titanic-data-split-rj

April 16, 2025

1 Titanic data split

1.1 Roksana Jandura IiAD nr. 416314

1.1.1 Wczytaj końcowy i przetworzony zbiór danych Titanic z poprzednich zajęć.

[1]:
```
import pandas as pd
df = pd.read_csv("titanic_cardinality.csv")
print(df.head(10))
```

```
   'pclass'  'survived'                                             'name'  \
0         1           1                    Allen, Miss. Elisabeth Walton
1         1           1                   Allison, Master. Hudson Trevor
2         1           0                     Allison, Miss. Helen Loraine
3         1           0             Allison, Mr. Hudson Joshua Creighton
4         1           0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
5         1           1                                  Anderson, Mr. Harry
6         1           1                Andrews, Miss. Kornelia Theodosia
7         1           0                           Andrews, Mr. Thomas Jr
8         1           1    Appleton, Mrs. Edward Dale (Charlotte Lamson)
9         1           0                           Artagaveytia, Mr. Ramon

    'sex'    'age'  'sibsp'  'parch'  'ticket'    'fare'  'cabin' 'embarked'  \
0  female  29.0000        0        0     24160  211.3375       B5          S
1    male   0.9167        1        2    113781  151.5500  C22 C26          S
2  female   2.0000        1        2    113781  151.5500  C22 C26          S
3    male  30.0000        1        2    113781  151.5500  C22 C26          S
4  female  25.0000        1        2    113781  151.5500  C22 C26          S
5    male  48.0000        0        0     19952   26.5500      E12          S
6  female  63.0000        1        0     13502   77.9583       D7          S
7    male  39.0000        0        0    112050    0.0000      A36          S
8  female  53.0000        2        0     11769   51.4792     C101          S
9    male  71.0000        0        0  PC 17609   49.5042      NaN          C

  'boat'  'body'                        'home.dest' CabinReduced
0      2     NaN                       St Louis, MO            B
1     11     NaN  Montreal, PQ / Chesterville, ON            C
2    NaN     NaN  Montreal, PQ / Chesterville, ON            C
3    NaN   135.0  Montreal, PQ / Chesterville, ON            C
```

```
4    NaN     NaN  Montreal, PQ / Chesterville, ON           C
5      3     NaN                     New York, NY           E
6     10     NaN                       Hudson, NY           D
7    NaN     NaN                      Belfast, NI           A
8      D     NaN              Bayside, Queens, NY           C
9    NaN    22.0              Montevideo, Uruguay           n
```

1.1.2 Zapoznaj się z funkcja train_test_split wchodzącą w skład biblioteki Scikit-learn. Zapisz swoje obserwacje.

Funkcja train_test_split() dzieli zbiór danych (np. cechy i etykiety) na część treningową i część testową. Parametry:

- test_size - ile danych przeznaczyć na test
- train_size - można podać zamiast test_size
- random_state - ustawienie losowości (dla powtarzalnych wyników)
- shuffle=True - czy mieszać dane przed podziałem

1.1.3 Stwórz zmienną do której zapiszesz listę z nazwami trzech kolumn – kabiny, zredukowane kabiny oraz płeć. Nazwij ją col_name.

[2]:
```
col_name = ["'cabin'", 'CabinReduced', "'sex'"]
```

1.1.4 Podziel zbiór na treningowy i testowy używając train_test_split.

Jako zmienną niezależną ustaw dane składające się z kolumn o nazwach zapisanych w col_name. Jako zmienną zależną ustaw kolumnę mówiącą o tym czy ktoś przeżył czy nie. Ustaw rozmiar zbioru testowego na 20 lub 30% całości. Parametr random_state ustaw jako 0.

Wyświetl wymiary zbiorów X_train, X_test, y_train, y_test używając .shape() i skomentuj wyniki.

[3]:
```
from sklearn.model_selection import train_test_split
X = df[col_name]
y = df["'survived'"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0 )
```

[4]:
```
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,)
```

Zbiór danych został podzielony na 70% treningowych i 30% testowych.

X_train shape: (916, 3) - oznacza, że zbiór treningowy zawiera 916 przykładów i 3 cechy (kolumny). Na tej części danych model będzie się uczył.

X_test shape: (393, 3) - zbiór testowy zawiera 393 przykładów i te same 3 cechy – na tej części sprawdzimy, jak dobrze model działa na nowych, niewidzianych danych.

y_train shape: (916,) - etykiety (czy ktoś przeżył – 0 lub 1) dla każdego z 916 przypadków w X_train

y_test shape: (393,) - etykiety odpowiadające danym testowym (X_test)

1.1.5 Zbadaj w zależności od kardynalności danej zmiennej czy rozkład etykiet dla poszczególnych cech w zbiorach testowych i treningowych jest równomierny.

[5]:
```
for col in col_name:
    unique_test = [x for x in X_test[col].unique() if x not in X_train[col].
 ↪unique()]
    print(f"{col}: {len(unique_test)} unikalnych wartości tylko w zbiorze␣
 ↪testowym")
```

```
'cabin': 37 unikalnych wartości tylko w zbiorze testowym
CabinReduced: 0 unikalnych wartości tylko w zbiorze testowym
'sex': 0 unikalnych wartości tylko w zbiorze testowym
```

Zmienna ‘cabin’ zawiera 37 unikalnych wartości tylko w zbiorze testowym. Oznacza, że model zobaczy w teście 37 nazw kabin, których nigdy wcześniej nie widział w zbiorze treningowym. Może to negatywnie wpłynąć na skuteczność predykcji, szczególnie jeśli kabina niesie ważną informację.

Zmienna ‘CabinReduced’ oraz ‘sex’ – w zbiorze testowym nie pojawiły się żadne nowe wartości względem zbioru treningowego. Oznacza to, że model zna wszystkie możliwe kategorie, a rozkład danych jest stabilny i spójny. Dzięki temu obie zmienne nadają się do modelowania i powinny pozytywnie wpłynąć na jakość predykcji.

1.1.6 Wykonaj kodowanie zmiennych kategorycznych do zmiennych liczbowych. Wykorzystaj pętlę for i metodę enumerate().

[6]:
```
mappings = {}

for col in col_name:
    unique_values = X_train[col].dropna().unique() #pomijam NaN

    #dla zmiennej CabinReduced wartości 'NaN' zostały skrócone do 'n'
    #wiec usuwam 'n'
    if col == 'CabinReduced':
        unique_values = [val for val in unique_values if val != 'n']

    mapping = {}
```

```
    for idx, val in enumerate(unique_values):
        mapping[val] = idx + 1

    mappings[col] = mapping

    print(f"\nSłownik dla zmiennej '{col}':")
    for k, v in list(mapping.items())[:10]:
        print(f"'{k}' : {v}")
```

```
Słownik dla zmiennej ''cabin'':
'E36' : 1
'C68' : 2
'E24' : 3
'C22 C26' : 4
'D38' : 5
'B50' : 6
'A24' : 7
'C111' : 8
'F' : 9
'C6' : 10

Słownik dla zmiennej 'CabinReduced':
'E' : 1
'C' : 2
'D' : 3
'B' : 4
'A' : 5
'F' : 6
'T' : 7
'G' : 8

Słownik dla zmiennej ''sex'':
'female' : 1
'male' : 2
```

1.1.7 Zastąp etykiety zmiennej (tu przykład dla kabina) słownikiem stworzonym w kroku 6. Do tego będzie potrzebne mapowanie.

[7]:
```
for col in col_name:
    map_col = f"{col}_map"
    mapping = mappings[col]

    X_train[map_col] = X_train[col].map(mapping)
    X_test[map_col] = X_test[col].map(mapping)

    print(f"\nZmienna: {col} (zbiór treningowy):")
```

```
    print(X_train[[map_col, col]].head(10))

    print(f"\nZmienna: {col} (zbiór testowy):")
    print(X_test[[map_col, col]].head(10))
```

```
Zmienna: 'cabin' (zbiór treningowy):
      'cabin'_map 'cabin'
501           NaN     NaN
588           NaN     NaN
402           NaN     NaN
1193          NaN     NaN
686           NaN     NaN
971           NaN     NaN
117           1.0     E36
540           NaN     NaN
294           2.0     C68
261           3.0     E24

Zmienna: 'cabin' (zbiór testowy):
      'cabin'_map 'cabin'
1139          NaN     NaN
533           NaN     NaN
459           NaN     NaN
1150          NaN     NaN
393           NaN     NaN
1189         25.0      G6
5             NaN     E12
231           NaN    C104
330           NaN     NaN
887           NaN     NaN

Zmienna: CabinReduced (zbiór treningowy):
      CabinReduced_map CabinReduced
501                NaN            n
588                NaN            n
402                NaN            n
1193               NaN            n
686                NaN            n
971                NaN            n
117                1.0            E
540                NaN            n
294                2.0            C
261                1.0            E

Zmienna: CabinReduced (zbiór testowy):
      CabinReduced_map CabinReduced
```

```
1139              NaN           n
533               NaN           n
459               NaN           n
1150              NaN           n
393               NaN           n
1189              8.0           G
5                 1.0           E
231               2.0           C
330               NaN           n
887               NaN           n

Zmienna: 'sex' (zbiór treningowy):
      'sex'_map   'sex'
501           1  female
588           1  female
402           1  female
1193          2    male
686           1  female
971           2    male
117           1  female
540           1  female
294           2    male
261           2    male

Zmienna: 'sex' (zbiór testowy):
      'sex'_map   'sex'
1139          2    male
533           1  female
459           2    male
1150          2    male
393           2    male
1189          1  female
5             2    male
231           2    male
330           2    male
887           2    male
```

1.1.8 Sprawdź liczbę brakujących wartości w zmodyfikowanych zbiorach. Zapisz wyniki i skomentuj.

[8]:
```
for col in col_name:
    map_col = f"{col}_map"

    missing_train = X_train[map_col].isna().sum()
    missing_test = X_test[map_col].isna().sum()
```

```
    print(f"{map_col} → braków w train: {missing_train}, braków w test:␣
↪{missing_test}")
```

```
'cabin'_map → braków w train: 702, braków w test: 354
CabinReduced_map → braków w train: 702, braków w test: 312
'sex'_map → braków w train: 0, braków w test: 0
```

W zmiennej 'cabin'_map występuje bardzo dużo braków: 702 w treningu i 354 w teście. Oznacza to, że wiele pasażerów nie miało przypisanej kabiny, lub że niektóre kabiny w teście nie były znane z treningu.

W zmiennej CabinReduced_map mamy również dużo braków, choć trochę mniej niż przy pełnej nazwie kabiny.Oznacza to, że informacja o pokładzie (np. A, B, C…) też była często niedostępna.

'sex'_map mamy brak braków = dane są kompletne i w pełni gotowe do użycia.

1.1.9 Zastąp brakujące wartości liczbą 0. Czy jest to najlepsze wyjście?

[9]:
```
for col in col_name:
    map_col = f"{col}_map"
    X_train[map_col] = X_train[map_col].fillna(0).astype(int)
    X_test[map_col] = X_test[map_col].fillna(0).astype(int)
    print(f"\nZmienna: {col} (zbiór treningowy):")
    print(X_train[[map_col, col]].head())

    print(f"\nZmienna: {col} (zbiór testowy):")
    print(X_test[[map_col, col]].head())

    missing_train = X_train[map_col].isna().sum()
    missing_test = X_test[map_col].isna().sum()

    print(f"{map_col} → braków w train: {missing_train}, braków w test:␣
↪{missing_test}")
```

```
Zmienna: 'cabin' (zbiór treningowy):
      'cabin'_map 'cabin'
501             0     NaN
588             0     NaN
402             0     NaN
1193            0     NaN
686             0     NaN

Zmienna: 'cabin' (zbiór testowy):
      'cabin'_map 'cabin'
1139            0     NaN
533             0     NaN
459             0     NaN
1150            0     NaN
```

```
393             0     NaN
'cabin'_map → braków w train: 0, braków w test: 0

Zmienna: CabinReduced (zbiór treningowy):
      CabinReduced_map CabinReduced
501                  0            n
588                  0            n
402                  0            n
1193                 0            n
686                  0            n

Zmienna: CabinReduced (zbiór testowy):
      CabinReduced_map CabinReduced
1139                 0            n
533                  0            n
459                  0            n
1150                 0            n
393                  0            n
CabinReduced_map → braków w train: 0, braków w test: 0

Zmienna: 'sex' (zbiór treningowy):
      'sex'_map   'sex'
501           1  female
588           1  female
402           1  female
1193          2    male
686           1  female

Zmienna: 'sex' (zbiór testowy):
      'sex'_map   'sex'
1139          2    male
533           1  female
459           2    male
1150          2    male
393           2    male
'sex'_map → braków w train: 0, braków w test: 0
```

W kontekście zmiennych kategorycznych, takich jak cabin czy CabinReduced, 0 może pełnić rolę "braku informacji". Model nie działa z NaN, więc uzupełnienie 0 zapewnia kompletność danych. Pozwala to zachować wszystkie wiersze –np. nie usuwamy wierszy z brakującymi informacjami.

Jednak 0 może być mylącą wartością, bo model może uznać to za konkratną wartość, a nie jej 'brak', może to prowadzić do błędnej interpetacji.

1.1.10 Porównaj ile unikalnych wartości jest w zbiorze treningowym, a ile w zbiorze testowym (funkcja len). Jaka jest różnica pomiędzy liczbą etykiet przed i po redukcji oraz mapowaniu? Czy cały proces, który został do tej pory wykonany może mieć wpływ na końcowy wynik predykcji i jakość modelu?

[10]:
```
for col in col_name:
    map_col = f"{col}_map"

    n_train_before = len(X_train[col].unique())
    n_test_before = len(X_test[col].unique())

    n_train_after = len(X_train[map_col].unique())
    n_test_after = len(X_test[map_col].unique())

    print(f"\nKolumna: {col}")
    print(f"  Trening (oryginalnie): {n_train_before} unikalnych etykiet")
    print(f"  Test (oryginalnie): {n_test_before} unikalnych etykiet")
    print(f"  Trening (po mapowaniu): {n_train_after}")
    print(f"  Test (po mapowaniu): {n_test_after}")
```

```
Kolumna: 'cabin'
  Trening (oryginalnie): 151 unikalnych etykiet
  Test (oryginalnie): 71 unikalnych etykiet
  Trening (po mapowaniu): 151
  Test (po mapowaniu): 35

Kolumna: CabinReduced
  Trening (oryginalnie): 9 unikalnych etykiet
  Test (oryginalnie): 8 unikalnych etykiet
  Trening (po mapowaniu): 9
  Test (po mapowaniu): 8

Kolumna: 'sex'
  Trening (oryginalnie): 2 unikalnych etykiet
  Test (oryginalnie): 2 unikalnych etykiet
  Trening (po mapowaniu): 2
  Test (po mapowaniu): 2
```

- ‘cabin’

Wiele kabin z testu nie pojawiło się w treningu, dlatego po mapowaniu mają wartość 0. Zmienna ma bardzo wysoką kardynalność i wiele nieznanych wartości w teście, co może wprowadzać błędy i szum do modelu. Lepiej jej nie używać w tej formie.

- CabinReduced

Rozkład etykiet jest stabilny i spójny między treningiem a testem. Zmienna została dobrze zakodowana, nie ma nieznanych wartości, czyli nadaje się do użycia w modelu. Taka redukcja upraszcza model i ogranicza ryzyko, że w zbiorze testowym pojawi się nieznana wcześniej wartość.

Pomimo uproszczenia zachowana zostaje część informacji (np. o lokalizacji kabiny na statku).

- ‘sex’

Jest to prosta, binarna zmienna (male, female) bez braków i bez różnic między zbiorami. Jest to bardzo dobra cecha predykcyjna – model ją w pełni rozumie.

Podsumowując:

Dzięki mapowaniu tekstów na liczby model w ogóle może zrozumieć dane. Zastąpienie braków 0 zapobiega błędom działania modelu, ale niesie ryzyko błędnej interpretacji, jeśli 0 oznacza coś innego niż „brak”. Uproszczenie zmiennej cabin do CabinReduced zmniejszyło kardynalność i pomogło modelowi lepiej generalizować. Pozostawienie oryginalnej cabin mogłoby prowadzić do overfittingu i złej jakości predykcji, zwłaszcza gdy w teście pojawiały się nowe kabiny.

Cały proces preprocessingowy ma kluczowe znaczenie – poprawił jakość danych, wyeliminował problemy z brakami, zmniejszył ryzyko błędów predykcji i przygotował dane do uczenia.

[ ]: