

Московский государственный технический университет
им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»



Лабораторная работа № 2

По курсу «методы машинного обучения в АСОИУ»

«Обработка признаков. Часть 1»

Выполнил:

студент ИУ5-23М
Семенов И.А.

Проверил:

Гапанюк Ю.Е.

Подпись:

29.02.2024

Москва, 2024

Описание задания

- Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Просьба не использовать датасет, на котором данная задача решалась в лекции.
- Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
 - ✓ устранение пропусков в данных;
 - ✓ кодирование категориальных признаков;
 - ✓ нормализация числовых признаков.

Текст программы и экранные формы с примерами выполнения программы



Кодирование категориальных признаков

```
[ ] import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Загрузка и предобработка данных

Используем данные из соревнования [Titanic](#)

```
[ ] # Будем использовать только обучающую выборку
data_loaded = pd.read_csv('data/titanic.csv', sep=",")
```

```
[ ] # размер набора данных
data_loaded.shape
```

(891, 12)

```
[ ] data_loaded.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
[ ] data_features = list(zip(
    # признаки
    [i for i in data_loaded.columns],
    zip(
        # типы колонок
        [str(i) for i in data_loaded.dtypes],
        # проверим есть ли пропущенные значения
        [i for i in data_loaded.isnull().sum()]
    ))
    # Признаки с типом данных и количеством пропусков
    data_features
```

```
[ ] [ ('PassengerId', ('int64', 0)),
      ('Survived', ('int64', 0)),
      ('Pclass', ('int64', 0)),
      ('Name', ('object', 0)),
      ('Sex', ('object', 0)),
      ('Age', ('float64', 177)),
      ('SibSp', ('int64', 0)),
      ('Parch', ('int64', 0)),
      ('Ticket', ('object', 0)),
      ('Fare', ('float64', 0)),
      ('Cabin', ('object', 687)),
      ('Embarked', ('object', 2))]
```

```
[ ] # Используем только некоторые признаки
cols_filter = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare',
               'Sex', 'Cabin', 'Embarked', 'Survived']
data = data_loaded[cols_filter]
data.head()
```

```
[ ]
```

	Pclass	Age	SibSp	Parch	Fare	Sex	Cabin	Embarked	Survived
0	3	22.0	1	0	7.2500	male	NaN	S	0
1	1	38.0	1	0	71.2833	female	C85	C	1
2	3	26.0	0	0	7.9250	female	NaN	S	1
3	1	35.0	1	0	53.1000	female	C123	S	1

```
[ ] # Заполним пропуски
data.dropna(subset=['Fare', 'Embarked'], inplace=True)
```

C:\ProgramData\Anaconda3\lib\site-packages\pandas\util_decorators.py:311: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
return func(*args, **kwargs)

```
[ ] # От каюты оставляет только первую букву
# и убираем кюты типа T так как их мало
data['Cabin'] = data['Cabin'].astype(str).str[0]
data = data[data['Cabin'] != 'T']
```

C:\Users\Paladin\AppData\Local\Temp\ipykernel_9088\3362610652.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
data['Cabin'] = data['Cabin'].astype(str).str[0]

```
[ ] # Заполним пропуски возраста средними значениями
def impute_na(df, variable, value):
    df[variable].fillna(value, inplace=True)
impute_na(data, 'Age', data['Age'].mean())
```

```
[ ] # Убедимся что нет пустых значений
data.isnull().sum()
```

```
[ ]
```

Pclass	0
Age	0
SibSp	0
Parch	0
Fare	0
Sex	0
Cabin	0
Embarked	0
Survived	0

✓ 1. Кодирование категорий целочисленными значениями - [label encoding](#)

Label Encoding (LE), который также называют integer encoding, предполагает, что значения категорий заменяются целыми числами в случайном порядке.

✓ Преимущества:

- Простота реализации.
- Не расширяется признаковое пространство (не появляется дополнительных колонок).

Недостатки:

- Не использует информацию о распределении значений категорий.
- Не подходит для линейных моделей, так как создает фиктивное отношение порядка между значениями.

```
[ ] from sklearn.preprocessing import LabelEncoder
```

```
[ ] le = LabelEncoder()  
cat_enc_le = le.fit_transform(data['Cabin'])
```

```
[ ] data['Cabin'].unique()
```

```
↪ array(['n', 'C', 'E', 'G', 'D', 'A', 'B', 'F'], dtype=object)
```

```
[ ] np.unique(cat_enc_le)
```

```
↪ array([0, 1, 2, 3, 4, 5, 6, 7])
```

```
[ ] le.inverse_transform([0, 1, 2, 3])
```

```
↪ array(['A', 'B', 'C', 'D'], dtype=object)
```

✓ 2. Кодирование категорий наборами бинарных значений - [one-hot encoding](#)

One-hot encoding предполагает, что значение категории заменяется на отдельную колонку, которая содержит бинарные значения.

✓ Преимущества:

- Простота реализации.
- Подходит для любых моделей, так как НЕ создает фиктивное отношение порядка между значениями.

Недостатки:

- Расширяется признаковое пространство.

```
[ ] from sklearn.preprocessing import OneHotEncoder
```

```
[ ] ohe = OneHotEncoder()  
cat_enc_ohe = ohe.fit_transform(data[['Cabin']])  
cat_enc_ohe
```

```
↪ <888x8 sparse matrix of type '<class 'numpy.float64'>'  
   with 888 stored elements in Compressed Sparse Row format>
```

```
🎮 cat_enc_ohe.todense()[0:10]
```

```
↪ matrix([[0., 0., 0., 0., 0., 0., 0., 1.],  
          [0., 0., 1., 0., 0., 0., 0., 0.],  
          [0., 0., 0., 0., 0., 0., 0., 1.],  
          [0., 0., 1., 0., 0., 0., 0., 0.],  
          [0., 0., 0., 0., 0., 0., 0., 1.],  
          [0., 0., 0., 0., 0., 0., 0., 1.],  
          [0., 0., 0., 0., 1., 0., 0., 0.],  
          [0., 0., 0., 0., 0., 0., 0., 1.],  
          [0., 0., 0., 0., 0., 0., 0., 1.],  
          [0., 0., 0., 0., 0., 0., 0., 1.]])
```

Использование библиотеки [Category Encoders](#)

```
[ ] from category_encoders.one_hot import OneHotEncoder as ce_OneHotEncoder
```

```
[ ] ce_OneHotEncoder1 = ce_OneHotEncoder()  
data_OHE = ce_OneHotEncoder1.fit_transform(data[data.columns.difference(['Survived'])])
```

data_OHE

	Age	Cabin_1	Cabin_2	Cabin_3	Cabin_4	Cabin_5	Cabin_6	Cabin_7	Cabin_8	Embarked_1	Embarked_2	Embarked_3	Fare	Parch	Pclass	Sex_1	S
0	22.000000	1	0	0	0	0	0	0	0	1	0	0	7.2500	0	3	1	
1	38.000000	0	1	0	0	0	0	0	0	0	1	0	71.2833	0	1	0	
2	26.000000	1	0	0	0	0	0	0	0	1	0	0	7.9250	0	3	0	
3	35.000000	0	1	0	0	0	0	0	0	1	0	0	53.1000	0	1	0	
4	35.000000	1	0	0	0	0	0	0	0	1	0	0	8.0500	0	3	1	
...	
886	27.000000	1	0	0	0	0	0	0	0	1	0	0	13.0000	0	2	1	
887	19.000000	0	0	0	0	0	0	1	0	1	0	0	30.0000	0	1	0	
888	29.620492	1	0	0	0	0	0	0	0	1	0	0	23.4500	2	3	0	
889	26.000000	0	1	0	0	0	0	0	0	0	1	0	30.0000	0	1	1	
890	32.000000	1	0	0	0	0	0	0	0	0	0	1	7.7500	0	3	1	

888 rows x 18 columns

Count encoding предполагает что значение категории заменяется на количество раз, которое оно встречается в категории.

В случае frequency encoding вместо количества используется доля (процент) от количества записей.

Преимущества:

- Простота реализации.
- Не расширяется признаковое пространство.

Недостатки:

- Если два значения встречаются одинаковое количество раз, то они будут заменены на одно и то же количество и становятся неразличимы.

Использование библиотеки [Category Encoders](#)

```
[ ] from category_encoders.count import CountEncoder as ce_CountEncoder
```

```
[ ] ce_CountEncoder1 = ce_CountEncoder()  
data_COUNT_ENC = ce_CountEncoder1.fit_transform(data[data.columns.difference(['Survived'])])
```

data_COUNT_ENC

	Age	Cabin	Embarked	Fare	Parch	Pclass	Sex	SibSp
0	22.000000	687	643	7.2500	0	3	576	1
1	38.000000	59	168	71.2833	0	1	312	1
2	26.000000	687	643	7.9250	0	3	312	0
3	35.000000	59	643	53.1000	0	1	312	1
4	35.000000	687	643	8.0500	0	3	576	0
...
886	27.000000	687	643	13.0000	0	2	576	0

```
[ ] data['Cabin'].unique()
```

```
⇒ array(['n', 'C', 'E', 'G', 'D', 'A', 'B', 'F'], dtype=object)
```

```
[ ] data_COUNT_ENC['Cabin'].unique()
```

```
⇒ array([687, 59, 32, 4, 33, 15, 45, 13], dtype=int64)
```

```
[ ] ce_CountEncoder2 = ce_CountEncoder(normalize=True)
data_FREQ_ENC = ce_CountEncoder2.fit_transform(data[data.columns.difference(['Survived'])])
```

```
[ ] data_FREQ_ENC
```

```
⇒
```

	Age	Cabin	Embarked	Fare	Parch	Pclass	Sex	SibSp
0	22.000000	0.773649	0.724099	7.2500	0	3	0.648649	1
1	38.000000	0.066441	0.189189	71.2833	0	1	0.351351	1
2	26.000000	0.773649	0.724099	7.9250	0	3	0.351351	0
3	35.000000	0.066441	0.724099	53.1000	0	1	0.351351	1
4	35.000000	0.773649	0.724099	8.0500	0	3	0.648649	0
...
886	27.000000	0.773649	0.724099	13.0000	0	2	0.648649	0
887	19.000000	0.050676	0.724099	30.0000	0	1	0.351351	0
888	29.620492	0.773649	0.724099	23.4500	2	3	0.351351	1
889	26.000000	0.066441	0.189189	30.0000	0	1	0.648649	0
890	32.000000	0.773649	0.086712	7.7500	0	3	0.648649	0

888 rows × 8 columns

```
[ ] data_FREQ_ENC['Cabin'].unique()
```

```
⇒ array([0.77364865, 0.06644144, 0.03603604, 0.0045045 , 0.03716216,
```

✓ Подключено к среде выполнения "Серверный ускоритель Python 3 на базе Google Compute Engine ()".

На основе полученных результатов можем сделать вывод о том, что качество яблока напрямую зависит от его зрелости. Также имеется прямая зависимость между зрелостью яблока и его хрупкостью. Сладость же и сочность почти не зависят друг от друга.