Titanic_Data_split

April 19, 2025

1 Inżynieria cech - data split

Inżynieria cech odgrywa istotną rolę w kontekście równomiernego podziału zbioru danych na zbiór treningowy i testowy. Jest to proces, który zapewnia spójność, poprawność i użyteczność danych zarówno w zbiorze treningowym, jak i testowym. Oto dlaczego:

Zachowanie równowagi cech: Podczas podziału zbioru danych na zbiór treningowy i testowy, inżynier danych/data scientist dba o to, aby oba zbiory zachowały równowagę cech. Oznacza to, że rozkład różnych cech w obu zbiorach jest podobny, co pomaga w uniknięciu obciążenia modelu podczas oceny jego skuteczności.

Utrzymywanie spójności cech i etykiet: Inżynieria cech pomaga zachować spójność między cechami a ich etykietami w obu zbiorach danych. Dzięki temu możliwe jest porównywanie wyników modelu na różnych zestawach danych i dokonywanie odpowiednich modyfikacji w procesie uczenia maszynowego.

[2]:
```
import pandas as pd
import numpy as np
import arff
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

1.1 Zad 1

Wczytaj końcowy i przetworzony zbiór danych Titanic z poprzednich zajęć.

[3]:
```
df = pd.read_csv('titanic.csv')
display(df.head())
```

```
   pclass  survived                                             name     sex  \
0     1.0         1                    Allen, Miss. Elisabeth Walton  female
1     1.0         1                   Allison, Master. Hudson Trevor    male
2     1.0         0                     Allison, Miss. Helen Loraine  female
3     1.0         0             Allison, Mr. Hudson Joshua Creighton    male
4     1.0         0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  female

       age  sibsp  parch  ticket      fare    cabin embarked boat   body  \
0  29.0000    0.0    0.0   24160  211.3375       B5        S    2    NaN
1   0.9167    1.0    2.0  113781  151.5500  C22 C26        S   11    NaN
2   2.0000    1.0    2.0  113781  151.5500  C22 C26        S  NaN    NaN
```

```
3  30.0000    1.0    2.0  113781  151.5500  C22 C26        S  NaN  135.0
4  25.0000    1.0    2.0  113781  151.5500  C22 C26        S  NaN    NaN

                        home.dest CabinReduced
0                     St Louis, MO            B
1  Montreal, PQ / Chesterville, ON            C
2  Montreal, PQ / Chesterville, ON            C
3  Montreal, PQ / Chesterville, ON            C
4  Montreal, PQ / Chesterville, ON            C
```

1.2 Zad 2

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

Funkcja train_test_split służy do podziału zbioru danych na zestawy treningowy i testowy. Funkcja ta umożliwia losowe rozdzielenie danych, co pozwala na obiektywne trenowanie modeli i ich późniejszą walidację.

Składnia:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X: Zmienna niezależna (features) – objaśniające y: Zmienna zależna (target) – objaśniana

test_size: Procent danych, które mają zostać przypisane do zestawu testowego (np. 0.2 oznacza 20% danych w zbiorze testowym, a reszta w treningowym)

random_state: ziarno losowości

1.3 Zad 3

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

[4]:
```
col_name = ["cabin", "CabinReduced", "sex"]
```

1.4 Zad 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.

[5]:
```
X = df[col_name]
y = df["survived"]

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2,␣
 ↪random_state = 0)
```

```
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: (1047, 3)
X_test shape: (262, 3)
y_train shape: (1047,)
y_test shape: (262,)
```

1.5 Komenatrz do wyników

Widzimy, że został dokonany podział na zbiór testowy i treningowy. Zbiór treningowy zajmuje 80% całości zbioru, natomiast zbiór testowy stanowi 20% całości. Zbiór został poprawnie podzielony ponieważ cześć testowa liczy 262, a treningowa 1047, co stanowi odpowiednio 20% i 80% z całości czyli z 1309.

Wymiary wyświetlają liczbę przykładów i liczbę cech (kolumn) w danych

1.6 Zad 5

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

[6]:
```
unique_test = [x for x in X_test['cabin'].unique() if x not in X_train['cabin'].
 ↪unique()]
print(f"Unikalne wartości w zbiorze testowym, ale nie w treningowym:␣
 ↪{len(unique_test)}")



for col in ['cabin', 'CabinReduced', 'sex']:
    unique_test = [x for x in X_test[col].unique() if x not in X_train[col].
 ↪unique()]
    print(f"{col} - unikalne w teście, nieobecne w treningu:␣
 ↪{len(unique_test)}")
    print(unique_test)
```

```
Unikalne wartości w zbiorze testowym, ale nie w treningowym: 24
cabin - unikalne w teście, nieobecne w treningu: 24
[nan, 'E12', 'C104', 'A31', 'D11', 'D48', 'D10 D12', 'B38', 'D45', 'C50', 'C31',
'B82 B84', 'A32', 'C53', 'B10', 'C70', 'A23', 'C106', 'E58', 'B11', 'F E69',
'B80', 'E39 E41', 'D22']
CabinReduced - unikalne w teście, nieobecne w treningu: 0
[]
sex - unikalne w teście, nieobecne w treningu: 0
[]
```

1.7 Zad 6

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

[8]:
```
encoding_dicts = {}


for i in ['cabin', 'CabinReduced', 'sex']:
    encoding_dict = {label: idx for idx, label in enumerate(df[i].dropna().
 ↪unique())} # słownik, gdzie klucze to unikalne wartości, a wartości to liczby
    encoding_dicts[i] = encoding_dict
    print(f"kolumna {i}: {encoding_dict}")
```

```
kolumna cabin: {'B5': 0, 'C22 C26': 1, 'E12': 2, 'D7': 3, 'A36': 4, 'C101': 5,
'C62 C64': 6, 'B35': 7, 'A23': 8, 'B58 B60': 9, 'D15': 10, 'C6': 11, 'D35': 12,
'C148': 13, 'C97': 14, 'B49': 15, 'C99': 16, 'C52': 17, 'T': 18, 'A31': 19,
'C7': 20, 'C103': 21, 'D22': 22, 'E33': 23, 'A21': 24, 'B10': 25, 'B4': 26,
'E40': 27, 'B38': 28, 'E24': 29, 'B51 B53 B55': 30, 'B96 B98': 31, 'C46': 32,
'E31': 33, 'E8': 34, 'B61': 35, 'B77': 36, 'A9': 37, 'C89': 38, 'A14': 39,
'E58': 40, 'E49': 41, 'E52': 42, 'E45': 43, 'B22': 44, 'B26': 45, 'C85': 46,
'E17': 47, 'B71': 48, 'B20': 49, 'A34': 50, 'C86': 51, 'A16': 52, 'A20': 53,
'A18': 54, 'C54': 55, 'C45': 56, 'D20': 57, 'A29': 58, 'C95': 59, 'E25': 60,
'C111': 61, 'C23 C25 C27': 62, 'E36': 63, 'D34': 64, 'D40': 65, 'B39': 66,
'B41': 67, 'B102': 68, 'C123': 69, 'E63': 70, 'C130': 71, 'B86': 72, 'C92': 73,
'A5': 74, 'C51': 75, 'B42': 76, 'C91': 77, 'C125': 78, 'D10 D12': 79, 'B82 B84':
80, 'E50': 81, 'D33': 82, 'C83': 83, 'B94': 84, 'D49': 85, 'D45': 86, 'B69': 87,
'B11': 88, 'E46': 89, 'C39': 90, 'B18': 91, 'D11': 92, 'C93': 93, 'B28': 94,
'C49': 95, 'B52 B54 B56': 96, 'E60': 97, 'C132': 98, 'B37': 99, 'D21': 100,
'D19': 101, 'C124': 102, 'D17': 103, 'B101': 104, 'D28': 105, 'D6': 106, 'D9':
107, 'B80': 108, 'C106': 109, 'B79': 110, 'C47': 111, 'D30': 112, 'C90': 113,
'E38': 114, 'C78': 115, 'C30': 116, 'C118': 117, 'D36': 118, 'D48': 119, 'D47':
120, 'C105': 121, 'B36': 122, 'B30': 123, 'D43': 124, 'B24': 125, 'C2': 126,
'C65': 127, 'B73': 128, 'C104': 129, 'C110': 130, 'C50': 131, 'B3': 132, 'A24':
133, 'A32': 134, 'A11': 135, 'A10': 136, 'B57 B59 B63 B66': 137, 'C28': 138,
'E44': 139, 'A26': 140, 'A6': 141, 'A7': 142, 'C31': 143, 'A19': 144, 'B45':
145, 'E34': 146, 'B78': 147, 'B50': 148, 'C87': 149, 'C116': 150, 'C55 C57':
151, 'D50': 152, 'E68': 153, 'E67': 154, 'C126': 155, 'C68': 156, 'C70': 157,
'C53': 158, 'B19': 159, 'D46': 160, 'D37': 161, 'D26': 162, 'C32': 163, 'C80':
164, 'C82': 165, 'C128': 166, 'E39 E41': 167, 'D': 168, 'F4': 169, 'D56': 170,
'F33': 171, 'E101': 172, 'E77': 173, 'F2': 174, 'D38': 175, 'F': 176, 'F G63':
177, 'F E57': 178, 'F E46': 179, 'F G73': 180, 'E121': 181, 'F E69': 182, 'E10':
183, 'G6': 184, 'F38': 185}
kolumna CabinReduced: {'B': 0, 'C': 1, 'E': 2, 'D': 3, 'A': 4, 'N': 5, 'T': 6,
'F': 7, 'G': 8}
kolumna sex: {'female': 0, 'male': 1}
```

1.8 Zad 7

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

[9]:
```
for i in ['cabin', 'CabinReduced', 'sex']:
    encoding_dict = {label: idx for idx, label in enumerate(df[i].dropna().
 ↪unique())}
    # Mapowanie na nowe kolumny w DataFrame
    df[f'{i}_Map'] = df[i].map(encoding_dict)

display(df[['cabin', 'cabin_Map', 'CabinReduced', 'CabinReduced_Map', 'sex',␣
 ↪'sex_Map']])
```

	cabin	cabin_Map	CabinReduced	CabinReduced_Map	sex	sex_Map
0	B5	0.0	B	0	female	0
1	C22 C26	1.0	C	1	male	1
2	C22 C26	1.0	C	1	female	0
3	C22 C26	1.0	C	1	male	1
4	C22 C26	1.0	C	1	female	0
…	…	…	…	…	…	…
1304	NaN	NaN	N	5	female	0
1305	NaN	NaN	N	5	female	0
1306	NaN	NaN	N	5	male	1
1307	NaN	NaN	N	5	male	1
1308	NaN	NaN	N	5	male	1

[1309 rows x 6 columns]

1.9 Zad 8

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

[ ]:
```
missing_values = df.isnull().sum()
print(missing_values)
```

```
pclass                0
survived              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
CabinReduced           0
cabin_Map           1014
CabinReduced_Map       0
sex_Map                0
dtype: int64
```

1.10 Zad 9

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

[ ]:
```
df = df.fillna(0)
display(df.head())

# Zastępując brakujące dane zerami mamy takie problemy jak: różny typ danych␣
 ↪oraz złą interpretacje wartości
```

```
   pclass  survived                                             name     sex  \
0     1.0         1                    Allen, Miss. Elisabeth Walton  female
1     1.0         1                   Allison, Master. Hudson Trevor    male
2     1.0         0                     Allison, Miss. Helen Loraine  female
3     1.0         0             Allison, Mr. Hudson Joshua Creighton    male
4     1.0         0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  female

       age  sibsp  parch  ticket      fare    cabin embarked boat   body  \
0  29.0000    0.0    0.0   24160  211.3375       B5        S    2    0.0
1   0.9167    1.0    2.0  113781  151.5500  C22 C26        S   11    0.0
2   2.0000    1.0    2.0  113781  151.5500  C22 C26        S    0    0.0
3  30.0000    1.0    2.0  113781  151.5500  C22 C26        S    0  135.0
4  25.0000    1.0    2.0  113781  151.5500  C22 C26        S    0    0.0

                        home.dest CabinReduced  cabin_Map  CabinReduced_Map  \
0                      St Louis, MO            B        0.0                 0
1  Montreal, PQ / Chesterville, ON            C        1.0                 1
2  Montreal, PQ / Chesterville, ON            C        1.0                 1
3  Montreal, PQ / Chesterville, ON            C        1.0                 1
4  Montreal, PQ / Chesterville, ON            C        1.0                 1

   sex_Map
0        0
1        1
2        0
3        1
4        0
```

1.11 Zad 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?

[ ]:
```
print("Unikalne wartości w zbiorze treningowym:")
print(X_train.nunique())
print("\nUnikalne wartości w zbiorze testowym:")
print(X_test.nunique())

print("\nSuma unikalnych wartości w zbiorze treningowym przed redukcją:")
print(X_train.nunique().sum())
print("\nSuma unikalnych wartości w zbiorze testowym przed redukcją:")
print(X_test.nunique().sum())
print("\nSuma unikalnych wartości w zbiorze treningowym po redukcji:")
print(X_train.nunique().sum() - X_train["CabinReduced"].nunique())
print("\nSuma unikalnych wartości w zbiorze testowym po redukcji:")
print(X_test.nunique().sum() - X_test["CabinReduced"].nunique())
```

```
Unikalne wartości w zbiorze treningowym:
cabin           163
CabinReduced      9
sex               2
dtype: int64

Unikalne wartości w zbiorze testowym:
cabin           48
CabinReduced     8
sex              2
dtype: int64

Suma unikalnych wartości w zbiorze treningowym przed redukcją:
174

Suma unikalnych wartości w zbiorze testowym przed redukcją:
58

Suma unikalnych wartości w zbiorze treningowym po redukcji:
165

Suma unikalnych wartości w zbiorze testowym po redukcji:
50
```

Redukcja liczby kategorii znacznie upraszcza dane i zmniejsza ryzyko overfittingu. Zamiast trenować model na 200+ różnych kabinach trenujemy go na kilku bardziej ogólnych grupach.

Redukcja i mapowanie zmiennych kategorycznych znacząco wpłynęły na uproszczenie zbioru danych. Liczba unikalnych etykiet dla zmiennej cabin spadła z ponad 200 do kilkunastu po redukcji i kodowaniu. Dzięki temu model jest mniej podatny na przeuczenie i lepiej generalizuje. Mapowanie etykiet na wartości liczbowe umożliwiło ich wykorzystanie w algorytmach ML. Należy jednak uważać na możliwą utratę informacji i brakujące etykiety w danych testowych.

Natomiast zmiana wartości w kolumnach może wpłynąć na jakość modelu.

- ryzyko utraty istotnych informacji
- ryzyko zmiany rozkładu danych (zmiana wyników modelu) Ważne jest, aby dokładnie analizować wpływ każdej zmiany na dane i model.