Foundations of Data Science

Master in Data Science

2021 / 2022

Exploratory Data Analysis

Exploratory Data Analysis

- “It is important to understand what you CAN DO before you learn to measure how WELL you seem to have done it.”

  John W. Tukey, Exploratory Data Analysis, 1977

Exploratory Data Analysis

- "It is important to understand what you CAN DO before you learn to measure how WELL you seem to have done it."

  John W. Tukey, Exploratory Data Analysis, 1977

- Investigative process to know the data
  - Look for anomalies: outliers, spurious values
  - Uncover patterns and potential associations
  - Generate hypotheses
  - Nowadays it is mostly graph-based

EDA

- Visual inspection
  - Get early understanding of the data just by ‘looking at it’
  - Get early insights such as
    ‘what are the columns / measurements’
    ‘how are these encoded’
    ‘how are missing values represented’
  - Can help spot initial data problems

```
titanic = sns.load_dataset('titanic')
titanic.head()
```

✓ 1.1s

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
titanic.info()
```

✓ 0.7s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.6+ KB
```

EDA - Summary Statistics

- Categorical data
  - Count the frequency of each category
  - With bivariate categorical data, count frequency of combinations
  - With multivariate categorical data, use several aggregations
    - Pandas: groupby, pivot_table, MultiIndex

```
titanic['class'].value_counts()
```

✓ 0.5s

```
Third     491
First     216
Second    184
Name: class, dtype: int64
```

```
titanic.groupby(['class', 'deck']).size()
```

✓ 0.7s

```
class   deck
First   A       15
        B       47
        C       59
        D       29
        E       25
        F        0
        G        0
Second  A        0
        B        0
        C        0
        D        4
        E        4
        F        8
        G        0
```

EDA - Summary Statistics

- Categorical data
  - Count the frequency of each category
  - With bivariate categorical data, count freque
  - With multivariate categorical data, use several aggregations
    - Pandas: groupby, pivot_table, MultiIndex

```python
age = pd.cut(titanic['age'], [0, 18, 60, 100])
titanic.pivot_table('survived', index=['sex', age],
    columns='class', aggfunc='sum')
```

✓ 0.1s

	class	First	Second	Third
sex	**age**			
female	(0, 18]	10.0	14.0	22.0
	(18, 60]	70.0	54.0	24.0
	(60, 100]	2.0	NaN	1.0
male	(0, 18]	4.0	9.0	11.0
	(18, 60]	35.0	5.0	27.0
	(60, 100]	1.0	1.0	0.0

EDA - Summary Statistics

- Univariate continuous data
  - Tukey's five number summary

  minimum value, lower quartile, median, upper quartile, maximum value

```
titanic.describe()
```

✓ 0.1s

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

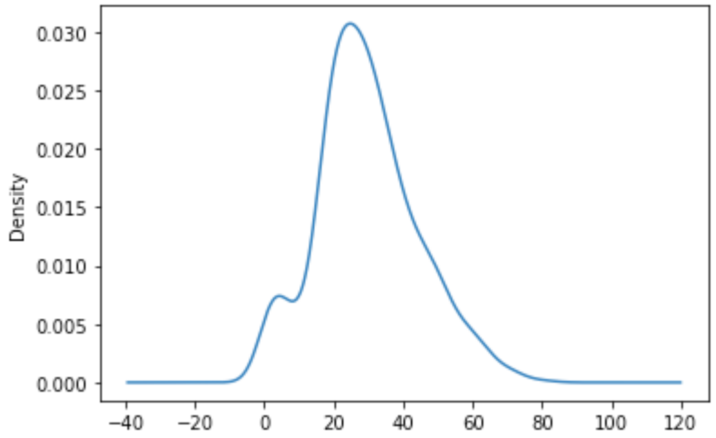
EDA - Summary Statistics

<u>Skewness</u>

Measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.

Skew = 0.389

Kurtosis = 0.178



<u>Kurtosis</u>

Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution.

Data sets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails.

EDA - Summary Statistics

- Bivariate/Multivariate continuous data
  - Covariance - a measure of the joint variability of two random variables
  - Correlation - a measure of linear correlation between two sets of data (Pearson's)

```
titanic.cov()
```

✓ 0.8s Python

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	0.236772	-0.137703	-0.551296	-0.018954	0.032017	6.221787	-0.132720	-0.048451
pclass	-0.137703	0.699015	-4.496004	0.076599	0.012429	-22.830196	0.038494	0.055347
age	-0.551296	-4.496004	211.019125	-4.163334	-2.344191	73.849030	2.012292	1.428550
sibsp	-0.018954	0.076599	-4.163334	1.216043	0.368739	8.748734	-0.136916	-0.315568
parch	0.032017	0.012429	-2.344191	0.368739	0.649728	8.661052	-0.138108	-0.230242
fare	6.221787	-22.830196	73.849030	8.748734	8.661052	2469.436846	-4.428757	-6.613861
adult_male	-0.132720	0.038494	2.012292	-0.136916	-0.138108	-4.428757	0.239723	0.097026
alone	-0.048451	0.055347	1.428550	-0.315568	-0.230242	-6.613861	0.097026	0.239723

```
titanic.corr()
```

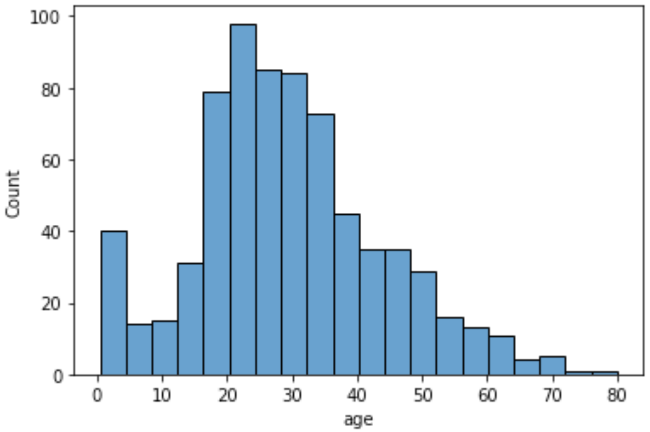
✓ 0.1s Python

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	-0.557080	-0.203367
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	0.094035	0.135207
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.280328	0.198270
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.253586	-0.584471
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.349943	-0.583398
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	-0.182024	-0.271832
adult_male	-0.557080	0.094035	0.280328	-0.253586	-0.349943	-0.182024	1.000000	0.404744
alone	-0.203367	0.135207	0.198270	-0.584471	-0.583398	-0.271832	0.404744	1.000000

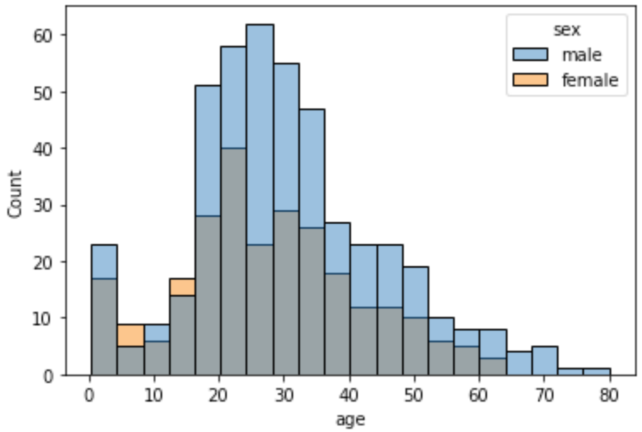
EDA - Visualizations

- Range and distribution
  - Histograms

```
sns.histplot(data=titanic, x='age')
✓ 0.5s
<AxesSubplot:xlabel='age', ylabel='Count'>
```



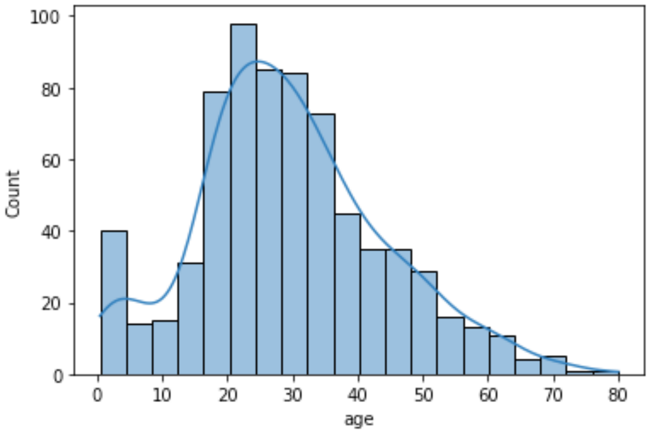

```
sns.histplot(data=titanic, x='age', hue='sex')
✓ 0.8s
<AxesSubplot:xlabel='age', ylabel='Count'>
```



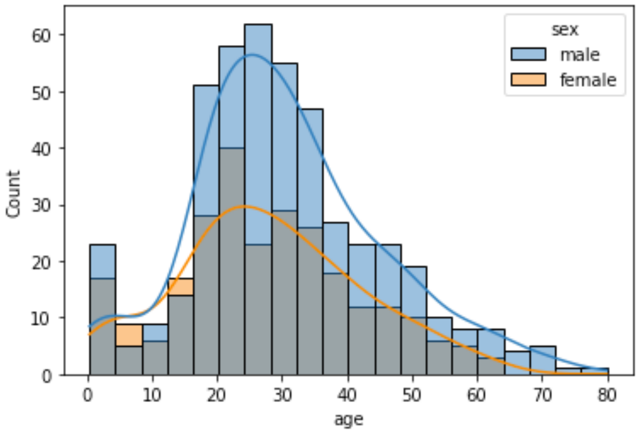
EDA - Visualizations

- Range and distribution
  - Kernel Density Estimation

```
sns.histplot(data=titanic, x='age', kde=True)
✓ 0.5s
<AxesSubplot:xlabel='age', ylabel='Count'>
```




```
sns.histplot(data=titanic, x='age', hue='sex', kde=
✓ 0.8s
<AxesSubplot:xlabel='age', ylabel='Count'>
```



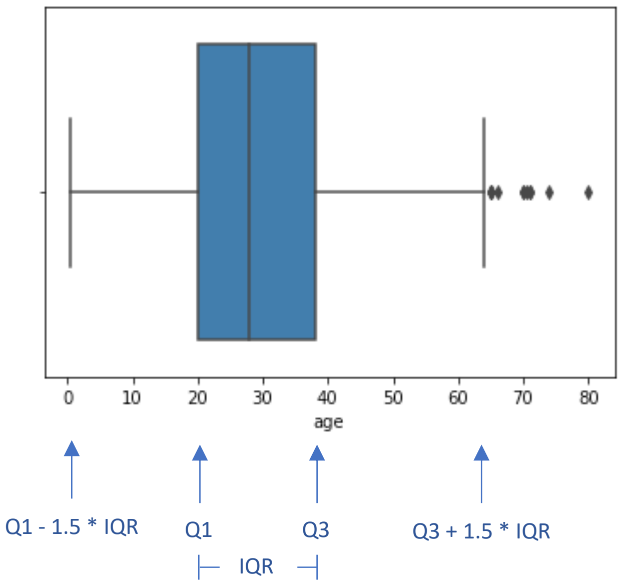
EDA - Visualizations

- Range and distribution
  - Box and whiskers

```
sns.boxplot(data=titanic, x='age')
```

✓ 0.3s

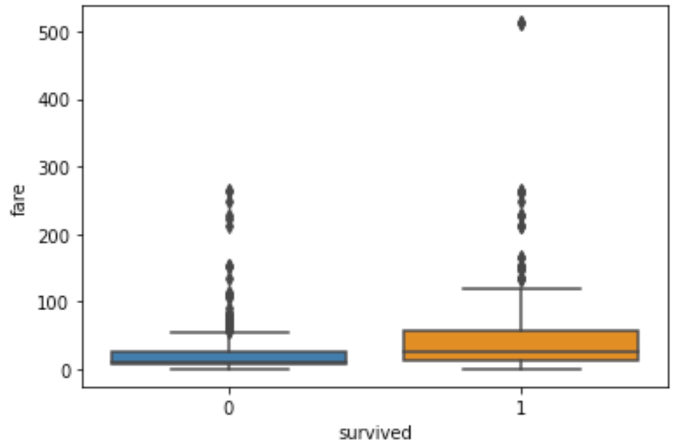
```
<AxesSubplot:xlabel='age'>
```




```
sns.boxplot(data=titanic, x='survived', y='fare')
```

✓ 0.3s

```
<AxesSubplot:xlabel='survived', ylabel='fare'>
```



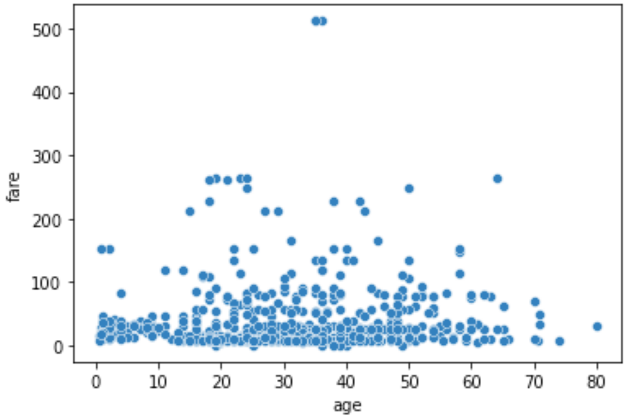
EDA - Visualizations

- Bivariate relations
  - Scatter

```
sns.scatterplot(data=titanic, x='age', y='fare')
```

✓ 0.4s

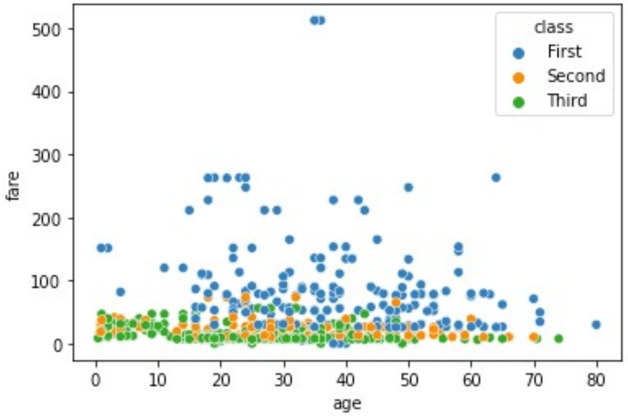
```
<AxesSubplot:xlabel='age', ylabel='fare'>
```




```
sns.scatterplot(data=titanic, x='age', y='fare', hue='class')
```

✓ 0.7s

```
<AxesSubplot:xlabel='age', ylabel='fare'>
```



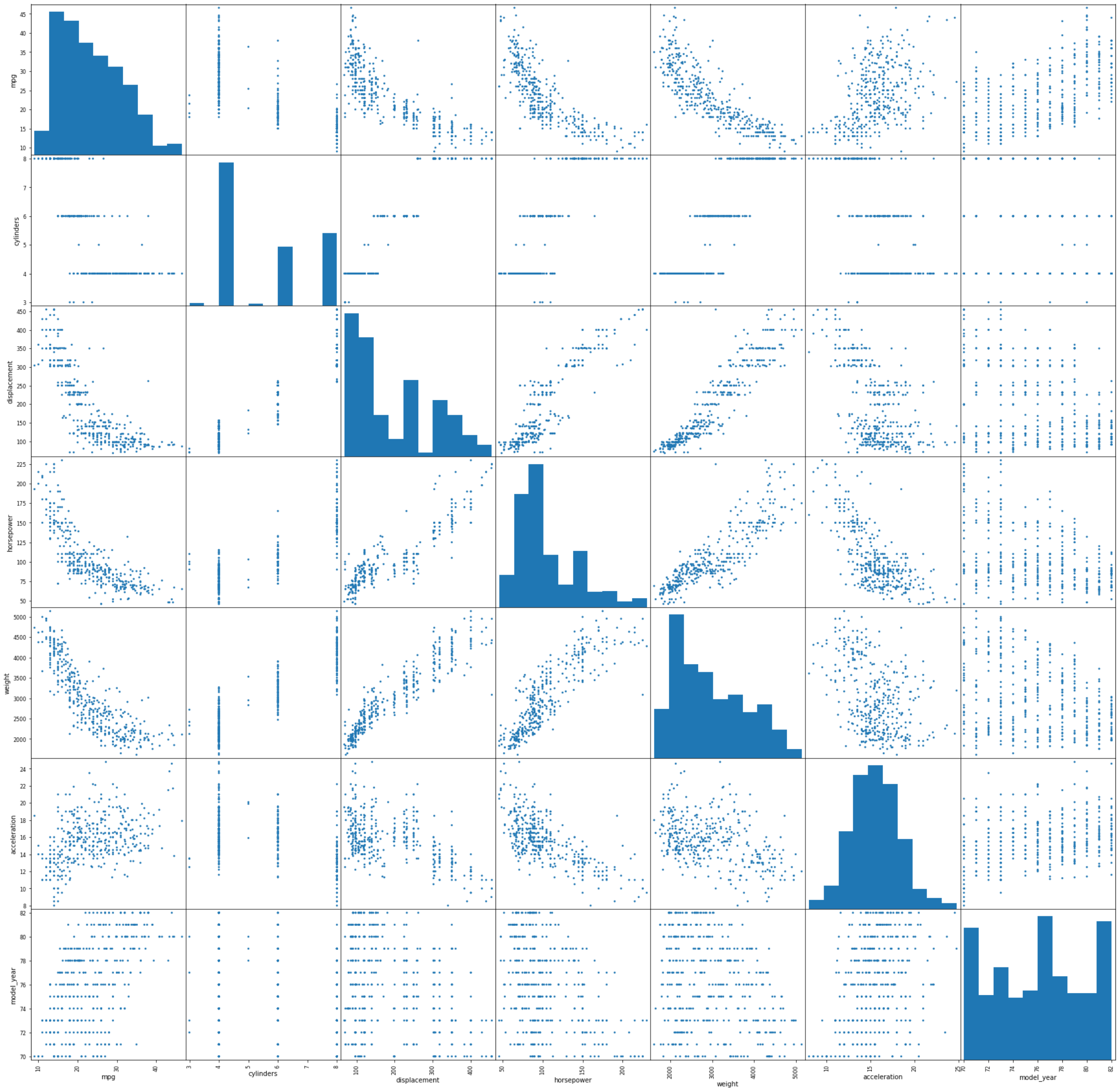
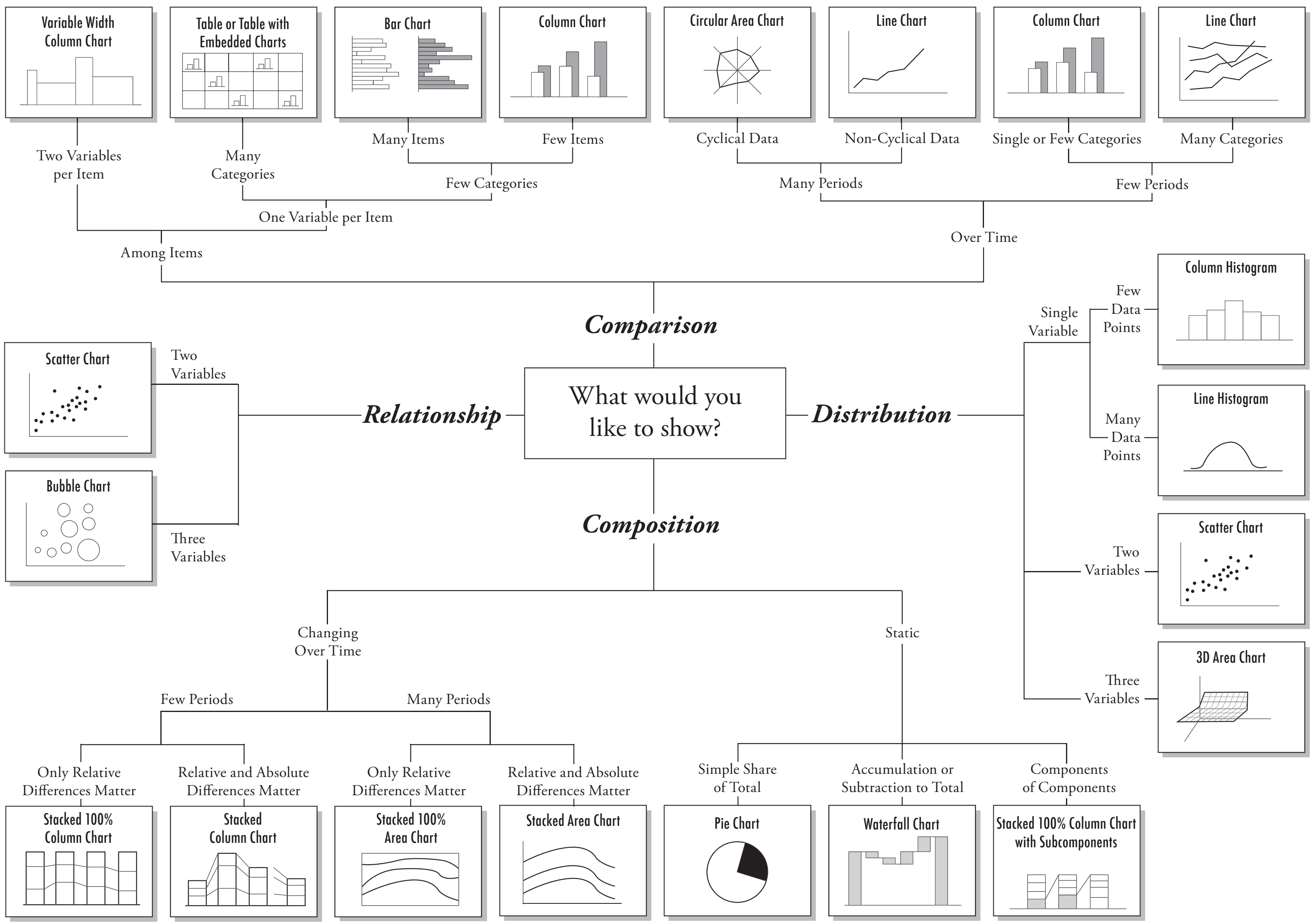

mpg
cylinders
displacement
horsepower
weight
acceleration
model_year

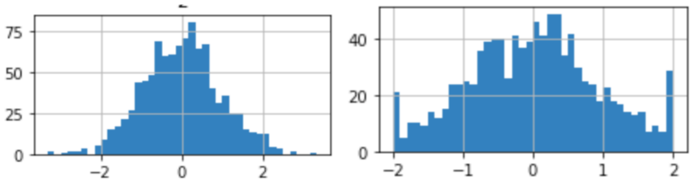
Chart Suggestions—A Thought-Starter

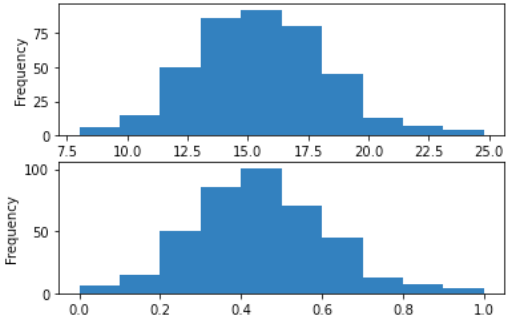


EDA – Transformations

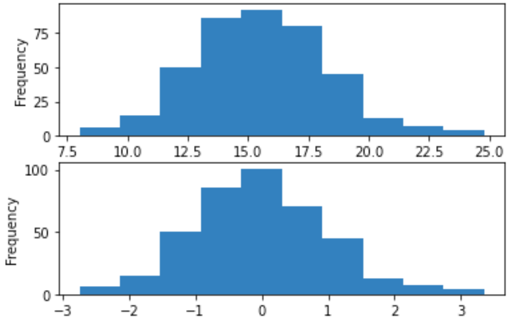
Scaling and Normalization

- Clipping

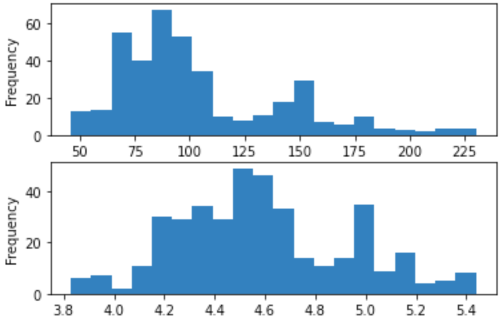




- Min-max scaling $x' = (x - x_{min})/(x_{max} - x_{min})$



- Z-score normalization $x' = (x - x_{mean})/std(x)$



- Log transformation $x' = log(x)$

Clipping

```
data = pd.DataFrame(np.random.randn(1000, 4))
data.describe()
```

✓ 0.1s

	0	1	2	3
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.006937	-0.001148	0.005534	-0.071845
std	0.997755	0.996284	0.956239	0.968821
min	-3.469708	-3.480299	-3.318964	-3.061939
25%	-0.667270	-0.635888	-0.626244	-0.753025
50%	0.001361	0.019188	0.019076	-0.085168
75%	0.678818	0.635179	0.591785	0.565218
max	3.320355	3.926691	3.343637	4.079920

```
data[np.abs(data) > 2] = np.sign(data) * 2
data.describe()
```

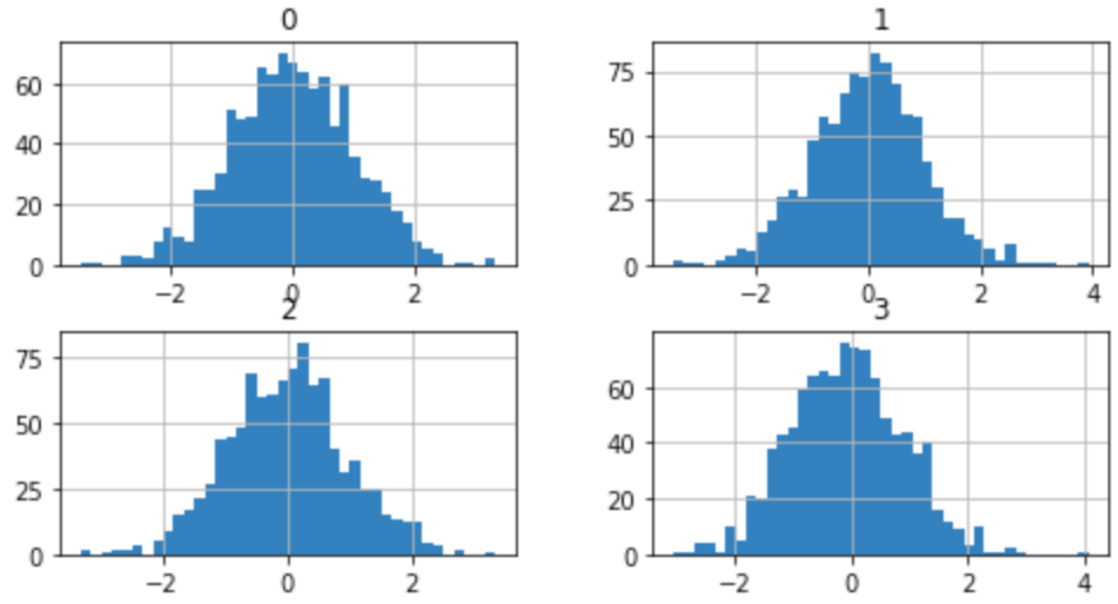
✓ 0.1s

	0	1	2	3
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.008778	-0.003545	0.006757	-0.072724
std	0.957944	0.938595	0.922174	0.929572
min	-2.000000	-2.000000	-2.000000	-2.000000
25%	-0.667270	-0.635888	-0.626244	-0.753025
50%	0.001361	0.019188	0.019076	-0.085168
75%	0.678818	0.635179	0.591785	0.565218
max	2.000000	2.000000	2.000000	2.000000

Clipping

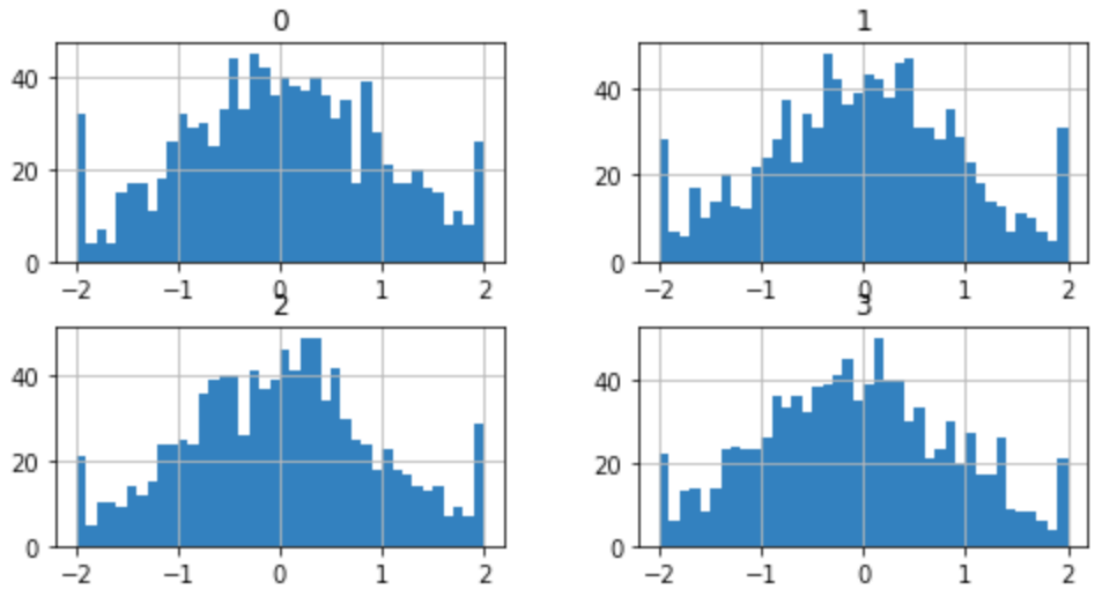
```
data = pd.DataFrame(np.random.randn(1000, 4))
data.describe()
```

✓ 0.1s




```
data[np.abs(data) > 2] = np.sign(data) * 2
data.describe()
```

✓ 0.1s

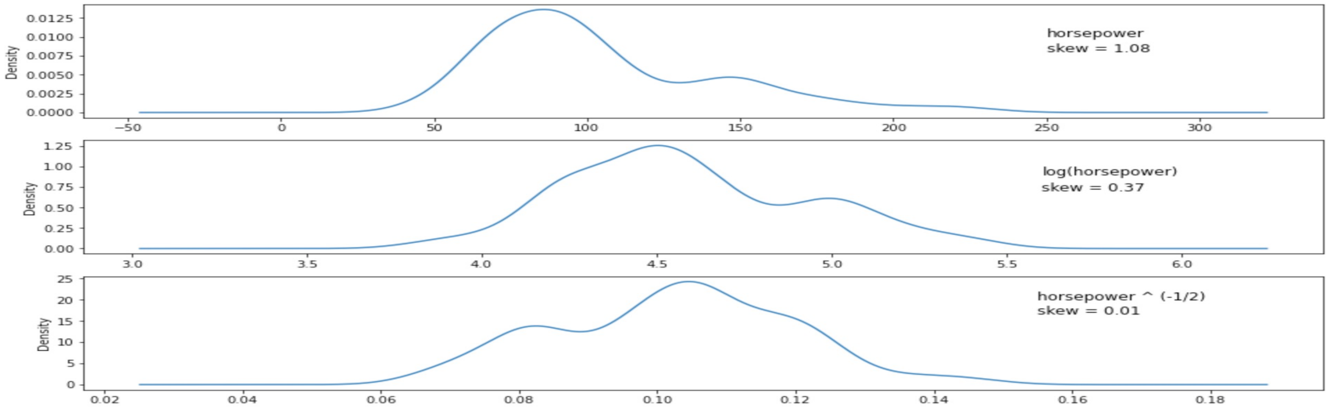


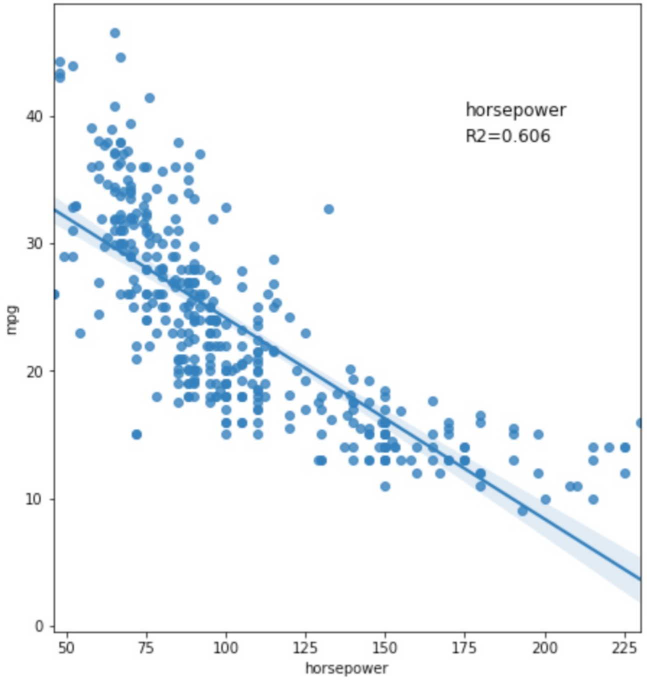
Tukey's ladder of powers

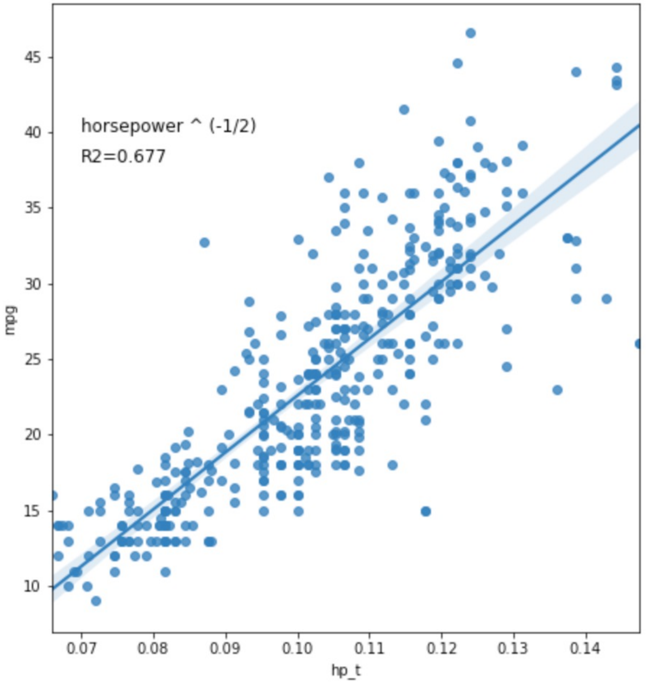
- How the distribution of $[0,+\infty)$ variables can be transformed to near symmetry using power transformations
  - Positively skewed data can be symmetrically transformed using negative powers
  - Negatively skewed data can be symmetrically transformed using positive powers

Power	Description	Skewness	Effect
-3	Inverse cubic	$+$	Drastic
-2	Inverse square	$+$	$\uparrow$
-1	Inverse	$+$	Reciprocal
$-\frac{1}{2}$	Inverse square root	$+$	$\downarrow$
$-\frac{1}{3}$	Inverse cubic root	$+$	Mild
$\log_b$	Logarithmic transformation	$+$	
$\frac{1}{3}$	Cubic root	$-$	Mild
$\frac{1}{2}$	Square root	$-$	$\uparrow$
1	Identity	$-$	None
2	Square	$-$	$\downarrow$
3	Cubic	$-$	Drastic

Example


horsepower
skew = 1.08
log(horsepower)
skew = 0.37
horsepower ^ (-1/2)
skew = 0.01
Density


horsepower
R2=0.606
mpg
horsepower

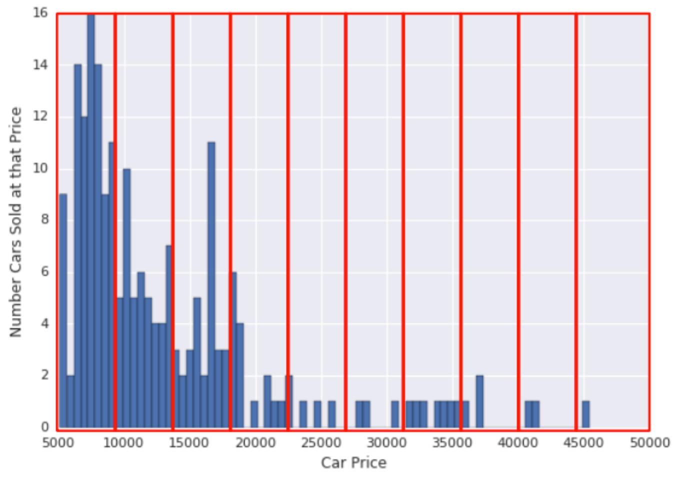

horsepower ^ (-1/2)
R2=0.677
mpg
hp_t

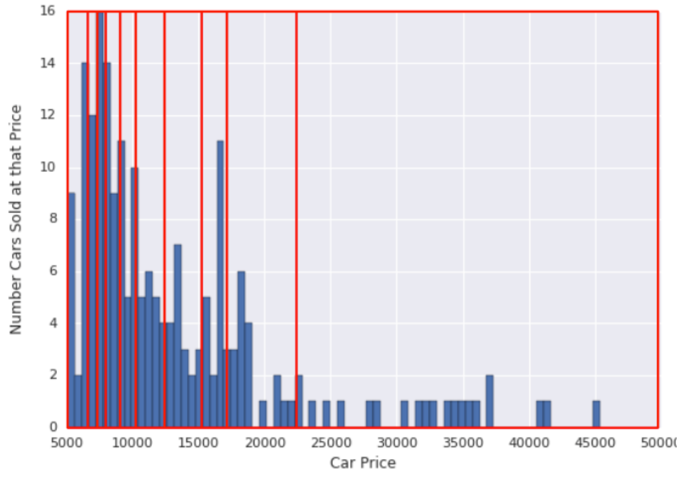
Binning

- Sometimes the continuous (raw) values are not informative, but value ranges (bins) can be

- Represent the data by bins
  - Equally spaced bins

  - Equally sized bins (quantile binning)
    Better representation of skewed values





Binning

```
ages = np.round(np.random.random([20])*90+10)
ages
```
✓ 0.8s

```
array([30., 92., 55., 74., 27., 36., 56., 73., 96., 19., 94., 85., 96.,
       92., 52., 97., 41., 36., 75., 77.])
```

```
age_bins = np.arange(10, 110, 10)
age_bins
```
✓ 0.1s

```
array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100])
```

The pandas function *cut* returns a special Categorical object

```
pd.cut(ages, age_bins)
```
✓ 0.6s

```
[(20, 30], (90, 100], (50, 60], (70, 80], (20, 30], ..., (90, 100], (40, 50], (30, 40], (70, 80], (70, 80]]
Length: 20
Categories (9, interval[int64]): [(10, 20] < (20, 30] < (30, 40] < (40, 50] ... (60, 70] < (70, 80] < (80, 90] < (90, 100]]
```

```
age_groups = pd.cut(ages, age_bins)
age_groups.codes
```
✓ 0.4s

```
array([ 1,  1,  7,  5,  8,  8, -1,  4,  4,  1,  5, -1, -1,  5,  4,  7,  1,
        2,  0,  1], dtype=int8)
```

See also:

```
pd.qcut
```

Categorical features

- Ordinal encoding

```
titanic['embark_town'].unique()
```

✓ 0.6s

```
array(['Southampton', 'Cherbourg', 'Queenstown', nan], dtype=object)
```

```
cities = list(titanic['embark_town'].unique())
titanic['embark_town'].map(lambda c: cities.index(c))
```

✓ 0.6s

```
0       0
1       1
2       0
3       0
4       0
```

Or use: CategoricalDtype
df. .astype('category')

What is the problem with this option?

In which cases could it work?

Categorical features

- Indicator features / One hot encoding

```
titanic['embark_town'].unique()
```

0.6s

```
array(['Southampton', 'Cherbourg', 'Queenstown', nan], dtype=object)
```

```
pd.get_dummies(titanic['embark_town'])
```

0.5s

	Cherbourg	Queenstown	Southampton
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1

Categorical features

- Dummy features

```
cities = list(titanic['embark_town'].unique())
pd.get_dummies(titanic['embark_town'], drop_first=True)
```

✓ 0.6s

	Queenstown	Southampton
0	0	1
1	0	0
2	0	1
3	0	1
4	0	1
5	1	0