



Data Science 101

Lesson 4 – Intro to Data Transformation in Python

Data Science & Machine Learning Team

10/23/2020

Yifei Yun

Yifei.Yun@hms.com

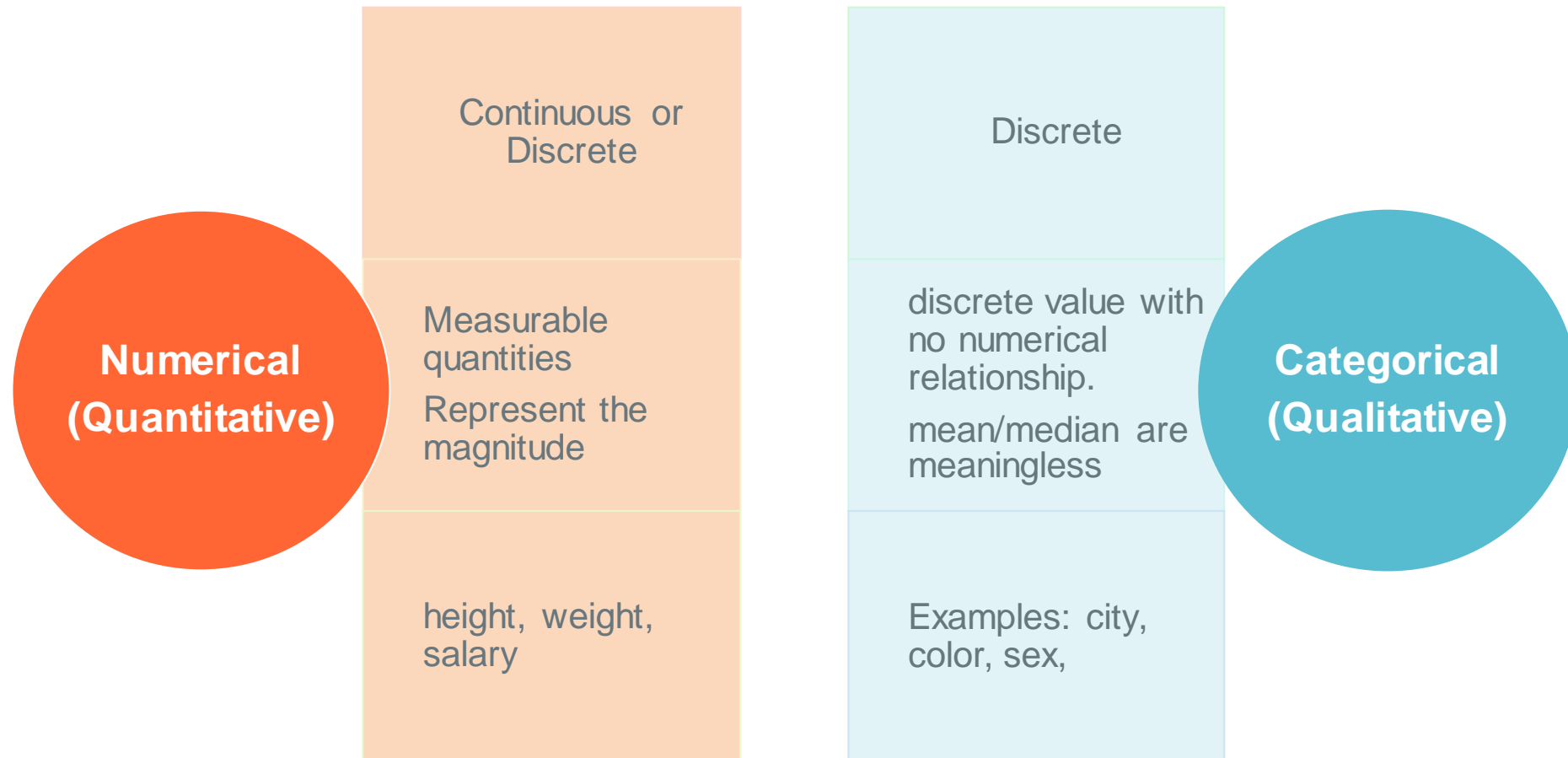
IT'S ALL ABOUT DATA

Agenda

- **Two Main Data Types**
- **Data Wrangling**
 - Aggregation
 - Stacking & Merge
 - Group by and Reshaping in data (Pivot)
- **Data Transformation**
 - Feature Scaling
 - Standardization (z-score normalization)
 - Normalization
 - Popular transformation (Log, Square Root)
- **Intro to Data Pipeline / ETL (broader process) with a simple pipeline example**

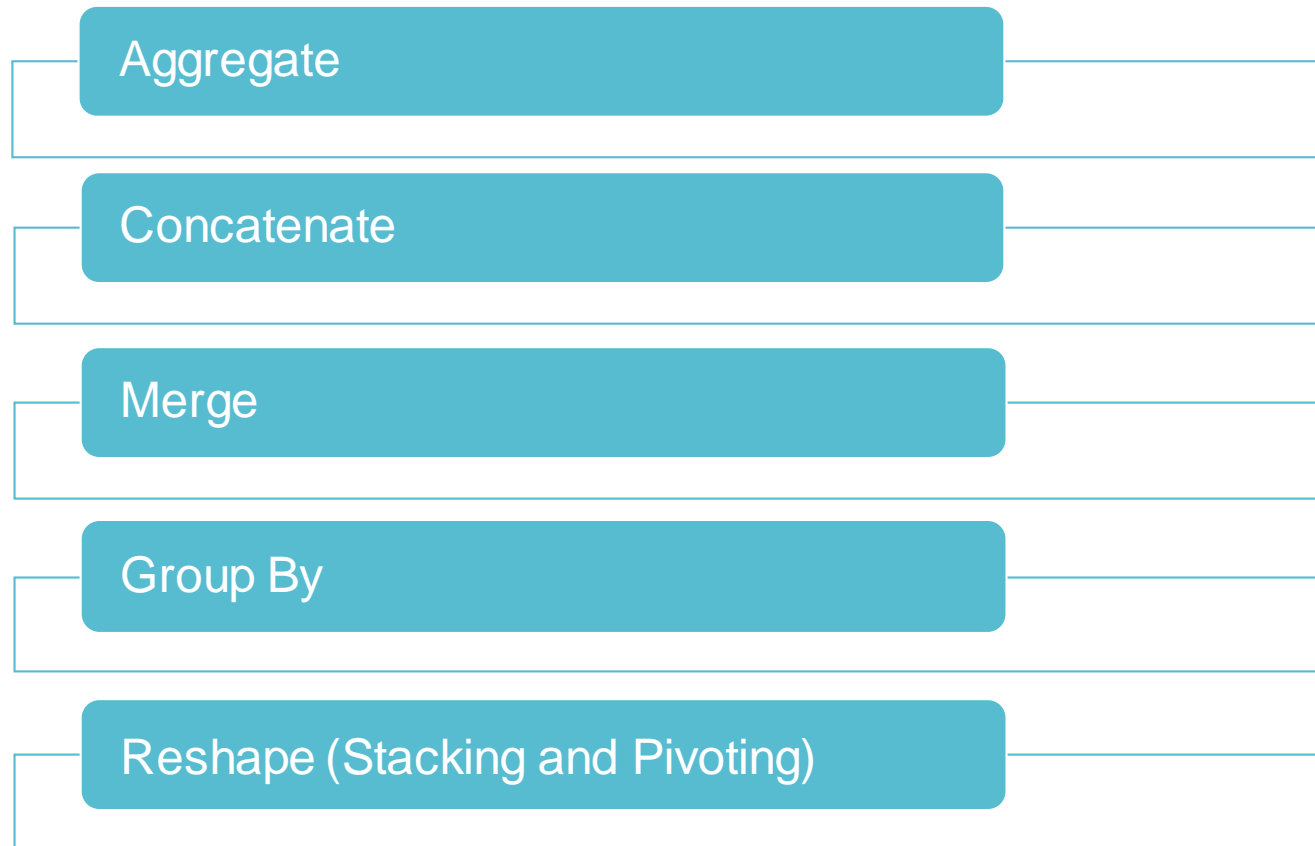
Two Main Data Types

In the ML world, data is nearly always split into two main groups: numerical and categorical



Data Wrangling

Data wrangling involves summarizing and processing the data in various formats like - **aggregation, concatenation, merging, grouping, reshaping** etc. for the purpose of **explanatory data analysis (EDA)** or getting them ready to be used with another set of data



Aggregate

- A single summary number gives insights into the nature of a potentially large dataset
- An essential piece of analysis of large data is efficient summarization like sum, mean, min, max, etc.
- `count()`, `first()`, `mean()`, `median()`, `std()`, `var()`, `sum()`, etc.

```
import pandas as pd
```

```
df = pd.read_csv('wine.csv', sep='\t')
```

Alcohol	Malic_acid	Color_Intensity	Wine_Class
12.79	2.67	1.26	C
13.45	3.70	1.46	C
11.64	2.06	1.35	B
13.52	3.17	0.55	C
11.56	2.05	1.87	B
13.05	3.86	1.62	B
12.47	1.52	3.28	B
12.51	1.73	1.48	B
13.17	5.19	1.55	C
11.66	1.88	1.15	B
13.11	1.90	1.56	C
13.05	1.65	1.44	A
13.29	1.97	1.66	A
14.06	1.63	2.10	A

DataFrame

```
df['Wine_Class'].count()
df['Malic_acid'].max()
df['Malic_acid'].std()
df[['Alcohol', 'Malic_acid']].mean()
df.agg({'Alcohol': 'mean', 'Malic_acid': 'median'})
df['Alcohol'][df['Wine_Class'] == 'A'].min()
```

Concatenate

Row-wise Concatenation

Alcohol	Malic_acid	Color_Intensity	Wine_Class	
51	13.83	1.65	2.29	A
55	13.56	1.73	2.45	A
24	13.50	1.81	1.66	A

Alcohol	Malic_acid	Color_Intensity	Wine_Class	
89	12.08	1.33	1.38	B
77	11.84	2.89	0.95	B
74	11.96	1.09	1.65	B
126	12.43	1.53	1.77	B

Alcohol	Malic_acid	Color_Intensity	Wine_Class	
51	13.83	1.65	2.29	A
55	13.56	1.73	2.45	A
24	13.50	1.81	1.66	A
89	12.08	1.33	1.38	B
77	11.84	2.89	0.95	B
74	11.96	1.09	1.65	B
126	12.43	1.53	1.77	B

```
pd.concat([df_A, df_B])
```

Fun facts:

- By default, Pandas concatenation takes place row-wise within DataFrame.
- Pandas concatenation preserves indices even if the result will have duplicate indices

Column-wise Concatenation

	Alcohol	Malic_acid	Color_Intensity	Wine_Class		Alcohol	Malic_acid	Color_Intensity	Wine_Class		
	51	13.83	1.65	2.29	A		89	12.08	1.33	1.38	B
	55	13.56	1.73	2.45	A		77	11.84	2.89	0.95	B
	24	13.50	1.81	1.66	A		74	11.96	1.09	1.65	B
							126	12.43	1.53	1.77	B

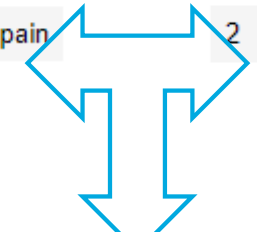
	Alcohol	Malic_acid	Color_Intensity	Wine_Class	Alcohol	Malic_acid	Color_Intensity	Wine_Class	
	24	13.50	1.81	1.66	A	NaN	NaN	NaN	NaN
	51	13.83	1.65	2.29	A	NaN	NaN	NaN	NaN
	55	13.56	1.73	2.45	A	NaN	NaN	NaN	NaN
	74	NaN	NaN	NaN	NaN	11.96	1.09	1.65	B
	77	NaN	NaN	NaN	NaN	11.84	2.89	0.95	B
	89	NaN	NaN	NaN	NaN	12.08	1.33	1.38	B
	126	NaN	NaN	NaN	NaN	12.43	1.53	1.77	B

```
pd.concat([df_A, df_B], axis = 1)
```

Merge

Pandas merge works similar as relational database “join” function.

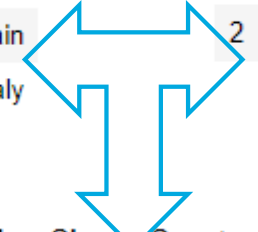
Wine_Class Country			Wine_Class Year		
0	A	France	0	A	2015
1	B	USA	1	B	2009
2	C	Spain	2	C	2010



Wine_Class Country Year			
0	A	France	2015
1	B	US	2009
2	C	Spain	2010

```
pd.merge(df1,df2,on='Wine_Class')
```

Wine_Class Country			Wine_Class Year		
0	A	France	0	A	2015
1	B	USA	1	B	2009
2	C	Spain	2	C	2010
3	D	Italy			



Wine_Class Country Year			
0	A	France	2015
1	B	USA	2009
2	C	Spain	2010
3	D	Italy	NaN

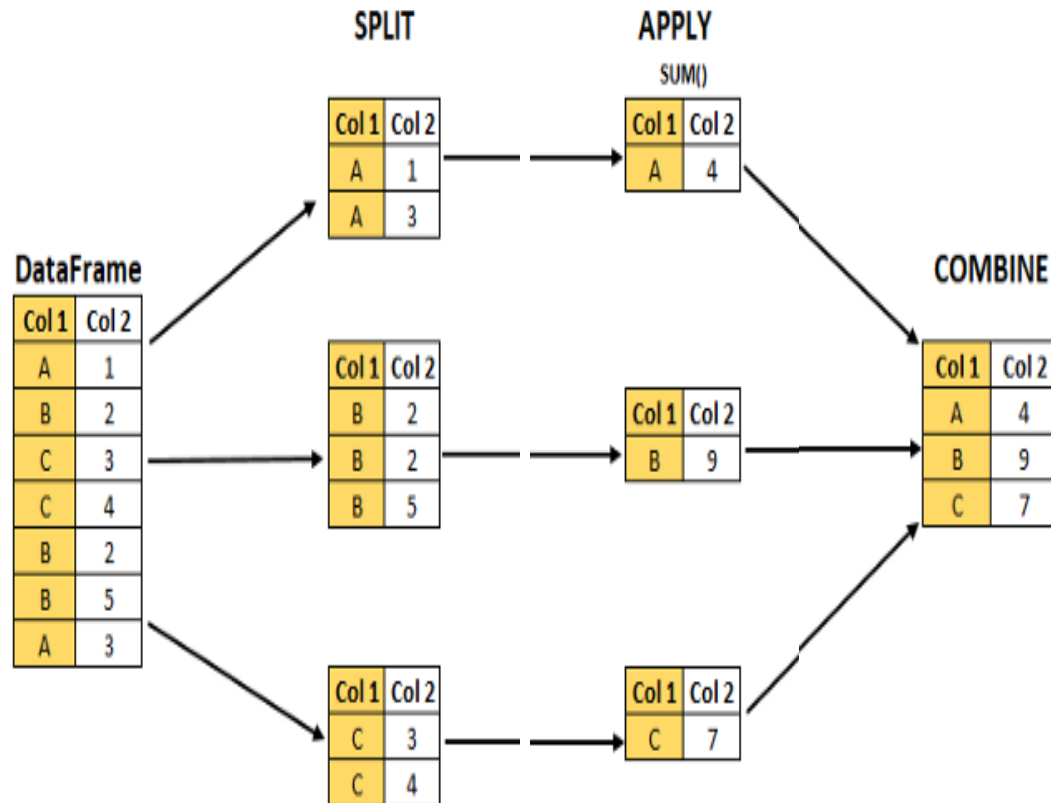
```
pd.merge(df1,df2,on='Wine_Class', how='left')
```

Fun fact:

- By default, Pandas merge performs data intersection in an “inner” join way.

Group By

The next level of data summarization which allows you quickly compute aggregates on subsets of data .



Python Code

```
df.groupby('Wine_Class').min()
```

Result

	Alcohol	Malic_acid	Color_Intensity
Wine_Class			
A	12.85	1.35	1.25
B	11.03	0.74	0.41
C	12.20	1.24	0.55

Python Code

```
df.groupby('Wine_Class').agg({'Alcohol':'max', 'Malic_acid':'mean'})
```

Result

	Alcohol	Malic_acid
Wine_Class		
A	14.83	2.010678
B	13.86	1.932676
C	14.34	3.333750

Reshaping (Un-Stacking)

- Un-Stacking – converts a dataset from a stacked multi-index to a simple two-dimensional representation

Alcohol	Malic_acid	Color_Intensity	Wine_Class	Country
13.28	1.64	1.36	A	France
12.77	2.39	0.64	C	Spain
13.58	2.58	1.54	C	Spain
14.23	1.71	2.29	A	France
12.81	2.31	0.83	C	Spain
11.76	2.68	1.05	B	USA
12.85	1.60	1.46	A	France
12.93	3.80	1.98	A	France
14.83	1.64	1.98	A	France
11.62	1.99	1.35	B	USA
13.73	1.50	2.38	A	France
12.52	2.43	1.22	B	USA

Python Code

```
df_C.groupby(['Wine_Class', 'Country'])['Alcohol'].mean()
```

Result

Wine_Class	Country	
A	France	13.744746
B	USA	12.278732
C	Spain	13.153750

Python Code

```
df_D.unstack()
```

Result

Country	France	Spain	USA
Wine_Class			
A	13.744746	NaN	NaN
B	NaN	NaN	12.278732
C	NaN	13.15375	NaN

Reshaping (Data Pivoting)

- Pivoting – think pivot table as a multidimensional version of group by aggregation. Split – Apply – Combine across a two-dimensional grid

Alcohol	Malic_acid	Color_Intensity	Wine_Class	Country
13.28	1.64	1.36	A	France
12.77	2.39	0.64	C	Spain
13.58	2.58	1.54	C	Spain
14.23	1.71	2.29	A	France
12.81	2.31	0.83	C	Spain
11.76	2.68	1.05	B	USA
12.85	1.60	1.46	A	France
12.93	3.80	1.98	A	France
14.83	1.64	1.98	A	France
11.62	1.99	1.35	B	USA
13.73	1.50	2.38	A	France
12.52	2.43	1.22	B	USA

Python Code

```
df_C.pivot_table('Alcohol', index='Wine_Class', columns='Country', aggfunc='mean')
```

Result

Country	France	Spain	USA
Wine_Class			
A	13.744746	NaN	NaN
B	NaN	NaN	12.278732
C	NaN	13.15375	NaN

Python Code

```
df_D.pivot_table('Alcohol', index="Wine_Class", columns=['Country', 'Year'], aggfunc = 'mean')
```

Result

Country	France	Spain	USA
Year	2015	2009	2009
Wine_Class			
A	13.744746	NaN	NaN
B	NaN	NaN	12.278732
C	NaN	13.15375	NaN

Data Transformation

Feature Scaling

Why feature scaling

- Highly varying in magnitudes, units and range.
- Features with high magnitudes will weigh in a lot more in the distance calculation than features with low magnitudes. To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.
- Deal with outliers

When to feature scaling:

- Rule of thumb Algorithm that **computes distance or assumes normality**, scale your features.
- K-means, K-nearest neighbor, gradient descent, linear discriminant analysis, PCA (since you want to find directions of maximizing the variance)), etc.

Data Transformation

Python Code

Feature Scaling

Standardization (z-score normalization): $z = \frac{x - \mu}{\sigma}$

- Rescaled to have the properties of a standard normal distribution.

Normalization:

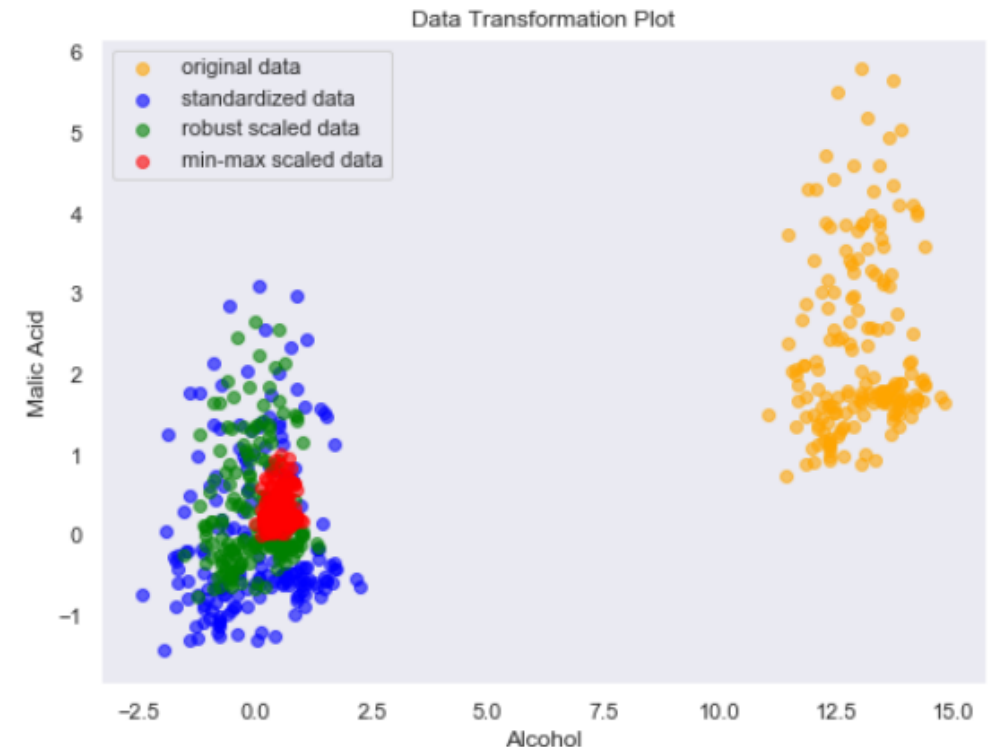
- Min-max scaling (0-1 range): $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$
 - In contrast to standardization we will end up with smaller SD which can suppress the effect of outliers. Often used in image processing
- Robust Scaling:
 - Subtract median and divided by interquartile range (75% value – 25% value)
 - Won't be influenced by large marginal outliers

```
from sklearn import preprocessing

std_scale = preprocessing.StandardScaler().fit(df[['Alcohol', 'Malic_acid']])
df_std = std_scale.transform(df[['Alcohol', 'Malic_acid']])

minmax_scale = preprocessing.MinMaxScaler().fit(df[['Alcohol', 'Malic_acid']])
df_minmax = minmax_scale.transform(df[['Alcohol', 'Malic_acid']])

robust_scale = preprocessing.RobustScaler().fit(df[['Alcohol', 'Malic_acid']])
df_robust = robust_scale.transform(df[['Alcohol', 'Malic_acid']])
```

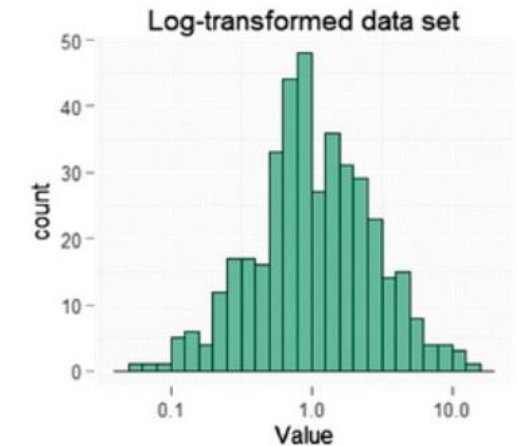
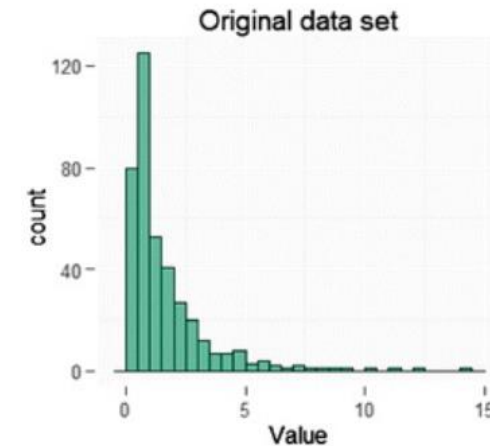


Data Transformation

Feature Scaling

Other useful transformations (Log, Square Root)

- Logarithm is a strong transformation with a major effect on distribution shape. It's commonly used for reducing right skewness and is often appropriate for measured variables. Cannot apply on zero or negative values.
- Square Root is a transformation with a moderate effect on distribution shape. It's weaker than logarithm and also used to reduce right skewness and also has the advantage of applying on zero values.



Intro to Data Pipeline

What is a pipeline

- Sequentially apply a list of data transformers and a final estimator.
- Intermediate steps of pipeline must implement fit and transform methods and the final estimator only needs to implement fit.

```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([('scaler',MinMaxScaler()),('rf',RandomForestClassifier())])

pipeline.fit(X_train , y_train)

pipeline.score(X_test, y_test)
```

Questions?