## Preprocessing data

For highly-skewed feature distributions such as 'capital-gain' and 'capital-loss', it is common practice to apply a [logarithmic transformation](https://en.wikipedia.org/wiki/Data_transformation_(statistics)) on the data so that the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers. Care must be taken when applying this transformation however: The logarithm of 0 is undefined, so we must translate the values by a small amount above 0 to apply the the logarithm successfully.

# Log-transform the skewed features

skewed = ['capital-gain', 'capital-loss']

features\_raw[skewed] = data[skewed].apply(lambda x: np.log(x + 1))

## Normalizing Numerical Features

# Import sklearn.preprocessing.StandardScaler

from sklearn.preprocessing import MinMaxScaler

# Initialize a scaler, then apply it to the features

scaler = MinMaxScaler()

numerical = ['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']

features\_raw[numerical] = scaler.fit\_transform(data[numerical])

## Convert Categorical Variables

* Use [pandas.get\_dummies()](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html?highlight=get_dummies#pandas.get_dummies) to perform one-hot encoding on the 'features\_raw' data.
* Convert the target label 'income\_raw' to numerical entries.
  + Set records with "<=50K" to 0 and records with ">50K" to 1.

# One-hot encode the 'features\_raw' data using pandas.get\_dummies()

features = pd.get\_dummies(features\_raw)

income = pd.factorize(pd.Categorical(income\_raw, categories = ['<=50K','>50K']))[0]

#### How to reverse the labels?