# Categorical variables

# Encoding categorical variables

## When is a variable categorical?

Categorical variables describe qualitative phenomena.

### Examples

- **Quantitative**: Stock price
- **Qualitative**: Industry sector of the stock

# Ordered and unordered categorical variables

Categorical variables can either be ordered or unordered.

## Examples

- **Ordered**: "Poor", "Good", "Best"
- **Unordered**: "French", "Indian", "American"

# Independent from data type

Numerical values can also be categorical variables.

## Examples

- Postal codes: “64295”, “55118”
  (Darmstadt is not 9177 better than Mainz)

# Machine learning models for categorical data

Whether categorical values must be transformed depends on the chosen model.

## Examples

- Decision trees can use categorical values as cut-offs
- CatBoost got its name from efficiently dealing with categorical variables
- Naive Bayes models work with contingency tables based on categorical values
- However: Most models expect numerical data

## One Hot Encoding

Each category value receives a bit.

| Person | City |
|---|---|
| 1 | DA |
| 2 | FFM |
| 3 | MZ |
| 4 | DA |

| Person | DA | FFM | MZ |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 |

# Your Task

- Use `sklearn.preprocessing.OneHotEncoder` to encode categorical data
- OKCupid Data, one-hot encoding for column `body_type`
- Bonus: Where are the feature names?
- Bonus: What to do with missing values?

## One Hot Encoding

As each category gets its own column, we have the equation:

$$e_1 + \ldots + e_k = 1$$

Here each $e_i$ is one column.

Thus, there's linear dependency between columns. This can lead to numerical problems for some algorithms.

# Dummy Coding

Solution for linear dependency: One category is implicitly represented by the zero vector.

- In python, two possible implementation: `pandas.get_dummies`, `OneHotEncoder(drop='first')`
- Why is `sklearn.preprocessing.OneHotEncoder` better for machine learning?

## Effect Coding

Implicit category is represented by a vector of -1.

- Not used much in practice, only in old textbooks on linear regeression
- Advantage: In linear regression, the intercept becomes the mean of the target variable.
- **Python**: `pandas.get_dummies`, then manually adjust values.

| Person | DA | FFM |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | -1 | -1 |
| 4 | 1 | 0 |

## Pros and Cons

| Method | Pro | Con |
| --- | --- | --- |
| One Hot | Missing data gets 0 vector | Redundant |
| Dummy | Not redundant | What to do with missing data? |
| Effect | Not redundant, easier interpretability | -1-vector is dense, computationally expensive |

## Problems with one hot encoding

- Incomplete vocabulary: Due to random sampling, some values just do not make it into your training set.
- Model size due to cardinality: What if you're categorical variable has millions of values, like e.g. a device id?
- Cold start: What to do with values that are not yet there during data collection of your training data, but appear later during the serving period?

# Your Task

- Apply one-hot encoding to column `location`.
- Create a train-test split.
- Identify variables not appearing in the training set.

## Solution with 'other' category

- Compute frequency per column.
- Categories below frequency cutoff become "other".
- Values not seen yet by the algorithm also go in "other"

## Solution with Feature Hashing

- Hash function maps a string to an integer in some range $[-m, m]$
- During hashing, **collisions** can appear: Two strings map to the same number

In case of encoding categorical variables:

- Collisions are not a bug, but a feature to reduce feature count

# Feature Hashing: Method

For desired n buckets:

- Choose a deterministic hash function (no random seed)
- String → Hash value → (Hash value mod n)

Example with n=16:

- "San Francisco" → 823883475 → 15
- San Francisco receives 1 in the 15th column, else 0.

# Feature Hashing: Extended

With (-1) as additional value, we get 2n features in n columns:

- String → Hash value → (Hash value mod 2n) - n

Example:

- "Palo Alto" → 834834297 → -14
- Palo Alto gets -1 in the 14th column, else 0.

## Feature Hashing: Summary

- No free lunch: You pay with a decrease of model accurucay due to bucket collision
- Loss of accuracy especially high with skewed inputs. For example, when hashing airports, Heathrow and Frankfurt Hahn can end up in the same category.

## Task: Feature Hashing

- Dataset: OK Cupid, column `location`
- **Python**: `sklearn.feature_extraction.FeatureHasher`

Tips:

- Use `.unique` to find distinct location values.
- Format data according to minimal working example.
- Then apply `FeatureHasher`.

# Bin Counting

Convert category values into statistics about the variable value:

- Instead of sparse one-hot, dense numeric representation.

# Bin Counting Example

Starting with counts for user clicks:

| User | Clicks | Non-clicks |
|---|---|---|
| Alice | 5 | 120 |
| Bob | 20 | 230 |

# Odds Ratio

Calculate 2-way contingency tables to get odds ratio:

- Odds Ratio is

$$\frac{P(Y=1|X=1)/P(Y=0|X=1))}{P(Y=1|X=0)/P(Y=0|X=0)}$$

- In implementations, usually only the numerator is used.

## Example: Odds Ratio

| | Click | Non-click | Total |
|---|---|---|---|
| Alice | 5 | 120 | 125 |
| Not Alice | 995 | 18880 | 19875 |
| Total | 1000 | 19000 | 20000 |

Then numerator of the odds ratio for Alice is

$$\frac{P(Y=1|X=1)}{P(Y=0|X=1)} = \frac{5/125}{120/125}$$

# Task: Bin Counting

- Calculate numerator odds ratio per `device id` in Avazu CTR dataset.
- Hint: Use `groupby` and `agg`.

## Bin Counting for rare categories

- Rare users get grouped in "other".
- Category emerges if count surpasses threshold.

# Bin Counting and Leakage

- Direct use of target variable info risks leakage.
- Solution: Three phases during data collection.