Tarea 1 Tema 2: Naive Bayes

Juan Manuel García Moyano
IABD
Informática y comunicaciones

Índice

1. Importación de librerías necesarias

He importado las librerías necesarias para el fin de entrenar los diferentes modelos del Naive Bayes. Además de las funcionalidades para realizar el mapa de confusión, el mapa de calor y la librería del Naive Bayes.

```
1 # Importación de la librerias necesarias
2 from sklearn.naive_bayes import GaussianNB, MultinomialNB, ComplementNB,
  BernoulliNB, CategoricalNB
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay
5 import pandas as pd
6 import seaborn as sns
7 import matplotlib.pyplot as plt
8 from google.colab import drive # Librería para Drive
```

2. Preproceso

2.1 Importación de los datos del dataset

La importación de los datos lo he realizado desde el drive. Para ello subí el dataset, luego monté mi drive en colab y leí el csv.

2.2 Mostrar las primeras y últimas filas del dataframe importado

En colab, si ponemos en la última línea una variable muestra las primeras y últimas filas pero si se hace desde un entorno diferente el cual no sea jupyter, deberíamos utilizar los métodos head y tail.

Aquí realizo el paso de codificar los atributos cuyo formato no son numéricos con la ayuda de LabelEncoder, ya que, nos va a hacer falta para próximos apartados.

Nota: no los vuelvo a trasformar a sus valores iniciales porque lo que muestro (los valores de la predicción) no tiene nada que con los valores transformados. Sino, si habría que volverlos a los valores que pertenezca.

2.3 Mostrar parámetros estadísticos de los datos (media, desviación típica, cuartiles, etc.)

Un dataframe tiene un método llamado describe que nos muestra los datos estadísticos de todos los atributos. Se debe de tener en cuenta que los datos estén en formato numérico.

2.4 Mostrar un mapa de calor que indique la correlación entre variables

Para realizar el mapa de calor he utilizado la librería seaborn. Para ello tuve de indicar la dimensión del mapa porque en un principio no se veía muy bien, ya que, los nombres de los atributos se juntaban. Por ello, primero lo redimensiono y después lo muestro con el método headmap(dataframe.corr(), square = True, annot =True). Vamos a desglosar los parámetros:

- dataframe.corr(): calcula la matriz de correlación entre las columnas numéricas del dataframe. La correlación mide la relación lineal entre dos variables, con valores que van desde:
  - 1: correlación positiva perfecta (ambas variables aumentas juntas).
  - -1: correlación negativa perfecta (una variable aumenta mientras la otra disminuye).
  - 0: no hay relación. Estas nos indican que deben ser eliminadas para el entrenamiento.
- square = True: fuerza que cada celda mapa de calor tenga forma cuadrada.
- Annot = True: muestra los valores numéricos de la correlación dentro de cada celda.

2.5 Seleccionar las características a tener en cuenta en el estudio

El estudio de se va realizar sin los atributos calidad e Id, porque si nos fijamos en el mapa de calor, la correlación entre ellos y el atributo calidad no es muy alta, por eso, he decidido en utilizar el máximo posible.

2.6 Separar datos entre datos de entrada y etiquetas (resultados)

El estudio de se va realizar sin los atributos calidad e Id. Para ello divido los datos en una variable (x) que va a tener los datos sin los atributos anteriores y otra variable (y) con los datos del atributo calidad solamente.

2.7 Separar datos entre entrenamiento y prueba (usando un 75% para entrenamiento y 25% para test)

Para separar los datos utilizo la funcionalidad train_test_split. A este le paso dos variables una con los datos de la variable x y la otra variable es y. X tiene los datos sin los atributos calidad e Id y la otra variable que es la que tiene todos los datos del atributo calidad. Además, esta función trae los parámetros por defecto de random_state (1) y test_size (0.25). Esto lo que hace que la división sea reproducible y representa la proporción del dataframe total asignada al conjunto de prueba, respectivamente.

Mediante el desempaquetado múltiple, lo que devuelve la función se lo asigno a 4 variables. Estas son:

- x_entrenamiento: tiene el 75% de los datos sin los atributos calidad e Id que van a servir para entrenar el modelo.
- x_test: tiene el otro 25% de los datos sin los atributos calidad e Id que van a servir hacer la predicción.
- y_entrenamiento: tiene el 75% de los datos de la calidad que van a servir para entrenar el modelo.
- y_test: tiene el otro 25% de los datos de la calidad que van a servir para calcular la probabilidad de acierto del modelo.

3. Entrenamiento y predicción

Para todos los modelos he intentado realizar lo mismo para que se lo más justo posible.

3.1 Elegir, instanciar y entrenar el modelo

Primero instancio el modelo deseado. Después, lo entreno con las variables anteriores de entrenamiento.

3.2 Realizar una predicción con los datos de prueba

Al modelo elegido hago que haga una predicción con los datos de x_test.

4. Evaluación

4.1 Mostrar el porcentaje de elementos correctamente clasificados

Con la ayuda de la función accuracy_score, que me sirve para comprobar la probabilidad de acertar que tiene el modelo. A esta función le paso los datos y_test (datos del atributo calidad) y la predicción realizada en el anterior apartado.

4.2 Mostrar la predicción realizada

Muestro los datos que he realizado con la predicción.

4.3 Representar gráficamente la clasificación obtenida

Aquí utilizo el método de ConfusionMatrixDisplay que es from_predictions. Se va a encargar de mostrarnos el mapa de confusión comparando los datos de test (y_test) y los datos de la predicción (y_prediccion).

5. Optimización

5.1 Finalmente prueba los distintos clasificadores y realiza una pequeña comparativa indicando cuál obtiene mejores resultados.

Como se puede observar, el que mejor resultado da es CategoricalNB con una probabilidad de 0.878260. El siguiente sería el modelo GaussianNB con una probabilidad total de 0.860869. Además, exceptuando el modelo GuassianNB se debe tener especial cuidado de que los datos que le pasemos como entrenamiento no sean negativos, para ello hay que escalar los datos. En este caso, lo he realizado con MinMaxScaler. Otro detalle a tener en cuenta es codificar los datos de tipo cadena a numéricos mediante el LabelEncoder. Un dato curioso es como tratar los datos con el modelo CategoricalNB, este utilizando los datos que he utilizado con los demás modelos no ha funcionado e incluso daba error sobre el axis = 1, aunque el error no tenga nada que ver la realidad.