



PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Preprocessing Data for Machine Learning

Sarah Guido

Senior Data Scientist



What is data preprocessing?

- Beyond cleaning and exploratory data analysis
- Prepping data for modeling
- Modeling in Python requires numerical input



Refresher on Pandas basics

```
In [1]: import pandas as pd
```

```
In [2]: hiking = pd.read_json("datasets/hiking.json")
```

```
In [3]: print(hiking.head())
```

	Accessible	Difficulty	Length	Limited_Access
0	Y	None	0.8 miles	N
1	N	Easy	1.0 mile	N
2	N	Easy	0.75 miles	N
3	N	Easy	0.5 miles	N
4	N	Easy	0.5 miles	N

Refresher on Pandas basics

```
In [4]: print(hiking.columns)
```

```
Index(['Accessible', 'Difficulty',  
      'Length', 'Limited_Access',  
      'Location', 'Name',  
      'Other_Details', 'Park_Name',  
      'Prop_ID', 'lat', 'lon'],  
      dtype='object')
```

```
In [5]: print(hiking.dtypes)
```

```
Accessible      object  
Difficulty      object  
Length          object  
Limited_Access  object  
Location        object  
Name            object  
Other_Details   object  
Park_Name       object  
Prop_ID         object  
lat             float64  
lon             float64  
dtype: object
```

Refresher on Pandas basics

```
In [6]: print(wine.describe())
```

	Type	Alcohol	Malic acid	Ash	Alcalinity of ash
count	178.000000	178.000000	178.000000	178.000000	178.000000
mean	1.938202	13.000618	2.336348	2.366517	19.494944
std	0.775035	0.811827	1.117146	0.274344	3.339564
min	1.000000	11.030000	0.740000	1.360000	10.600000
25%	1.000000	12.362500	1.602500	2.210000	17.200000
50%	2.000000	13.050000	1.865000	2.360000	19.500000
75%	3.000000	13.677500	3.082500	2.557500	21.500000
max	3.000000	14.830000	5.800000	3.230000	30.000000



Removing missing data

```
In [7]: 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

```
In [8]: print(df.dropna())
```

	A	B	C
1	4.0	7.0	3.0
4	5.0	9.0	7.0



Removing missing data

```
In [9]: 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

```
In [10]: 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

```
In [11]: 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

```
In [12]: 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

```
In [13]: 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

```
In [14]: print(df[df["B"] == 7])
```

	A	B	C
1	4.0	7.0	3.0
3	NaN	7.0	NaN



Removing missing data

```
In [15]: 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

```
In [16]: print(df["B"].isnull().sum())
```

```
2
```

```
In [17]: print(df[df["B"].notnull()])
```

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



PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Let's practice!



PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Working With Data Types

Sarah Guido

Senior Data Scientist



Why are types important?

```
In [1]: print(volunteer.dtypes)
```

```
opportunity_id      int64
content_id          int64
vol_requests        int64
event_time          int64
title               object
hits                int64
summary             object
is_priority          object
category_id         float64
...
```

- object: string/mixed types
- int64: integer
- float64: float
- datatype64 (or timedelta): datetime



Converting column types

```
In [2]: print(df)
```

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

```
In [3]: print(df.dtypes)
```

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



Converting column types

```
In [4]: print(df)
```

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

```
In [5]: df["C"] = df["C"].astype("float")
```

```
In [6]: print(df.dtypes)
```

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



PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Let's practice!



PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Training and Test Sets

Sarah Guido

Senior Data Scientist



Splitting up your dataset

```
In [1]: from sklearn.model_selection import train_test_split
```

```
In [2]: X_train, X_test, y_train, y_test = train_test_split(X, y)
```

	X_train	y_train
0	1.0	n
1	4.0	n
2	7.0	n
3	2.0	n
4	5.0	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

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



Stratified sampling

```
In [3]: y["labels"].value_counts()
```

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

```
In [4]: X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y)
```

```
In [5]: y_train["labels"].value_counts()
```

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

```
In [6]: y_test["labels"].value_counts()
```

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



PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Let's practice!