Data Preprocessing

Hongchang Gao

Spring 2024

Review Data Understanding

- Tabular Data
  - Numerical values
    - Line plot
    - Hist plot
    - Boxplot
    - Scatter plot
  - Categorial values
    - Bar plot
    - Pie plot
- Temporal data
- Spatial data
- Graph data

Outline

- Missing values
- Categorical features
- Normalization

Why is data processing important?

- 1. Missing values

```
0   ham  Go until jurong point, crazy.. Available only ...       NaN
1   ham                      Ok lar... Joking wif u oni...       NaN
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...       NaN
3   ham  U dun say so early hor... U c already then say...       NaN
4   ham  Nah I don't think he goes to usf, he lives aro...       NaN
```

Why is data processing important?

- 2. Categorical features

```
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

Why is data processing important?

- 3. Different scales

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

Outline

- Missing values
- Categorical features
- Feature scaling

Missing Values

```
   longitude  latitude  housing_median_age
0      False     False               False
1      False     False               False
2      False     False               False
3      False     False               False
4      False     False               False
```

- Find the missing values
  - DataFrame.isnull()
    - Return a boolean same-sized object indicating if the values are NA

```python
import pandas as pd

df = pd.read_csv('housing.csv')
# print(df)

print(df.shape)

df.isnull().sum()
```

```
(20640, 10)

longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
```

Missing Values

- Methods:
  - 1. Remove the feature with a lot of missing values

```python
import pandas as pd

df = pd.read_csv('housing.csv')

print(df.isnull().sum()/df.shape[0])
```

```
longitude             0.000000
latitude              0.000000
housing_median_age    0.000000
total_rooms           0.000000
total_bedrooms        0.010029
population            0.000000
households            0.000000
median_income         0.000000
median_house_value    0.000000
ocean_proximity       0.000000
```

Missing Values

- Methods: Remove the feature with a lot of missing values

```
import pandas as pd

df = pd.read_csv('housing.csv')

print(df.columns)

df = df.drop('total_bedrooms', axis=1)
print(df.columns)
```

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'population', 'households', 'median_income', 'median_house_value',
       'ocean_proximity'],
      dtype='object')
```

Missing Values

- Methods: Fill in the missing values
  - Numerical values
    - Fill in the missing values with mean or median

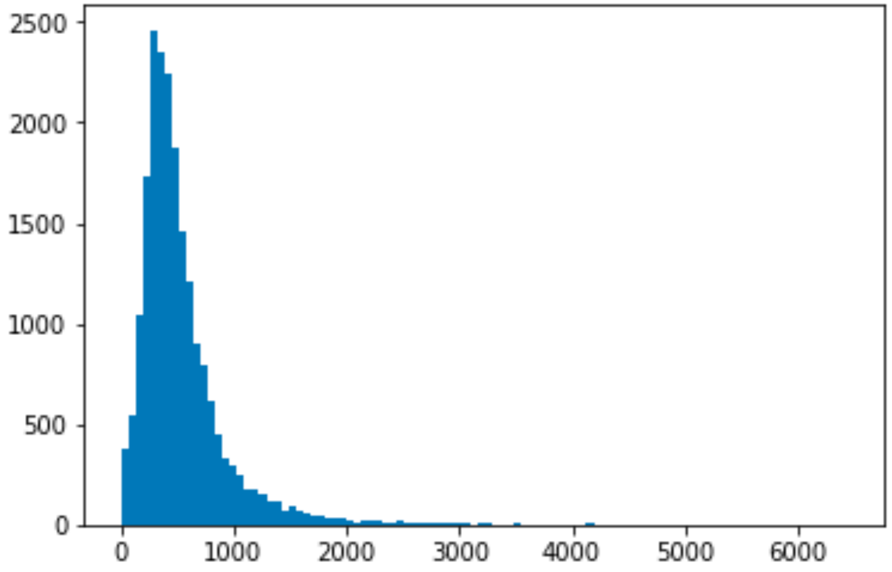
```
mean_val = df['total_bedrooms'].mean()
median_val = df['total_bedrooms'].median()

print(mean_val)
print(median_val)

df['total_bedrooms'] = df['total_bedrooms'].fillna(mean_val)
print(df.isnull().sum())
```

```
537.8705525375639
438.0
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

Missing Values

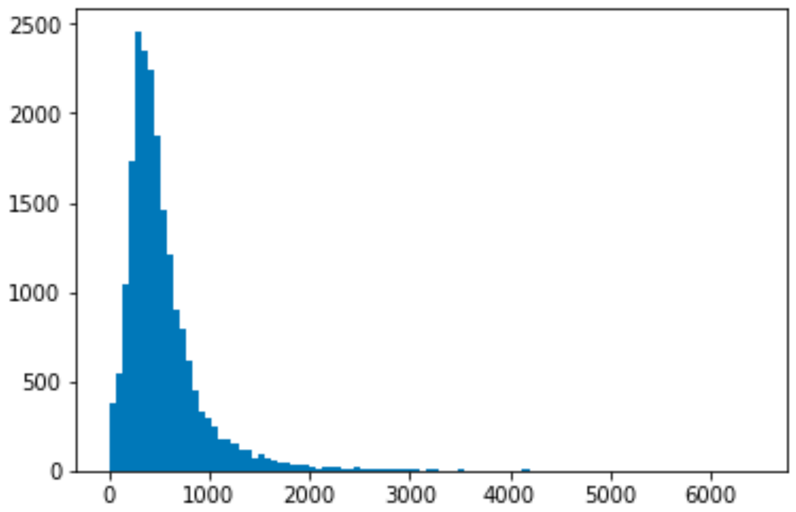
- Methods: Fill in the missing values
  - Numerical values
    - Mean or Median value? Check the distribution

```
import pandas as pd
import matplotlib.pyplot as plt


df = pd.read_csv('housing.csv')

plt.hist(df['total_bedrooms'].values, 100)

plt.show()
```

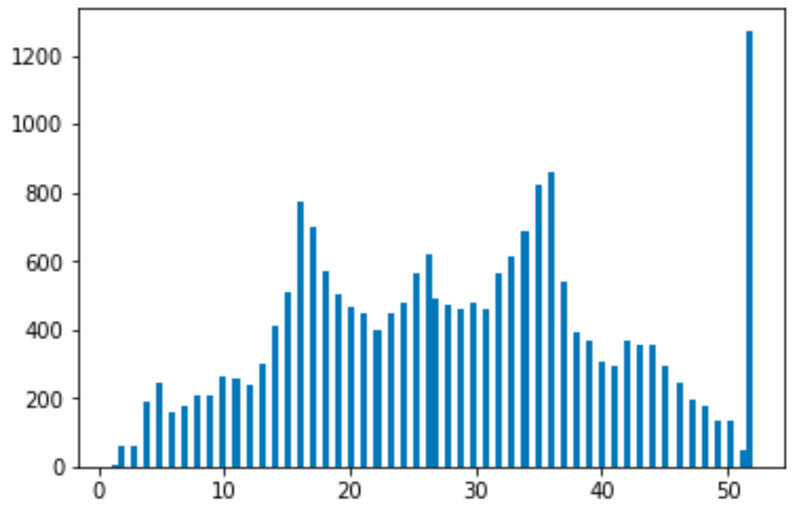


Missing Values

- Methods: Fill in the missing values
  - Check the distribution
    - Long tail distribution: median
    - Non-long tail distribution: mean or median



Long tail distribution
many values far from the "head" or central part of the distribution



Non-long tail distribution

Missing Values

- Example

[1, 2, 3, 5, 4, 200]

Mean=?

Median=?

[1, 2, 3, 5, 2, 4]

Mean=?

Median=?

Missing Values

- Methods: Fill in the missing values
  - Categorical features?
    - Impossible to compute mean or median values

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                  41          880           129.0
1    -122.22     37.86                  21         7099          1106.0
2    -122.24     37.85                  52         1467           190.0
3    -122.25     37.85                  52         1274           235.0
4    -122.25     37.85                  52         1627           280.0

   population  households  median_income  median_house_value ocean_proximity
0         322         126         8.3252              452600             NaN
1        2401        1138         8.3014              358500             NaN
2         496         177         7.2574              352100             NaN
3         558         219         5.6431              341300             NaN
4         565         259         3.8462              342200        NEAR BAY
```

Missing Values

- Methods: Fill the missing values
  - Categorical features?
    - Add a new categorical value

```
import pandas as pd

df = pd.read_csv('housing_missing.csv')

print(df['ocean_proximity'].unique())

filling_value = 'PA'
df['ocean_proximity'] = df['ocean_proximity'].fillna(filling_value)

print(df['ocean_proximity'].unique())
```

```
[nan 'NEAR BAY' '<1H OCEAN' 'INLAND' 'NEAR OCEAN' 'ISLAND']
['PA' 'NEAR BAY' '<1H OCEAN' 'INLAND' 'NEAR OCEAN' 'ISLAND']
```

Outline

- Missing values
- Categorical features
- Feature scaling

Categorical Values

- Convert categorical values to numerical values

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

```
['NEAR BAY' '<1H OCEAN' 'INLAND' 'NEAR OCEAN' 'ISLAND']
```

Categorical Values

- Label Encoding
  - each categorical feature is converted into an integer value

NEAR BAY	0
<1H OCEAN	1
INLAND	2
NEAR OCEAN	3
ISLAND	4

Categorical Values

- Label Encoding
  - each categorical feature is converted into an integer value

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('housing.csv')
print(df["ocean_proximity"].value_counts())

labelencoder = LabelEncoder()
df['ocean_proximity'] = labelencoder.fit_transform(df['ocean_proximity'])

print(df["ocean_proximity"].value_counts())
```

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
```

```
0    9136
1    6551
4    2658
3    2290
2       5
```

Categorical Values

- One-Hot Encoding
  - Each category is mapped with a vector containing either 0 or 1

```
['NEAR BAY' '<1H OCEAN' 'INLAND' 'NEAR OCEAN' 'ISLAND']
```

NEAR BAY	1	0	0	0	0
<1H OCEAN	0	1	0	0	0
INLAND	0	0	1	0	0
NEAR OCEAN	0	0	0	1	0
ISLAND	0	0	0	0	1

Categorical Values

- Example

[‘Apple’, ‘Orange’, ‘Banana’, ‘Kiwi’]

Categorical Values

- One-Hot Encoding
  - Each category is mapped with a vector containing either 0 or 1

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

df = pd.read_csv('housing.csv')
print(df["ocean_proximity"][0])

onehotencoder = OneHotEncoder(sparse=False)
result = onehotencoder.fit_transform(df[['ocean_proximity']])
print(result[0,:])
```

```
NEAR BAY
[0. 0. 0. 1. 0.]
```

Categorical Values

- Ordinal Encoding:
  - the categorical feature is ordinal
  - retaining the order is important

rating	
Poor	1
Good	2
Very Good	3
Excellent	4

Categorical Values

- Example

Degree	
High school	
Bachelor	
Master	
PhD	

Categorical Values

- Ordinal Encoding:
  - the categorical feature is ordinal
  - retaining the order is important

```
import pandas as pd

data = {'rating': ['Poor', 'Good', 'Very Good', 'Excellent']}
df = pd.DataFrame(data)
print(df)

coding_map = {'Poor': 1, 'Good': 2, 'Very Good': 3, 'Excellent': 4}
df['rating'] = df.rating.map(coding_map)
print(df)
```

```
      rating
0       Poor
1       Good
2  Very Good
3  Excellent
   rating
0       1
1       2
2       3
3       4
```

Outline

- Missing values
- Categorical features
- Feature scaling

Feature Scaling

- Different features have different scales

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

$$\|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2}$$

The feature with large scale dominates the distance

Feature Scaling

- Example

$$x = (1, 1000)\ y = (2, 2000)$$

$$x = (1, 4)\ y = (3, 5)$$

Feature Scaling Methods

- Min-max normalization
  - Sensitive to outliers

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
np.set_printoptions(precision=4)

df = pd.read_csv('housing.csv')
X = df.values[0:5, 5:9].astype(dtype=np.float32)
print('Original data')
print(X)

x_min = X.min(axis=0)
x_max = X.max(axis=0)
print('min and max')
print(x_min)
print(x_max)

X = (X-x_min)/(x_max-x_min)
print('Scaling data')
print(X)
```

```
Original data
[[   322.         126.           8.3252 452600.    ]
 [  2401.        1138.           8.3014 358500.    ]
 [   496.         177.           7.2574 352100.    ]
 [   558.         219.           5.6431 341300.    ]
 [   565.         259.           3.8462 342200.    ]]
min and max
[   322.         126.           3.8462 341300.    ]
[  2401.        1138.           8.3252 452600.    ]
Scaling data
[[0.     0.     1.     1.    ]
 [1.     1.     0.9947 0.1545]
 [0.0837 0.0504 0.7616 0.097 ]
 [0.1135 0.0919 0.4012 0.    ]
 [0.1169 0.1314 0.     0.0081]]
```

Feature Scaling Methods

- Example

$$\mathbf{x} = [1, 2, 3, 4]$$

Feature Scaling Methods

- Z-Score normalization (Standardization)
  - Good for normal distribution

$$x' = \frac{x-\bar{x}}{\sigma}$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Exercise:

$$\mathbf{x} = [1, 2, 3, 4]$$

Feature Scaling Methods

- Z-Score normalization (Standardization)

$$x' = \frac{x - \bar{x}}{\sigma}$$

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
np.set_printoptions(precision=4)

df = pd.read_csv('housing.csv')
X = df.values[0:5, 5:9].astype(dtype=np.float32)
print('Original data')
print(X)

x_mean = np.mean(X, axis=0)
x_std = np.std(X, axis=0)

print('Mean and Std')
print(x_mean)
print(x_std)

X = (X-x_mean)/x_std
print('Scaling data')
print(X)
```

```
Original data
[[   322.         126.           8.3252 452600.    ]
 [  2401.        1138.           8.3014 358500.    ]
 [   496.         177.           7.2574 352100.    ]
 [   558.         219.           5.6431 341300.    ]
 [   565.         259.           3.8462 342200.    ]]
Mean and Std
[   868.4        383.8          6.6747 369340.    ]
[   771.2972     379.6785       1.719   42118.336 ]
Scaling data
[[-0.7084 -0.679   0.9602  1.9768]
 [ 1.987   1.9864  0.9463 -0.2574]
 [-0.4828 -0.5447  0.339  -0.4093]
 [-0.4024 -0.4341 -0.6001 -0.6657]
 [-0.3934 -0.3287 -1.6454 -0.6444]]
```