```