

✓ Práctica 1.1. Preprocesado de Audiology_soft

- **Asignatura:** Análisis Automático de Datos para las Ciencias Biomédicas, Medioambientales, Agroalimentarias
- **Estudiantes:** Mabrouka Salmi <z12salsm@uco.es>; Alba Marquez Rodriguez<z32maroa@uco.es>
- **Máster:** Inteligencia Computacional e Internet de las Cosas
- **Universidad:** Escuela Politécnica Superior de Córdoba -UCO
- **Curso:** 2023/2024


Cargue la base de datos Audiology_soft. [Audiology_soft](#) es una base de datos de la UCI que ya tiene algún preprocesamiento realizado.

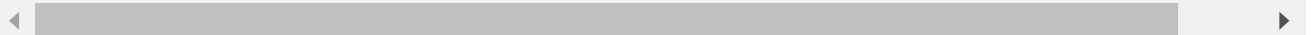
Trata sobre clasificación de enfermedades de oído (24 enfermedades) a partir de 10 atributos. Aplique los siguientes filtros, explicando cómo queda la base de datos en cada paso. Haga uso de capturas de pantallas y de lo que necesite para poder explicar el resultado:

1. Eliminar con el filtro Remove (o desde la pestaña Preprocess, elige los que quieres borrar y pulsa en Remove) aquellos atributos que tengan demasiados valores perdidos como para hacer imputaciones a partir de medias o moda. Muestra cómo queda la base de datos al eliminarlos.
2. Usar el filtro filters→unsupervised→attributes→ReplaceMissingValues para reemplazar valores perdidos de atributos que sean factibles para imputar. Muestra cómo queda la base de datos al aplicar el filtro.
3. Usar el filtro para pasar de filters→unsupervised→attributes→nominalToBinary, haciendo que los nominales binarios (con dos valores) se pasen también a dos numéricos y no queden como un solo atributo numérico con valores 0 o 1. Muestra cómo queda la base de datos al aplicar el filtro.

✓ 1.Cargando datos e importando las bibliotecas necesarias

```
# library necessary for reading the .arff datasets
!pip install liac-arff
```

 Requirement already satisfied: liac-arff in /usr/local/lib/python3.10/dist-packages (



```

from scipy.io import arff
import pandas as pd
import numpy as np

from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

```

✓ Cargar la base de datos Audiology_soft

Subimos primero el archivo a los archivos de Google Collab, y lo cargamos

```

# file_path = '/content/drive/MyDrive/TMUNI/audiology_soft_.arff'
file_path = 'audiology_soft_.arff'

# Train dataset is downloaded from: "https://github.com/geantrindade/TESHierarchicalDatas
data, meta = arff.loadarff(file_path)

# Turn train and test datasets into a dataframe
data = pd.DataFrame(data)

data.head() # return the 4 first rows of the dataframe

```



	age_gt_60	air	airBoneGap	ar_c	ar_u	bone	boneAbnormal	bser
0	b't'	b'mild'	b'f'	b'normal'	b'normal'	b'unmeasured'	b'f'	b'?
1	b't'	b'mild'	b'f'	b'normal'	b'normal'	b'unmeasured'	b'f'	b'?
2	b't'	b'normal'	b'f'	b'normal'	b'normal'	b'unmeasured'	b'f'	b'?
3	b't'	b'mild'	b'f'	b'normal'	b'normal'	b'unmeasured'	b'f'	b'?
4	b't'	b'mild'	b'f'	b'normal'	b'normal'	b'unmeasured'	b'f'	b'?

```

# Visualizar información sobre los atributos
print(meta)

```



```

Dataset: audiology
age_gt_60's type is nominal, range is ('f', 't')
air's type is nominal, range is ('mild', 'moderate', 'normal', 'profound', 's
airBoneGap's type is nominal, range is ('f', 't')
ar_c's type is nominal, range is ('absent', 'elevated', 'normal')
ar_u's type is nominal, range is ('absent', 'elevated', 'normal')
bone's type is nominal, range is ('mild', 'moderate', 'normal', 'unmeasured')
boneAbnormal's type is nominal, range is ('f', 't')
bser's type is nominal, range is ('degraded', 'normal')
history_buzzing's type is nominal, range is ('f', 't')
history_dizziness's type is nominal, range is ('f', 't')
class's type is nominal, range is ('acoustic_neuroma', 'bells_palsy', 'cochle

```

Hay columnas que están codificas, antes de empezar el preprocesamiento se van a decodificar:

```
# Decodificar las columnas de bytes a cadenas de texto
data = data.apply(lambda x: x.str.decode('utf-8') if x.dtype == 'object' else x)

# Mostrar las primeras filas del DataFrame decodificado
print(data.head())
```

```

age_gt_60    air airBoneGap    ar_c    ar_u    bone boneAbnormal bser \
0          t    mild          f  normal  normal  unmeasured          f    ?
1          t    mild          f  normal  normal  unmeasured          f    ?
2          t  normal          f  normal  normal  unmeasured          f    ?
3          t    mild          f  normal  normal  unmeasured          f    ?
4          t    mild          f  normal  normal  unmeasured          f    ?

history_buzzing history_dizziness    class
0              f              f    cochlear_age
1              f              f    cochlear_age
2              f              f  cochlear_age_and_noise
3              f              f  cochlear_age_and_noise
4              f              f    cochlear_age
```

✓ 1. Eliminar atributos con valores perdidos

Eliminar con el filtro Remove (o desde la pestaña Preprocess, elige los que quieres borrar y pulsa en Remove) aquellos atributos que tengan demasiados valores perdidos como para hacer imputaciones a partir de medias o moda. Muestra cómo queda la base de datos al eliminarlos.

En este ejemplo, eliminamos columnas con más del X% de valores perdidos siendo X un threshold, en este caso pondremos un 30% de los valores perdidos.

```
# Reemplazar '?' con NaN
data.replace('?', np.nan, inplace=True)

# Mostrar las primeras filas del DataFrame con los valores perdidos reemplazados
print(data.head())
```

```

age_gt_60    air airBoneGap    ar_c    ar_u    bone boneAbnormal bser \
0          t    mild          f  normal  normal  unmeasured          f  NaN
1          t    mild          f  normal  normal  unmeasured          f  NaN
2          t  normal          f  normal  normal  unmeasured          f  NaN
3          t    mild          f  normal  normal  unmeasured          f  NaN
4          t    mild          f  normal  normal  unmeasured          f  NaN

history_buzzing history_dizziness    class
0              f              f    cochlear_age
1              f              f    cochlear_age
2              f              f  cochlear_age_and_noise
3              f              f  cochlear_age_and_noise
4              f              f    cochlear_age
```

```
# Establecer threshold - valor umbral y eliminar columnas
threshold = 0.3
data = data.dropna(thresh=len(data) * threshold, axis=1)
```

```
# Mostrar base de datos tras preprocesado
print(data.head())
```

```
↩  age_gt_60    air airBoneGap    ar_c    ar_u    bone boneAbnormal \
0         t    mild          f  normal  normal  unmeasured      f
1         t    mild          f  normal  normal  unmeasured      f
2         t  normal          f  normal  normal  unmeasured      f
3         t    mild          f  normal  normal  unmeasured      f
4         t    mild          f  normal  normal  unmeasured      f

    history_buzzing history_dizziness    class
0                f                f    cochlear_age
1                f                f    cochlear_age
2                f                f  cochlear_age_and_noise
3                f                f  cochlear_age_and_noise
4                f                f    cochlear_age
```

✓ 2. Reemplazar valores perdidos de atributos factibles

```
# Creamos un SimpleImputer con estrategia 'most_frequent' para reemplazar con la moda
imputer = SimpleImputer(strategy='most_frequent')
```

```
# Aplicamos el imputer a nuestro DataFrame
data = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
```

```
# Mostramos las primeras filas del DataFrame con los valores perdidos reemplazados
print(data.head())
```

```
↩  age_gt_60    air airBoneGap    ar_c    ar_u    bone boneAbnormal \
0         t    mild          f  normal  normal  unmeasured      f
1         t    mild          f  normal  normal  unmeasured      f
2         t  normal          f  normal  normal  unmeasured      f
3         t    mild          f  normal  normal  unmeasured      f
4         t    mild          f  normal  normal  unmeasured      f

    history_buzzing history_dizziness    class
0                f                f    cochlear_age
1                f                f    cochlear_age
2                f                f  cochlear_age_and_noise
3                f                f  cochlear_age_and_noise
4                f                f    cochlear_age
```

✓ 3. Convertir atributos nominales binarios

Usar el filtro para pasar de filters→unsupervised→attributes→nominalToBinary, haciendo que los nominales binarios (con dos valores) se pasen también a dos numéricos y no queden como

un solo atributo numérico con valores 0 o 1. Muestra cómo queda la base de datos al aplicar el filtro.

```
binary_columns = []

for column in data.columns:
    unique_values = data[column].unique()
    if len(unique_values) == 2:
        binary_columns.append(column)

print("Columnas binarias:", binary_columns)
```

⇒ Columnas binarias: ['age_gt_60', 'airBoneGap', 'boneAbnormal', 'history_buzzing', 'hi

```
# Obtén las columnas binarias identificadas previamente
binary_columns = ['age_gt_60', 'airBoneGap', 'boneAbnormal', 'history_buzzing', 'history_

# Aplica one-hot encoding a las columnas binarias
data = pd.get_dummies(data, columns=binary_columns)

# Dropear las columnas que terminan con '_f'
columns_to_drop = [col for col in data.columns if col.endswith('_f')]
data.drop(columns=columns_to_drop, inplace=True)

# Renombrar las columnas que terminan con '_t'
data.columns = [col.replace('_t', '') for col in data.columns]

# Muestra las primeras filas del DataFrame con variables binarias convertidas
print(data.head())
```

⇒

	air	ar_c	ar_u	bone	class	age_gt_60	\
0	mild	normal	normal	unmeasured	cochlear_age	True	
1	mild	normal	normal	unmeasured	cochlear_age	True	
2	normal	normal	normal	unmeasured	cochlear_age_and_noise	True	
3	mild	normal	normal	unmeasured	cochlear_age_and_noise	True	
4	mild	normal	normal	unmeasured	cochlear_age	True	

	airBoneGap	boneAbnormal	history_buzzing	history_dizziness
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False