Chapters 2&3: Data Preparation

MATH2319

1. **Designing the Analytics Base Table**
2. **Identifying Data Quality Issues**
3. **Handling Data Quality Issues**
   - Handling Missing Values
   - Handling Outliers
4. **Data Preparation**
   - Normalization
   - Binning
   - Sampling
5. **Best Practices with Python**

Case Study: Car Insurance Fraud

In spite of having a fraud investigation team that investigates up to 30% of all claims made, a car insurance company is still losing too much money due to fraudulent claims.

- *Data Requirements:* A large collection of historical claims marked as *'fraudulent'* and *'non-fraudulent'*.

Designing the Analytics Base Table

- The basic structure in which we capture historical datasets is the **analytics base table** (**ABT**)

Descriptive Features						Target Feature
----	----	----	----	----	----	----
----	----	----	----	----	----	----
----	----	----	----	----	----	----
----	----	----	----	----	----	----
----	----	----	----	----	----	----

Figure: The general structure of an **analytics base table**—descriptive features and a target feature. Usually we will have **one-row-per-subject**.

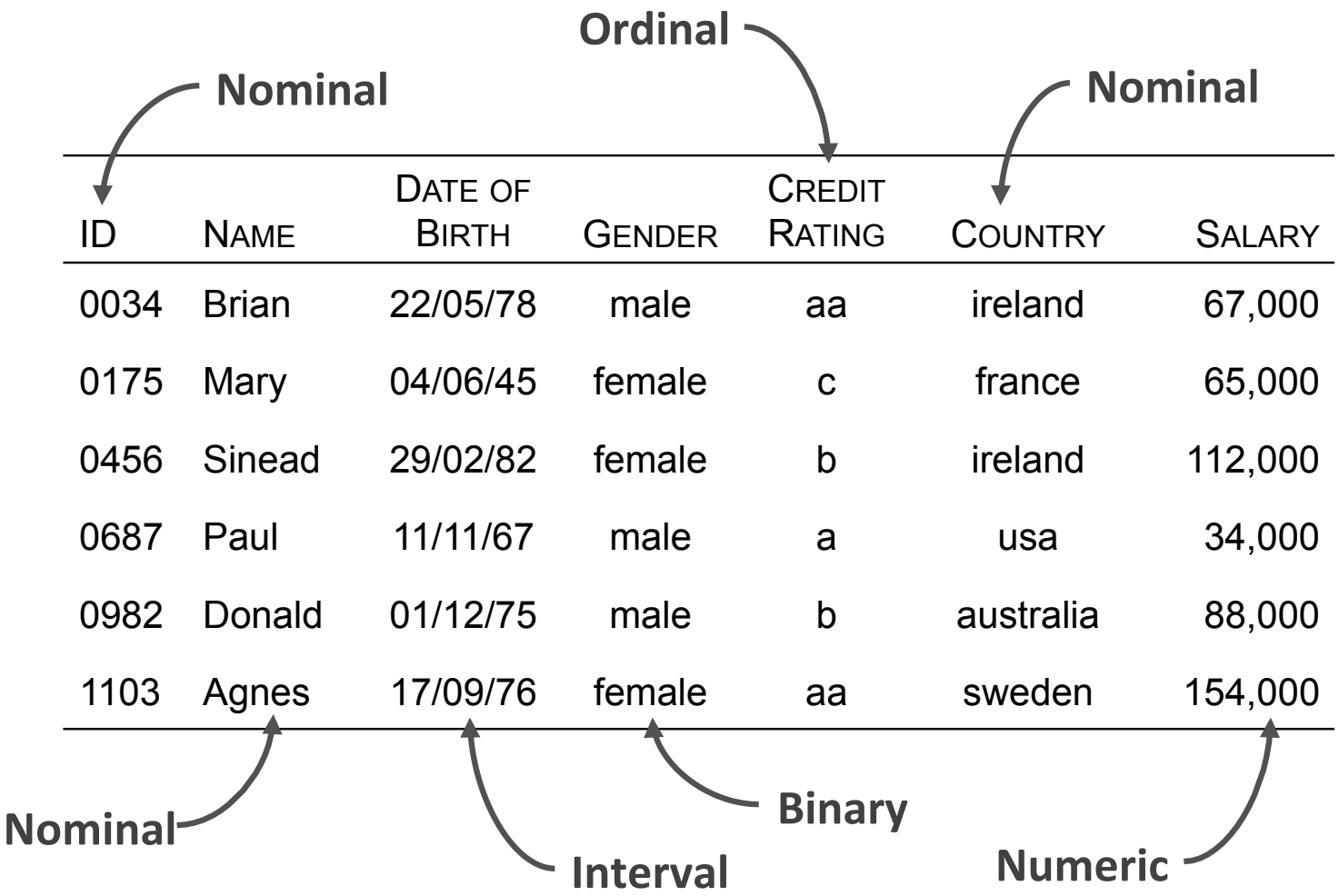


Figure: Sample descriptive feature data illustrating different types.

- The features in an ABT can be of two types:
  - **Raw features**
  - **Derived features**
- There are a number of common derived feature types:
  - **Aggregates** (such as total income of a customer)
  - **Flags** (such as fraudulent or not)
  - **Ratios** (such as loan-to-income ratio)
  - **Mappings** (a popular one is a log transformation to compress the range of a feature such as salaries)

Case Study: Car Insurance Fraud

- The following table illustrates the structure of the final ABT that was designed for the car insurance claims fraud detection solution.
- The table also shows the first four instances.
- If we examine the table closely, we see a number of strange values (for example, $-9\,999$) and a number of missing values.

Table: The ABT for the car insurance claims fraud detection solution

ID	Type	Inc.	Marital Status	Num. Clmnts.	Injury Type	Hospital Stay	Claim Amt.
1	CI	0		2	Soft Tissue	No	1 625
2	CI	0		2	Back	Yes	15 028
3	CI	54 613	Married	1	Broken Limb	No	-9 999
4	CI	0		3	Serious	Yes	270 200
⋮						⋮	

ID	Total Claimed	Num. Claims	Num. Claims 3 Months	Avg. Claims Per Year	Avg. Claims Ratio	Num. Soft Tissue	% Soft Tissue
1	3 250	2	0	1	1	2	1
2	60 112	1	0	1	1	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
⋮					⋮		

ID	Unsucc. Claims	Claim Amt. Rec.	Claim Div.	Claim to Prem.	Region	Fraud Flag
1	2	0	0	32.5	MN	1
2	0	15 028	0	57.14	DL	0
3	0	572	0	-89.27	WAT	0
4	0	270 200	0	30.186	DL	0
⋮					⋮	

Identifying Data Quality Issues

- A **data quality issue** is loosely defined as anything *unusual* about the data in an ABT.
- The most common data quality issues are:
  - **missing values** (sometimes ?, -1, -9999 etc. will denote a missing value)
  - **irregular cardinality** (e.g., US, U.S., etc. all mean the same)
  - **outliers** (e.g., a customer with an age of 225)

Handling Data Quality Issues

Handling Missing Values

- Approach 1: Drop all rows that have at least one missing value.
- Approach 2: Impute missing values, that is, make up reasonable values for them.

- **Imputation** replaces missing feature values with a plausible estimated value based on the feature values that are present.
- The most common approach to imputation is to replace missing values for a feature with a measure of the central tendency of that feature such as
  - Mean or median for a numerical feature
  - Mode for a categorical feature

Handling Outliers

- The easiest way to handle outliers is to use a **clamp transformation** that clamps all values above an upper threshold and below a lower threshold to these threshold values, thus removing the offending outliers

$$a_i = \begin{cases} lower & \text{if } a_i < lower \\ upper & \text{if } a_i > upper \\ a_i & otherwise \end{cases} \tag{1}$$

where a_i is a specific value of feature a, and *lower* and *upper* are the lower and upper thresholds.

Data Preparation

- Some data preparation techniques change the way data is represented just to make it more compatible with certain machine learning algorithms.
  - Normalization
  - Binning
  - Sampling

Normalization

- **Normalization** techniques can be used to change a continuous feature to fall within a specified range while maintaining the relative differences between the values for the feature.

- We use **range normalization** to convert a feature value into the range $[low, high]$ as follows:

$$a_i' = \frac{a_i - min(a)}{max(a) - min(a)} \times (high - low) + low \tag{2}$$

- Another way to normalize data is to **standardize** it into **standard scores**.
- A standard score measures how many standard deviations a feature value is from the mean for that feature.
- We calculate a standard score as follows:

$$a_i^{'} = \frac{a_i - \overline{a}}{sd(a)} \tag{3}$$

Example: The result of normalising a small sample of the HEIGHT and SPONSORSHIP EARNINGS features from the professional basketball squad dataset.

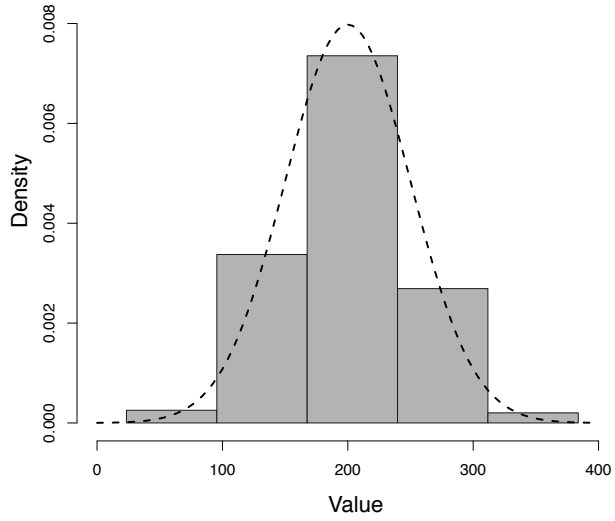
	HEIGHT			SPONSORSHIP EARNINGS		
	Values	Range	Standard	Values	Range	Standard
	192	0.500	-0.073	561	0.315	-0.649
	197	0.679	0.533	1,312	0.776	0.762
	192	0.500	-0.073	1,359	0.804	0.850
	182	0.143	-1.283	1,678	1.000	1.449
	206	1.000	1.622	314	0.164	-1.114
	190	0.429	-0.315	1,179	0.694	0.512
	178	0.000	-1.767	1,078	0.632	0.322
	196	0.643	0.412	47	0.000	-1.615
	201	0.821	1.017	1111	0.652	0.384
Max	206			1,678		
Min	178			47		
Mean	193			907		
Std Dev	8.26			532.18		

Binning

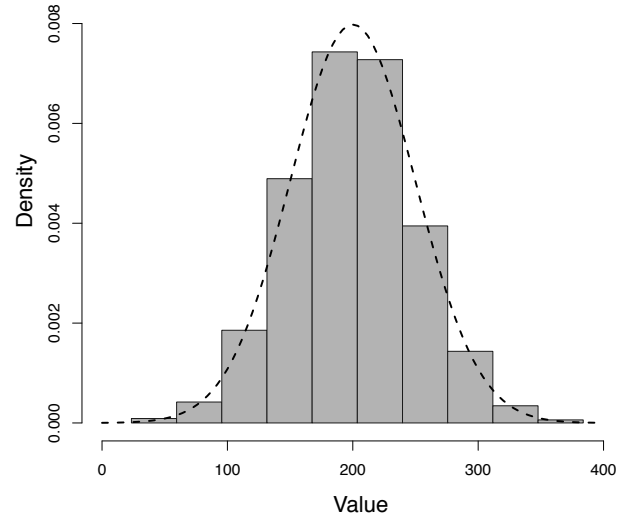
- **Binning** involves converting a continuous feature into a categorical feature.
- To perform binning, we define a series of ranges (called **bins**) for the continuous feature that correspond to the levels of the new categorical feature we are creating.
- We will introduce two of the more popular ways of defining bins:
- Two popular ways to define bins:
  - **equal-width binning**
  - **equal-frequency binning**

- Deciding on the number of bins can be difficult. The general trade-off is this:
  - If we set the number of bins to a very low number we may lose a lot of information
  - If we set the number of bins to a very high number then we might have very few instances in each bin or even end up with empty bins.

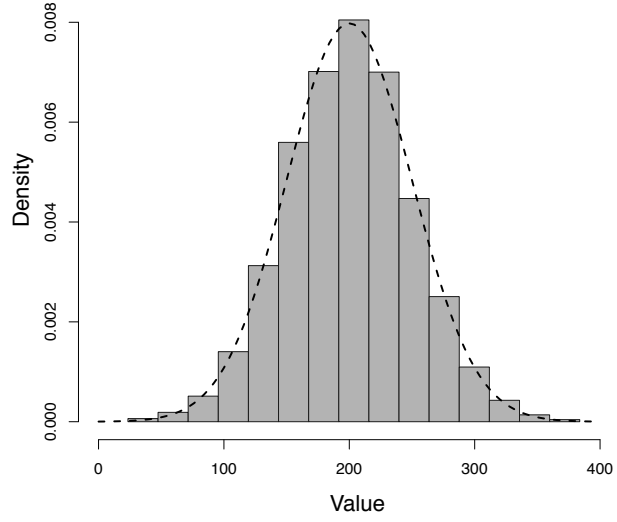
- The **equal-width binning** algorithm splits the range of the feature values into b bins each of size $\frac{range}{b}$.



(a) 5 Equal-width bins

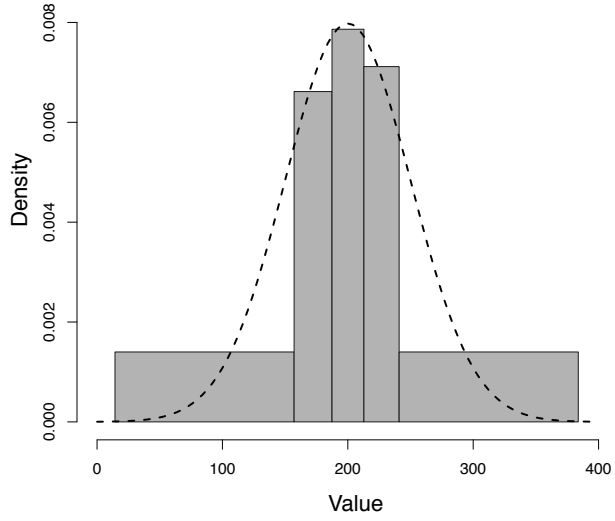


(b) 10 Equal-width bins

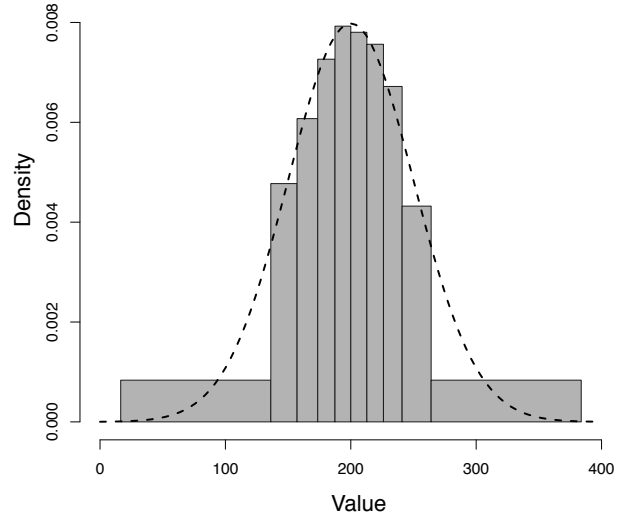


(c) 15 Equal-width bins

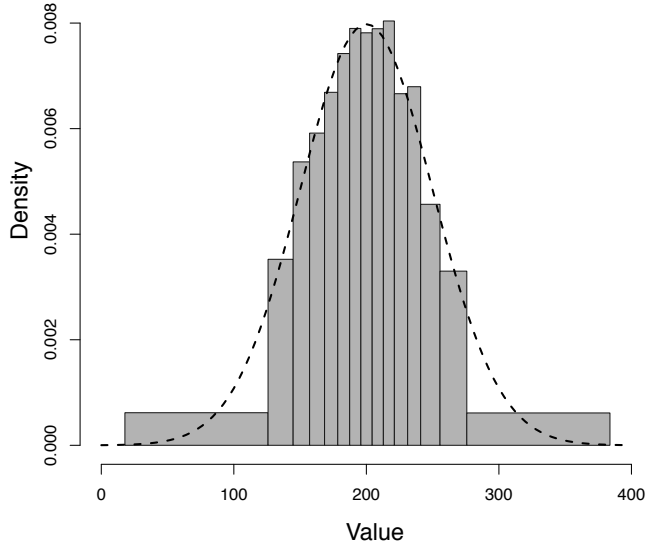
- **Equal-frequency binning** first sorts the continuous feature values into ascending order and then places an equal number of instances into each bin, starting with bin 1.
- The number of instances placed in each bin is simply the total number of instances divided by the number of bins, b.
- Example: 400 instances and 5 bins means each bin will contain 80 instances each.



(d) 5 Equal-frequency bins



(e) 10 Equal-frequency bins



(f) 15 Equal-frequency bins

Sampling

- Sometimes the dataset we have is so large that we do not use all the data available to us in an ABT and instead **sample** a smaller percentage from the larger dataset.
- Two common types of sampling:
  - **random sampling**
  - **under-sampling**

- Our recommended default, **random sampling** randomly selects a proportion of $s\%$ of the instances from a large dataset to create a smaller set.
- Random sampling is a good choice in most cases as the random nature of the selection of instances should avoid introducing bias.

- Sometimes we would like a sample with respect to the levels of a particular feature in the original dataset.
- To do this, we can use **under-sampling**.
- This technique is especially relevant in datasets with a **class imbalance** problem.

- Example: 100 fraudulent cases vs. 2500 non-fraudulent cases:
  - Keep all the 100 fraudulent cases
  - Randomly select 500 non-fraudulent cases out of the 2500
  - Final dataset will have 600 cases

Best Practices with Python

First Steps: Before undertaking more involved data preparation steps, we need to perform some basic steps:

1. **ID Columns for Rows**: Sometimes the dataset at hand will have a unique value for each row, such as Customer ID, Patient ID, Record ID, etc. Such features are irrelevant in machine learning and they must be removed (for example, by setting this ID column to data frame row index).
2. **Constant Features:** Sometimes a dataset will have constant features (that have only one unique value). Such features are also irrelevant for machine learning, so they need to be removed.
3. **Other Irrelevant Features:** While being problem-specific, any feature that is not relevant for "learning" needs to be removed, e.g., the name of the database the row comes from.
4. **Redundant Features:** A descriptive feature is "redundant" if it conveys the same information as another feature, e.g., salary in US and AU dollars.
5. **Date and Time Features:** Date features such as birthdays cannot be used as they are and they must be transformed, e.g., birthdays must be transformed to "age".

Next Steps: The following steps will be necessary for data preparation in this specific order in general:

1. Outliers and unusual values (such as a negative age) are taken care of: they are either clamped, imputed, dropped, or set to missing values.
2. Missing values are imputed or the rows containing them are dropped (most ML methods will give an error if you have even one single missing value in your dataset).
3. Any categorical descriptive feature is encoded to be numeric as follows:
   - One-hot-encoding for nominals
   - One-hot-encoding or integer-encoding for ordinals
4. All descriptive features (which are all numeric at this point) are scaled.
5. In case of a classification problem, the target feature is label-encoded (in case of a binary problem, the positive class is encoded as 1).
6. If the dataset has too many observations, only a small random subset of entire dataset is selected to be used during model tuning and model comparison.
7. Before fitting any Scikit-Learn models, any Pandas series or data frame is converted to a NumPy array using the *values* method in Pandas.

1. **Designing the Analytics Base Table**
2. **Identifying Data Quality Issues**
3. **Handling Data Quality Issues**
   - Handling Missing Values
   - Handling Outliers
4. **Data Preparation**
   - Normalization
   - Binning
   - Sampling
5. **Best Practices with Python**