CS353 Machine Learning Lab

Lab-2 (05/02/21)

Shumbul Arifa (181CO152)

Task:

Perform naive bayes classification on breast cancer standard dataset.

Attributes:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area - 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension (“coastline approximation” - 1)

The mean, standard error, and “worst” or largest (mean of the three worst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field 10 is Radius SE, field 20 is Worst Radius.

class:

- WDBC-Malignant
- WDBC-Benign

Results:

Accuracy Obtained = 94.73 %

Breast Cancer dataset

The breast cancer dataset is a classic and very easy binary classification dataset.
The dataset is described below:

Features	Quantity
Classes	2
Samples per Class	212(M),357(B)
Samples Total	569
Dimensionality	30
Features	real, positive

Importing Libraries

```
%matplotlib inline
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from scipy.stats import norm
from sklearn.metrics import confusion_matrix, accuracy_score, classification_r
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
```

Loading dataset

```
data = load_breast_cancer()
X, y, col_names = data['data'], data['target'], data['feature_names']
X = pd.DataFrame(X, columns=col_names)
```

```
X.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	cor
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	56(
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	(
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	(
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	(
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	(
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	(
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	(
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	(

```
X.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	me conca poi
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10

Splitting Data

We are using X-y split method with test size 20 % and random state 5.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rand
```

```
X_train.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	co p
306	13.20	15.82	84.07	537.3	0.08511	0.05251	0.001461	0.0
410	11.36	17.57	72.49	399.8	0.08858	0.05313	0.027830	0.0
197	18.08	21.84	117.40	1024.0	0.07371	0.08642	0.110300	0.0
376	10.57	20.22	70.15	338.3	0.09073	0.16600	0.228000	0.0
244	19.40	23.50	129.10	1155.0	0.10270	0.15580	0.204900	0.0

```
X_test.head()
```

mean mean mean mean mean mean mean

Implementing Naive Bayes Classifier

Naive Bayes is a simple probabilistic model. We assume that the dataset is representative of samples in the real world, and assume they have similar distributions. We then look at each value given for the validation dataset, and see how many rows in the training data have similar values. We take this probability as the probability that the validation data belongs to a particular sample.

1. Using python from scratch
2. Using sklearn standard library

Part - 1: Using python from scratch

```
means = X_train.groupby(y_train).apply(np.mean)
stds = X_train.groupby(y_train).apply(np.std)
probs = X_train.groupby(y_train).apply(lambda x: len(x))/X_train.shape[0]
```

```
means
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity
0	17.418841	21.443110	114.933720	971.482927	0.102330	0.141595	0.156754
1	12.172498	17.749175	78.246357	465.044330	0.092968	0.079921	0.044955

```
stds
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity
0	3.112415	3.833583	21.172813	352.429935	0.012418	0.051954	0.071969
1	1.803035	3.871083	11.979191	136.467688	0.013783	0.033661	0.041980

```
probs
```

```
0    0.36044
1    0.63956
dtype: float64
```

Function to perform naive baiyes classification

```
y_pred=[]

for sample in range(X_test.shape[0]): #for each sample in the validation data
  p = {} #start by making an empty dictionary to hold probabilities of belong

  for each_class in np.unique(y_train):
    p[each_class] = probs.iloc[each_class] #the standard probability that it w

    for index, param in enumerate(X_test.iloc[sample]): #enum returns the inde
      p[each_class] *= norm.pdf(param, means.iloc[each_class, index], stds.ilo
      #if we were to bin and solve, we'd get similar results, but it wouldn't

  y_pred.append(pd.Series(p).values.argmax())
```

Computing Accuracy

```
print("Accuracy of our model using python from scrath is: ")
print(accuracy_score(y_test, y_pred))
```

```
Accuracy of our model using python from scrath is:
0.9385964912280702
```

Confusion Matrix

```
confusion_matrix(y_test, y_pred)
```

```
array([[44,  4],
       [ 3, 63]])
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        48
           1       0.94      0.95      0.95        66

    accuracy                           0.94       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.94      0.94      0.94       114
```

Part - 2: Using classifier from sklearn

```
model = GaussianNB()
```

```
model.fit(X_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Computing Accuracy

```
print("Accuracy of our model using sklearn std library is: ")
sk_pred = model.predict(X_test)
print(accuracy_score(y_test, sk_pred))
```

```
Accuracy of our model using sklearn std library is:
0.9473684210526315
```

Confusion Matrix

```
print(classification_report(y_test,sk_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.92      0.94        48
           1       0.94      0.97      0.96        66

    accuracy                           0.95       114
   macro avg       0.95      0.94      0.95       114
weighted avg       0.95      0.95      0.95       114
```

```
plot_confusion_matrix(model, X_test, y_test, display_labels=data.target_names
plt.show()
```