# Inżyniera cech - data split

## 1. Zbiór danych Titanic

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

In [2]:
```
df = pd.read_csv("Zbior_danych_Titanic_reduced.csv")
df.head(2)
```

Out[2]:

| | pclass | survived | name | sex | age | sibsp | parch | ticket | fare | cabin | e |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 1 | Allen, Miss. Elisabeth Walton | female | 29.0000 | 0.0 | 0.0 | 24160 | 211.3375 | B5 | |
| **1** | 1.0 | 1 | Allison, Master. Hudson Trevor | male | 0.9167 | 1.0 | 2.0 | 113781 | 151.5500 | C22 C26 | |

## 2. train_test_split - moje obserwacje

Jest to funkcja do podziału danych na zbiory **treningowe** i **testowe**. Umożliwia ona szybkie przygotowanie danych do trenowania i testowania modeli ML. Można ustawić proporcję podziału przez `test_size` lub `train_size`. Dzięki parametrowi `random_state` wyniki można łatwo powtórzyć. Przydatną opcją jest też `stratify`, która pozwala zachować proporcje klas w obu zbiorach. Funkcja działa na różnych typach danych i jest bardzo wygodna w użyciu.

## 3. Zmienna, do której zapisano listę z nazwami trzech kolumn – kabiny, zredukowane kabiny oraz płeć.

In [3]:
```
col_name = ['cabin', 'CabinReduced', 'sex']
```

## 4. Podział zbioru na treningowy i testowy.

In [4]:
```
# Zmienne niezależne i zależna
X = df[col_name]
y = df['survived']

# Podział danych
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

# Wymiary zbiorów
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

```
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (916, 3)
X_test shape: (393, 3)
y_train shape: (916,)
y_test shape: (393,)
```

Wymiary pokazują, ile próbek trafiło do zbioru treningowego i testowego. Przy `test_size` = 0.3 dane zostały podzielone w proporcji ok 70% do 30%. Każdy wiersz odpowiada jednej osobie, a kolumny to wybrane cechy (przy zmiennej niezależnej X, są to 3 cechy; dla zmiennej zależnej y - nie mamy podanje liczby kolumn ale wiemy, że jest to tylko ta jedna kolumna). Liczba wierszy w X_train i y_train musi się zgadzać, podobnie w testowych.

## 5. Czy rozkład etykiet poszczególnych cech w zbiorach testowych i treningowych jest równomierny - w zależności od kardynalności danej zmiennej.

In [5]:
```
for col in col_name:
    print(f"\nZmienna: {col}")
    unique_test = [x for x in X_test[col].unique() if x not in X_train[col].uniq
    print("Unikalne etykiety w zbiorze testowym, których nie ma w zbiorze trenin
    print("Liczba:", len(unique_test))
```

```
Zmienna: cabin
Unikalne etykiety w zbiorze testowym, których nie ma w zbiorze treningowym: [nan,
'E12', 'C104', 'A31', 'D11', 'D48', 'D10 D12', 'B38', 'D45', 'C50', 'C31', 'B82 B
84', 'A32', 'C53', 'B10', 'C70', 'A23', 'C106', 'C46', 'E58', 'B11', 'F E69', 'B8
0', 'E39 E41', 'D22', 'E40', 'A19', 'C32', 'B79', 'C45', 'B22', 'B39', 'C47', 'B1
01', 'A7', 'E52', 'F38']
Liczba: 37

Zmienna: CabinReduced
Unikalne etykiety w zbiorze testowym, których nie ma w zbiorze treningowym: []
Liczba: 0

Zmienna: sex
Unikalne etykiety w zbiorze testowym, których nie ma w zbiorze treningowym: []
Liczba: 0
```

Zmienna `cabin` w oryginalnej formie jest trudna do modelowania ze względu na dużą liczbę unikalnych i rzadko powtarzających się wartości. Redukcja do `CabinReduced` pozwala zachować część informacji, jednocześnie ograniczając ryzyko pojawienia się niewidzianych wcześniej kategorii w zbiorze testowym. W przypadku zmiennych o niskiej kardynalności, jak `sex`, problem ten nie występuje i dane rozkładają się równomiernie.

## 6. Kodowanie zmiennych kategorycznych do zmiennych liczbowych.

In [6]:
```
dictionary = {}

for col in col_name:
    unique_vals = X_train[col].dropna().unique() # biore niepuste wartosci z kol
    val2num = {val: i+1 for i, val in enumerate(unique_vals)}
    val2num[np.nan] = 0  # NaN jako 0
```

```
    dictionary[col] = val2num
    print(f"\n{col}_map = {val2num}")
```

```
cabin_map = {'E36': 1, 'C68': 2, 'E24': 3, 'C22 C26': 4, 'D38': 5, 'B50': 6, 'A2
4': 7, 'C111': 8, 'F': 9, 'C6': 10, 'C87': 11, 'E8': 12, 'B45': 13, 'C93': 14, 'D
28': 15, 'D36': 16, 'C125': 17, 'B35': 18, 'T': 19, 'B73': 20, 'B57 B59 B63 B66':
21, 'A26': 22, 'A18': 23, 'B96 B98': 24, 'G6': 25, 'C78': 26, 'C101': 27, 'D9': 2
8, 'D33': 29, 'C128': 30, 'E50': 31, 'B26': 32, 'B69': 33, 'E121': 34, 'C123': 3
5, 'B94': 36, 'A34': 37, 'D': 38, 'C39': 39, 'D43': 40, 'E31': 41, 'B5': 42, 'D1
7': 43, 'F33': 44, 'E44': 45, 'D7': 46, 'A21': 47, 'D34': 48, 'A29': 49, 'D35': 5
0, 'A11': 51, 'B51 B53 B55': 52, 'D46': 53, 'E60': 54, 'C30': 55, 'D26': 56, 'E6
8': 57, 'A9': 58, 'B71': 59, 'D37': 60, 'F2': 61, 'C55 C57': 62, 'C89': 63, 'C12
4': 64, 'C23 C25 C27': 65, 'C126': 66, 'E49': 67, 'F E46': 68, 'E46': 69, 'D19':
70, 'B58 B60': 71, 'C82': 72, 'B52 B54 B56': 73, 'C92': 74, 'E45': 75, 'F G73': 7
6, 'C65': 77, 'E25': 78, 'B3': 79, 'D40': 80, 'C91': 81, 'B102': 82, 'B61': 83,
'F G63': 84, 'A20': 85, 'B36': 86, 'C7': 87, 'B77': 88, 'D20': 89, 'C148': 90, 'C
105': 91, 'E38': 92, 'B86': 93, 'C132': 94, 'C86': 95, 'A14': 96, 'C54': 97, 'A
5': 98, 'B49': 99, 'B28': 100, 'B24': 101, 'C2': 102, 'F4': 103, 'A6': 104, 'C8
3': 105, 'B42': 106, 'A36': 107, 'C52': 108, 'D56': 109, 'C116': 110, 'B19': 111,
'E77': 112, 'F E57': 113, 'E101': 114, 'B18': 115, 'C95': 116, 'D15': 117, 'E33':
118, 'B30': 119, 'D21': 120, 'E10': 121, 'C130': 122, 'D6': 123, 'C51': 124, 'D3
0': 125, 'E67': 126, 'C110': 127, 'C103': 128, 'C90': 129, 'C118': 130, 'C97': 13
1, 'D47': 132, 'E34': 133, 'B4': 134, 'D50': 135, 'C62 C64': 136, 'E17': 137, 'B4
1': 138, 'C49': 139, 'C85': 140, 'B20': 141, 'C28': 142, 'E63': 143, 'C99': 144,
'D49': 145, 'A10': 146, 'A16': 147, 'B37': 148, 'C80': 149, 'B78': 150, nan: 0}

CabinReduced_map = {'N': 1, 'E': 2, 'C': 3, 'D': 4, 'B': 5, 'A': 6, 'F': 7, 'T':
8, 'G': 9, nan: 0}

sex_map = {'female': 1, 'male': 2, nan: 0}
```

W zadaniu 6 wykonuję kodowanie zmiennych kategorycznych na zbiorze treningowym, ponieważ to na tym zbiorze uczymy nasz model. Wartości w zbiorze treningowym są podstawą do nauczenia algorytmu, dlatego musimy przygotować dane, by były zgodne z wymaganiami modelu. Następnie, kiedy mamy już stworzone mapowanie, możemy zastosować je również do zbioru testowego, by model mógł sprawdzić swoje przewidywania na nowych danych.

## 7. Zastąpione etykiety zmiennej - słownikiem stworzonym w kroku 6.

In [7]:
```
for col in col_name:
    X_train[f"{col}_map"] = X_train[col].map(dictionary[col])
    X_test[f"{col}_map"] = X_test[col].map(dictionary[col])

display(X_train.head(20))
display(X_test.head(20))
```

| | cabin | CabinReduced | sex | cabin_map | CabinReduced_map | sex_map |
|---|---|---|---|---|---|---|
| **501** | NaN | N | female | 0 | 1 | 1 |
| **588** | NaN | N | female | 0 | 1 | 1 |
| **402** | NaN | N | female | 0 | 1 | 1 |
| **1193** | NaN | N | male | 0 | 1 | 2 |
| **686** | NaN | N | female | 0 | 1 | 1 |
| **971** | NaN | N | male | 0 | 1 | 2 |
| **117** | E36 | E | female | 1 | 2 | 1 |
| **540** | NaN | N | female | 0 | 1 | 1 |
| **294** | C68 | C | male | 2 | 3 | 2 |
| **261** | E24 | E | male | 3 | 2 | 2 |
| **587** | NaN | N | male | 0 | 1 | 2 |
| **489** | NaN | N | female | 0 | 1 | 1 |
| **2** | C22 C26 | C | female | 4 | 3 | 1 |
| **405** | NaN | N | male | 0 | 1 | 2 |
| **1284** | NaN | N | male | 0 | 1 | 2 |
| **338** | NaN | N | male | 0 | 1 | 2 |
| **356** | NaN | N | male | 0 | 1 | 2 |
| **985** | NaN | N | male | 0 | 1 | 2 |
| **182** | NaN | N | female | 0 | 1 | 1 |
| **1027** | NaN | N | male | 0 | 1 | 2 |

| | cabin | CabinReduced | sex | cabin_map | CabinReduced_map | sex_map |
|---|---|---|---|---|---|---|
| **1139** | NaN | N | male | 0.0 | 1 | 2 |
| **533** | NaN | N | female | 0.0 | 1 | 1 |
| **459** | NaN | N | male | 0.0 | 1 | 2 |
| **1150** | NaN | N | male | 0.0 | 1 | 2 |
| **393** | NaN | N | male | 0.0 | 1 | 2 |
| **1189** | G6 | G | female | 25.0 | 9 | 1 |
| **5** | E12 | E | male | NaN | 2 | 2 |
| **231** | C104 | C | male | NaN | 3 | 2 |
| **330** | NaN | N | male | 0.0 | 1 | 2 |
| **887** | NaN | N | male | 0.0 | 1 | 2 |
| **531** | NaN | N | male | 0.0 | 1 | 2 |
| **790** | NaN | N | male | 0.0 | 1 | 2 |
| **427** | NaN | N | male | 0.0 | 1 | 2 |
| **1260** | NaN | N | female | 0.0 | 1 | 1 |
| **251** | B57 B59 B63 B66 | B | female | 21.0 | 5 | 1 |
| **486** | NaN | N | male | 0.0 | 1 | 2 |
| **436** | NaN | N | female | 0.0 | 1 | 1 |
| **31** | A31 | A | male | NaN | 6 | 2 |
| **186** | NaN | N | female | 0.0 | 1 | 1 |
| **1051** | NaN | N | male | 0.0 | 1 | 2 |

## 8. Liczba brakujących wartości w zmodyfikowanych zbiorach.

In [8]:
```
for col in col_name:
  print(f"Liczba brakujących wartości w kolumnie {col}_map: \nX_train: {X_train[
```

```
Liczba brakujących wartości w kolumnie cabin_map:
X_train: 0,
X_test: 42
Liczba brakujących wartości w kolumnie CabinReduced_map:
X_train: 0,
X_test: 0
Liczba brakujących wartości w kolumnie sex_map:
X_train: 0,
X_test: 0
```

Braki w cabin_map w zbiorze testowym pojawiły się, ponieważ wcześniej ta kolumna zawierała NaN, które nie zostały zamienione podczas kodowania. Dlatego musiałam je później uzupełnić – zastępując NaN wartością 0.

CabinReduced_map i sex_map: braków brak (wszystkie wartości przypisane). Wnioskuję, że dane są dobrze przygotowane do dalszej analizy – brakujące dane są zakodowane jako 0, więc nie spowodują błędów w modelu.

## 9. Brakujące wartości zastępuję liczbą 0. Czy jest to najlepsze wyjście?

In [9]:
```
X_test['cabin_map'] = X_test['cabin_map'].fillna(0)
display(X_test.head(20))
```

| | cabin | CabinReduced | sex | cabin_map | CabinReduced_map | sex_map |
|---|---|---|---|---|---|---|
| **1139** | NaN | N | male | 0.0 | 1 | 2 |
| **533** | NaN | N | female | 0.0 | 1 | 1 |
| **459** | NaN | N | male | 0.0 | 1 | 2 |
| **1150** | NaN | N | male | 0.0 | 1 | 2 |
| **393** | NaN | N | male | 0.0 | 1 | 2 |
| **1189** | G6 | G | female | 25.0 | 9 | 1 |
| **5** | E12 | E | male | 0.0 | 2 | 2 |
| **231** | C104 | C | male | 0.0 | 3 | 2 |
| **330** | NaN | N | male | 0.0 | 1 | 2 |
| **887** | NaN | N | male | 0.0 | 1 | 2 |
| **531** | NaN | N | male | 0.0 | 1 | 2 |
| **790** | NaN | N | male | 0.0 | 1 | 2 |
| **427** | NaN | N | male | 0.0 | 1 | 2 |
| **1260** | NaN | N | female | 0.0 | 1 | 1 |
| **251** | B57 B59 B63 B66 | B | female | 21.0 | 5 | 1 |
| **486** | NaN | N | male | 0.0 | 1 | 2 |
| **436** | NaN | N | female | 0.0 | 1 | 1 |
| **31** | A31 | A | male | 0.0 | 6 | 2 |
| **186** | NaN | N | female | 0.0 | 1 | 1 |
| **1051** | NaN | N | male | 0.0 | 1 | 2 |

Uzupełniłam brakujące wartości w kolumnie cabin_map w zbiorze testowym zerem, tak jak zrobiłam to wcześniej w zbiorze treningowym. Dzięki temu dane są kompletne i model nie będzie miał problemu z brakami.

W przypadku zmiennych jakościowych (takich jak cabin, CabinReduced, sex) można bezpiecznie przypisać brakującym wartościom (NaN) kod 0, o ile 0 nie reprezentuje żadnej rzeczywistej kategorii. W takiej sytuacji 0 działa jedynie jako techniczny kod oznaczający brak informacji.

Natomiast jeśli mamy już zakodowane kolumny, w których 0 jest przypisany do konkretnej etykiety, to przypisywanie 0 także dla braków danych (NaN) może prowadzić do nieprawidłowej interpretacji.

## 10. Porównanie ile unikalnych wartości jest w zbiorze treningowym, a ile w zbiorze testowym.

In [10]:
```
for col in col_name:
    print(f"\nZmienna: {col}")
    print("TRAIN - liczba unikalnych etykiet (oryginalne):", len(X_train[col].un
    print("TEST  - liczba unikalnych etykiet (oryginalne):", len(X_test[col].uni
    print("TRAIN - liczba unikalnych etykiet (mapowane):", len(X_train[f"{col}_m
    print("TEST  - liczba unikalnych etykiet (mapowane):", len(X_test[f"{col}_ma
```

```
Zmienna: cabin
TRAIN - liczba unikalnych etykiet (oryginalne): 151
TEST  - liczba unikalnych etykiet (oryginalne): 71
TRAIN - liczba unikalnych etykiet (mapowane): 151
TEST  - liczba unikalnych etykiet (mapowane): 35

Zmienna: CabinReduced
TRAIN - liczba unikalnych etykiet (oryginalne): 9
TEST  - liczba unikalnych etykiet (oryginalne): 8
TRAIN - liczba unikalnych etykiet (mapowane): 9
TEST  - liczba unikalnych etykiet (mapowane): 8

Zmienna: sex
TRAIN - liczba unikalnych etykiet (oryginalne): 2
TEST  - liczba unikalnych etykiet (oryginalne): 2
TRAIN - liczba unikalnych etykiet (mapowane): 2
TEST  - liczba unikalnych etykiet (mapowane): 2
```

`cabin` : bardzo dużo unikalnych wartości (151 w treningowym, 71 w testowym), ale po mapowaniu w teście zostało już tylko 35 – oznacza to, że wiele kategorii nie występuje w zbiorze testowym.

`CabinReduced` : liczba etykiet znacznie mniejsza, co upraszcza dane i może poprawić jakość modelu, pomaga w uniknięciu braków po kodowaniu.

`sex` : stabilna liczba kategorii (2), bez zmian po mapowaniu.

### WNIOSKI

Kodowanie zmiennej cabin wykonałam osobno na zbiorze treningowym i testowym. Przez to w X_test pojawiły się brakujące wartości (NaN), ponieważ nie wszystkie etykiety z testu były znane modelowi uczonemu na trainie. Takie sytuacje mogą prowadzić do problemów przy predykcji, dlatego musiałam uzupełnić je zerem. Gdybym zrobiła kodowanie na całym zbiorze przed podziałem, uniknęłabym tych braków.