unir MÉXICO

Nombre: **González Trillo Rodolfo Arturo**

Asignatura: **Aprendizaje automático**

Actividad: **2**

Fecha: **20 de septiembre de 2021**

Clasificación con máquina de vectores de soporte y redes de neuronas

Resumen

Se emplearon los modelos de clasificación de máquina de vectores de soporte y redes neuronales, para la clasificación de los datos de Sharma (2017) y predecir la categoría `price_range`. El modelo mejor ajustado fue el de redes neuronales que alcanzó una calificación f1 de 0.91 y el otro fue de 0.87.

Introducción

Los modelos de máquina de soporte de vectores son modelos de aprendizaje supervisado. Dado un conjunto de entrenamiento, con las categorías definidas,los algoritmo genera fronteras con distinta forma que maximizan la distancia entre las categorías en el híperespacio conformado por las variables de los datos.

Los modelos de clasificación de redes neuronales se basan el uso de neuronas, funciones matemáticas basadas en sus equivalentes biológicos. Estos modelos también son de aprendizaje supervisado y por lo tal requiere entrenamiento.

En este trabajo vamos a probar estos dos modelos, con una base de datos de ventas de teléfonos celulares. Dadas las características de los modelos se intentará predecir a que categoría de precio pertenecen.

Bibliotecas

Durante la elaboración de este proyecto se utilizó **Python 3** con las siguientes bibliotecas:

- **`Pandas`**: Manejo y limpieza de la base de datos.
- **`Numpy`**: Uso de funciones para arreglos.
- **`Sklearn`**: Ajuste y validación de modelos de máquinas de soporte de vectores y redes neuronales de clasificación.
- **`Pyplot`**: Visualización de los datos.
- **`Seaborn`**: Extensiones a la visualización de datos.

Importar datos

Los datos utilizados provienen de 2017. Es una tabla de datos acerca de características de los móviles y su precio. La tabla de de datos tiene 2000 entradas y 21 variables.

```
[2]: df = pd.read_csv("data/train.csv")
     df.shape
```

```
[2]: (2000, 21)
```

Estadística descriptiva

Las variables son todas de tipo numérico, lo que a *priori* hace creer que son todas variables de este tipo. Sin embargo, esto es falso, pues un análisis más profundo muestra que algunas de las variables son de hecho categóricas, pero han pasado ya por

un proceso de discretización, se puede apreciar esto en tabla 1. Las variables que tienen mínimo 0 y máximo 1 son variables categóricas que indican la presencia o no de un característica.

```
[3]: df.dtypes
```

```
[3]: battery_power      int64
     blue               int64
     clock_speed      float64
     dual_sim           int64
     fc                 int64
     four_g             int64
     int_memory         int64
     m_dep            float64
     mobile_wt          int64
     n_cores            int64
     pc                 int64
     px_height          int64
     px_width           int64
     ram                int64
     sc_h               int64
     sc_w               int64
     talk_time          int64
     three_g            int64
     touch_screen       int64
     wifi               int64
     price_range        int64
```

```
[5]: stats = df.describe().round(0).astype(int)
     stats
```

Output en tabla 1.
No es idéntica a como está presentada en `act2.ipynb`, aquí ha sido reorganizada para mejorar su visibilidad.

De los datos estadísticos de las variables (tabla 1) se destacan las siguientes observaciones:

- `sc_h` y `sc_w`: Son el alto y el ancho de la pantalla en pulgadas. Los valores van desde 0 —probablemente signifique sin pantalla— hasta 19'' ¡Lo que significa que no todos los equipos son teléfonos celulares!, suponemos que también hay tabletas.
- `dual_sim`, `four_g`,`touch_screen` y `wifi`: Son todas variables categóricas (con valores 0 y 1). Los 2 últimos cuartiles están marcados con 1, lo que signifca que al menos el 50 % de los teléfonos posee estas características.
- `three_g`: La variable del dispositivo con conexión de este tipo. 75 % de los equipos tienen este tipo de conectividad. El hecho de que existan equipos sólo con 3g indica que hay equipos viejos en la tabla, pues los primeros moviles con 4G comenzaron a ser vendidos en 2009 (Torstensson, 2009).
- `blue`: Es si el equipo está equipado con *bluetooth*. Al parecer en 2017 esto era algo raro, pues al menos 25 % cuentan con esta característica.
- `px_heigt` es el alto de la resolución de la pantalla. Hay pantallas que tienen 0 en este valor. No se sabe que representa ésto realmente, pero la hipótesis es que se trata de teléfonos con pantalla de cristal líquido numéricas.

Variables categóricas y numéricas

Mostramos las variables categóricas:

```
[7]: mask = (stats.loc['min']==0) & (stats.loc['max']==1) &
     ↪  (df.dtypes=='int64')

     categorical = stats.columns[mask]
     categorical
```

```
[7]: Index(['blue', 'dual_sim', 'four_g', 'three_g',
     ↪  'touch_screen', 'wifi'], dtype='object')
```

y las variables numéricas:

```
[8]: numerical = stats.columns[mask==False]
     numerical
```

```
[8]: Index(['battery_power', 'clock_speed', 'fc', 'int_memory',
     ↪  'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
     ↪  'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time',
     ↪  'price_range'],
     dtype='object')
```

Tratamiento de datos faltantes

Verificamos si el arreglo tienen valores faltantes:

```
[9]: df.isna().sum()
```

Tabla 1: Estadísticos de las variables numéricas.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000	2000	2000	2000	2000	2000	2000
mean	1239	0	2	1	4	1	32
std	439	1	1	1	4	0	18
min	501	0	0	0	0	0	2
25 %	852	0	1	0	1	0	16
50 %	1226	0	2	1	3	1	32
75 %	1615	1	2	1	7	1	48
max	1998	1	3	1	19	1	64

	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram
count	2000	2000	2000	2000	2000	2000	2000
mean	1	140	5	10	645	1252	2124
std	0	35	2	6	444	432	1085
min	0	80	1	0	0	500	256
25 %	0	109	3	5	283	875	1208
50 %	0	141	4	10	564	1247	2146
75 %	1	170	7	15	947	1633	3064
max	1	200	8	20	1960	1998	3998

	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
count	2000	2000	2000	2000	2000	2000	2000
mean	12	6	11	1	1	1	2
std	4	4	5	0	1	1	1
min	5	0	2	0	0	0	0
25 %	9	2	6	1	0	0	1
50 %	12	5	11	1	1	1	2
75 %	16	9	16	1	1	1	2
max	19	18	20	1	1	1	3

Output del código 5.

```
[9]: battery_power    0
     blue             0
     clock_speed      0
     dual_sim         0
     fc               0
     four_g           0
     int_memory       0
     m_dep            0
     mobile_wt        0
     n_cores          0
     pc               0
     px_height        0
     px_width         0
     ram              0
     sc_h             0
     sc_w             0
     talk_time        0
     three_g          0
     touch_screen     0
     wifi             0
     price_range      0
     dtype: int64
```

Como se puede apreciar, no hay ninguna variable con datos faltantes.

Escalado e histogramas

Ninguna de las distribuciones presenta forma de distribución normal (véase figura 1), por lo que se decidió no estandarizar las distribuciones sino escalar escalar los datos con el método *minmax*, haciendo que los valores de todas las variables estén en el intervalo $[-1, 1]$. De acuerdo a la documentación, scikit-learn (s.f.), este procedimiento también es adecuado para los métodos de clasificación con máquina de vectores de soporte y de redes neuronales.

```
[10]: normalized_df = 2*(df-df.min())/(df.max()-df.min()) - 1
```

Los histogramas se grafican con:

```
[13]: for column in x:
          ax = sns.displot(normalized_df, x = column, hue=y,
          ↪  multiple='stack')
```

Output en figura 1

De la figura 1 se pueden realizar las siguientes observaciones sobre los histogramas.

- `price_range`: Muestra que hay 4 categorías de precios, y que los datos están repartidos uniformemente entre ellas.
- `blue` y `n_cores`: Son representativas de la mayoría de las variables, distribuciones uniformes y los datos repartidos de forma equitativa en las categorías de los rangos de precio.

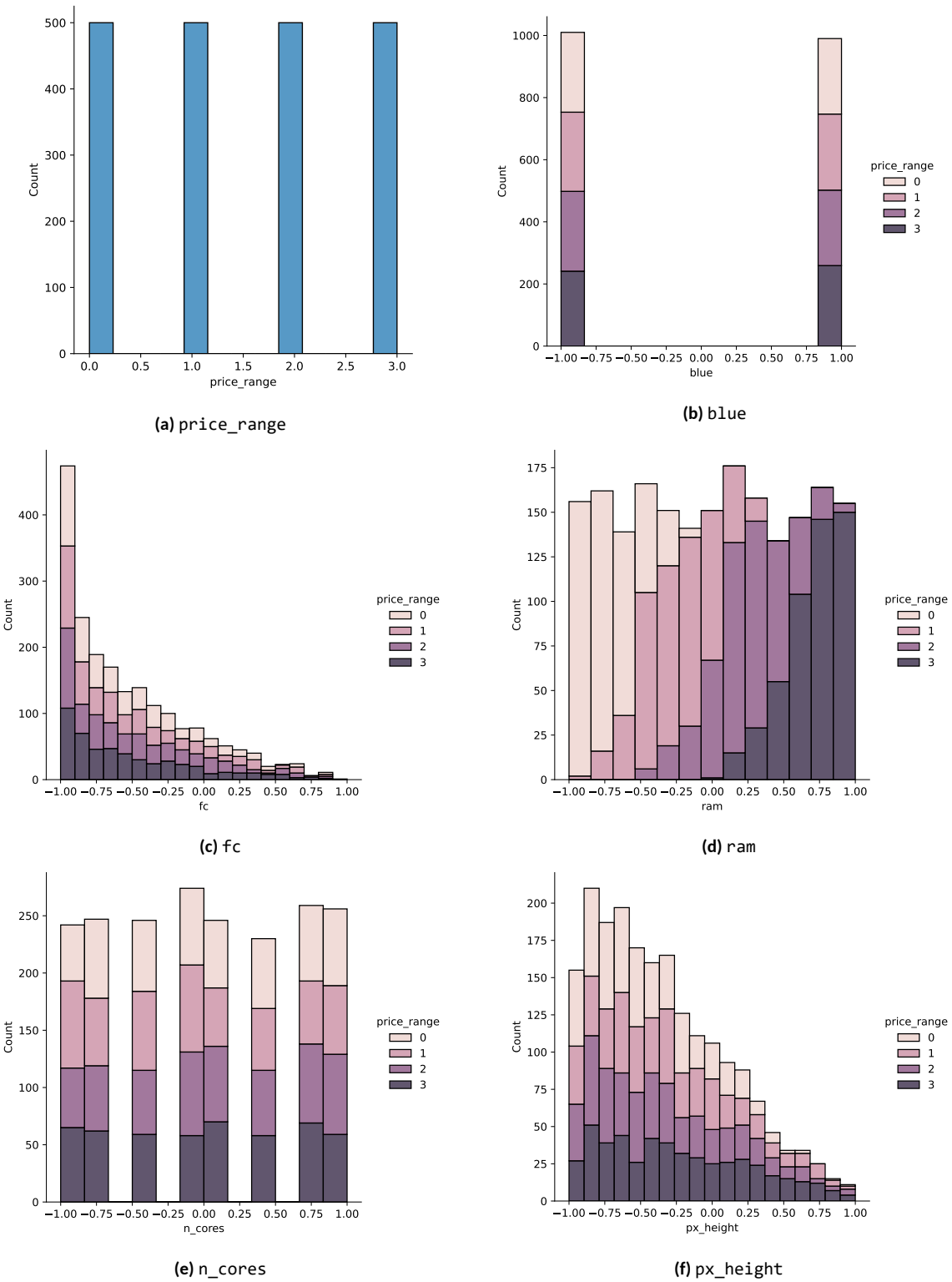
- `fc` y `px_height`: Se observa el otro grupo de variables con distribución con asimetría positiva.
- `ram`: Es la única variable que parece tener buena separación de las categorías de precio a lo largo de su intervalo. Por esta razón, `ram` es la variable con más potencial para poder clasificar los datos.

Referencias

scikit-learn. (s.f.). 6.3. Preprocessing data. https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing

Sharma, A. (2017). *Mobile Price Classification* [kaggle]. https://www.kaggle.com/iabhishekofficial/mobile-price-classification

Torstensson, Å. (2009). TeliaSonera first in the world with 4G services. https://web.archive.org/web/20100706212217/http://www.teliasonera.com/News-and-Archive/Press-releases/2009/TeliaSonera-first-in-the-world-with-4G-services/



(a) `price_range` **(b)** `blue`

(c) `fc` **(d)** `ram`

(e) `n_cores` **(f)** `px_height`

Figura 1: Histogramas de algunas variables relevantes.