CUERPO ACADÉMICO DE PROBABILIDAD Y ESTADÍSTICA BUAP

Decimocuarta Semana Internacional de la Estadística y la Probabilidad
14-18 de junio de 2021

CONSTRUCCIÓN DE UN MODELO DE MAQUINA DE SOPORTE VECTORIAL PARA LA PREDICCIÓN DEL CONSUMO DE DROGAS.

LAURA VIANEY DÍAZ CASTRO [a], FRANCISCO SERGIO SALEM SILVA [b]

[a]Facultad de Matemáticas
Universidad Veracruzana, Xalapa, Veracruz, México.
e-mail: zS16012126@estudiantes.uv.mx

[b]Facultad de Matemáticas
Universidad Veracruzana, Xalapa, Veracruz, México.
e-mail: frsalem@uv.mx

En el presente trabajo se hace uso del aprendizaje automático para crear un modelo predictivo, usando una base de datos que contiene registros de 1885 personas de distintas edades. Esta base de datos contiene características que nos permiten construir un modelo para hacer una evaluación del riesgo de ser consumidor de algunas drogas y así predecir qué tan susceptible es una persona, (dependiendo de su edad) a consumir éstas. El modelo que construimos es una máquina de soporte vectorial. Este modelo lo construimos usando la biblioteca scikit-learn de Python. La base de datos ('drug.csv') contiene 18 problemas de clasificación. Cada una de las variables de etiquetas independientes contiene siete clases: "Nunca utilizado", Ütilizado durante una década atrás", Ütilizado en la última década", Ütilizado en el último año", Ütilizado en el último mes", Ütilizado en la última semana", y Üsado en el último día". En primera instancia se modificó la base (limpieza de datos) con la biblioteca de Python llamada Pandas y se visualizaron los datos con apoyo de Seaborn y Matplotlib.

Keywords: Aprendizaje de máquina, Máquina de soporte vectorial, Base de datos.

1. Introducción

La base de datos contiene registros de 1885 encuestados. Para cada encuestado se conocen 12 atributos: mediciones de personalidad que incluyen NEO-FFI-R (neuroticismo, extraversión, apertura a la experiencia, amabilidad y conciencia), BIS-11 (impulsividad) e ImpSS (búsqueda de sensaciones), nivel de educación, edad, género, país de residencia y etnia. Todos los atributos de entrada son originalmente categóricos y están cuantificados. Después de la cuantificación, los valores de todas las características de entrada pueden considerarse como valores reales. Además, se preguntó a los participantes sobre el uso de 18 drogas legales e ilegales (alcohol, anfetaminas, nitrito de amilo, benzodiacepinas, cannabis, chocolate, cocaína, cafeína, crack, éxtasis, heroína, ketamina, máximos legales, LSD, metadona, hongos, abuso de nicotina y sustancias volátiles y un fármaco ficticio (Semeron) que se introdujo para identificar a los reclamantes en exceso. Para cada fármaco, deben seleccionar una de las respuestas: nunca usaron el fármaco, lo usaron hace más de una década o en la última década, año, mes, semana o día. La base de datos contiene 18 problemas de clasificación. Cada una de las variables de etiquetas independientes contiene siete clases: 'Nunca utilizado', 'Utilizado durante una década atrás', 'Utilizado en la última década', 'Utilizado en el último año', 'Utilizado en el último mes', 'Utilizado en la última semana', y 'Usado en el último día'.

2. Problemas planteados que se pueden resolver

- Siete clasificaciones de clase para cada droga ilegal por separado.
- El problema se puede transformar en clasificación binaria mediante la unión de parte de las clases en una nueva clase. Por ejemplo, 'Nunca utilizado', 'Usado en una década atrás' clase de formulario 'No usuario' y todas las demás clases forman la clase

’Usuario’.

- Evaluación del riesgo de ser consumidor de drogas para cada droga.

Descargamos la base de datos desde[**2**]

3. Tratamiento de datos

Ahora se guarda el archivo con todos los datos de tipo flotante

```
[21] # guardamos el archivo
     drug.to_csv('drug.csv', index=False)
```

4. MODELO DE CLASIFICACIÓN

El cerebro de los jóvenes sigue creciendo y desarrollándose hasta alrededor de los 25 años, incluyendo la corteza prefrontal, que cumple un rol importante para tomar decisiones. Consumir drogas durante la juventud puede interferir con los procesos de desarrollo del cerebro. También puede afectar la toma de decisiones. Es más probable hacer cosas riesgosas, como tener sexo sin protección y conducir peligrosamente.

Mientras más temprano los jóvenes empiecen a consumir drogas, mayores serán las posibilidades de continuar consumiendo en el futuro. El abuso de drogas en la juventud puede contribuir al desarrollo de problemas como enfermedades cardíacas, presión arterial alta y trastornos del sueño. [**4**]

Es por ello por lo que tomaremos como objetivo principal el consumo de drogas dependiendo de la edad a la que están, o fueron consumidas, para así poder decir si volverán a hacerlo. Usaremos un modelo SVM (Support Vector Machine) este encuentra el mejor hiperplano para dividir las clases. Esto se hace maximizando la distancia entre el hiperplano y las muestras más cercanas de cada clase, que se llaman vectores de soporte.

Este método lineal también se puede usar para modelar clases no lineales usando el truco del kernel. Esto El método mapea las características en un espacio de mayor dimensión en el que el hiperplano está determinado. Este hiperplano del que hemos estado hablando también se conoce como la decisión superficie, y lo visualizaremos al entrenar nuestros modelos.

```
[1]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

     %config InlineBackend.figure_format='retina'
     sns.set() # Se toman los valores por defecto de matplotlib
     plt.rcParams['figure.figsize'] = (9, 6)
     plt.rcParams['axes.labelpad'] = 10
     sns.set_style("darkgrid")
     #Ajustamos el parametro para el tamaño de las figuras
     plt.rcParams['figure.figsize'] = (8, 8)
```

Ahora cargamos la base que se construyó en la sección anterior (base limpia).

```
[2] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Alrededor del año 3.000 a.C ya se utilizaban algunos opiáceos: en Asia el cáñamo, en América hojas de coca como analgésico o en la sociedad Azteca algunos hongos como el peyote. Aunque no se conozca con exactitud cuál fue la primera droga que se usó, el alcohol está entre las primeras ya que es probable que cuando se almacenaba la miel y ésta fermentaba se produjese el primer vino. Desde entonces, las bebidas fermentadas han sido consumidas a lo largo de la historia convirtiéndose en un importante comercio colonial. Pero la droga con más adicción a lo largo y ancho del planeta es la cafeína. Aunque la planta existía hace miles de años en Etiopía, su difusión empezó en Arabia en el siglo X. Cuenta la leyenda que un monje al observar cómo quedaban de excitados los animales después de comer sus frutos decidió probarlos. En el siglo XVIII el café se propagó por Europa. [**1**]

Es por esto por lo que tendremos como variables objetivos el Alcohol y el café, así analizamos en la descripción de nuestro dataframe la media de cada droga y vemos quien está siendo más utilizada.

Aquí, el objetivo es predecir la clase de una muestra utilizando las métricas disponibles. En el caso más simple, solo hay dos clases posibles, lo que significa que estamos haciendo clasificación binaria. Este es el caso del problema que nos ocupa donde intentamos predecir si una persona consume drogas o no. Si se tuviera más de dos etiquetas de clase estriamos haciendo clasificación de clases múltiples.

Llamamos a nuestra Base y construimos un dataFrame con ella.

```
[3] #Llanamos la base que procesamos
    d2_df = pd.read_csv('/content/drive/MyDrive/drug.csv')
```

```
[4] d2_df.describe()
```

	EDAD	GENERO	ALCOHOL	AMPHET	AMYL	BENZOS	CAFF	CANNABIS	CHOC	COKE	CRACK
count	1885.00000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000	1885.000000
mean	0.03461	-0.000256	4.635013	1.340584	0.606897	1.465252	5.483820	2.989390	5.106631	1.161273	0.297613
std	0.87836	0.482588	1.331253	1.783587	1.064210	1.867275	1.114648	2.287438	1.089315	1.513031	0.837053
min	-0.95197	-0.482460	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	-0.95197	-0.482460	4.000000	0.000000	0.000000	0.000000	5.000000	1.000000	5.000000	0.000000	0.000000
50%	-0.07854	-0.482460	5.000000	0.000000	0.000000	0.000000	6.000000	3.000000	5.000000	0.000000	0.000000
75%	0.49788	0.482460	6.000000	2.000000	1.000000	3.000000	6.000000	5.000000	6.000000	2.000000	0.000000
max	2.59171	0.482460	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000

Para fines prácticos cambiaremos los valores de la columna EDAD. Quedando de la siguiente forma equivalente: 18-24=1, 25-34=2, 35-44=3, 45-54=4, 55-64=5, 65+=6. Además, necesitamos pasar estos últimos valores a punto flotante para ello utilizamos el siguiente comando.

```
[5] #Hacemos el cambio
    d3_df = d2_df.replace({-0.95197: int(1), -0.07854: int(2), 0.49788: int(3), 1.09449: int(4), 1.82213: int(5), 2.59171: int(6)})
```

El conjunto de datos de entrenamiento son los datos de los que partimos para entrenar a nuestro algoritmo, es decir, las características y las etiquetas. Un conjunto

de prueba es un conjunto de datos que se usa para evaluar un modelo desarrollado a partir de un conjunto de entrenamiento.

```
[7] #Dividimos los datos en conjunto de entrenamiento y prueba
    from sklearn.model_selection import train_test_split

    features = ['ALCOHOL', 'CAFF']
    X_train, X_test, y_train, y_test = train_test_split(
        d3_df[features].values, d3_df['EDAD'].values,
        test_size=None, random_state=0)
```

```
[8] #Para que el modelo funcione mejor vamos a escalar los datos de entrada
    #para que las caracteristicas esten en el mismo orden
    from sklearn.preprocessing import StandardScaler

    scaler = StandardScaler()
    X_train_std = scaler.fit_transform(X_train)
    X_test_std = scaler.transform(X_test)
```

Mantenemos parte de los datos como un conjunto de pruebas. El comando ”train-test-split”se encargará de dividir matrices en subconjuntos de train y test aleatorios. Primero colocamos los valores de las características elegidas, después los valores de las edades. Como elegimos None en test-size por defecto se establecerá en 0.25* y para random-state controla la mezcla aplicada a los datos antes de aplicar la división. El comando StandardScaler se encarga de estandarizar características eliminando la media y escalando a la varianza de la unidad de medida. Mediante la transformación se almacenan la media y la desviación estándar para utilizarlas en datos posteriores. Muchos elementos utilizados en la función objetivo de un algoritmo de aprendizaje. Ya que el kernel RBF de Support Vector Machines asumen que todas las características se centran alrededor de 0 y tienen varianza en el mismo orden. Cabe aclarar que, si una característica tiene una varianza que es órdenes de magnitud mayor que otras, puede dominar la función objetivo y hacer que el estimador no pueda aprender de otras características correctamente como se esperaba.

5. La clase support vector machine

Importamos el support vector machine class de la biblioteca scikit-learn y ajustamos el modelo en los datos de entrenamiento. El parámetro C controla la penalización por clasificación errónea*, lo que permite que la varianza y el sesgo del modelo sea revisado. [**3**]

```
[9] from sklearn.svm import SVC

    svm = SVC(kernel='linear', C=1, random_state=0)
    svm.fit(X_train_std, y_train)
```

```
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001,
    verbose=False)
```

C=1 nos da por default. break-ties=False por default se devuelve la primera clase entre las clases empatadas. cache-size=200 especifica el tamaño del cache del kernel en MB. class-weight=None por default se supone que todas las clases tienen un peso uno. coef0=0.0 es el termino independiente en función del kernel. decision-function-shape=’ovr’ por default devuelve una función de decisión one-vs-rest de forma como los demás clasificadores. degree=3 es el grado de la función del núcleo polinomial. Ignorado por todos los demás núcleos. gamma = ’scale’ usa 1 / (n-features * X.var ()) como valor de gamma. max-iter=-1 por default no hay limite de iteraciones. probability=False por default para habilitar estimaciones de probabilidad. Esto debe estar habilitado antes de llamar al ajuste, ralentizará ese método, ya que utiliza internamente una validación cruzada de 5 veces. shrinking=True por default para usar la heurística de reducción. tol=0.001 Tolerancia al criterio de parada. verbose=False por default habilita la salida detallada.

Antes de guardar el modelo completo, haremos parte por parte para verificar que todo esté en orden.

```
[10] #Vemos la exactitud en el modelo
     from sklearn.metrics import accuracy_score
     y_pred = svm.predict(X_test_std)
     acc = accuracy_score(y_test, y_pred)
     print('accuracy = {:.1f}%'.format(acc*100))
```

```
accuracy = 35.0%
```

Una matriz de confusión es una tabla resumida que se utiliza para evaluar el rendimiento de un modelo de clasificación. Por definición es una matriz C donde la ijesima entrada Cij es igual al número de observaciones que se sabe que están en el grupo i y que se predice que están en el grupo j.

```
[11] #Calculamos la matriz de confusion
     from sklearn.metrics import confusion_matrix

     print('Registro de porcentaje de exactitud por clase:')
     cmat = confusion_matrix(y_test, y_pred)
     scores = cmat.diagonal() / cmat.sum(axis=1) * 100
     print('EDAD = 1 : {:.2f}%'.format(scores[0]))
     print('EDAD = 6 : {:.2f}%'.format(scores[1]))
```

```
Registro de porcentaje de exactitud por clase:
EDAD = 1 : 100.00%
EDAD = 6 : 0.00%
```

Parece que el modelo simplemente clasifica cada muestra como 0, lo que claramente no es útil en absoluto. Usemos un diagrama de contorno para mostrar la clase predicha en cada punto del espacio de características. Esto se conoce comúnmente como la trama de las regiones de decisión.

Lo que apreciamos en el gráfico es como la función trazó las regiones de decisión con las 300 muestras pasadas como argumentos, a pesar de ser muchos, no obstruyen demasiado y el gráfico es funcional. Pero notamos que ha clasificado a toda la región con el color correspondiente a la clase 2. Por ello entrenaremos el kernel (kernel='rbf') y veremos si obtenemos una región mejor clasificada.

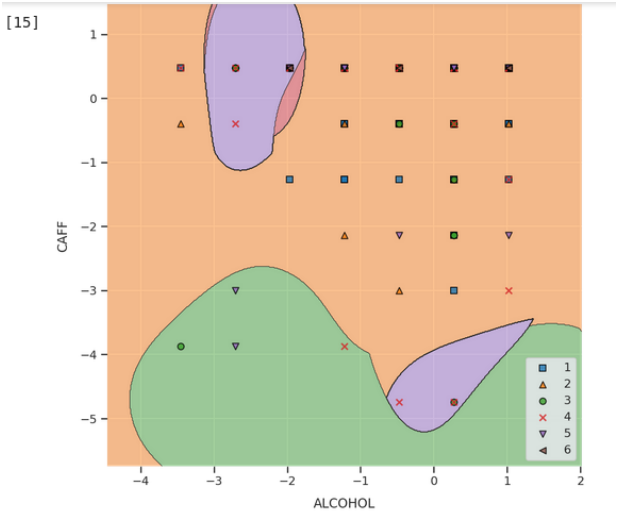
```
[13] svm = SVC(kernel='rbf', C=1, random_state=0)
     svm.fit(X_train_std,y_train)

     SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
         decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
         max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001,
         verbose=False)
```

```
[14] from sklearn.metrics import accuracy_score
     from sklearn.metrics import confusion_matrix
     from IPython.display import display
     from mlxtend.plotting import plot_decision_regions

     def drug_model_fit(clf, X_test, y_test):

         y_pred = clf.predict(X_test)
         acc = accuracy_score(y_test, y_pred, normalize=True) * 100
         print('total accuracy = {:.1f}%'.format(acc))

         print('percent accuracy score per class:')
         cmat = confusion_matrix(y_test, y_pred)
         scores = cmat.diagonal() / cmat.sum(axis=1) * 100
         print('EDAD = 1 : {:.2f}%'.format(scores[0]))
         print('EDAD = 2 : {:.2f}%'.format(scores[1]))
         print()

         fig = plt.figure(figsize=(8, 8))
         N_samples = 300
         X, y = X_test[:N_samples], y_test[:N_samples]
         plot_decision_regions(X, y.astype(np.integer), clf=clf)

         plt.xlabel('ALCOHOL')
         plt.ylabel('CAFF')
         plt.legend(loc='lower right')
```

6. Interpretación

Visualizamos el modelo entrenamos

```
[15] drug_model_fit(svm, X_test_std, y_test)

     total accuracy = 36.2%
     percent accuracy score per class:
     EDAD = 1 : 98.79%
     EDAD = 2 : 0.00%
```



Como entrenamos el kernel, podemos apreciar una clasificación mucho más clara y legible para nuestro estudio. podemos observar que la región que ocupa mayor espacio es la clase naranja que corresponde a la edad de 25-34 años, es decir la persona que se encuentre en este intervalo es más propenso a consumir estas drogas.

7. Conclusiones

Cabe mencionar que nuestro modelo se hizo para el consumo de café y alcohol, pero se puede construir un modelo que considere el consumo de cualquier otra droga incluida en la base. Se puede observar que este modelo no es muy bueno ya la métrica accuracy nos da un valor de 35 % sin embargo se puede mejor el modelo usando técnicas de validación cruzada y curvas de validación, estas nos permiten seleccionar parámetros óptimos para nuestro modelo. [**3**]

Referencias

[1] Agustí, F. *SER Historia: "Las drogas y su origen"*, Artículo recuperado de Cadena Ser, sitio web: https://cadenaser.com/programa/2013/06/14/ser-historia/1371165434-850215.html, 2013.

[2] ARXIV. *The Five Factor Model of personality and evaluation of drug consumption risk*, Base de datos recuperada de, Cornell University, sitio web: `https://arxiv.org/abs/1506.06297:~:text=The%20Five%20Factor%20Model%20of%20personality%20and%20evaluation%20of%20drug%20consumption%20risk,-E.&text=An%20online%20survey%20methodology%20was,ImpSS)%2C%20and%20demographic%20information`, 2015.

[3] Galea, A. *Beginning Data Science with Python and Jupyter*, Birmingham, England. Pack Publishing, 2018.

[4] NIH. *Drogas y Menores de Edad*, Artículo recuperado de, MedLinePlus, sitio web: https://medlineplus.gov/spanish/drugsandyoungpeople.html, 2020.

[5] Rossant, C. *IPython Interactive Computing and Visualization Cookbook*, Birmingham, England: Pack Publishing, 2014.