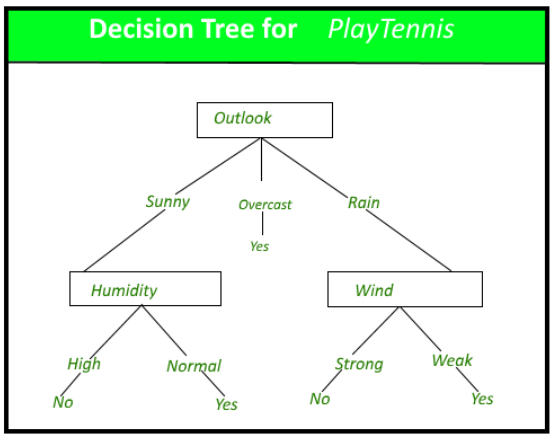
Discretization

- This is a process of converting continuous data into a set of data intervals.
- Continuous attribute values are substituted by small interval labels.
- This makes the data easier to study and analyze.
- If a continuous attribute is handled by a data mining task, then its discrete values can be replaced by constant quality attributes.
- This improves the efficiency of the task.
- This method is also called data reduction mechanism as it transforms a large dataset into a set of categorical data.
- Discretization also uses decision tree-based algorithms to produce short, compact and accurate results when using discrete values.



- It is a process of transforming continuous data into set of small intervals. Most Data Mining activities in the real world require continuous attributes. Yet many of the existing data mining frameworks are unable to handle these attributes.

- Also, even if a data mining task can manage a continuous attribute, it can significantly improve its efficiency by replacing a constant quality attribute with its discrete values.

    - For **example**, (1-10, 11-20) (age:- young, middle age, senior).

Transformation thru Discretization (Binning)

Data-mining applications often involve quantitative data (Numerical or Continuous). However, learning from quantitative data is often less effective and less efficient than learning from qualitative data (Categorical or Discrete). Discretization addresses this issue by transforming quantitative data into qualitative data.

- Binning, also known as quantization is used for transforming continuous numeric features into discrete ones (categories).

- Numerical input variables may have a highly skewed or non-standard distribution.
    - This could be caused by outliers in the data, multi-modal distributions, highly exponential distributions, and more.
- Many machine learning algorithms prefer or perform better when numerical input variables have a standard probability distribution.
- Discretization is the process through which we can transform continuous variables, models or functions into a discrete form.
- We do this by creating a set of contiguous intervals (or bins) that go across the range of our desired variable/model/function.
    - Discretization is also known as Binning

Why Discretization is Important

Mathematical problems with continuous data have an infinite number of degrees of freedom (DoF) but our calculations cannot go on forever. Data Scientists require using Discretization for a number of reasons. Many of the top contributions on Kaggle use discretization for some of the following reasons:

1. Fits the problem statement

- Often, it is easier to understand continuous data (such as weight) when divided and stored into meaningful categories or groups.
    - For example, we can divide a continuous variable, weight, and store it in the following groups :
      Under 100 lbs *(light),* ***between 140–160 lbs*** *(mid), and* ***over 200 lbs*** *(heavy)*
- This will make the structure more useful as no objective difference between variables falling under the same category (weight class in our case).
    - In our example, weights of *85 lbs* and *56 lbs* convey the same information (the object is light). Therefore, discretization helps make our data easier to understand if it fits the problem statement.

2. Interprets features

- Continuous features have a smaller chance of correlating with the target variable due to infinite degrees of freedom and may have a complex non-linear relationship. Thus, it may be harder to interpret an such a function.
- After discretizing a variable, groups corresponding to the target can be interpreted.

3. Incompatible with models/methods

- Certain models may be incompatible with continuous data, for example, alternative decision-tree models such as a Random-Forest model is not suitable for continuous features.
- Feature engineering methods, for example any entropy-based methods may not work with continuous data, thus we would discretize variables to work with different models & methods.

Signal-to-Noise Ratio

- When we discretize a model, we are fitting it to bins and reducing the impact of small fluctuation in the data. Often, we would consider small fluctuations as noise. We can reduce this noise through discretization.
- This is the process of "smoothing", wherein each bin smoothens fluctuations, thus reducing noise in the data.

Approaches to Discretization

- Unsupervised:
    - Fixed width binning
        - We manually create fix width bins based on some rules and domain knowledge.
        - on the basis of some domain knowledge, rules or constraints. Just like the name indicates, in fixed-width binning, we have specific fixed widths for each of the bins which are usually pre-defined by the user analyzing the data. Each bin has a pre-fixed range of values which should be assigned to that bin
    - Equal-Width
    - Equal-Frequency
    - Statistical binning or Binning based on Rounding (Divide and round/floor/ceiling)
    - Adaptive Binning

- Based on the statistical information of the data boundary is decided

  — K-Means
  — Quantile based

- Supervised:
  — Decision Trees

Equal Width Bin:

In Equal width, we divide the data in equal widths.

$$width=(max-min)/N$$

- N is user defined

- Thus, i^{th} interval range will be $[A + (i-1)w, A + iw]$ where i = 1, 2, 3…..N

Example: Convert the list into 3 Equal Width Bins:
List = 5, 10, 11, 13, 15, 35, 50 ,55, 72, 92, 204, 215
Result: 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3
Width = 215-5/3 = 210/3 = 70

Bin 1: 5 to 70+5=75
Bin2 : 75 to 145 (75+70)
Bin3: 145 to 215 (145+70)

Result:
bin1: 5,10,11,13,15,35,50,55,72
bin2: 92
bin3: 204,215

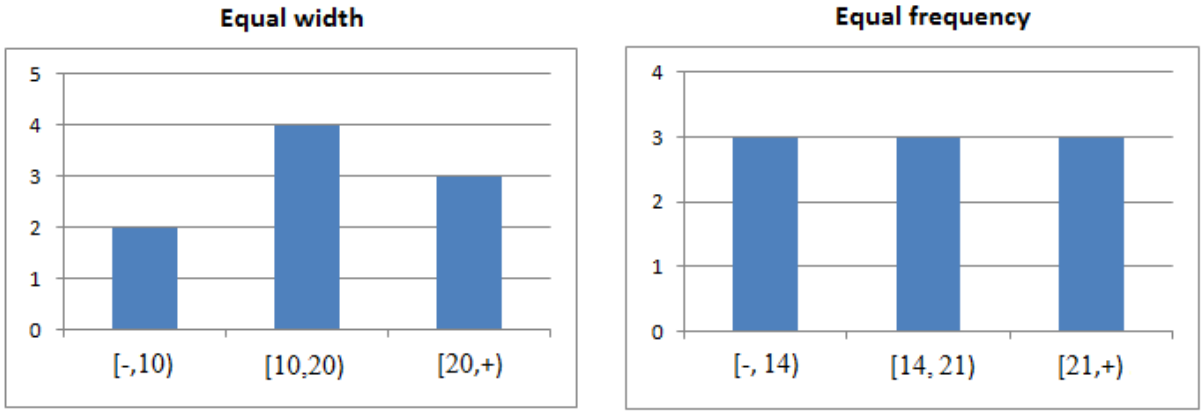
Challenges:
- How to define N?
- To create the sorted order list
- Skewed data cannot be handled well by this method

2- Equal Frequency Binning

The algorithm divides the data into *k* groups which each group contains approximately same number of values. For the both methods, the best way of determining k is by looking at the histogram and try different intervals or groups.

Challenge:
- Equal frequency may not be possible due to repeated values.

- **Data :** 0, 4, 12, 16, 16, 18, 24, 26, 28
- **Equal width**
  - Bin 1: 0, 4 [-,10)
  - Bin 2: 12, 16, 16, 18 [10,20)
  - Bin 3: 24, 26, 28 [20,+)
- **Equal frequency**
  - Bin 1: 0, 4, 12 [-, 14)
  - Bin 2: 16, 16, 18 [14, 21)
  - Bin 3: 24, 26, 28 [21,+)



The drawback in using fixed-width binning is that

- due to manually deciding the bin ranges, we can end up with irregular bins which are not uniform based on the number of data points or values which fall in each bin.
- Some of the bins might be densely populated and some of them might be sparsely populated or even empty!

Adaptive Binning

- In Fixed-Width Binning, bin ranges are manually decided. So, we usually end up in creating irregular bins which are not uniform based on the number of data points or values which fall under each bin. Some of the bins might be densely populated and some of them might be sparsely populated or even empty.
    - For example, bins 0, 5 and 8 are empty in our case.

- In Adaptive Binning, data distribution itself decides bin ranges for itself. No manual intervention is required. So, the bins which are created are uniform in terms of number of data points in it.

Quantile based binning

- It is a good strategy to use for adaptive binning.

- Quantiles are specific values or cut-points which help in partitioning the continuous valued distribution of a specific numeric field into discrete contiguous bins or intervals.

- Thus, **q-Quantiles** help in partitioning a numeric attribute into q equal partitions.
- Popular examples of quantiles include the *2-Quantile* known as the *median* which divides the data distribution into two equal bins,
- *4-Quantiles* known as the *quartiles* which divide the data into 4 equal bins [used in Box plot]
- *10-Quantiles* also known as the *deciles* which create 10 equal width bins. Let's now look at the data distribution for the developer Income field.

  Note :

- The algorithm assumes that the data quantiles are (mostly) unique and they divide *N* observations into *k* groups that each have approximately *N/k* observations.
- The repeated value can occupy more than one quantile value
    - If there are more than *N/k* repeated values,
    - if a particular value is repeated more than 2*N/k* times
- For example,
    - The diastolic blood pressure data is given with 5209 observations
    - It is to be divided into 10 groups that each have approximately 10% of the data.
    - The value 80 appears in 711^{th} position (which is about 14% of the data)
    - His value will reside in 2^{nd} quartile

    - If 80 is repeated from 711^{th} position to 1500^{th} position in such a way that it is appeared in 2^{nd} and 3^{rd} both quartiles

- If you do not want to split the value 80 across two bins, merge these bins and it will result in nine bins instead 10. (https://blogs.sas.com/content/iml/2014/11/05/binning-quantiles-rounded-data.html#prettyPhoto)

Case Study

- Data set: **2016 FreeCodeCamp Developer\Coder survey**
    - about various attributes pertaining to coders and software developers.

	ID.x	EmploymentField	Age	Income
0	cef35615d61b202f1dc794ef2746df14	office and administrative support	28.0	32000.0
1	323e5a113644d18185c743c241407754	food and beverage	22.0	15000.0
2	b29a1027e5cd062e654a63764157461d	finance	19.0	48000.0
3	04a11e4bcb573a1261eb0d9948d32637	arts, entertainment, sports, or media	26.0	43000.0
4	9368291c93d5d5f5c8cdb1a575e18bec	education	20.0	6000.0

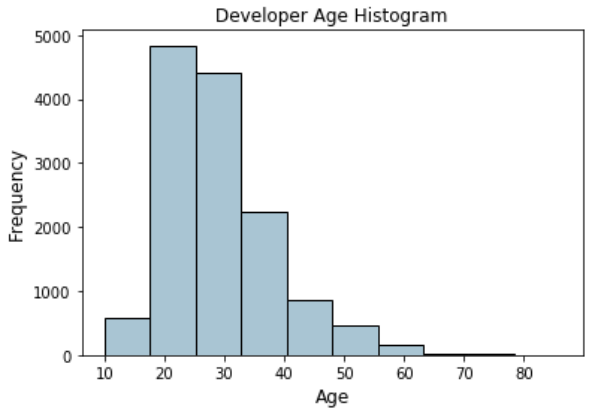
Sample attributes from the FCC coder survey dataset

- The ID.x variable is basically a unique identifier for each coder\developer who took the survey and the other fields are pretty self-explanatory.

Fixed-Width Binning

- Just like the name indicates, in fixed-width binning, we have specific fixed widths for each of the bins which are usually pre-defined by the user analyzing the data.
- Each bin has a pre-fixed range of values which should be assigned to that bin on the basis of some domain knowledge, rules or constraints.
- Binning based on rounding is one of the ways, where you can use the rounding operation which we discussed earlier to bin raw values.

- **Feature:Age** frequency distribution shown in the histogram indicate slightly right sqewed values (Developers are of less age)



Histogram depicting developer age distribution

We will now assign these raw age values into specific bins based on the following scheme

```
Age Range: Bin
---------------
 0 -  9  : 0
10 - 19  : 1
20 - 29  : 2
30 - 39  : 3
40 - 49  : 4
50 - 59  : 5
60 - 69  : 6
... and so on
```

Age Bin = Floor (Age/10)

	ID.x	Age	Age_bin_round
1071	6a02aa4618c99fdb3e24de522a099431	17.0	1.0
1072	f0e5e47278c5f248fe861c5f7214c07a	38.0	3.0
1073	6e14f6d0779b7e424fa3fdd9e4bd3bf9	21.0	2.0
1074	c2654c07dc929cdf3dad4d1aec4ffbb3	53.0	5.0
1075	f07449fc9339b2e57703ec7886232523	35.0	3.0

Binning by rounding

But what if we need more flexibility? What if we want to decide and fix the bin widths based on our own rules\logic? Binning based on custom ranges will help us achieve this.

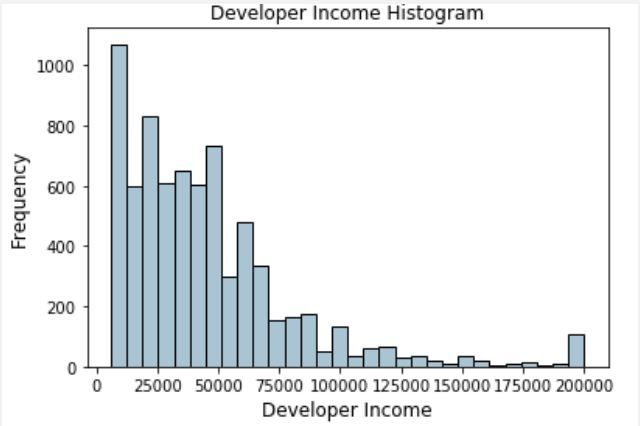
Let's define some custom age ranges for binning developer ages using the following scheme.

```
Age Range : Bin
---------------
 0 -  15  : 1
16 -  30  : 2
31 -  45  : 3
46 -  60  : 4
61 -  75  : 5
75 - 100  : 6
```

	ID.x	Age	Age_bin_round	Age_bin_custom_range	Age_bin_custom_label
1071	6a02aa4618c99fdb3e24de522a099431	17.0	1.0	(15, 30]	2
1072	f0e5e47278c5f248fe861c5f7214c07a	38.0	3.0	(30, 45]	3
1073	6e14f6d0779b7e424fa3fdd9e4bd3bf9	21.0	2.0	(15, 30]	2
1074	c2654c07dc929cdf3dad4d1aec4ffbb3	53.0	5.0	(45, 60]	4
1075	f07449fc9339b2e57703ec7886232523	35.0	3.0	(30, 45]	3

Custom binning scheme for developer ages

Adaptive Binning



Histogram depicting developer income distribution

- The above distribution depicts a right skew in the income with lesser developers earning more money and vice versa.
- Let's take a *4-Quantile* or a *quartile* based adaptive binning scheme.

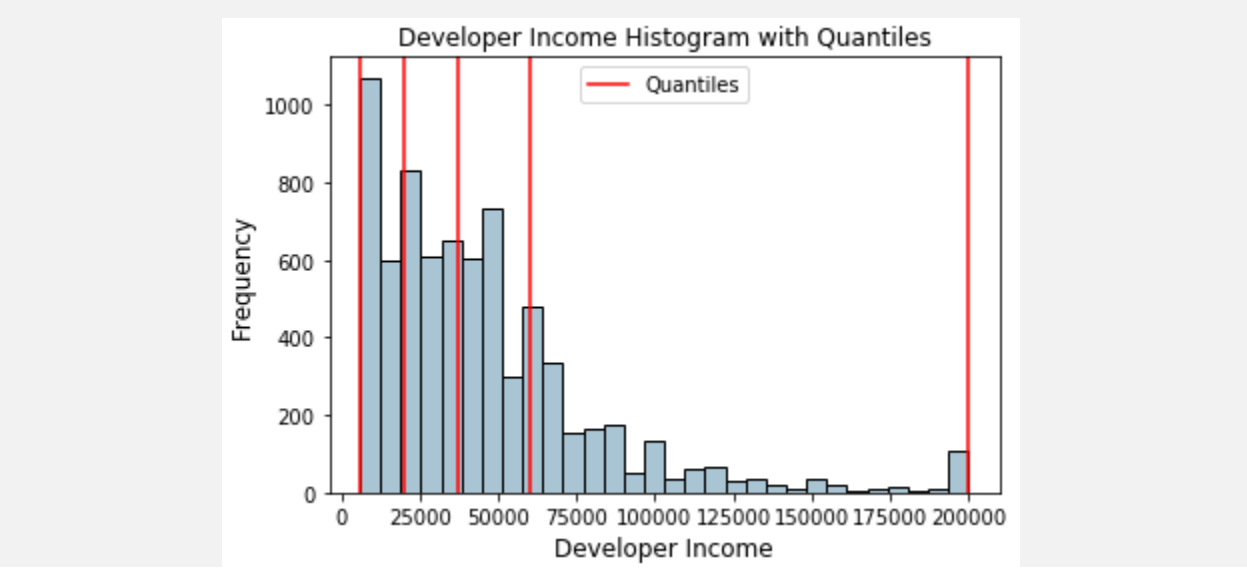
quartiles obtained :

- Income range [0..200000]
- quantile_list = [0, .25, .5, .75, 1.]

Quantile	Value
0.00	6000.0 (smallest value)
0.25	20000.0
0.50	37000.0
0.75	60000.0
1.00	200000.0

	ID.x	Age	Income	Income_quantile_range	Income_quantile_label
4	9368291c93d5d5f5c8cdb1a575e18bec	20.0	6000.0	(5999.999, 20000.0]	0-25Q
5	dd0e77eab9270e4b67c19b0d6bbf621b	34.0	40000.0	(37000.0, 60000.0]	50-75Q
6	7599c0aa0419b59fd11ffede98a3665d	23.0	32000.0	(20000.0, 37000.0]	25-50Q
7	6dff182db452487f07a47596f314bddc	35.0	40000.0	(37000.0, 60000.0]	50-75Q
8	9dc233f8ed1c6eb2432672ab4bb39249	33.0	80000.0	(60000.0, 200000.0]	75-100Q

Quantile based bin ranges and labels for developer incomes



Histogram depicting developer income distribution with quartile values

- The red lines in the distribution above depict the quartile values and our potential bins.
- This will result in uniform distribution of the values

Kmeans clustering binning

K-means clustering-based binning

For the k-means clustering, we implement the proposed technique in following steps:

1. Initial groups are obtained using the k-means algorithm on the first lot (Training Data).
2. The 1st instance from the second lot (Test Data set) is obtained and the Euclidean distances between this vector and centroid of each group obtained from the first phase is calculated. The device is assigned to the group to which it is nearest.
3. To follow the k-means algorithm in this case,
    - The average sum of distance of this point and each point of that group to the centroid should be calculated.
        - This process will be repeated for every group and the device should be assigned to the group, which gives the minimum average sum.
        - Clearly, such an approach will increase the complexity of binning process to a high degree and make it unusable in production environment.
    - Another approach for the best approximation is : the device is assigned to whichever groups centroid it is closest.
4. The centroid of the group to which the device is added, is recalculated and updated for that group.
5. This procedure is carried out for all the Data of the new Data_set

Limitations:

- K-Means doesn't improve the value spread
- It can handle outliers, however a centroid bias may exist.
- Can be combined with categorical encoding

Extra Reading : https://www.cs.colostate.edu/~malaiya/turakhia_paper.pdf

Tree based

- Decision Tree does not improve the value spread
- It can handle outliers well as trees are robust to outliers.
- Creates monotonic relationships

Extra reading : https://hiweller.github.io/colordistance/binning-methods.html