Barrios_Lascar_Actividad_1

December 4, 2023

Técnicas de Inteligencia Artificial

Actividad 1. Laboratorio: Árboles de decisión, reglas y ensemble learning

Presentado por: **Ramón Barrios Láscar** Fecha: 01/12/2023

0.1 Importación de librerias necesarias

[1]:
```
#Para esta actividad se importarán las siguientes librerías:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler # ????
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
from mlxtend.plotting import plot_decision_regions
```

0.2 Cargar el Dataset

[2]:
```
#Código para cargar el Dataset
url = 'https://raw.githubusercontent.com/oscar-unir/TIA/main/datasets/car/
 ↪Laboratorio_dataset_car.csv'
df = pd.read_csv(url, sep=';', header='infer', encoding='utf-8')
```

RBL: Se carga el dataset directamente de la URL anotada en el notebook base, así se evita depender de el archivo local y su estructura.

0.3 Explique el problema a resolver.

Descripción del problema. Tipo de problema (justifique). Variable objetivo, variables de entrada. Utilidad de su posible solución. Elementos adicionales que considere relevantes:

RBL: Según la especificaciones en el documento visa04lab, página 3, se deberán usar dos algoritmos de clasificación, por lo que se trata entonces de un *problema de clasificación*. Antes de explorar detalladamente el dataset no se puede decir más, por lo que esto se tratará más adelante.

0.4 Caracterización del Dataset

Se incluye una descripción de los datos con:

- Número de clases de la variable objetivo, indicando que representan dichas clases y el tipo de valor que toman.
- Número de instancias en total.
- Número de instancias pertenecientes a cada clase.
- Número de atributos de entrada, su significado y tipo.
- ¿Hay algún valor de atributo desconocido?

Se incorporá una descripción (EDA) del conjunto de datos utilizado. Se analiza el dataset proporcionando, se muestra al menos algunas de sus características mediante tablas y al menos algunas de ellas en modo gráfico (p.ej., histogramas, diagramas de dispersión, diagramas de cajas y bigotes, etc.)

[3]:
```
print(f"El dataset está compuesto de %d instancias y %d atributos␣
 ↪(filas×columnas)"%(df.shape[0],df.shape[1],))
print(f"Con los siguientes atributos:")
for col in df.columns:
    print(f"• {col}")
```

```
El dataset está compuesto de 1750 instancias y 7 atributos (filas×columnas)
Con los siguientes atributos:
• Buying
• Maintenance
• Doors
• Person
• lug_boot
• safety
• class
```

RBL: Como reporta el código, el dataset tiene 1749 instancias, con atributos:

Buying

Maintenance

Doors

Person

lug_boot

safety

class

[4]:
```
for col in df.columns:
    print(f"Valores para la columna '{col}':")
    print(df[col].value_counts())
    print()
print("Tipos:")
```

```
df.dtypes
```

```
Valores para la columna 'Buying':
vhigh    443
med      438
low      437
high     432
Name: Buying, dtype: int64

Valores para la columna 'Maintenance':
low      447
vhigh    437
med      434
high     432
Name: Maintenance, dtype: int64

Valores para la columna 'Doors':
2        444
5more    437
3        435
4        434
Name: Doors, dtype: int64

Valores para la columna 'Person':
4       587
more    585
2       578
Name: Person, dtype: int64

Valores para la columna 'lug_boot':
big      585
med      583
small    582
Name: lug_boot, dtype: int64

Valores para la columna 'safety':
high    590
med     582
low     578
Name: safety, dtype: int64

Valores para la columna 'class':
unacc    1215
acc       390
good       75
vgood      70
Name: class, dtype: int64
```

Tipos:

[4]:
```
Buying         object
Maintenance    object
Doors          object
Person         object
lug_boot       object
safety         object
class          object
dtype: object
```

RBL: Se tienen las columnas ya mencionadas, que se detallan:

Buying: texto (categórica), con cuatro posibilidades, low, 437; med, 438; high, 432; y vhigh, 443 (parece ser el "precio de compra")

Maintenance: texto (categórica), con cuatro posibilidades, low, 447; med, 434; high, 432; y vhigh, 437 (parece ser el "costo de mantenimiento")

Doors: texto (categórica), con cuatro posibilidades, 2, 444; 3, 435; 4, 434; y 5more, 437 (esto es el número de puertas, con 5more siendo "5 o más puertas")

Person: texto (categórica), con tres posibilidades, 2, 578; 4, 587; y more, 585 (esto es el número de pasajeros, con more siendo "más de cuatro pasajeros")

lug_boot: texto (categórica), con tres posibilidades, small, 582; med, 583; y big, 585; (esto es el tamaño del baúl)

safety: texto (categórica), con tres posibilidades, low, 578; med, 582; y high, 590 (esto es qué tan seguro se considera al vehículo)

class: texto (categórica), con cuatro posibilidades, unacc, 1215; acc, 390; good, 75; y vgood, 70 (esto parece ser "qué tan atractivo para la compra es", con unacc o "no", acc o "aceptable", good o "bien", y vgood o "muy")

El texto así es uno de los valores del atributo y el número así es "cuántas instancias en esa clase". Nótese que como todos los atributos son categóricos, no hay mínimo, ni media ni máximo, y la moda es simplemente el valor más común para ese atributo.

Teniendo que class es un atributo de "no deseable" a "muy deseable", se creería que este atributo se puede tomar como objetivo, y como ya se mencionó, se puede clasificar en cuatro clases (la pregunta inicial pendiente)

RBL: Este es un dataset completamente limpio, sin nulos o "N/A"

Encontradas en Buying con valores: 432+437+438+443 = 1750 (todas)

Encontradas en Maintenance con valores: 447+434+432+437 = 1750 (todas)

Encontradas en Doors con valores: 444+435+434+437 = 1750 (todas)

Encontradas en Person con valores: 578+587+585 = 1750 (todas)

Encontradas en lug_boot con valores: 582+583+585 = 1750 (todas)

Encontradas en safety con valores: 578+582+590 = 1750 (todas)

Encontradas en class con valores: 1215+390+75+70 = 1750 (todas)

0.5 Preprocesamiento del dataset. Transformaciones previas necesarias para la modelación

[5]:
```
# Código que realice las transformaciones necesarias para poder realizar los␣
 ↪procesos de modelación. Ej.One hot enconding
# Primero renombro las columnas para que todas sigan como la misma nomenclatura
try:
    df.rename(columns={'lug_boot':'TrunkSize', 'safety':'Safety', 'class':
 ↪'Class', 'Person':'Passengers'} , inplace=True)
except KeyError:
    print(f"No se renombró")
df
```

[5]:

	Buying	Maintenance	Doors	Passengers	TrunkSize	Safety	Class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc
…	…	…	…	…	…	…	…
1745	low	low	5more	more	med	high	vgood
1746	low	low	5more	more	med	high	vgood
1747	low	low	5more	more	big	low	unacc
1748	low	low	5more	more	big	med	good
1749	low	low	5more	more	big	high	vgood

[1750 rows x 7 columns]

RBL: Hay 4 valores para Buying, 4 para Maintenance, 4 para Doors, 3 para Person, 3 para TrunkSize, 3 para Safety, y 4 valores para Class, lo que da 4+4+4+3+3+3+4=25, 25 features si los codificamos todos.

RBL: Se usa un truco para hacer encoding de los atributos: si se usa Pandas groupBy (para hacer lo que hace StringIndexer de PySpark) y se usan los índices de los grupos, esos grupos son únicos y sirven como integer encoding para los siguientes pasos.

[6]:
```
# Encoding usando groupBy de pandas, primero hacemos una copia
idxDF = df.copy()
idxDF
```

[6]:

	Buying	Maintenance	Doors	Passengers	TrunkSize	Safety	Class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc

```
4      vhigh       vhigh      2           2       med    med  unacc
…      …           …    …          …         …    …     …
1745     low         low  5more       more       med   high  vgood
1746     low         low  5more       more       med   high  vgood
1747     low         low  5more       more       big    low  unacc
1748     low         low  5more       more       big    med   good
1749     low         low  5more       more       big   high  vgood

[1750 rows x 7 columns]
```

[7]:
```
# Encode Buying usando groupBy de pandas
idxDF['Buying'] = df.groupby('Buying').ngroup()
idxDF
```

[7]:
```
      Buying Maintenance  Doors Passengers TrunkSize Safety  Class
0          3       vhigh      2          2     small    low  unacc
1          3       vhigh      2          2     small    med  unacc
2          3       vhigh      2          2     small   high  unacc
3          3       vhigh      2          2       med    low  unacc
4          3       vhigh      2          2       med    med  unacc
…      …           …    …          …         …    …     …
1745       1         low  5more       more       med   high  vgood
1746       1         low  5more       more       med   high  vgood
1747       1         low  5more       more       big    low  unacc
1748       1         low  5more       more       big    med   good
1749       1         low  5more       more       big   high  vgood

[1750 rows x 7 columns]
```

[8]:
```
# Encode Maintenance usando groupBy de pandas
idxDF['Maintenance'] = df.groupby('Maintenance').ngroup()
idxDF
```

[8]:
```
      Buying  Maintenance  Doors Passengers TrunkSize Safety  Class
0          3            3      2          2     small    low  unacc
1          3            3      2          2     small    med  unacc
2          3            3      2          2     small   high  unacc
3          3            3      2          2       med    low  unacc
4          3            3      2          2       med    med  unacc
…      …           …    …          …         …    …     …
1745       1            1  5more       more       med   high  vgood
1746       1            1  5more       more       med   high  vgood
1747       1            1  5more       more       big    low  unacc
1748       1            1  5more       more       big    med   good
1749       1            1  5more       more       big   high  vgood

[1750 rows x 7 columns]
```

[9]:
```
# Encode Doors usando groupBy de pandas
idxDF['Doors'] = df.groupby('Doors').ngroup()
idxDF
```

[9]:

	Buying	Maintenance	Doors	Passengers	TrunkSize	Safety	Class
0	3	3	0	2	small	low	unacc
1	3	3	0	2	small	med	unacc
2	3	3	0	2	small	high	unacc
3	3	3	0	2	med	low	unacc
4	3	3	0	2	med	med	unacc
…	…	…	…	…	…	…	…
1745	1	1	3	more	med	high	vgood
1746	1	1	3	more	med	high	vgood
1747	1	1	3	more	big	low	unacc
1748	1	1	3	more	big	med	good
1749	1	1	3	more	big	high	vgood

[1750 rows x 7 columns]

[10]:
```
# Encode Passengers usando groupBy de pandas
idxDF['Passengers'] = df.groupby('Passengers').ngroup()
idxDF
```

[10]:

	Buying	Maintenance	Doors	Passengers	TrunkSize	Safety	Class
0	3	3	0	0	small	low	unacc
1	3	3	0	0	small	med	unacc
2	3	3	0	0	small	high	unacc
3	3	3	0	0	med	low	unacc
4	3	3	0	0	med	med	unacc
…	…	…	…	…	…	…	…
1745	1	1	3	2	med	high	vgood
1746	1	1	3	2	med	high	vgood
1747	1	1	3	2	big	low	unacc
1748	1	1	3	2	big	med	good
1749	1	1	3	2	big	high	vgood

[1750 rows x 7 columns]

[11]:
```
# Encode TrunkSize usando groupBy de pandas
idxDF['TrunkSize'] = df.groupby('TrunkSize').ngroup()
idxDF
```

[11]:

	Buying	Maintenance	Doors	Passengers	TrunkSize	Safety	Class
0	3	3	0	0	2	low	unacc
1	3	3	0	0	2	med	unacc
2	3	3	0	0	2	high	unacc
3	3	3	0	0	1	low	unacc

```
4          3            3      0            0           1    med  unacc
…      …          …    …          …        …    …    …
1745       1            1      3            2           1   high  vgood
1746       1            1      3            2           1   high  vgood
1747       1            1      3            2           0    low  unacc
1748       1            1      3            2           0    med   good
1749       1            1      3            2           0   high  vgood

[1750 rows x 7 columns]
```

[12]:
```
# Encode Safety usando groupBy de pandas
idxDF['Safety'] = df.groupby('Safety').ngroup()
idxDF
```

[12]:
```
      Buying  Maintenance  Doors  Passengers  TrunkSize  Safety  Class
0          3            3      0           0          2       1  unacc
1          3            3      0           0          2       2  unacc
2          3            3      0           0          2       0  unacc
3          3            3      0           0          1       1  unacc
4          3            3      0           0          1       2  unacc
…      …          …    …          …        …    …    …
1745       1            1      3           2          1       0  vgood
1746       1            1      3           2          1       0  vgood
1747       1            1      3           2          0       1  unacc
1748       1            1      3           2          0       2   good
1749       1            1      3           2          0       0  vgood

[1750 rows x 7 columns]
```

[13]:
```
# Encode Class usando groupBy de pandas
idxDF['Class'] = df.groupby('Class').ngroup()
idxDF
```

[13]:
```
      Buying  Maintenance  Doors  Passengers  TrunkSize  Safety  Class
0          3            3      0           0          2       1      2
1          3            3      0           0          2       2      2
2          3            3      0           0          2       0      2
3          3            3      0           0          1       1      2
4          3            3      0           0          1       2      2
…      …          …    …          …        …    …    …
1745       1            1      3           2          1       0      3
1746       1            1      3           2          1       0      3
1747       1            1      3           2          0       1      2
1748       1            1      3           2          0       2      1
1749       1            1      3           2          0       0      3

[1750 rows x 7 columns]
```

[14]:
```
idxDF.dtypes
```

[14]:
```
Buying         int64
Maintenance    int64
Doors          int64
Passengers     int64
TrunkSize      int64
Safety         int64
Class          int64
dtype: object
```

RBL: Se confirma que con el truco todas las features no solo están condificadas a enteros, si no que python las reconoce como columnas enteras en el dataframe.

0.6 División del dataset en datos de entrenamiento y datos de test

[15]:
```
#Código que realice la división en entrenamiento y test, de acuerdo con la␣
 ↪estretgia de evluación planeada. Describa cuál es.
X = idxDF[['Buying', 'Maintenance', 'Doors', 'Passengers', 'TrunkSize',␣
 ↪'Safety']]
X
```

[15]:

	Buying	Maintenance	Doors	Passengers	TrunkSize	Safety
0	3	3	0	0	2	1
1	3	3	0	0	2	2
2	3	3	0	0	2	0
3	3	3	0	0	1	1
4	3	3	0	0	1	2
…	…	…	…	…	…	…
1745	1	1	3	2	1	0
1746	1	1	3	2	1	0
1747	1	1	3	2	0	1
1748	1	1	3	2	0	2
1749	1	1	3	2	0	0

[1750 rows x 6 columns]

[16]:
```
y = idxDF[['Class']]
y
```

[16]:

	Class
0	2
1	2
2	2
3	2
4	2
…	…

```
1745      3
1746      3
1747      2
1748      1
1749      3

[1750 rows x 1 columns]
```

[17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
↪random_state=25)
```

[18]:
```
X_train.shape
```

[18]:
```
(1400, 6)
```

[19]:
```
X_test.shape
```

[19]:
```
(350, 6)
```

0.7 Ajuste de los modelos de clasificación propuestos

Justifique la selección de las dos propuestas de modelación seleccionadas:

[20]:
```
#Código de ajuste del modelo de clasificación 1

# Modelo con K-Neighbors Classifier
knnModel = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
# knnModel.fit(X_train, y_train)                    # Genera el mensaje:
                                                    # DataConversionWarning: A␣
↪column-vector y was passed when a 1d array
                                                    # was expected. Please change␣
↪the shape of y to (n_samples,), for
                                                    # example using ravel().
knnModel.fit(X_train, y_train.values.ravel())       # Corrección y ya no genera␣
↪el mensaje

# Predict the results using KNN
y_pred = knnModel.predict(X_test)
```

[21]:
```
#Código de ajuste del modelo de clasificación 2
```

0.8 Evaluación de cada modelo

Al menos incluya:

- Instancias clasificadas correctamente
- Instancias clasificadas incorrectamente
- TP Rate

- FP Rate
- Matriz de confusión

[22]:
```
#Código para mostrar la evaluación del modelo de clasificación 1
# Confusion matrix ----------------------------------------
print("Confusion Matrix")
knnMatrix = confusion_matrix(y_test, y_pred)
print(knnMatrix)

# Classification report -----------------------------------
print("Classification Report")
knnReport = classification_report(y_test, y_pred)
print(knnReport)
print("Con los valores codificados:")
groups = df.groupby('Class')
for i in range(len(list(groups.groups.keys()))):
    print(f"{i:>12} = {list(groups.groups.keys())[i]}")

# Accuracy of the model
print()
knnAccuracy = accuracy_score(y_test, y_pred)
print(f"KNN Classification Accuracy of the model: {knnAccuracy:.2%}")
```

```
Confusion Matrix
[[ 50   1  18   0]
 [  9   5   2   0]
 [  1   0 247   1]
 [  4   1   1  10]]
Classification Report
              precision    recall  f1-score   support

           0       0.78      0.72      0.75        69
           1       0.71      0.31      0.43        16
           2       0.92      0.99      0.96       249
           3       0.91      0.62      0.74        16

    accuracy                           0.89       350
   macro avg       0.83      0.66      0.72       350
weighted avg       0.88      0.89      0.88       350

Con los valores codificados:
           0 = acc
           1 = good
           2 = unacc
           3 = vgood

KNN Classification Accuracy of the model: 89.14%
```

Construya un párrafo con los principales hallazgos.

[23]:
```
#Código  para mostrarla evaluación del modelo de clasificación 2
```

Construya un párrafo con los principales hallazgos.

0.9 Comparación del desempeño de modelos

[24]:
```
#Código para mostrar la comparación de métricas de desempeño de las dos␣
↪propuestas en tabla
```

[25]:
```
#Código para mostrar la comparación de métricas de desempeño de las dos␣
↪propuestas en gráfica
```

Construya un párrafo con los principales hallazgos.

0.10 Discusión de los resultados obtenidos y argumentos sobre cómo se podrían mejorar de dichos resultados

Realice en este espacio todo el anásis de resultados final incluyendo: ventajas y desventajas de cada modelo propuesto, Resultados comparados. Conclusiones objetivas y significantes con base a las diferentes métricas escogidas. Recomendaciones de mejora de las propuestas: