Lista 8 - Classificador de Bayes Aplicado a um Problema Multivariado

Vítor Gabriel Reis Caitité

August 1, 2021

1 Experimento

Neste exercício será resolvido um problema de classificação multivariado utilizando o classificador de bayes. A base de dados será uma real chamada heart [1], que contém os 13 atributos descritos abaixo:

```
- 1. idade
- 2. sexo
- 3. tipo de dor no peito  (4 values)
- 4. pressão arterial em repouso
- 5. colesterol sérico em mg / dl
- 6. açúcar no sangue em jejum > 120 mg / dl
- 7. resultados eletrocardiográficos de repouso (valores 0,1,2)
- 8. frequência cardíaca máxima alcançada
- 9. angina induzida por exercício
- 10. oldpeak = depressão de ST induzida por exercício em relação ao repouso
- 11. a inclinação do segmento ST de pico de exercício
- 12. número de vasos principais (0-3) coloridos por fluoroscopia
- 13. thal: 3 = normal; 6 = defeito corrigido; 7 = defeito reversível
```

Com esse *database* deseja-se prever a variável que indica ausência (1) ou presença (2) de doença cardíaca.

1.1 Carregar base de dados

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from scipy.stats import multivariate_normal
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪accuracy_score
#from warnings import simplefilter
```

```
#simplefilter(action='ignore', category=FutureWarning)
```

[2]:
```
# Loading dataset:
df = pd.read_csv("~/Documents/UFMG/10/Reconhecimento de padrões/list/
↪pattern-recognition-exercises/list_8/databases/heart.dat", header=None,  sep =␣
↪' ');
df.head()
```

[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	70.0	1.0	4.0	130.0	322.0	0.0	2.0	109.0	0.0	2.4	2.0	3.0	3.0	2
1	67.0	0.0	3.0	115.0	564.0	0.0	2.0	160.0	0.0	1.6	2.0	0.0	7.0	1
2	57.0	1.0	2.0	124.0	261.0	0.0	0.0	141.0	0.0	0.3	1.0	0.0	7.0	2
3	64.0	1.0	4.0	128.0	263.0	0.0	0.0	105.0	1.0	0.2	2.0	1.0	7.0	1
4	74.0	0.0	2.0	120.0	269.0	0.0	2.0	121.0	1.0	0.2	1.0	1.0	3.0	1

1.2 Separação em Conjunto de Treino e Teste

Como solicitado no exercício deve-se separar os dados em um conjunto de treinamento com 90% dos dados e um conjunto de testes com 10% dos dados de forma aleatória.

[3]:
```
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
# Separate data between training and test:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
```

1.3 Desenvolvimento e Aplicação do Classificador

[4]:
```
# Bynary Bayes Classifier
class bayes_classifier:

    # To initialize the parameters from the Bayes algorithm:
    def __init__(self):
        self.p_ci = None
        self.X_train = None
        self.y_train = None

    # Training the model
    def fit(self, X, y):
        self.X_train = X
        self.y_train = y
        # Calculate P[C1], P[C2], ... , P[Cn]:
        n = np.unique(y).shape[0]
        self.p_ci =  np.zeros(n)
        for i in range(0,n):
            n_elements = np.count_nonzero(self.y_train==np.unique(self.
↪y_train)[i])
            total_elements = self.y_train.shape[0]
            self.p_ci[i] = n_elements/total_elements
```

```
    def predict(self, X):
        # Calculate PDFs:
        n = np.unique(self.y_train).shape[0]
        mean_ci = np.zeros(self.X_train.shape[1])
        pdf = np.zeros(n)
        y = np.zeros(X.shape[0])
        index = 0;
        for x in X:
            for i in range(0,n):
                indexes = np.where(self.y_train==np.unique(self.y_train)[i])
                for col in range(0, self.X_train.shape[1]):
                    mean_ci[col] = np.mean(self.X_train[indexes,col])
                cov = np.cov(self.X_train, rowvar=False)
                pdf[i] = multivariate_normal.pdf(x, mean_ci, cov)

            K = (pdf[1] * self.p_ci[1])/(pdf[0] * self.p_ci[0])
            if K >= 1:
                y[index] = 2
            else:
                y[index] = 1
            index += 1
        return y
```

[5]:
```
clf = bayes_classifier()
clf.fit(X_train.to_numpy(), y_train.to_numpy())
y_pred = clf.predict(X_test.to_numpy())
```

1.4 Cálculo da Acurácia e da Matriz de Confusão

[6]:
```
df_confusion = confusion_matrix(y_test.to_numpy(), y_pred)
print("Matriz de Confusão:")
print(df_confusion)
```

```
Matriz de Confusão:
[[18  0]
 [ 2  7]]
```

[7]:
```
print("Acurácia: " + str(accuracy_score(y_test,y_pred)))
```

```
Acurácia: 0.9259259259259259
```

2 Repetição do Experimento Utilizando 70% dos Dados para Treinamento

2.1 Separação em Conjunto de Treino e Teste

Como solicitado, separou-se o dataset original em um conjunto de treinamento com 70% dos dados e um conjunto de testes com 30% dos dados de forma aleatória.

[8]:
```
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
# Separate data between training and test:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

2.2 Treinamento e Aplicação do Classificador

[9]:
```
clf = bayes_classifier()
clf.fit(X_train.to_numpy(), y_train.to_numpy())
y_pred = clf.predict(X_test.to_numpy())
```

2.3 Cálculo da Acurácia e da Matriz de Confusão

[10]:
```
df_confusion = confusion_matrix(y_test.to_numpy(), y_pred)
print("Matriz de Confusão:")
print(df_confusion)
```

```
Matriz de Confusão:
[[40  7]
 [ 3 31]]
```

[11]:
```
print("Acurácia: " + str(accuracy_score(y_test,y_pred)))
```

```
Acurácia: 0.8765432098765432
```

3 Repetição do Experimento Utilizando 20% dos Dados para Treinamento

3.1 Separação em Conjunto de Treino e Teste

Como solicitado, separou-se o dataset original em um conjunto de treinamento com 20% dos dados e um conjunto de testes com 80% dos dados de forma aleatória.

[12]:
```
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
# Separate data between training and test:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8)
```

3.2 Treinamento e Aplicação do Classificador

[13]:
```
clf = bayes_classifier()
clf.fit(X_train.to_numpy(), y_train.to_numpy())
y_pred = clf.predict(X_test.to_numpy())
```

3.3 Cálculo da Acurácia e da Matriz de Confusão

[14]:
```
df_confusion = confusion_matrix(y_test.to_numpy(), y_pred)
print("Matriz de Confusão:")
print(df_confusion)
```

```
Matriz de Confusão:
[[102  23]
 [ 15  76]]
```

[15]:
```
print("Acurácia: " + str(accuracy_score(y_test,y_pred)))
```

```
Acurácia: 0.8240740740740741
```

4 Conclusão

Neste trabalho pôde-se desenvolver um classificador de Bayes e aplicá-lo a um problema real de previsão de doença cardíaca. Utilizando 90% da base de dados para treinamento obteve-se uma acurácia de 0.926, o que foi considerado um bom resultado. Contudo, apenas 10% dos dados para se realizar a validação (o que equivale a 27 amostras) pode ser avaliado como pouco, além de não permitir examinar o modelo mais a fundo.

Assim, o experimento foi repetido utilizando dessa vez 70% dos dados para treinamento e 30% para teste. Como pôde ser visto, obteve-se também um bom resultado, alcançando uma acurácia de 0.876 e uma matriz de confusão bem equilibrada.

Por fim, foi repetido o processo novamente, desta vez utlizando somente 20% dos dados para treinamento e os outros 80% para teste. Neste teste, encontrou-se uma acurácia de 0.824. Apesar dessa ser um valor não tão distante dos obtidos anteriormente, vale citar que a baixa quantidade de dados tornou o modelo muito mais sensível aos dados que foram utilizados para treinamento. Observou-se que repetindo o treinamento algumas vezes com diferentes amostras, hora obtia-se um bom modelo, e hora obtia-se um modelo com acurácias consideravelmente mais baixas.

5 Referências

[1] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.