titanic_split

April 19, 2025

[11]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
```

[12]:
```
df = pd.read_csv('titanic_cardinal.csv')
df.head()
```

[12]:

	Unnamed: 0	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	CabinReduced
0	0	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	C
1	1	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.55	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	C
2	2	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.55	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON	C
3	3	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.55	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	C
4	4	1	1	Anderson, Mr. Harry	male	48.0000	0	0	19952	26.55	E12	S	3	NaN	New York, NY	E

Funkcja `train_test_split` służy do losowego podziału danych na zbiór treningowy i

testowy, co pozwala na ocenę działania modelu na nowych danych.

0.1 3.

[13]:
```
col_name = ['cabin', 'CabinReduced', 'sex']
```

0.2 4. Dzielenie zbioru na treningowy i testowy

[14]:
```
X_train, X_test, Y_train, Y_test = train_test_split(df[col_name],␣
 ↪df['survived'], test_size=0.2, random_state=42)

print(f'X_train.shape: {X_train.shape}')
print(f'X_test.shape: {X_test.shape}')
print(f'Y_train.shape: {Y_train.shape}')
print(f'Y_test.shape: {Y_test.shape}')
X_train.head()
```

```
X_train.shape: (1046, 3)
X_test.shape: (262, 3)
Y_train.shape: (1046,)
Y_test.shape: (262,)
```

[14]:

	cabin	CabinReduced	sex
770	NaN	n	female
543	NaN	n	male
289	E67	E	male
10	C62 C64	C	female
147	C83	C	male

Zbiory treningowe mają 1046 wierszy czyli jest to ~80% wszystkich wierszy. Zbiory testowe mają 262 wiersze ~20% wszystkich wierszy.

Zbiory X mają trzy kolumny - trzy zmienne objaśniające. Zbiory Y mają jedną kolumnę - jedna zmienna objaśniana.

0.3 5.

[15]:
```
for col in col_name:
    unique_test = [x for x in X_test[col].unique() if x not in X_train[col].
 ↪unique()]
    unique_train = [x for x in X_train[col].unique() if x not in X_test[col].
 ↪unique()]

    print(f"\nZmienna: {col}")
    print(f"Kardynalność: {len(df[col].unique())}")
    print(f"Unikalne w testowym: {len(unique_test)}")
    print(f"Unikalne w treningowym: {len(unique_train)}")
```

```
Zmienna: cabin
Kardynalność: 187
Unikalne w testowym: 27
Unikalne w treningowym: 137

Zmienna: CabinReduced
Kardynalność: 9
Unikalne w testowym: 1
Unikalne w treningowym: 1

Zmienna: sex
Kardynalność: 2
Unikalne w testowym: 0
Unikalne w treningowym: 0
```

- `cabin` ma rozkład nierównomierny, prawdopodobnie przez to że ma bardzo dużo etykiet
- `CabinReduced` i `sex` mają rozkład równomierny

0.4 6. i 7.

[16]:
```python
for feature in ['cabin', 'CabinReduced', 'sex']:
    unique_values = df[feature].unique()
    dicc = {word: i + 1 for i, word in enumerate(unique_values)}
    for key, value in dicc.items():
        print(f"'{key}': {value},")
    X_train[feature] = X_train[feature].map(dicc).fillna(0)
    X_test[feature] = X_test[feature].map(dicc).fillna(0)

print('\nTreningowy:')
display(X_train.iloc[:5])
print('Testowy:')
display(X_train.iloc[-5:])
```

```
'C22 C26': 1,
'E12': 2,
'D7': 3,
'A36': 4,
'C101': 5,
'nan': 6,
'C62 C64': 7,
'B35': 8,
'A23': 9,
'B58 B60': 10,
'D15': 11,
'C6': 12,
'D35': 13,
'C148': 14,
'C97': 15,
```

```
'B49': 16,
'C99': 17,
'C52': 18,
'T': 19,
'A31': 20,
'C7': 21,
'C103': 22,
'D22': 23,
'E33': 24,
'A21': 25,
'B10': 26,
'B4': 27,
'E40': 28,
'B38': 29,
'E24': 30,
'B51 B53 B55': 31,
'B96 B98': 32,
'C46': 33,
'E31': 34,
'E8': 35,
'B61': 36,
'B77': 37,
'A9': 38,
'C89': 39,
'A14': 40,
'E58': 41,
'E49': 42,
'E52': 43,
'E45': 44,
'B22': 45,
'B26': 46,
'C85': 47,
'E17': 48,
'B71': 49,
'B20': 50,
'A34': 51,
'C86': 52,
'A16': 53,
'A20': 54,
'A18': 55,
'C54': 56,
'C45': 57,
'D20': 58,
'A29': 59,
'C95': 60,
'E25': 61,
'C111': 62,
'C23 C25 C27': 63,
```

```
'E36': 64,
'D34': 65,
'D40': 66,
'B39': 67,
'B41': 68,
'B102': 69,
'C123': 70,
'E63': 71,
'C130': 72,
'B86': 73,
'C92': 74,
'A5': 75,
'C51': 76,
'B42': 77,
'C91': 78,
'C125': 79,
'D10 D12': 80,
'B82 B84': 81,
'E50': 82,
'D33': 83,
'C83': 84,
'B94': 85,
'D49': 86,
'D45': 87,
'B69': 88,
'B11': 89,
'E46': 90,
'C39': 91,
'B18': 92,
'D11': 93,
'C93': 94,
'B28': 95,
'C49': 96,
'B52 B54 B56': 97,
'E60': 98,
'C132': 99,
'B37': 100,
'D21': 101,
'D19': 102,
'C124': 103,
'D17': 104,
'B101': 105,
'D28': 106,
'D6': 107,
'D9': 108,
'B80': 109,
'B5': 110,
'C106': 111,
```

```
'B79': 112,
'C47': 113,
'D30': 114,
'C90': 115,
'E38': 116,
'C78': 117,
'C30': 118,
'C118': 119,
'D36': 120,
'D48': 121,
'D47': 122,
'C105': 123,
'B36': 124,
'B30': 125,
'D43': 126,
'B24': 127,
'C2': 128,
'C65': 129,
'B73': 130,
'C104': 131,
'C110': 132,
'C50': 133,
'B3': 134,
'A24': 135,
'A32': 136,
'A11': 137,
'A10': 138,
'B57 B59 B63 B66': 139,
'C28': 140,
'E44': 141,
'A26': 142,
'A6': 143,
'A7': 144,
'C31': 145,
'A19': 146,
'B45': 147,
'E34': 148,
'B78': 149,
'B50': 150,
'C87': 151,
'C116': 152,
'C55 C57': 153,
'D50': 154,
'E68': 155,
'E67': 156,
'C126': 157,
'C68': 158,
'C70': 159,
```

```
'C53': 160,
'B19': 161,
'D46': 162,
'D37': 163,
'D26': 164,
'C32': 165,
'C80': 166,
'C82': 167,
'C128': 168,
'E39 E41': 169,
'D': 170,
'F4': 171,
'D56': 172,
'F33': 173,
'E101': 174,
'E77': 175,
'F2': 176,
'D38': 177,
'F': 178,
'F G63': 179,
'F E57': 180,
'F E46': 181,
'F G73': 182,
'E121': 183,
'F E69': 184,
'E10': 185,
'G6': 186,
'F38': 187,
'C': 1,
'E': 2,
'D': 3,
'A': 4,
'n': 5,
'B': 6,
'T': 7,
'F': 8,
'G': 9,
'male': 1,
'female': 2,
```

Treningowy:

	cabin	CabinReduced	sex
770	6	5	2
543	6	5	1
289	156	2	1
10	7	1	2
147	84	1	1

Testowy:

	cabin	CabinReduced	sex
1095	6	5	1
1130	185	2	1
1294	6	5	1
860	6	5	2
1126	6	5	1

0.5 8.

[8]:
```
print("X_train NaNs:\n", X_train.isna().sum())
print("X_test NaNs:\n", X_test.isna().sum())
```

```
X_train NaNs:
 cabin           0
CabinReduced    0
sex             0
dtype: int64
X_test NaNs:
 cabin           0
CabinReduced    0
sex             0
dtype: int64
```

NaN zostały zamienione na 0.

Tak, zamienienie NaN na 0 jest najlepszym wyjściem. Alternatywą jest usunięcie wierszy z NaN, przez co zostałyby stracone dodatkowe informacje w nich.

0.6 10.

[9]:
```
uniq_df = pd.DataFrame()

for feature in col_name:
    uniq_df[feature] = [
        len(X_train[feature].unique()),
        len(X_test[feature].unique()),
    ]

uniq_df.index = ['X_train', 'X_test']
display(uniq_df)

print('cabin:', len(df['cabin'].unique()))
print('CabinReduced:', len(df['CabinReduced'].unique()))
print('Różnica:', len(df['cabin'].unique()) - len(df['CabinReduced'].unique()))
```

	cabin	CabinReduced	sex
X_train	161	8	2
X_test	51	8	2

```
cabin: 187
CabinReduced: 9
Różnica: 178
```

Różnica w liczbie etykiet przed i po redukcji wynosi 178. Samo mapowanie nie zmienia liczby etykiet, natomiast redukcja może pozytywnie wpłynąć na jakość modelu, ponieważ ogranicza liczbę unikalnych (często rzadkich) wartości, które mogłyby działać jak wartości odstające.