Creado por:

Isabel Maniega

In [1]:
```
# pip install pandas
```

In [2]:
```
# pip install scikit-learn
```

In [3]:
```
# pip install seaborn
```

In [10]:
```
import pandas as pd
from sklearn.datasets import fetch_openml
import seaborn as sns
from sklearn.impute import SimpleImputer
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

Cargar el titanic con sklearn

In [6]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]
df.head()
```

Out[6]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

Mostrar las columnas sin datos

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype   
---  ------     --------------  -----   
 0   pclass     1309 non-null   float64 
 1   name       1309 non-null   object  
 2   sex        1309 non-null   category
 3   age        1046 non-null   float64 
 4   sibsp      1309 non-null   float64 
 5   parch      1309 non-null   float64 
 6   ticket     1309 non-null   object  
 7   fare       1308 non-null   float64 
 8   cabin      295 non-null    object  
 9   embarked   1307 non-null   category
 10  boat       486 non-null    object  
 11  body       121 non-null    float64 
 12  home.dest  745 non-null    object  
dtypes: category(2), float64(6), object(5)
memory usage: 115.4+ KB
```

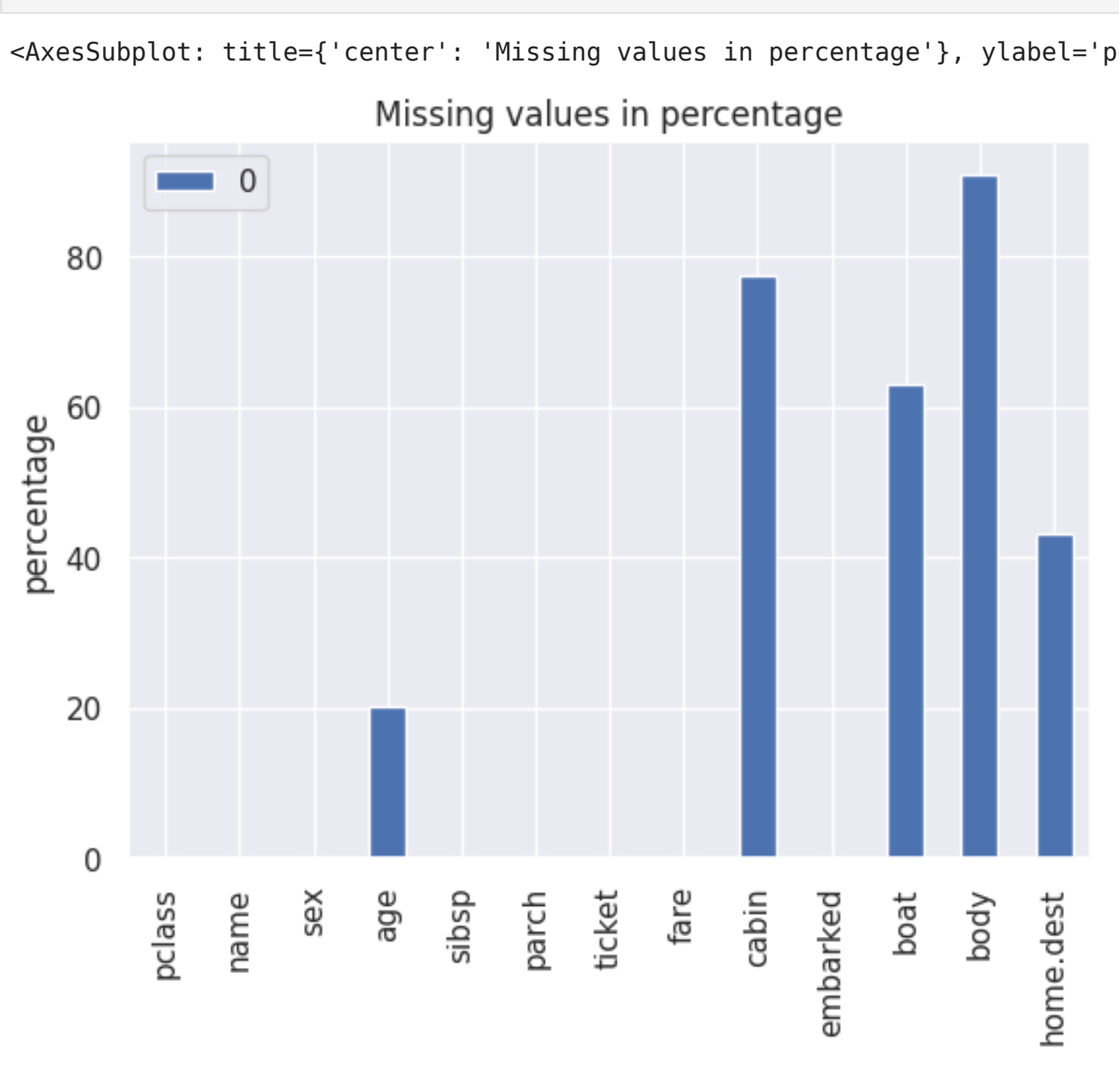
In [8]:
```
df.isnull().sum()
```

Out[8]:
```
pclass          0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

In [11]:
```
# Visualización de los datos

sns.set()
miss_vals = pd.DataFrame(df.isnull().sum()/ len(df)*100)
miss_vals.plot(kind="bar",
               title="Missing values in percentage",
               ylabel="percentage")
```

Out[11]:
```
<AxesSubplot: title={'center': 'Missing values in percentage'}, ylabel='percentage'>
```



Procedimiento para valores nulos

Existen dos maneras:

- Eliminar la columna
- Asignamos a los valores la media, mediana, moda, etc

Eliminación

- Eliminamos los valores nulos:

In [13]:
```
print(f"Size of the dataset: {df.shape}")
df.drop(["cabin", "boat", "body", "home.dest"], axis=1, inplace=True)
df.dropna(inplace=True)
print(f"Size of the dataset: {df.shape}")
```

```
Size of the dataset: (1309, 13)
Size of the dataset: (1043, 9)
```

Sustitución

- Sustituir por el valor más común (media):

In [14]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]
df.head()
```

Out[14]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

In [15]:
```
print(f"Número de valores nulos de la columna edad: {df.age.isnull().sum()}")
```

```
Número de valores nulos de la columna edad: 263
```

In [16]:
```
df["age"].fillna(df["age"].mean(), inplace=True)
print(f"Número de valores nulos de la columna edad: {df.age.isnull().sum()}")
```

```
Número de valores nulos de la columna edad: 0
```

- Otra opción: Simple transformación con Sklearn

In [17]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]
df.head()
```

Out[17]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

In [18]:
```
print(f"Número de valores nulos de la columna edad: {df.age.isnull().sum()}")
```

```
Número de valores nulos de la columna edad: 263
```

In [20]:
```
imp = SimpleImputer(strategy="mean")
df["age"] = imp.fit_transform(df[["age"]])
print(f"Número de valores nulos de la columna edad: {df.age.isnull().sum()}")
```

```
Número de valores nulos de la columna edad: 0
```

In [22]:
```
print("Tipos de datos con valores Nulos:")
for col in df.columns[df.isnull().any()]:
    print(col, df[col][df[col].isnull()].values[0])
```

```
Tipos de datos con valores Nulos:
fare nan
cabin None
embarked nan
boat None
body nan
home.dest None
```

- Modificamos los None:

In [23]:
```
imp = SimpleImputer(missing_values=None, strategy="most_frequent")
df["cabin"] = imp.fit_transform(df[["cabin"]])
print(f"Número de valores nulos de la columna cabina: {df.cabin.isnull().sum()}")
```

```
Número de valores nulos de la columna cabina: 0
```

In [24]:
```
def get_parameters(df):
    parameters = {}
    for col in df.columns[df.isnull().any()]:
        if df[col].dtype == "float64" or df[col].dtype == "int64" or df[col].dtype == "int32":
            strategy = "mean"
        else:
            strategy = "most_frequent"
        missing_values = df[col][df[col].isnull()].values[0]
        parameters[col] = {"missing_values": missing_values, "strategy": strategy}
    return parameters
get_parameters(df)
```

Out[24]:
```
{'fare': {'missing_values': nan, 'strategy': 'mean'},
 'embarked': {'missing_values': nan, 'strategy': 'most_frequent'},
 'boat': {'missing_values': None, 'strategy': 'most_frequent'},
 'body': {'missing_values': nan, 'strategy': 'mean'},
 'home.dest': {'missing_values': None, 'strategy': 'most_frequent'}}
```

In [25]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]
df.head()
```

Out[25]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

In [26]:
```
parameters = get_parameters(df)

for col, param in parameters.items():
    missing_values = param["missing_values"]
    strategy = param["strategy"]
    imp = SimpleImputer(missing_values=missing_values, strategy=strategy)
    df[col] = imp.fit_transform(df[[col]])

df.isnull().sum()
```

Out[26]:
```
pclass       0
name         0
sex          0
age          0
sibsp        0
parch        0
ticket       0
fare         0
cabin        0
embarked     0
boat         0
body         0
home.dest    0
dtype: int64
```

In [27]:
```
df.head()
```

Out[27]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	160.809917	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	160.809917	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	13	160.809917	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	13	135.000000	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	13	160.809917	Montreal, PQ / Chesterville, ON

Crear nuevas características (Feature Engineering)

- Sibsp: pasajeros que viajan con hermanos
- Parch: viajeros que viajan con niños

Calculamos el número de pasajeros que viajan solos:

In [29]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]
df.head()
```

Out[29]:

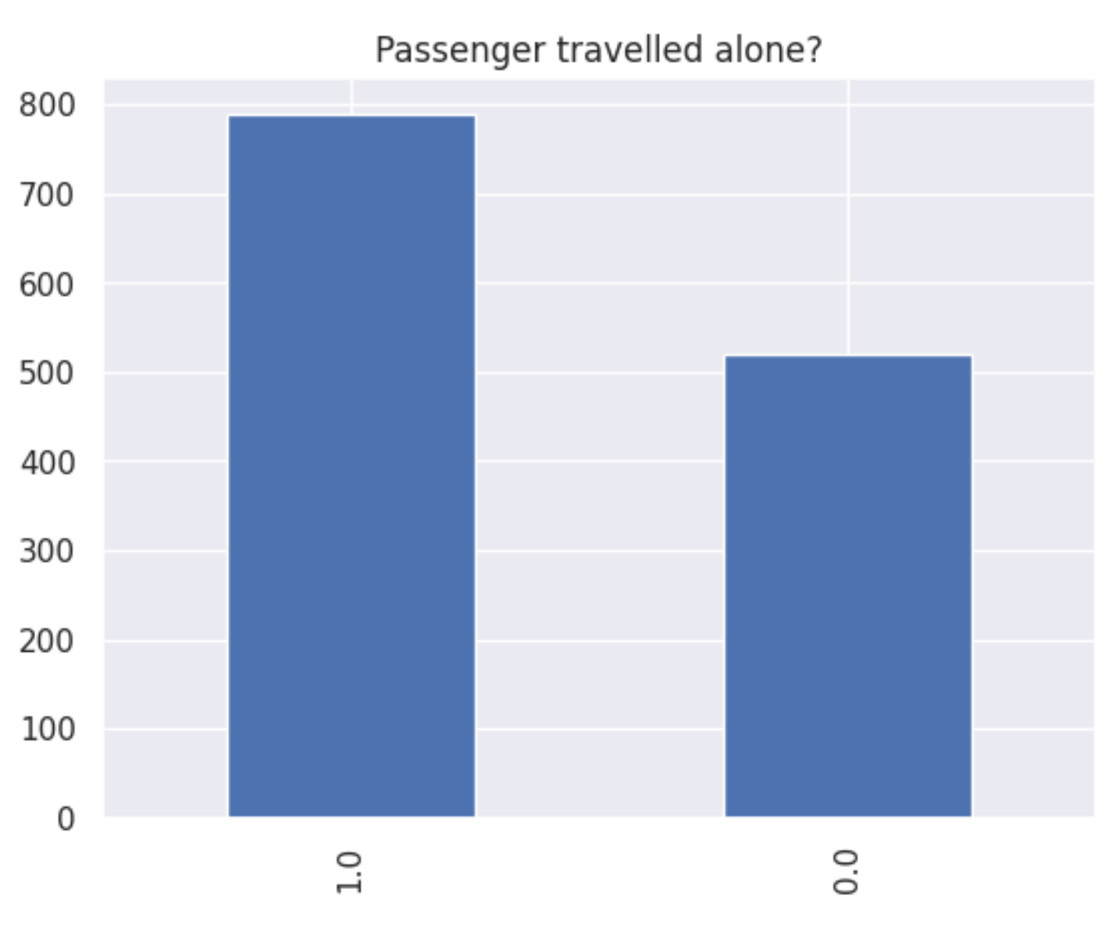
	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

In [30]:
```
df["family"] = df["sibsp"] + df["parch"]

df.loc[df["family"] > 0, "travelled_alone"] = 0
df.loc[df["family"] == 0, "travelled_alone"] = 1

df["travelled_alone"].value_counts().plot(title="Passenger travelled alone?", kind="bar")
```

Out[30]:
```
<AxesSubplot: title={'center': 'Passenger travelled alone?'}>
```



Encode categorical features

- scikit-learn: OneHotEncoder()
- pandas: get_dummies()

In [31]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]
df.head()
```

Out[31]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

In [32]:
```
df[["female", "male"]] = OneHotEncoder().fit_transform(df[['sex']]).toarray()
df[["sex", "female", "male"]]
```

Out[32]:

	sex	female	male
0	female	1.0	0.0
1	male	0.0	1.0
2	female	1.0	0.0
3	male	0.0	1.0
4	female	1.0	0.0
...	...	...	...
1304	female	1.0	0.0
1305	female	1.0	0.0
1306	male	0.0	1.0
1307	male	0.0	1.0
1308	male	0.0	1.0

1309 rows × 3 columns

Eliminaremos uno de las columnas para evitar la colinealidad

In [34]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]
df["sex"] = OneHotEncoder().fit_transform(df[['sex']]).toarray()[:, 1]
df.head()
```

Out[34]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	0.0	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	1.0	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	0.0	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	1.0	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	0.0	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

0 == female; 1== male

- Pandas:

In [35]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]

df["sex"] = pd.get_dummies(df["sex"], drop_first=True)
df.head()
```

Out[35]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	0	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	1	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	0	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	1	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	0	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

0 == female; 1== male

Encoding all categorical features

In [36]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]

cat_cols = df.select_dtypes(include=["category"]).columns
print(f"Columnas Categoricas: {cat_cols}")
```

```
Columnas Categoricas: Index(['sex', 'embarked'], dtype='object')
```

In [38]:
```
for col in cat_cols:
    fill_value = df[col].mode()[0]
    df[col].fillna(fill_value, inplace=True)

    append_to = list(df[col].unique())

    print(append_to)

    df[append_to] = OneHotEncoder().fit_transform(df[[col]]).toarray()

    df.drop(col, axis=1, inplace=True)
    df.drop(append_to[0], axis=1, inplace=True)

print(df.columns)
df[["male", "C", "Q"]].head()
```

```
['female', 'male']
['S', 'C', 'Q']
Index(['pclass', 'name', 'age', 'sibsp', 'parch', 'ticket', 'fare', 'cabin',
       'boat', 'body', 'home.dest', 'male', 'C', 'Q'],
      dtype='object')
```

Out[38]:

	male	C	Q
0	0.0	0.0	1.0
1	1.0	0.0	1.0
2	0.0	0.0	1.0
3	1.0	0.0	1.0
4	0.0	0.0	1.0

In [39]:
```
df.head()
```

Out[39]:

	pclass	name	age	sibsp	parch	ticket	fare	cabin	boat	body	home.dest	male	C	Q
0	1.0	Allen, Miss. Elisabeth Walton	29.0000	0.0	0.0	24160	211.3375	B5	2	NaN	St Louis, MO	0.0	0.0	1.0
1	1.0	Allison, Master. Hudson Trevor	0.9167	1.0	2.0	113781	151.5500	C22 C26	11	NaN	Montreal, PQ / Chesterville, ON	1.0	0.0	1.0
2	1.0	Allison, Miss. Helen Loraine	2.0000	1.0	2.0	113781	151.5500	C22 C26	None	NaN	Montreal, PQ / Chesterville, ON	0.0	0.0	1.0
3	1.0	Allison, Mr. Hudson Joshua Creighton	30.0000	1.0	2.0	113781	151.5500	C22 C26	None	135.0	Montreal, PQ / Chesterville, ON	1.0	0.0	1.0
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	25.0000	1.0	2.0	113781	151.5500	C22 C26	None	NaN	Montreal, PQ / Chesterville, ON	0.0	0.0	1.0

MinMaxScaler

MinMaxScaler() pone todos los valores númericos de 0 a 1:

In [42]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]

num_cols = df.select_dtypes(include=["int64", "int32", "float64"]).columns
print(num_cols)
```

```
Index(['pclass', 'age', 'sibsp', 'parch', 'fare', 'body'], dtype='object')
```

In [43]:
```
for col in num_cols:
    fill_value = df[col].mean()
    df[col].fillna(fill_value, inplace=True)

minmax = MinMaxScaler()

df[num_cols] = minmax.fit_transform(df[num_cols])
df[num_cols]
```

Out[43]:

	pclass	age	sibsp	parch	fare	body
0	0.0	0.361169	0.000	0.000000	0.412503	0.488715
1	0.0	0.009395	0.125	0.222222	0.295806	0.488715
2	0.0	0.022964	0.125	0.222222	0.295806	0.488715
3	0.0	0.373695	0.125	0.222222	0.295806	0.409786
4	0.0	0.311064	0.125	0.222222	0.295806	0.488715
...	...	...	...	...	...	...
1304	1.0	0.179540	0.125	0.000000	0.028213	1.000000
1305	1.0	0.372206	0.125	0.000000	0.028213	0.488715
1306	1.0	0.329854	0.000	0.000000	0.014102	0.926606
1307	1.0	0.336117	0.000	0.000000	0.014102	0.488715
1308	1.0	0.361169	0.000	0.000000	0.015371	0.488715

1309 rows × 6 columns

StandardScaler

StandardScaler() poner todos los valores tengan una media de 0 y de desviación de 1

In [44]:
```
df = fetch_openml("titanic", version=1, as_frame=True)["data"]

num_cols = df.select_dtypes(include=["int64", "int32", "float64"]).columns
print(num_cols)
```

```
Index(['pclass', 'age', 'sibsp', 'parch', 'fare', 'body'], dtype='object')
```

In [45]:
```
for col in num_cols:
    fill_value = df[col].mean()
    df[col].fillna(fill_value, inplace=True)

ss = StandardScaler()

df[num_cols] = ss.fit_transform(df[num_cols])
df[num_cols].head()
```

Out[45]:

	pclass	age	sibsp	parch	fare	body
0	-1.546098	-0.068420	-0.479087	-0.445000	3.442480	0.000000
1	-1.546098	-2.249092	0.481288	1.866526	2.286476	0.000000
2	-1.546098	-2.164974	0.481288	1.866526	2.286476	0.000000
3	-1.546098	0.009230	0.481288	1.866526	2.286476	-0.872539
4	-1.546098	-0.379021	0.481288	1.866526	2.286476	0.000000

In [46]:
```
df[num_cols].describe()
```

Out[46]:

	pclass	age	sibsp	parch	fare	body
count	1.309000e+03	1.309000e+03	1.309000e+03	1.309000e+03	1.309000e+03	1.309000e+03
mean	-1.737003e-16	1.519878e-16	-8.142201e-18	1.628440e-17	4.342507e-17	-2.578364e-17
std	1.000382e+00	1.000382e+00	1.000382e+00	1.000382e+00	1.000382e+00	1.000382e+00
min	-1.546098e+00	-2.307330e+00	-4.790868e-01	-4.449995e-01	-6.437751e-01	-5.402590e+00
25%	-3.520907e-01	-6.119712e-01	-4.790868e-01	-4.449995e-01	-4.911082e-01	0.000000e+00
50%	8.419164e-01	2.758687e-16	-4.790868e-01	-4.449995e-01	-3.643001e-01	0.000000e+00
75%	8.419164e-01	3.974806e-01	4.812878e-01	-4.449995e-01	-3.906640e-02	0.000000e+00
max	8.419164e-01	3.891737e+00	7.203909e+00	9.956864e+00	9.262219e+00	5.652087e+00

Usando el método de describe() podemos ver la media y la desvicación estandar de las columnas escaladas.

La media no parece ser igual a 0 pero, de hecho 4.342507e-17 es igual 0,000000000000000043425. Esto es tan cercano a 0 que puede considerarse igual a 0. Lo mismo ocurre con la desviación estándar que es tan cercana a 1 que puede considerarse igual a 1.

Creado por:

Isabel Maniega