Práctica 1

Inteligencia de Negocio

Amin Kasrou Aouam

UNIVERSIDAD
DE GRANADA

2020-11-10

Índice

Práctica 1

Introducción

En esta práctica, usaremos distintos algoritmos de aprendizaje automático para resolver un problema de clasificación.

Procesado de datos

Antes de proceder con el entrenamiento de los distintos modelos, debemos realizar un preprocesado de los datos, para asegurarnos que nuestros modelos aprenden de un *dataset* congruente.

La integridad de la lógica del preprocesado se encuentra en el archivo *preprocessing.py*, cuyo contenido mostramos aquí:

```python
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold


def replace_values(df):
    columns = ["BI-RADS", "Margin", "Density", "Age"]
    for column in columns:
        df[column].fillna(value=df[column].mean(), inplace=True)
    return df


def process_na(df, action):
    if action == "drop":
        return df.dropna()
    elif action == "fill":
        return replace_values(df)
    else:
        print("Unknown action selected. The choices are: ")
        print("fill: fills the na values with the mean")
        print("drop: drops the na values")
        exit()


def encode_columns(df):
    label_encoder = LabelEncoder()
    encoded_df = df.copy()
    encoded_df["Shape"] = label_encoder.fit_transform(df["Shape"])
    encoded_df["Severity"] = label_encoder.fit_transform(df["Severity"
        ])
    return encoded_df


def split_train_target(df):
    train_data = df.drop(columns=["Severity"])
    target_data = df["Severity"]
    return train_data, target_data


def split_k_sets(df):
    k_fold = KFold(shuffle=True, random_state=42)
    return k_fold.split(df)


def parse_data(source, action):
    df = read_csv(filepath_or_buffer=source, na_values="?")
    processed_df = process_na(df=df, action=action)
    encoded_df = encode_columns(df=processed_df)
    test_data, target_data = split_train_target(df=encoded_df)
    return test_data, target_data
```

A continuación, mostraremos cada uno de los pasos que realizamos para obtener el *dataset* final:

Valores nulos

Nuestro *dataset* contiene valores nulos, representados mediante un signo de interrogación (?). Optamos por evaluar 2 estrategias:

1. Eliminar los valores nulos

```
df = read_csv(filepath_or_buffer="../data/mamografia.csv",
    na_values="?")
processed_df = process_na(df=df, action="drop")
print("DataFrame sin preprocesamiento: ")
print(df.describe())
print("DataFrame sin preprocesamiento: ")
print(processed_df.describe())
```

```
DataFrame sin preprocesamiento:
          BI-RADS         Age      Margin     Density
count  959.000000  956.000000  913.000000  885.000000
mean     4.296142   55.487448    2.796276    2.910734
std      0.706291   14.480131    1.566546    0.380444
min      0.000000   18.000000    1.000000    1.000000
25%      4.000000   45.000000    1.000000    3.000000
50%      4.000000   57.000000    3.000000    3.000000
75%      5.000000   66.000000    4.000000    3.000000
max      6.000000   96.000000    5.000000    4.000000
DataFrame sin preprocesamiento:
          BI-RADS         Age      Margin     Density
count  847.000000  847.000000  847.000000  847.000000
mean     4.322314   55.842975    2.833530    2.909091
std      0.703762   14.603754    1.564049    0.370292
min      0.000000   18.000000    1.000000    1.000000
25%      4.000000   46.000000    1.000000    3.000000
50%      4.000000   57.000000    3.000000    3.000000
75%      5.000000   66.000000    4.000000    3.000000
max      6.000000   96.000000    5.000000    4.000000
```

   Observamos que el número de instancias disminuye considerablemente, hasta un máximo de 112, en el caso del *BI-RADS*. Aún así, los valores de la media y desviación estándar no se ven afectados de forma considerable.

2. Imputar su valor con la media

```
df = read_csv(filepath_or_buffer="../data/mamografia.csv",
    na_values="?")
processed_df = process_na(df=df, action="fill")
print("DataFrame sin preprocesamiento: ")
print(df.describe())
print("DataFrame sin preprocesamiento: ")
print(processed_df.describe())
```

```
DataFrame sin preprocesamiento:
           BI-RADS         Age      Margin     Density
count  961.000000  961.000000  961.000000  961.000000
mean     4.296142   55.487448    2.796276    2.910734
std      0.705555   14.442373    1.526880    0.365074
min      0.000000   18.000000    1.000000    1.000000
25%      4.000000   45.000000    1.000000    3.000000
50%      4.000000   57.000000    3.000000    3.000000
75%      5.000000   66.000000    4.000000    3.000000
max      6.000000   96.000000    5.000000    4.000000
DataFrame sin preprocesamiento:
           BI-RADS         Age      Margin     Density
count  961.000000  961.000000  961.000000  961.000000
mean     4.296142   55.487448    2.796276    2.910734
std      0.705555   14.442373    1.526880    0.365074
min      0.000000   18.000000    1.000000    1.000000
25%      4.000000   45.000000    1.000000    3.000000
50%      4.000000   57.000000    3.000000    3.000000
75%      5.000000   66.000000    4.000000    3.000000
max      6.000000   96.000000    5.000000    4.000000
```

Esta alternativa nos permite mantener el número de instancias en todas las columnas, sin alterar la media ni la desviación típica.

Valores no números

La mayoría de algoritmos de aprendizaje automática trabaja con datos numéricos, desafortunadamente nuestro *dataset* contiene dos columnas con datos descriptivos.

Procedemos a convertirlos en valores numéricos mediante un *LabelEncoder*:

```
encoded_df = encode_columns(df=processed_df)
print(encoded_df.head())
```

```
   BI-RADS   Age  Shape  Margin   Density  Severity
0      5.0  67.0      1     5.0  3.000000         1
1      4.0  43.0      4     1.0  2.910734         1
2      5.0  58.0      0     5.0  3.000000         1
3      4.0  28.0      4     1.0  3.000000         0
4      5.0  74.0      4     5.0  2.910734         1
```

Vemos como las columnas **Shape** y **Severity** se componen ahora únicamente de valores numéricos.

Separación de datos

Como último paso, separamos la columna objetivo de los demás datos.

```
test_data, target_data = split_train_target(df=encoded_df)
print("Datos de entrenamiento: ")
print(test_data.head())
print("Datos objetivo: ")
print(target_data.head())
```

```
Datos de entrenamiento:
   BI-RADS   Age  Shape  Margin   Density
0      5.0  67.0      1     5.0  3.000000
1      4.0  43.0      4     1.0  2.910734
2      5.0  58.0      0     5.0  3.000000
3      4.0  28.0      4     1.0  3.000000
4      5.0  74.0      4     5.0  2.910734
Datos objetivo:
0    1
1    1
2    1
3    0
4    1
Name: Severity, dtype: int64
```