Práctica 2:
Preparación de datos

Curso 2020/2021

PEDRO MANUEL FLORES CRESPO

Tratamiento Inteligente de Datos

Máster en Ingeniería Informática

Índice

1. Introducción

El objetivo de esta práctica es entender la importancia de la preparación de los datos en el proceso de extracción de patrones. El conjunto de datos a utilizar consiste en una serie de accidentes de tráfico que ocurrieron en Estados Unidos de los que disponemos información sobre:

- Mes, día y hora del accidente.
- Número de vehículos y no conductores involucrados.
- Población del área donde se dio el accidente.
- Tipo de colisión.
- Tipo de carretera, la condición de su superficie, si había algún tipo de control y límite de velocidad.
- Condiciones de luz y atmosféricas.
- Si estuvo involucrado algún autobús escolar o algún tipo de viandante o ciclista.
- Si había alcohol involucrado.
- Región del país donde sucedió.
- Si hubo fallecidos, lesiones y daños materiales.

Nuestro objetivo es poder calcular la clase del accidente en función de estos últimos atributos: fallecidos (`FATALITIES`), lesiones (`INJURY CRASH`) y daños materiales (`PRPTYDMG CRASH`). Para ello, construimos esta nueva variable, `clase` del siguiente modo:

$$\texttt{clase} = 3*\texttt{FATALITIES} + 2*\texttt{INJURY CRASH} + \texttt{PRPTYDMG CRASH}.$$

Así, los accidentes de clase 3 fueron los que tuvieron algún fallecido, en los de clase 2 solo hubo lesiones y en los de clase 1 solamente hubo daños materiales.

Algunas de las variables contienen datos perdidos. En ese caso, se proporciona una segunda variable denominada `*_I` (salvo en la velocidad que es `*_H`) que rellena dichos valores según algún criterio. Durante el desarrollo de la práctica, usaremos distintas técnicas de preprocesamiento para obtener la clase del accidente en función de la información proporcionada.

2. Desarrollo

En primer lugar, destacar que la práctica se ha desarrollado con el lenguaje de programación `Python`. Además, se ha utilizado la herramienta *Google Colaboratory* para llevar a cabo todas las pruebas. El cuaderno generado con todo el procedimiento se adjunta junto con el resto de documentación de la práctica. Una vez cargados los datos (en formato `csv`), se ha creado una nueva columna `clase` como se ha indicado anteriormente. Luego creamos los dos conjunto de datos por separado con los datos sin inputar e inputados (eliminando en ambos casos las columnas que se han usado para crear la variable clase). A partir de ellos, se harán las modificaciones necesarias con el prepocesamiento de los datos. Destacar que el árbol de decisión utilizado es el disponible en la biblioteca `sklearn` [1]. Para comprobar la bondad de los resultados obtenidos se ha dividido el conjunto de los datos disponibles en en uno de entrenamiento (80 %) y otro de test (20 %). En las siguientes secciones, nos centraremos sobre todo en mostrar y comentar los resultados obtenidos ya que, el proceso se puede consultar en el cuaderno antes mencionado.

2.1. Datos sin procesar

Si aplicamos el algoritmo de decisión a los datos sin procesos, es decir, tal cual nos los encontramos incluyendo valores perdidos obtenemos los resultados de la tabla 1.

clase/pred	1	2	3
1	3 231	2 424	49
2	2 344	2 969	68
3	45	57	6

Bien clasificados	6 206
Exactitud	55.45 %

Tabla 1: Resultados de aplicar el algoritmo a los datos sin procesar.

Vemos que el algoritmo ha sido capaz de clasificar correctamente algo más de la mitad de los datos.

2.2. Datos procesados _I

Por su parte, si aplicamos el mismo algritmo a los datos que nos dan ya inputados obtenemos el resultado de 2.

clase/Pred	1	2	3
1	3 264	2 384	56
2	2 371	2 931	79
3	38	64	6

Bien clasificados	6 201
Exactitud	55.4 %

Tabla 2: Resultados de aplicar el algoritmo a los datos preprocesados ya proporcionados.

Vemos que los resultados obtenidos son más o menos los mismos que en el caso anterior, incluso un poco peor.

2.3. Discretización

Para tratar de mejorar los resultados, primero vamos a llevar a cabo una discretización de algunas variables (a partir de conjunto de datos `_I`). Más concretamente:

- `HOUR_I`: se han creado tres categorías, centrándonos en si el accidente sucedió en hora punta o no.

```
autos_DISC['HOUR_I']= pd.cut(x=autos_DISC['HOUR_I'], bins=[-1,12,17,24],
                      labels=[1,2,3])
```

- `SPDLIM_H`: también se han dividido en tres categorías según sea la velocidad baja, media o alta.

```
autos_DISC['SPDLIM_H'] = pd.cut(x=autos_DISC['SPDLIM_H'], bins=[-1,30,50,80],
                         labels=[1,2,3])
```

- `WKDY_I`: las categorías sería día entre semana, sábado y domingo.

```
autos_DISC['WKDY_I'] = pd.cut(x=autos_DISC['WKDY_I'], bins=[0,1,6,7],
                       labels=[1,2,3])
```

Los resultados obtenidos los vemos en la tabla 3.

clase/Pred	1	2	3
1	3 296	2 359	49
2	2 426	2 879	76
3	38	58	12

Bien clasificados	6 187
Exactitud	55.28 %

Tabla 3: Resultados de aplicar el algoritmo tras una discretización.

En este caso no hemos conseguido mejorar ninguno de los resultados anteriores.

2.4. Valores perdidos

Si nos fijamos en los datos sin inputar, vemos que hay una gran cantidad de valores perdidos en los mismos. Por ejemplo en `PROFILE` hay 16 125 en `ALIGN` 7 433 o en `SPD_LIM` 8 798 entre otros. Primero vamos a ver qué ocurre cuando eliminamos las instancias que presentan algún valor perdido.

clase/Pred	1	2	3
1	3 291	2 378	35
2	2 337	2 956	88
3	37	64	7

Bien clasificados	6 254
Exactitud	55.87 %

Tabla 4: Resultados tras eliminar instancias con valores perdidos.

Como vemos en las tablas 4 no se han mejorado los resultados. Comprobemos el resultados tras eliminar en esta ocasión las columnas que presentan algún valor perdido.

clase/Pred	1	2	3
1	3 519	2 157	28
2	2 719	2 611	51
3	38	65	5

Bien clasificados	6 135
Exactitud	54.81 %

Tabla 5: Resultados tras eliminar las columnas con valores perdidos.

Según las tablas de 5, seguimos sin obtener unos resultados mejores.

2.5. Rellenar valores perdidos

Como no hemos obtenido buenos resultados eliminando las instancias con valores perdidos, usemos una estrategia diferente. En esta ocasión, lo que haremos será rellenar dichos valores según la media (hora y límite de velocidad) o la moda (el resto de características). Tras ello, los resultados son los que se muestran en la tabla 6. Vemos que obtenemos los mismos resultados que en los casos anteriores.

2.6. PCA

Nuestro objetivo ahora es reducir el número de características a las que aplicar el árbol de decisión. El conjunto de datos original presenta unas 24 características. Para reducirlas, vamos a partir del conjunto de datos discretizados de la sección 2.3. A este conjunto, vamos a aplicarle una análisis de componentes principales y nos quedamos solamente con 7.

clase/Pred	1	2	3
1	3 227	2 441	36
2	2 382	2 934	65
3	48	55	5

Bien clasificados	6 166
Exactitud	55.09 %

Tabla 6: Resultados tras rellenar los valores perdidos con la media o con la moda.

clase/Pred	1	2	3
1	3 787	1 900	17
2	1 901	3 422	58
3	24	78	6

Bien clasificados	7 215
Exactitud	64.46 %

Tabla 7: Resultados aplicando PCA con 7 componentes.

En este caso, como vemos en la tabla 7, sí hemos obtenido mejores resultados que anteriormente y hemos conseguido una exactitud de casi el 65 %, algo menos de un 10 % superior que en el mejor de los casos anteriores.

2.7. Balanceo de los datos

Para acabar, vamos a ver qué ocurre si utilizamos un conjunto de entrenamiento de datos balanceado. Si contamos las instancias que hay de cada una de las clases vemos la siguiente distribución:

- Clase 1: 28 536 instancias.
- Clase 2: 26 897 instancias.
- Clase 3: 531 instancias.

Observamos que la clase 3 es muy poco usual frente a las otras dos. Por ello, lo que vamos a hacer es ver qué ocurre si enfrentamos la clase minoritaria frente a las otras dos. Primero partimos del conjunto de datos discretizado (sección 2.3) agrupando en una clase las que hasta ahora teníamos como 1 y 2. Nuestro objetivo es por ello enfrentar la clase 1 con la 3. Los resultados se obtienen en la tabla 8.

clase/Pred	1	3
1	10 949	143
3	96	12

Bien clasificados	10 954
Exactitud	97.86 %

Tabla 8: Resultados enfrentando la clase minoritaria frente al resto.

En este caso sí hemos obtenido bueno resultados, clasificando correctamente el 97.94 % de las muestras del conjunto de test que hemos creado. Nuestro segundo paso es aplicar el algoritmo de clasificación al conjunto con solo los valores 1 y 3 en la variable clase pero que esté balanceado. Es decir, vamos aplicar el árbol de decisión a un conjunto con 531 datos de la clase 3 y 530 de la clase 1.

En las tablas 9 obtenemos que es capaz de clasificar bien sobre el 60 % de los datos, lo que sigue mejorando las estimaciones iniciales. En este último caso destacamos que, al haber pocas instancias, puede depender mucho del orden de selección de las características a la hora de construir el árbol. Por ejemplo, cambiando la semilla se ha llegado a conseguir una exactitud del 70 % mientras que en el resto de situaciones usando el conjunto completo más o menos se mantienen los resultados en el mismo rango.

clase/Pred	1	3
1	69	37
3	47	60

Bien clasificados	129
Exactitud	60.56 %

Tabla 9: Resultados enfrentando la clase minoritaria frente al resto con conjuntos balanceados.

3. Conclusiones

En conclusión vemos que los árboles obtenidos no han sido capaces de obtener una precisión demasiado alta. La mejor ha sido cuando se ha aplicado PCA al conjunto de datos obteniendo una exactitud del 65 %. Sin embargo, si vemos las clases 1 y 2 como una única clase y la enfrentamos a la 3, hemos obtenido una exactitud del 97 %. Esta diferencia se puede deber a que el árbol no es capaz de distinguir correctamente las clases 1 y 2. De hecho, si observamos las matrices de confusión, las equivocaciones entre estas dos clases están en el mismo rango que los aciertos.

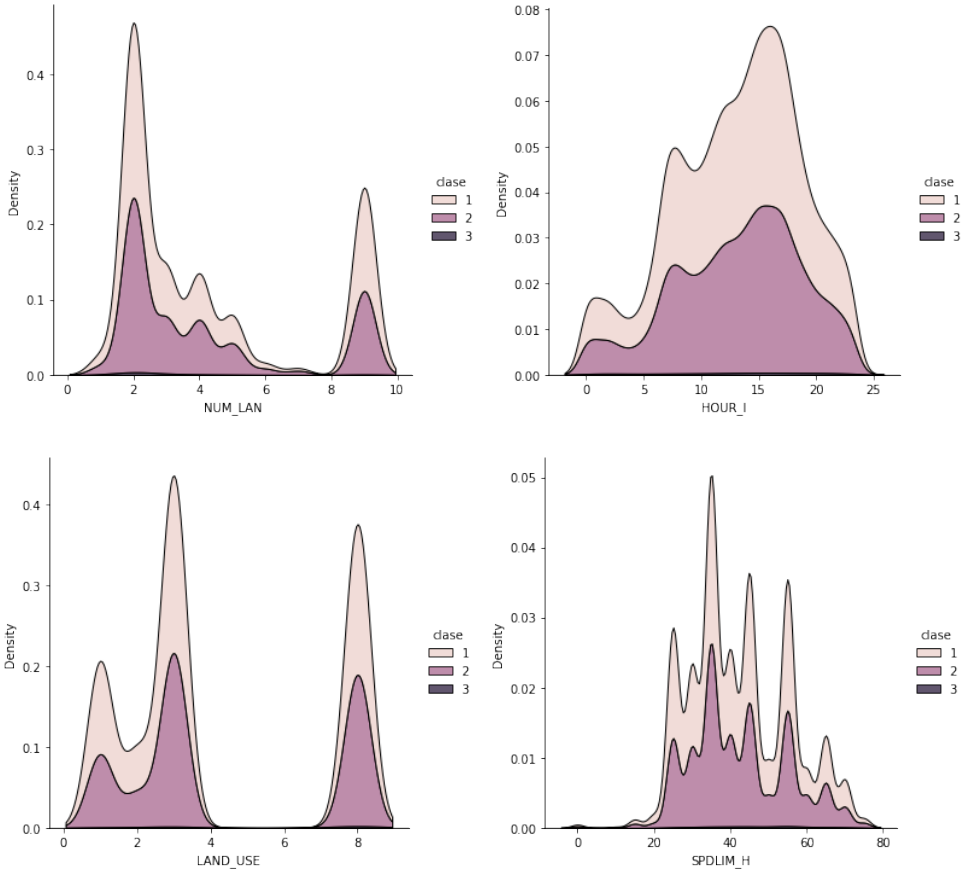

Figura 1: Densidad de las clases respecto de algunas características.

Si nos fijamos en las figuras 1, donde se ha dibujado la densidad de algunas variables según la clase, vemos que tanto la clase 1 como la 2 son bastante parecidas. Este hecho puede hacer que el árbol no sea capaz de diferenciarlas ya que ambas presentan las mismas “concentraciones” de instancias en los mismos valores

Referencias

1. Decision Tres scikit-learn: https://scikit-learn.org/stable/modules/tree.html#tree-classification
2. How to create a confusion matrix in Python using scikit-learn: https://www.educative.io/edpresso/how-to-create-a-confusion-matrix-in-python-using-scikit-learn
3. How To Discretize/Bin a Variable in Python with NumPy and Pandas? https://cmdlinetips.com/2019/12/how-to-discretize-bin-a-variable-in-python/
4. pandas.DataFrame.dropna https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html
5. sklearn.decomposition.PCA https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
6. Visualizing distributions of data https://seaborn.pydata.org/tutorial/distributions.html