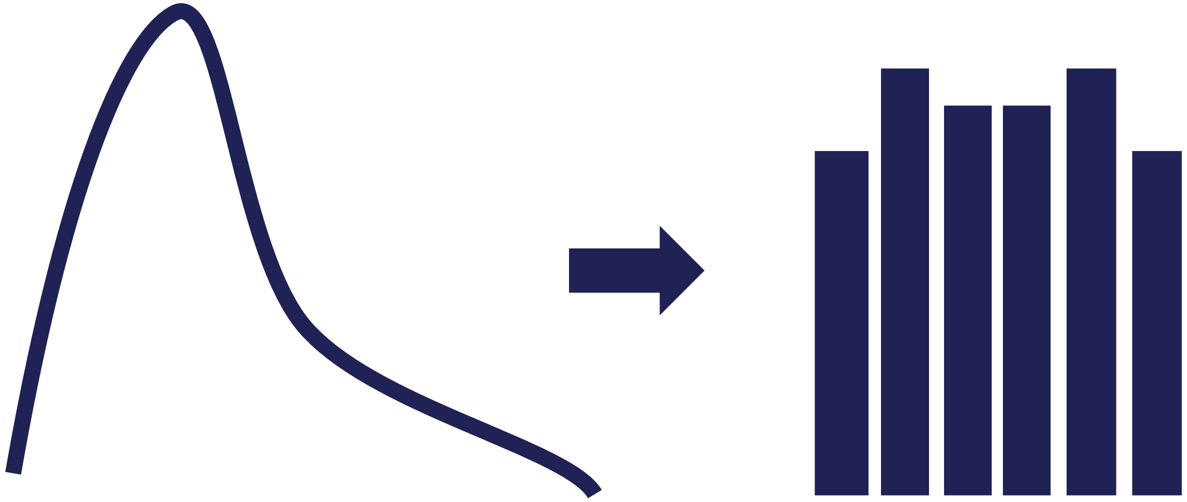
Train In Data

Wrap up

Discretization



Continuous variable

Data sorted in intervals

Discretization or binning consists in sorting continuous variables into discrete intervals.

Discretization: why use it?

- Improve model performance.
- Reduce training time.
- Improve value spread.
- Mitigate the effect of outliers.
- Create simpler features (for us humans).

Limitations of discretization

- **Discretization can also lead to a loss of information.**
- For example by combining values that are strongly associated with different classes (target values) into the same bin.

Discretization task

The aim of the discretization algorithm is to find the minimum number of intervals without incurring in information loss.

Discretization methods

Unsupervised

- Equal-width
- Equal-frequency
- Arbitrary
- Binarization
- K means

Given the number of intervals, they find the interval limits.

Supervised

- Decision Trees
- Chi-Merge
- CAIM

Find the optimal number of bins and their limits.

Basic discretization methods

Unsupervised

- **Equal-width**
- **Equal-frequency**
- **Arbitrary**
- Binarization
- K means

Given the number of intervals, they find the interval limits.

Supervised

- Decision Trees
- Chi-Merge
- CAIM

Find the optimal number of bins and their limits.

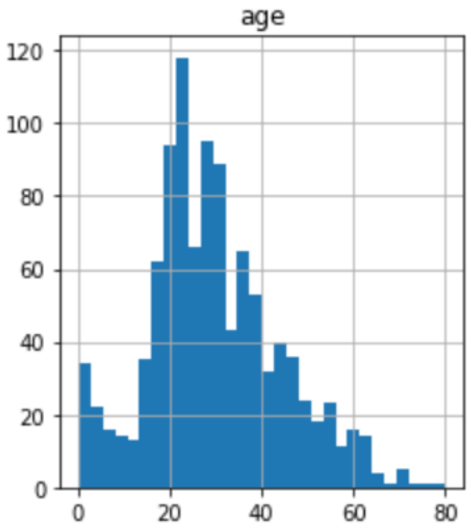
Discretization effects on distribution

- Equal-width discretization preserves the original variable shape.

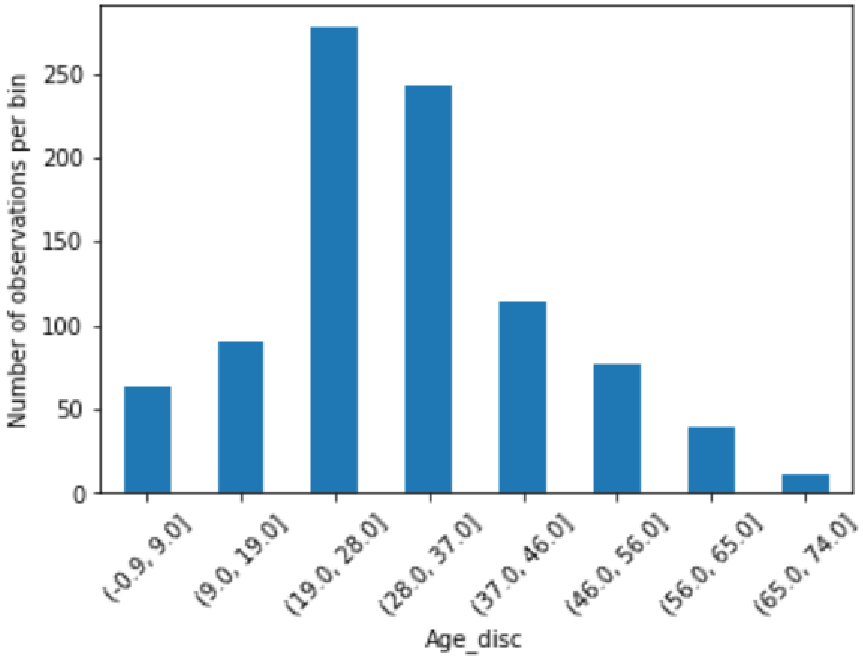
- Equal-frequency discretization returns an homogeneous value distribution.

- Arbitrary discretization may or may not change the shape of the original variable depending on how we create the intervals.

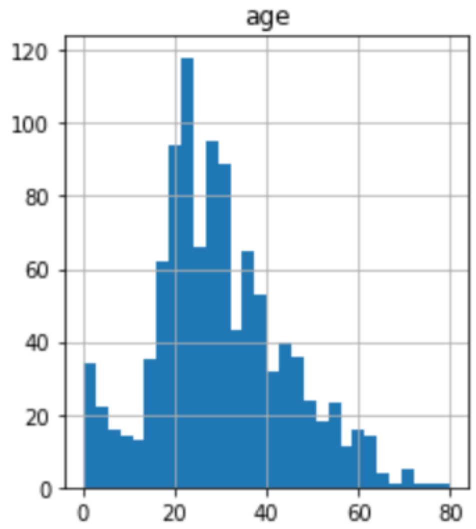
Discretization effects on distribution



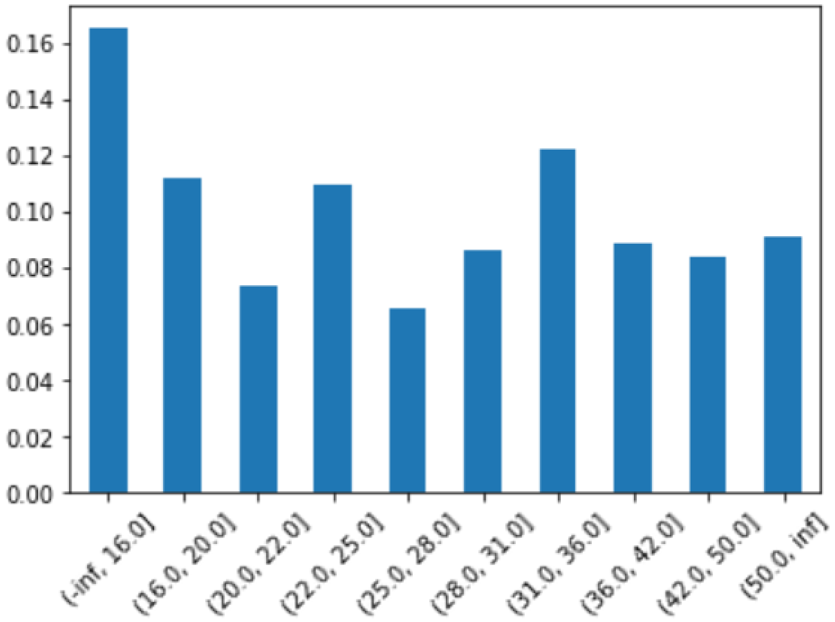
Age after equal-width discretization



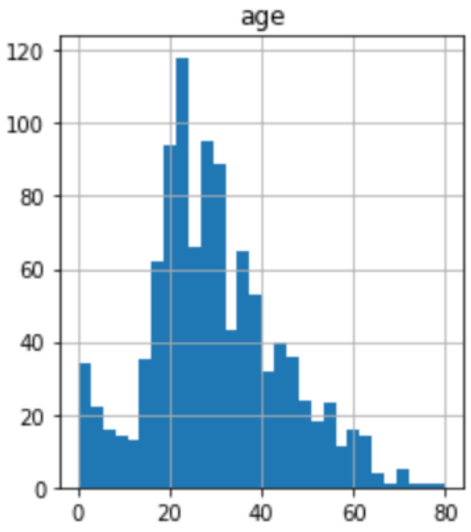
Discretization effects on distribution



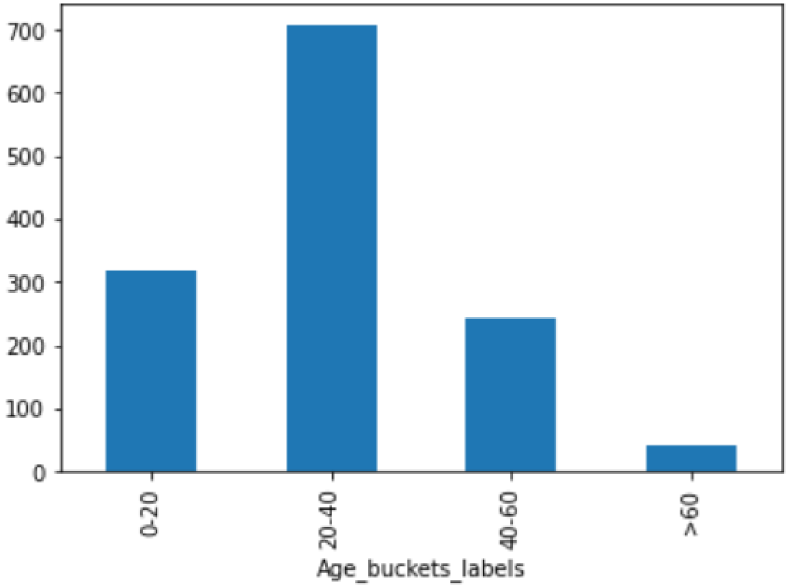
Age after equal-frequency discretization



Discretization effects on distribution



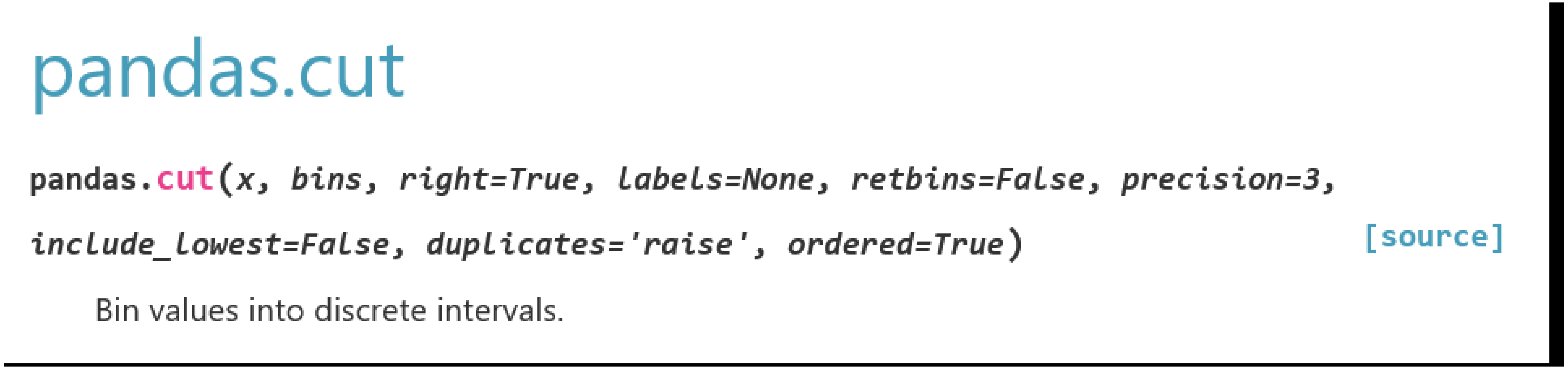
Age after arbitrary discretization



Discretization + encoding

- After discretization, we commonly use the intervals as categories.
- If the intervals are integers, it is the equivalent of ordinal encoding.
- But, we can follow up with any encoding that we like.

Discretization with pandas

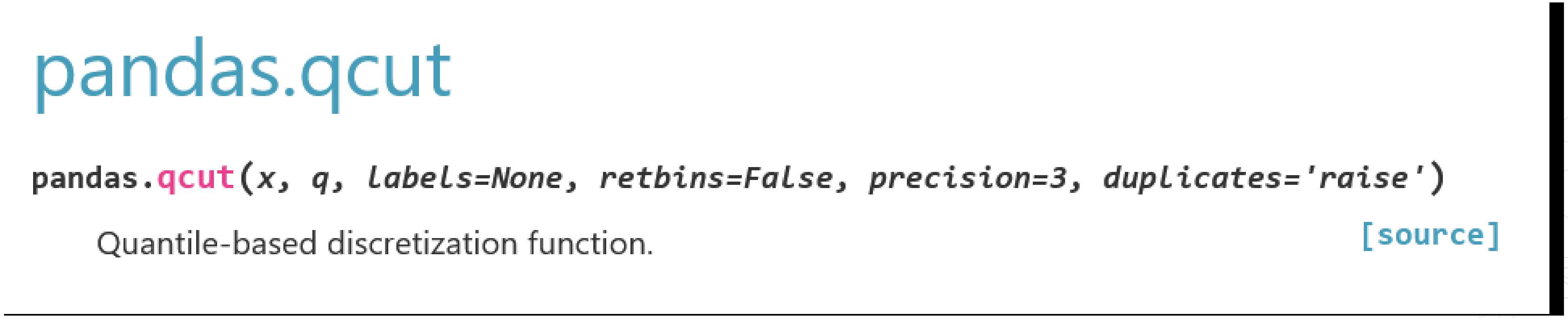


pandas.cut

```
pandas.cut(x, bins, right=True, labels=None, retbins=False, precision=3,
include_lowest=False, duplicates='raise', ordered=True)
```

[source]

Bin values into discrete intervals.



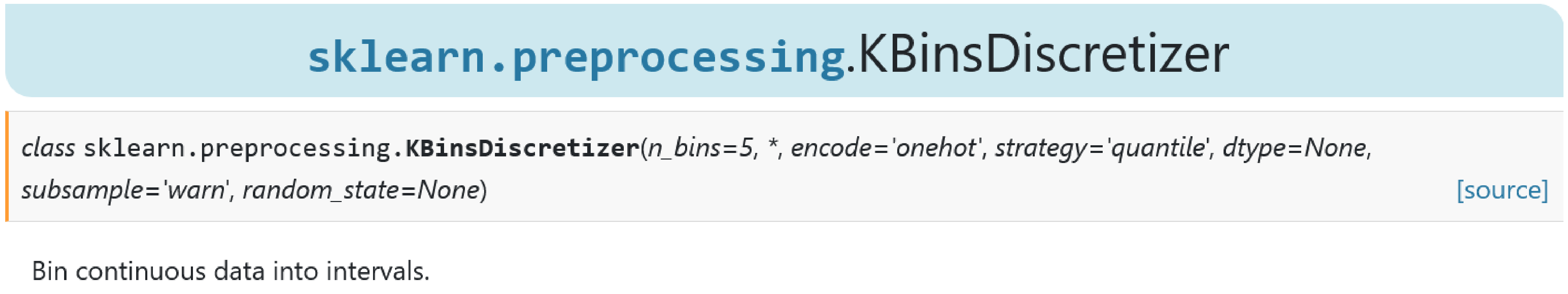
pandas.qcut

```
pandas.qcut(x, q, labels=None, retbins=False, precision=3, duplicates='raise')
```

[source]

Quantile-based discretization function.

Discretization with sklearn

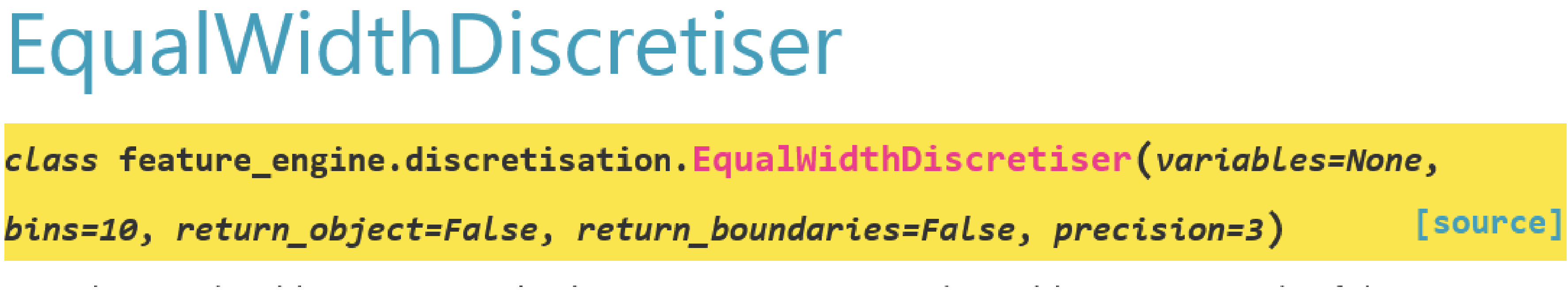


sklearn.preprocessing.KBinsDiscretizer

class `sklearn.preprocessing.`**`KBinsDiscretizer`**(*n_bins=5*, ***, *encode='onehot'*, *strategy='quantile'*, *dtype=None*, *subsample='warn'*, *random_state=None*) [source]

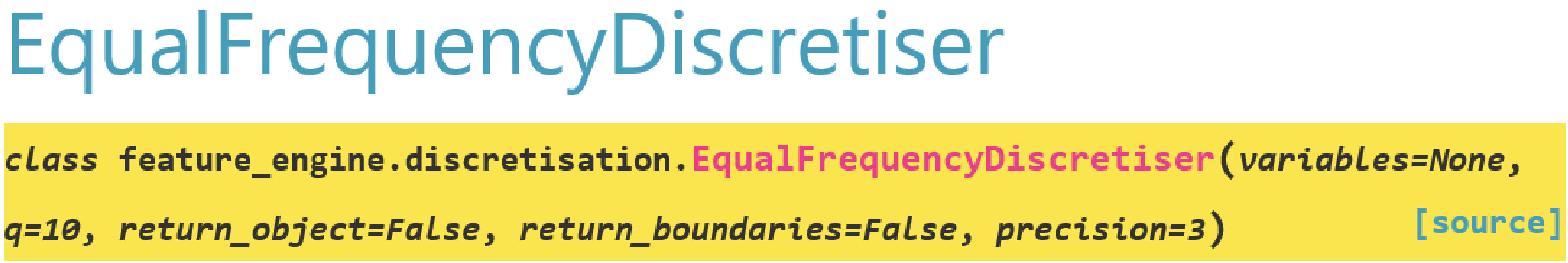
Bin continuous data into intervals.

Discretization with Feature-engine



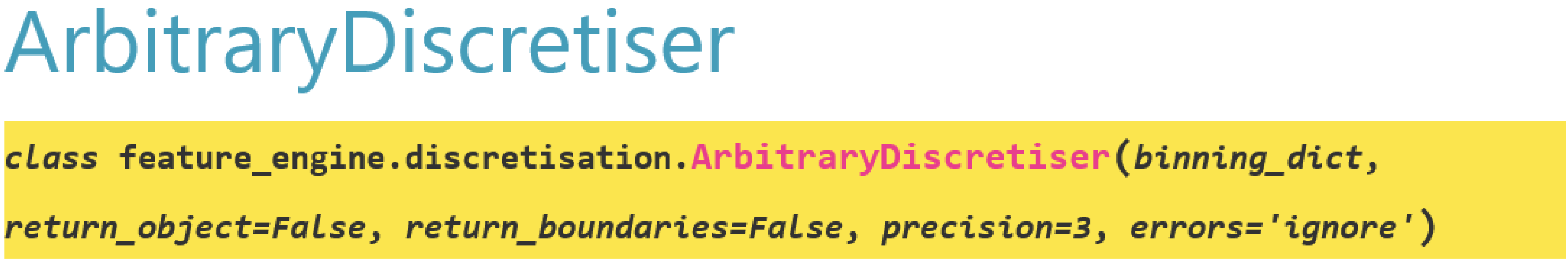
EqualWidthDiscretiser

```
class feature_engine.discretisation.EqualWidthDiscretiser(variables=None,
bins=10, return_object=False, return_boundaries=False, precision=3)      [source]
```



EqualFrequencyDiscretiser

```
class feature_engine.discretisation.EqualFrequencyDiscretiser(variables=None,
q=10, return_object=False, return_boundaries=False, precision=3)      [source]
```



ArbitraryDiscretiser

```
class feature_engine.discretisation.ArbitraryDiscretiser(binning_dict,
return_object=False, return_boundaries=False, precision=3, errors='ignore')
```

Accompanying Jupyter Notebook

- How to perform discretization:
  - Pandas
  - Scikit-learn
  - Feature-engine

Train In Data

THANK YOU

www.trainindata.com