

Introduction to preprocessing

PREPROCESSING FOR MACHINE LEARNING IN PYTHON



James Chapman

Curriculum Manager, DataCamp

What is data preprocessing?

- After exploratory data analysis and data cleaning
- Preparing data for modeling
- **Example:** transforming categorical features into numerical features (dummy variables)

Why preprocess?

- Transform dataset so it's suitable for modeling
- Improve model performance
- Generate more reliable results



Recap: exploring data with pandas

```
import pandas as pd
hiking = pd.read_json("hiking.json")
print(hiking.head())
```

```
   Prop_ID      Name  ... lat lon
0    B057  Salt Marsh Nature Trail  ... NaN NaN
1    B073      Lullwater  ... NaN NaN
2    B073      Midwood  ... NaN NaN
3    B073    Peninsula  ... NaN NaN
4    B073    Waterfall  ... NaN NaN
```

Recap: exploring data with pandas

```
print(hiking.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 11 columns):
#   Column          Non-Null Count  Dtype
--  --
0   Prop_ID         33 non-null    object
1   Name            33 non-null    object
2   Location        33 non-null    object
3   Park_Name       33 non-null    object
4   Length          29 non-null    object
5   Difficulty       27 non-null    object
6   Other_Details   31 non-null    object
7   Accessible      33 non-null    object
8   Limited_Access  33 non-null    object
9   lat             0 non-null     float64
10  lon             0 non-null     float64
dtypes: float64(2), object(9)
memory usage: 3.0+ KB
```

Recap: exploring data with pandas

```
print(wine.describe())
```

	Type	Alcohol	...	Alcalinity of ash
count	178.000000	178.000000	...	178.000000
mean	1.938202	13.000618	...	19.494944
std	0.775035	0.811827	...	3.339564
min	1.000000	11.030000	...	10.600000
25%	1.000000	12.362500	...	17.200000
50%	2.000000	13.050000	...	19.500000
75%	3.000000	13.677500	...	21.500000
max	3.000000	14.830000	...	30.000000

Removing missing data

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.dropna())
```

	A	B	C
1	4.0	7.0	3.0
4	5.0	9.0	7.0

Removing missing data

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.drop([1, 2, 3]))
```

	A	B	C
0	1.0	NaN	2.0
4	5.0	9.0	7.0

Removing missing data

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.drop("A", axis=1))
```

	B	C
0	NaN	2.0
1	7.0	3.0
2	NaN	NaN
3	7.0	NaN
4	9.0	7.0

Removing missing data

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.isna().sum())
```

A	1
B	2
C	2

dtype: int64

```
print(df.dropna(subset=["B"]))
```

	A	B	C
1	4.0	7.0	3.0
3	NaN	7.0	NaN
4	5.0	9.0	7.0

Removing missing data

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.dropna(thresh=2))
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
4	5.0	9.0	7.0

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Working With Data Types

PREPROCESSING FOR MACHINE LEARNING IN PYTHON



James Chapman

Curriculum Manager, DataCamp

Why are types important?

```
print(volunteer.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 665 entries, 0 to 664
Data columns (total 35 columns):
#   Column                Non-Null Count  Dtype
--  --
0   opportunity_id        665 non-null   int64
1   content_id            665 non-null   int64
2   vol_requests          665 non-null   int64
3   event_time            665 non-null   int64
4   title                 665 non-null   object
..  ...
34  NTA                   0 non-null     float64
dtypes: float64(13), int64(8), object(14)
memory usage: 182.0+ KB
```

- `object` : string/mixed types
- `int64` : integer
- `float64` : float
- `datetime64` : dates and times

Converting column types

```
print(df)
```

```
   A      B      C
0  1  string  1.0
1  2 string2  2.0
2  3 string3  3.0
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
--  --
0    A      3 non-null      int64
1    B      3 non-null      object
2    C      3 non-null      object
dtypes: int64(1), object(2)
memory usage: 200.0+ bytes
```

Converting column types

```
print(df)
```

	A	B	C
0	1	string	1.0
1	2	string2	2.0
2	3	string3	3.0

```
df["C"] = df["C"].astype("float")  
print(df.dtypes)
```

```
A      int64  
B      object  
C     float64  
dtype: object
```


Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Training and test sets

PREPROCESSING FOR MACHINE LEARNING IN PYTHON



James Chapman

Curriculum Manager, DataCamp

Why split?

1. Reduces *overfitting*
2. Evaluate performance on a holdout set

Splitting up your dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```

X_train y_train
0      1.0      n
1      4.0      n
      ...
5      5.0      n
6      6.0      n

X_test y_test
0      9.0      y
1      1.0      n
2      4.0      n
```

Stratified sampling

- Dataset of 100 samples: 80 **class 1** and 20 **class 2**
- Training set of 75 samples: 60 **class 1** and 15 **class 2**
- Test set of 25 samples: 20 **class 1** and 5 **class 2**

Stratified sampling

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)
```

```
y["labels"].value_counts()
```

```
class1    80  
class2    20  
Name: labels, dtype: int64
```

Stratified sampling

```
y_train["labels"].value_counts()
```

```
class1    60  
class2    15  
Name: labels, dtype: int64
```

```
y_test["labels"].value_counts()
```

```
class1     20  
class2      5  
Name: labels, dtype: int64
```

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON