Dataset preparation II:

Data transformation

Mario Martin

CS-UPC

October 7, 2019

Recap

- We have seen that data for *Data mining* should be represented as a **table**.
- We have seen ways to represent non-tabular data into tables: Images, temporal series, documents, transactions, etc.
- We have seen how to inspect your data:
  - Visually: *Boxplots*, *Scatterplots*
  - Numerically: Correlation, regression
- And clean your raw data:
  - Find and impute *Missing Data*
  - Finding and correcting *errors*
  - Finding and considering *outliers*

Today

- Transformation of data:
  - Values in the table
  - Columns of the table
  - Rows of the table

Today

- Transformation of data:
  - **Values in the table**
  - Columns of the table
  - Rows of the table

Value transformation

Value transformation

- It means to change the values in the table
- Several reason to do that:
  - Data homogenization
  - Re-scaling of data
  - Change of distribution

Value transformation

- It means to change the values in the table
- Several reason to do that:
  - **Data homogenization**
  - Re-scaling of data
  - Change of distribution

Data homogenization

- We have seen that there are a lot of possible values for a column:
  - Numerical
  - Categorical
- Some Data mining algorithms only work with numerical features
- Different ways to transform Categorical Data into numerical:
  - Label Encoding
  - One-Hot encoding
  - Mean encoding

Ordinal (Label) encoding

- When categories have an implicit order
- Examples:
  - *Age*: Child, Young, Adult, Old
  - *Monthly Income*: "0-4k", "4-10k", "10k-20k", "$>$20k"

Ordinal (Label) encoding

- Example of transformation for *Age* attribute:

	...	*Age*	...
o_1	...	Child	...
o_2	...	Young	...
o_3	...	Adult	...
o_4	...	Child	...
o_5	...	Old	...
...	...	...	...

$\Longrightarrow$

	...	*Age*	...
o_1	...	1	...
o_2	...	2	...
o_3	...	3	...
o_4	...	1	...
o_5	...	4	...
...	...	...	...

- Coding in python and pandas:

```
Age_dict = {'Child':1, 'Young':2, 'Adult':3, 'Old':4}
df['Age'] = df.Age.map[Age_dict]
```

One-Hot encoding

- When categories have no order related to target and there are not a lot of categories (f.i. *color*)

One-Hot encoding

- When categories have no order related to target and there are not a lot of categories (f.i. *color*)
- Ordinal encoding is not right because it induces different distances between modalities

	...	*color*	...
o_1	...	red	...
o_2	...	blue	...
o_3	...	brown	...
o_4	...	red	...
o_5	...	green	...
...	...	...	...

$\not\Rightarrow$

	...	*color*	...
o_1	...	1	...
o_2	...	2	...
o_3	...	3	...
o_4	...	1	...
o_5	...	4	...
...	...	...	...

One-Hot encoding

- One-Hot encoding: Create *dummy* variables, one for modality

	...	*color*	...
o_1	...	red	...
o_2	...	blue	...
o_3	...	brown	...
o_4	...	red	...
o_5	...	green	...
...	...	...	...

$\Longrightarrow$

	...	*red?*	*blue?*	*brown?*	*green?*	...
o_1	...	1	0	0	0	...
o_2	...	0	1	0	0	...
o_3	...	0	0	1	0	...
o_4	...	1	0	0	0	...
o_5	...	0	0	0	1	...
...	...	...	...	...	...	...

- When too many modalities you can group some values to reduce number of modalities,
- Always for binary categories (only one column 0-1)
- Coding in `python` and `pandas`:

```
df2 = pd.get_dummies(df, columns =['color'])
```

Mean encoding

- When goal is to predict a label, modalities can be transformed according to joint appearance with labels

	...	*color*	...	*label*
o_1	...	red	...	yes
o_2	...	blue	...	yes
o_3	...	brown	...	no
o_4	...	red	...	no
o_5	...	green	...	yes
...	...	...	...	...

$\Longrightarrow$

	...	*color*	...	*label*
o_1	...	0.5	...	yes
o_2	...	1	...	yes
o_3	...	0	...	no
o_4	...	0.5	...	no
o_5	...	1	...	yes
...	...	...	...	...

Categorical to numeric

- Not the whole story. A lot of other methods depending on the data.

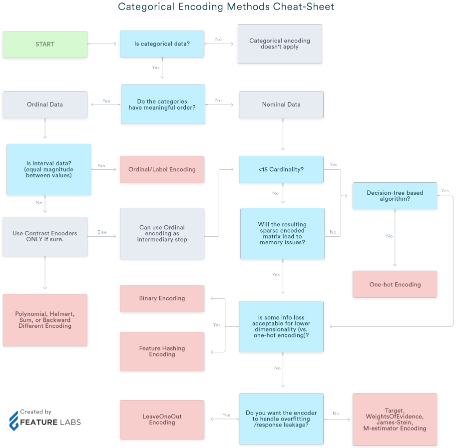


Figure: source

Value transformation

- It means to change the values in the table
- Several reason to do that:
  - Data homogenization
  - **Re-scaling of data**
  - Change of distribution

Transforming values of columns

- All variables should be in the same range to avoid bias in computation of distances.
- Mandatory for some algorithms like *k-NN*, *SVM* or Neural Networks
- Can be seen as change of units
- Common procedures: Normalization or Standardization
- Caveat: Different assumptions!

Transforming values of columns

- Normalization $[0-1]$:

$$v(i) = \frac{v(i) - min(v)}{max(v) - min(v)}$$

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(data)
d2 = scaler.transform(data)
```

- Standardization $N(0,1)$:

$$v(i) = \frac{v(i) - mean(v)}{std(v)}$$

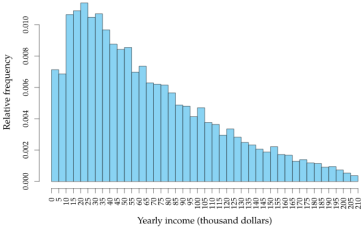
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(data)
d2 = scaler.transform(data)
```

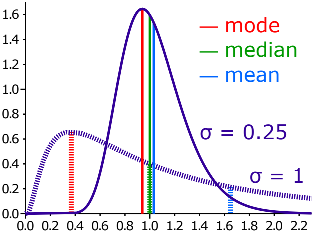
Value transformation

- It means to change the values in the table
- Several reason to do that:
  - Data homogenization
  - Re-scaling of data
  - **Change of distribution**

Value transformation

- In some cases distribution of data has heavy tails

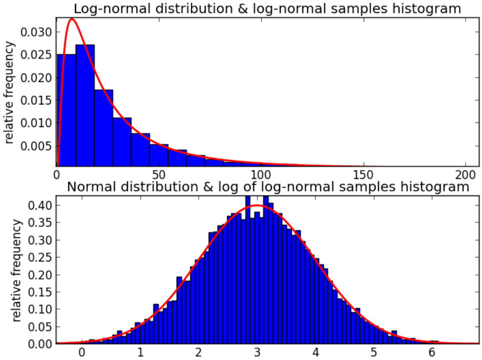




- In these cases, we have problems distinguishing between most of data.
- Data appear as *outliers*
- Solutions:
  - Apply Log of column (log-normalization)
  - Replace values by quantiles

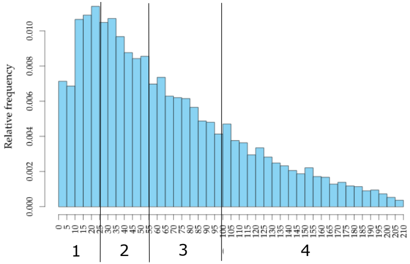
Value transformation

- Effect of log-normalization:



Value transformation

- Replace value according the position in the quantile
- Effect of quantiles separation when using 4 values



- You can use a higher degree of quantization if needed

Value transformation

Today

- Transformation of data:
  - Values in the table
  - **Columns of the table**
  - Rows of the table