Clustering and the Explanation of Clusters

Let us begin with an example. Consider the data on most countries of the world, almost 200 of them, which we describe with a few dozen socioeconomic features. A sample of such data is, for example, in the table below, but in reality our data matrix is much larger.

Country	Life expectancy	Average years of schooling	GDP per capita	Share of urban population	Unvaccinated infants (measles)
Australia	82.5	13.2	42 822.0	89.4	7.0
Chile	82.0	9.9	21 665.0	89.5	6.0
Cuba	79.6	11.8	7455.0	77.1	1.0
Denmark	80.4	12.7	44 519.0	87.7	10.0
Greece	81.1	10.5	24 808.0	78.0	3.0
Slovenia	80.6	12.1	28 664.0	49.7	6.0
Switzerland	83.1	13.4	56 364.0	73.9	7.0
Venezuela	74.4	9.4	15 129.0	89.0	11.0

Table 5: Socioeconomic profiling of the countries.

We can use this data to perform *clustering*, that is, using a procedure in which we divide the countries into groups (clusters) according to their similarity. In this chapter, we will try to present clustering procedures systematically, but only a bit later.

Namely, we have already encountered a visual approach to cluster discovery in the chapter on dimensionality reduction (an example of such a data map is shown in Figure 33), but skipped the all-important aspect of cluster discovery, which deals with explaining clusters. Hence, we start with the explanation aspect, with a note that the procedures that we will use can be applied to results of any kind of clustering.

Figure 33: Countries in a t-SNE graph. To build the graph, we used the *Human Development Index* data set from 2014 with 50 features. Among the countries in the graph, we selected a smaller group, which is represented with highlighted dots.

On the data map in Figure 33, we can discern a few groups. We selected one of them, with eight countries, and want to know what these countries have in common. We have several options:

- We can list the names of the countries. For eight countries, this will probably be the first thing we can do, but it would be a prob-

lem if the selected group were larger and, with lots of their features in the profile, less transparent.

- We look at where these countries are on the map. The map offers us additional information that is not contained in the data, but comes in handy, especially if we are versed in geography.
- We can make use of data on groups of countries, for example Mediterranean, Asian, South American, and then determine whether most of the countries we selected belong to any of these groups.
- We can think about in which features the selected countries differ from all the others. What would be most useful here is an ordered (ranked) list of discriminating attributes. Of course, socioeconomic knowledge should help us here in understanding this ordered list.
- If we have many attributes, organizing the attributes into groups could help us as well. For example, we could determine attributes that are related to healthcare, financial indicators, education, and the like. Then we might be able to determine for the selected countries that they are, say, among the wealthiest countries in the world, or among the countries with the most developed education systems, and so on.

The above are different ways of explaining resulting groups. In some, we used knowledge about the (selected) data instances (countries), while in others we tried to describe the selected data instances (differentially) by using the data. In most of the somewhat better explanations, additional knowledge or information that was not present in the data itself came in handy. Such knowledge is also called *domain knowledge*. In all of the above ideas, except perhaps in the first two, we will nevertheless have to use some computational procedures for ranking attributes or determining group enrichments. And it is precisely these procedures that we will discuss next.

Group Enrichment Analysis

Let us simplify the above example and assume that our data contained only 12 countries and that among them we found a group of five, as shown in Table 6. We notice that among the selected countries there are quite a few Mediterranean ones. In fact, the table contains five Mediterranean countries (France, Greece, Croatia, Italy, Spain), and among them as many as four countries are from our selection. Our selection also includes Portugal, which is not a Mediterranean country. We ask whether such a large (or larger) number of Mediterranean countries in our selection would also be obtained by chance.

The phenomenon that some group appears in a selection more often than we would expect if the selection were arbitrary is called group enrichment.

That is, what is the probability that, in a random selection of five countries from all twelve, we would select at least four Mediterranean countries?

Let us use combinatorics. Our data contain a finite set of $N = 12$ countries, among which there are $K = 5$ Mediterranean countries. From this set we selected $n = 5$ countries, of which $k = 4$ were Mediterranean. The probability that in the selection we obtain exactly k Mediterranean countries is equal to

$$P(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}.$$

This expression is known as the hypergeometric distribution, which describes probabilities when sampling without replacement from a finite set. In our case, the random variable X, which counts the number of Mediterranean countries in the selection, follows a hypergeometric distribution with parameters N, K, and n.

In the hypergeometric distribution, $\binom{K}{k}$ is the number of ways in which we choose k countries from all K Mediterranean countries, $\binom{N-K}{n-k}$ is the number of ways in which we choose the remaining $n - k$ countries from all non-Mediterranean countries, and $\binom{N}{n}$ is the number of all possible selections of n countries from the entire set of N countries. The fraction thus represents the proportion of all possible selections in which there are exactly k Mediterranean countries.

In our case, we observed $X = 4$, since among the five selected countries four are Mediterranean. Since we are interested in whether this is an unusually large proportion, we compute the probability that by chance we would obtain at least four Mediterranean countries:

$$P(X \geq 4) = P(X = 4) + P(X = 5).$$

We obtain

$$P(X = 4) = \frac{\binom{5}{4}\binom{7}{1}}{\binom{12}{5}} = \frac{5 \cdot 7}{792} = \frac{35}{792},$$

and

$$P(X = 5) = \frac{\binom{5}{5}\binom{7}{0}}{\binom{12}{5}} = \frac{1 \cdot 1}{792} = \frac{1}{792}.$$

Together, therefore,

$$P(X \geq 4) = \frac{35}{792} + \frac{1}{792} = \frac{36}{792} = \frac{1}{22} \approx 0.0455.$$

This means that, in a completely random selection of five countries from twelve, we would obtain at least four Mediterranean countries with probability approximately 4.5%. Such a result is therefore

Enrichment analysis is especially common in bioinformatics, where it is used to interpret lists of genes. For example, if a set of genes is found to be active in some condition, enrichment analysis can identify biological processes, pathways, or gene groups that occur in this list more often than expected by chance. The same idea, however, is completely general: any selected set of objects can be compared with predefined groups of objects.

Table 6: Countries and their selection. Selected countries are marked in the column "Selection" with 1, the others with 0.

Country	Selection
Austria	0
France	1
Greece	1
Croatia	0
Italy	1
Germany	0
Netherlands	0
Portugal	1
Poland	0
Spain	1
Sweden	0
Switzerland	0

relatively unlikely and suggests that Mediterranean countries are enriched in our selection relative to what we would expect by chance.

We can also compute the expected number of Mediterranean countries in a random selection of five countries. For the hypergeometric distribution, the expected value is

$$E(X) = n\frac{K}{N} = 5 \cdot \frac{5}{12} = \frac{25}{12} \approx 2.08.$$

In a random selection, we would therefore on average expect about two Mediterranean countries, whereas we observed as many as four. In this way, too, we see that our result differs from what is expected under random selection.

Of course, instead of the Mediterranean group we could also take some other group of countries. For example, EU countries. Among the twelve countries in our data, there are nine such countries, and among the five selected countries all are members of the EU. If we denote by X the number of EU countries in the selection, then $N = 12$, $K = 9$, $n = 5$, and $k = 5$. The probability that in a random selection of five countries we would obtain only EU countries is

$$P(X = 5) = \frac{\binom{9}{5}\binom{3}{0}}{\binom{12}{5}} = \frac{126}{792} \approx 0.159.$$

This result is therefore not particularly unlikely, so in this case we could not speak of pronounced enrichment.

We are actually not only interested in the probability of picking a specific number of countries belonging to a certain group, but rather in the probability that this number would be the same or higher, that is, the probability that, under random selection, we would do similarly or better. A bit more on this in our next section on implementation.

Enrichment is thus the phenomenon where elements of a particular group appear in a selection more often than would be expected by chance. Enrichment analysis is the procedure by which we quantitatively evaluate this deviation and check whether it can be attributed to chance or whether it indicates an actual pattern in the data.

Figure 34: A YAML notation with an example of defining groups of countries.

```
Mediterranean:
  - France
  - Greece
  - Croatia
  - Italy
  - Spain

Balkan:
  - Greece
  - Croatia

Western European:
  - France
  - Germany
  - Netherlands
  - Austria
  - Switzerland
```

Enrichment Analysis in Code

With enrichment analysis, we can therefore explain which group of examples the selected examples belong to, in cases where this membership is (strongly) different from random. For this, in addition to the data, we need to prepare the domain knowledge in the form of groups, for our example, in the notation shown in Figure 34.

To compute enrichment, we can use the function `hypergeom.sf` from the `scipy.stats` library, where `sf` is the so-called survival func-

tion and returns the probability that the random variable X is greater than a given value k.

```
def enrichment(all_items, selected_items, group):
    group = set(group) & all_items

    N = len(all_items)
    n = len(selected_items)
    K = len(group)
    k = len(group & selected_items)

    p_value = hypergeom.sf(k - 1, N, K, n) if K > 0 else 1.0
    expected = n * K / N if N > 0 else 0.0
    fold = (k / expected) if expected > 0 else float("nan")

    return {
        "K": K,
        "k": k,
        "expected": expected,
        "fold": fold,
        "p_value": p_value,
    }
```

Statistical significance describes how likely it is that the observed result arose merely due to chance. We estimate this with the p-value: the smaller it is, the less likely it is that the result is a consequence of random selection.

The input to our enrichment function `enrichment` is the set of all objects, the set of selected objects, and the group for which we compute enrichment, or the probability that the selected objects would belong to this group if the selection were random. The function returns the distribution parameters K and k, the expected value (how many elements from the given group we would expect among the selected objects under random selection), the enrichment factor (the ratio of the actual and expected value), and finally the p-value, which estimates the statistical significance of the observed overlap.

In the main part of our implementation, we read the data on the selection, the data on groups of countries, compute enrichment, and print the results:

```
import yaml
import pandas as pd

df = pd.read_excel("selected-countries.xlsx")

all_items = set(df["Country"].astype(str).str.strip())
selected_items = set(
    df.loc[df["Selection"] == 1, "Country"].astype(str).str.strip()
)

with open("country-groups.yaml", "r", encoding="utf-8") as f:
    groups = yaml.safe_load(f)
```

```
results = []
for name, group in groups.items():
    r = enrichment(all_items, selected_items, group)
    r["group"] = name
    results.append(r)

results.sort(key=lambda x: x["p_value"])

for r in results:
    print(
        f"{r['group']+":":20s} "
        f"k={r['k']:2d}, K={r['K']:2d}, "
        f"E={r['expected']:.2f}, "
        f"fold={r['fold']:.2f}, "
        f"P={r['p_value']:4f}"
    )
```

Enrichment analysis for our selection of countries and the groups of countries with which we represent domain knowledge therefore returns a list of groups ordered by p-value:

```
Mediterranean:       k= 4, K= 5, E=2.08, fold=1.92, P=0.0455
Eurozone:            k= 5, K= 9, E=3.75, fold=1.33, P=0.1591
Coastal:             k= 5, K=10, E=4.17, fold=1.20, P=0.3182
Balkan:              k= 1, K= 2, E=0.83, fold=1.20, P=0.6818
Western European:    k= 1, K= 5, E=2.08, fold=0.48, P=0.9735
```

In fact, we are interested only in the groups that are statistically significant and whose random probability is very small. For this, however, we need to set a threshold, usually denoted by α (e.g. $\alpha = 0.05$), below which the p-value is regarded as sufficiently small. Groups with $p < \alpha$ are thus treated as enriched, while the others are treated as a consequence of random selection.

How, then, do we approach the explanation of possible groups from Figure 33? One way is by creating an interactive visualization, where we would allow the user to select points or countries from the visualization and then display the enriched groups. Even better would be the automatic recognition of clusters of points in the visualization and then the annotation of these clusters with the names of enriched groups. Something similar to the (admittedly invented) explanation in Figure 35.

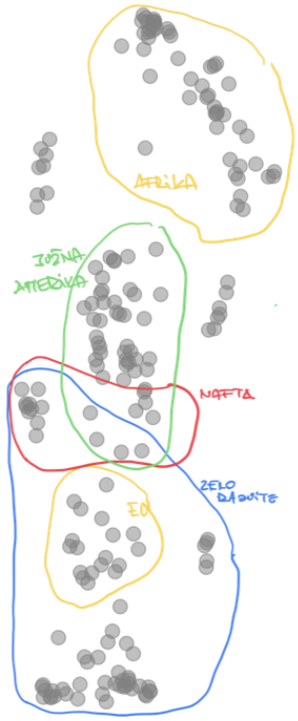


Figure 35: An example of a possible annotation of the t-SNE arrangement of countries into a two-dimensional map. Would you know how to program such an explanation of the map?

The parameter α determines the probability of a type I error, that is, that we incorrectly declare a result statistically significant even though it is in fact a consequence of chance.

Multiple Testing

In the above example we tested only a few groups of countries. In real data, however, the number of groups can be much larger. For example, countries could be annotated with many geographic, political, economic, cultural, and historical labels. If we test each of

these groups separately, then some groups may appear statistically significant simply by chance.

This problem is often referred to as *the multiple testing problem*. If we test many groups separately, some groups may appear statistically significant simply by chance. Suppose that we test 100 groups and use the threshold $\alpha = 0.05$. Even if none of the groups is truly enriched, we would still expect about five of them to have $p < 0.05$ just by random variation. Therefore, when the number of tested groups is large, the raw p-values can be misleading.

One simple correction is the Bonferroni correction. If we test m groups, we replace the threshold α with

$$\alpha' = \frac{\alpha}{m}.$$

The Bonferroni correction is named after Carlo Emilio Bonferroni, and its basis comes from his work on probability inequalities in 1936. So it is about 90 years old.

The Bonferroni correction is very conservative: it strongly reduces the chance of false discoveries, but it may also hide some truly enriched groups.

A less conservative and often more useful approach is to control the *false discovery rate* (FDR). The FDR is the expected proportion of false discoveries among all discoveries that we report as significant. For example, if we control the FDR at level $\alpha = 0.05$, we accept that, among the groups we report as enriched, about 5% may be false discoveries.

The false discovery rate (FDR) was introduced by Yoav Benjamini and Yosef Hochberg in 1995, in their paper "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing."

The most common procedure for controlling the FDR is the Benjamini–Hochberg correction. We can understand it as a way of adjusting the p-values. We first sort all m p-values from the smallest to the largest:

$$p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}.$$

Each sorted p-value is then multiplied by a factor that depends on its rank:

$$p^*_{(i)} = p_{(i)} \frac{m}{i}.$$

Thus, the smallest p-value is multiplied by m, the second smallest by $m/2$, the third by $m/3$, and so on. After this, the adjusted values are made monotone, so that they do not decrease as we move from smaller to larger original p-values. The resulting adjusted p-values can then be compared with the usual threshold α. Groups with adjusted p-values below α are reported as significant.

Compared with the Bonferroni correction, the Benjamini–Hochberg correction is usually less strict, so it often detects more enriched groups, while still limiting the expected proportion of false discoveries among the reported results.

In practice, enrichment analysis should therefore report not only raw p-values, but also corrected p-values, or an FDR score, especially when many groups are tested. For a small number of groups, this

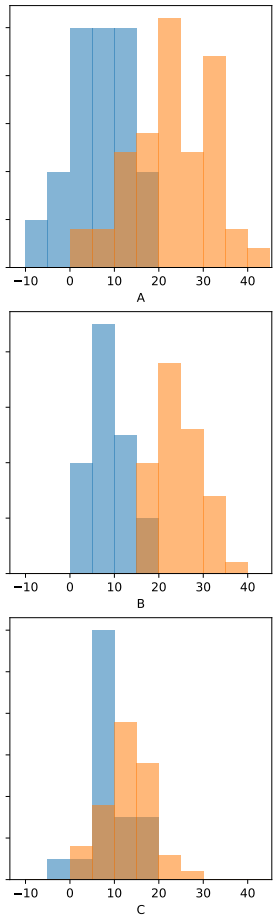
correction may not change the conclusions much. For thousands of groups, however, it can completely change which explanations we should trust.

Continuous Features Associated with Groups

Let us return to our selection of the group of countries from Figure 33, but this time focus on their properties. We would like to determine in which features the selected group differs from all the others. In the data from which we built the visualization, there were 50 features, so it would make sense to obtain an ordered list, where at the top are those that are most strongly associated with our selection. We therefore need some estimator of the association between the group and the informativeness of the feature, which takes larger values for features in which the selected countries differ most from the remaining ones.

Let us begin with a hypothetical example and show, for three features that we have simply named A, B, and C, what their distribution of values is in the selected set of countries and how these values are distributed in all the other countries (Figure 36). Which feature is the most informative? Which one, then, best separates the examples from the selected cluster from all the other examples? The distributions are separated for features A and B, but the overlap is smaller for feature B. The overlap is large for feature C.

Figure 36: Distribution of variables A, B, and C in the selected group of examples (orange) and in all other examples (blue).



Clearly, the most information about the selected group is provided by feature B. Intuitively, we can say that better features are those for which the two distributions are as far apart from each other as possible and at the same time as little dispersed as possible. The separation can be estimated by the difference between the mean values of the two groups, and the dispersion by the variance. Since we want a single estimator that takes both into account, we can use the t-statistic. It measures the difference between the means relative to the dispersion of the data in the two groups. For feature X we write it as

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}},$$

where $\bar{x}_1$ and $\bar{x}_2$ are the means in the selected and remaining groups, s_1^2 and s_2^2 are the variances, and n_1 and n_2 are the sizes of the two groups. A larger absolute value of t means greater separability between the groups and thus greater informativeness of the feature.

For ranking features, the absolute value of t will be sufficient, but if we also want to estimate the statistical significance of the observed difference, we convert t into a p-value. We obtain this from the t-

The statistic t was introduced by William Sealy Gosset in 1908. He published it under the pseudonym Student in the article "The probable error of a mean", since he worked for the Guinness brewery, which did not allow its employees to publish scientific work under their own names. That is why today we speak of Student's t-distribution and the t-test.

distribution as the probability that, under the assumption that there is no difference between the groups, we would obtain such a large or even larger value of $|t|$. In doing so, we assume that t follows a t-distribution with the appropriate number of degrees of freedom (e.g. $\nu \approx n_1 + n_2 - 2$) and compute the tail probability. For a two-sided test,

$$p = 2 \cdot P(T \geq |t|).$$

The smaller the p-value, the less likely it is that the observed difference is a consequence of chance. In Python, we compute this as:

```
from scipy.stats import t
p_value = 2 * t.sf(abs(t_stat), df)
```

With code that would properly read the data, allow us to select some group of countries, and then compute the t-statistic or its corresponding p-value, we could obtain an output, an example of which is shown below:

```
0.0000 v Inequality in life expectancy (%)
0.0000 v Infant mortality (per 1000 births)
0.0006 v Youth not in school or employment (%)
0.0012 v Infants exclusively breastfed (%)
0.0014 v Income inequality (%)
0.0294 ^ Share of women in parliament (%)
0.0769 ^ Paid maternity leave (days)
```

In addition to the p-value, this also indicates whether the value of the feature in the selected group is lower or higher. Of course, it would be very appropriate if we developed a suitable user interface for such an analysis, which would allow us to explore the data and their groups in a simple way.

What if the Variables Are Categorical?

The above assumes that all features are continuous. But what if they are categorical (for example, gender, region, or type of country)? In this case, instead of the t estimator, we can use the χ^2 statistic, which measures the deviation between observed and expected frequencies in a contingency table.

The χ^2 test was introduced by Karl Pearson in 1900 as a goodness-of-fit test. It later became one of the standard tools for analyzing categorical data, especially contingency tables.

Say that among the 12 countries, 8 are OECD members and 4 are not, and that the data table (see Table) contains a feature that reports this membership. Let there be 4 OECD members and 1 non-member in the selected group of five countries. These ratios can be represented in the contingency table 7.

Table 7: Contingency table of OECD membership.

	Selected	Others
OECD	4	4
Non-OECD	1	3
Total	5	7

If OECD membership were not associated with the selection, we would compute the expected frequencies as

$$E_{ij} = \frac{(\text{row sum}) \cdot (\text{column sum})}{\text{total sum}}.$$

Thus, for OECD members in the selected group, we obtain

$$E = \frac{8 \cdot 5}{12} = 3.33,$$

and for non-OECD members,

$$E = \frac{4 \cdot 5}{12} = 1.67.$$

We will quantitatively measure the deviation between observed and expected frequencies with the statistic χ^2, which combines the contributions of all cells of the contingency table. The statistic χ^2 is computed as

$$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}},$$

where O_{ij} is the observed frequency and E_{ij} the expected frequency. In our example, we obtain

$$\chi^2 = \frac{(4 - 3.33)^2}{3.33} + \frac{(4 - 4.67)^2}{4.67} + \frac{(1 - 1.67)^2}{1.67} + \frac{(3 - 2.33)^2}{2.33} \approx 0.69.$$

The larger the value of χ^2, the larger the deviation between observed and expected frequencies, and thus the stronger the association between the categorical feature and the selected group. If we also want to estimate statistical significance, we convert the value of χ^2 into a p-value using the χ^2 distribution with the appropriate number of degrees of freedom, but in principle we may do this only if the expected frequencies in all cells are sufficiently large (as a rule, at least around 5).

The statistic χ^2 was introduced by Karl Pearson in 1900 and presented in the article *On the criterion that a given system of deviations...* as a method for testing agreement between observed and expected frequencies.

In Python, we compute our example with the following code:

```
import numpy as np
from scipy.stats import chi2_contingency

O = np.array([
    [4, 4],
    [1, 3]
])

chi2_stat, p_value, df, E = chi2_contingency(O, correction=False)
print(f"chi2 = {chi2_stat:.4f}")
print(f"p    = {p_value:.4f}")
```

The function `chi2_contingency` returns the value of χ^2, its corresponding p-value, the number of degrees of freedom, and the matrix of expected frequencies. The program returns

```
chi2 = 0.6857
p    = 0.4076
```

and we can conclude that OECD membership is not a feature that is associated with our selection of countries.

Variables Can Also Be Arranged into Groups

Figure 37: Socioeconomic features arranged into groups.

```
Children and youth:
  - Child labor (%)
  - Youth not in school or employment (%)
  - Child malnutrition (%)
  - Child mortality (per 1000)
  - Infants exclusively breastfed (%)
  - Infants w/o vaccination (DTP) (%)
  - Infants w/o vaccination (measles) (%)

Health and inequality:
  - Inequality in life expectancy (%)
  - Life expectancy index

Economic structure:
  - Employed in agriculture (%)
  - Employed in services (%)
  - Share of urban population (%)
```

Explaining a group by ranking variables of course requires interpretation. We must know well the meaning of the variables and know what they refer to, and interpret the results in accordance with the p-values and the order of the variables in the ranking. Huh, even the previous sentence is quite long and perhaps complicated. Interpretation, then, is not simple, and requires some extra knowledge.

What if here, too, we help ourselves with domain knowledge and, for example, arrange the variables into groups? For example, as shown in Figure 37. Our goal here is to determine whether the group of selected countries is characterized by features that belong to a particular group of features, and then report on the enrichment of these groups. In doing so, we combine everything we have already introduced in this chapter, that is, ranking of features, their selection (with respect to some upper bound α for the p-value), and the computation of enrichment of groups of features. The output of the procedure is therefore the probabilities that the selected groups of features are represented in the observed cluster more often than would be expected by chance.

In the above procedure, we introduced a new parameter, the threshold α. We can avoid this. Instead of computing enrichment in the above way, for a given group of features we can observe where in the list of ranked features the features from this group appear. We are interested in cases where most of the features from the group are near the top of the ordered list (presence) or near the bottom (absence).

The Mann–Whitney test and the corresponding statistic were proposed in 1947 by Henry B. Mann and Donald R. Whitney as a nonparametric alternative to the t-test, which does not assume normality of the data and is based on ranks.

The estimator that we introduce for such a computation, is called the Mann–Whitney statistic, which is in fact related to the area under the ROC curve, better known in machine learning (the so-called *area under the curve*, or AUC). It measures how often features from a given group are ranked higher than features outside the group, and can be interpreted as the probability that a randomly selected feature from the group will be ranked higher than a randomly selected feature outside the group. Values of the statistic close to 1 indicate a

concentration of features at the top of the list, values close to 0 at the bottom, and a value around 0.5 corresponds to a random arrangement.

Let us look at a short example. Suppose we have eight ranked features,

$$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8,$$

ordered from the most to the least informative. Let us have a group of features

$$G = \{x_1, x_3, x_4\},$$

and denote the remaining features by $\bar{G} = \{x_2, x_5, x_6, x_7, x_8\}$.

The Mann–Whitney statistic compares all pairs (g, o), where $g \in G$ and $o \in \bar{G}$, and counts how many times the feature from the group is ranked higher (has a smaller rank) than the feature outside the group. In our example, there are $3 \cdot 5 = 15$ such pairs. The number of "wins" of the group is

$$U = 5 + 4 + 4 = 13,$$

since x_1 is before all five, x_3 is before four (but not before x_2), and x_4 is likewise before four. We normalize the number of wins by the number of all possible pairs, and obtain

$$\text{AUC} = \frac{U}{|G| \cdot |\bar{G}|} = \frac{13}{15} \approx 0.867.$$

This means that the probability that a randomly selected feature from the group will be ranked higher than a randomly selected feature outside the group is approximately 86.7%, which indicates a pronounced presence of the group at the top of the list. The implementation of AUC has linear complexity (a walk through the ordered list), if we do not take sorting into account, but we can also use the implementation from the `scipy.stats` library:

```
from scipy.stats import mannwhitneyu

group_ranks = [1, 3, 4]
other_ranks = [2, 5, 6, 7, 8]

U, p = mannwhitneyu(group_ranks, other_ranks, alternative="less")
auc = 1 - U / (len(group_ranks) * len(other_ranks))

print(f"U = {U}")
print(f"AUC = {auc:.3f}")
print(f"p = {p:.4f}")
```

The code above also returns the p-value, which tells us how likely it is that such a favorable arrangement of features would be obtained by chance.

Table 8: Enrichment of feature groups in socioeconomic data.

AUC	p	Group
1.000	0.0008	health_and_inequality
0.738	0.0434	inequality
0.721	0.0321	demography
0.678	0.0846	education

A possible output of such an analysis would be a list such as the one shown in Table 8. This shows, for example, that the selected countries are special from the perspective of health, inequality, demography, and education, which is admittedly a rather abstract but perhaps suitable explanation. To understand it, however, we certainly need additional knowledge, especially about whether these areas are represented positively or negatively and in what way.

An analysis of this kind should also include information about the desirable direction of features (e.g., higher GDP is better, a lower unemployment rate is better, etc.), and all this should be sensibly incorporated into the user interface, which enables traceability or "drilling down" — from abstract results to concrete data with which we can explain why and how we arrived at these results.

This idea is related to the FAIR principles: data and analyses should be *Findable, Accessible, Interoperable, and Reusable*. In our context, this means that explanations should not only report abstract results, but also allow the user to trace them back to the data, annotations, and computations from which they were obtained. See Wilkinson et al. (2016).

What About Clustering?

Above, we have written at some length about explanations of clusters and also a little about the user interfaces needed for this. For the latter, let us note that those that would truly allow excellent explanations of clusters are rather rare and are found more in specialist tools, that is, those intended for data analyses from selected domains. But back to clustering. We will assume that the reader knows the field well, also from previous courses, and here we will only mention the key approaches. Therefore, here we only give Table with the main approaches and describe a few selected methods.

Group	Methods	Advantages	Disadvantages
Partitioning	k-means, k-medoids	fast, simple, applicable to very large numbers of examples	determining k, sensitive to initialization, spherical clusters
Hierarchical	agglomerative, divisive	visualization with a dendrogram	slow, suitable only for smaller data sets
Density-based	DBSCAN, OPTICS	clusters of arbitrary shape, handle noise, identify outliers	strongly sensitive to the method's meta-parameters
Model-based	Gaussian mixtures	probabilistic, examples can be assigned to multiple clusters	assumptions about the data distribution, local optima
Network-based	Louvain, Leiden	suitable for networks, communities	dependent on graph construction
Projection / embedding-based	t-SNE, PCA, spectral clustering	can detect complex structure, visualization	not directly intended for finding clusters, additional analysis is required

Table 9: **Main groups of clustering methods.** An overview of key approaches with typical methods and their basic advantages and disadvantages.

Hierarchical Clustering

Hierarchical clustering builds a hierarchy of clusters from the data: at initialization it assumes that each example is its own cluster, and then the algorithm gradually merges the most similar clusters until only one remains. The input to the method is a set of examples described by attributes, together with the choice of a distance measure between examples (e.g. Euclidean, Manhattan, cosine) and a distance measure between clusters (e.g. *single*, *complete*, *average linkage*, *Ward*). With *single linkage* we consider the smallest distance between elements of two clusters, with *complete linkage* the largest, with *average linkage* the average over all pairs, and with Ward's method the increase in within-cluster variance after merging. The result can be nicely displayed as a dendrogram, which is a tree representation of the sequence of merges; if we "cut" the dendrogram at a selected height, we obtain a concrete division of examples into several clusters. The problem with this display is that, by rotating individual branches, it can be shown in different ways.

Hierarchical clustering generally does not optimize a single global estimator for the entire solution, but builds a sequence of locally best merges according to the chosen distance measure. The main estimator in the procedure is therefore the distance between examples or clusters, and in Ward's method the increase in the sum of squared deviations from the centroid. The advantage of the method is that we do not need to determine the number of clusters in advance and that we obtain the result in a clear hierarchical form, while its disadvantages are computational complexity and sensitivity to the choice of distance, linkage method, and attribute scaling.

The k-Means Method

The k-means method is a partitioning clustering method that divides the data into a predetermined number K of clusters. Each cluster is represented by its leader, that is, the centroid or mean vector of the examples in the cluster. The input to the method is a set of examples described by numerical attributes, the selected number of clusters K, and a distance measure (most commonly Euclidean). The algorithm begins by choosing the initial leaders, and then iteratively repeats two steps: it assigns each example to the nearest leader, and then updates the leaders as the centroid values of the corresponding clusters. The procedure ends when the partition no longer changes or when the changes stabilize.

The output of the method is a partition of examples into K clusters and the corresponding leaders. The method optimizes the compact-

ness of clusters, that is, the sum of squared distances of examples to their leaders (SSE):

$$\sum_{i=1}^{K} \sum_{x \in C_i} \|x - v^{(i)}\|^2,$$

where the optimal leaders are equal to the centroids of the clusters. The algorithm typically converges quickly, but only to a local optimum, so the result depends on the initial choice of leaders.

The advantage of the method is its efficiency and suitability for large data sets, while its disadvantages are the need to specify K in advance, sensitivity to initialization and the distance measure, and the fact that the method works best for approximately spherical clusters of comparable size.

Silhouette

The silhouette is an estimator of the quality of a partition of data into groups, which simultaneously takes into account cohesion within clusters and separability between them. We can use it to assess the quality of a clustering for a given number of groups K, or to choose an appropriate K for methods such as k-means or hierarchical clustering. The input to the method is a partition of the data into groups and a selected distance measure between examples, and the output is the silhouette coefficient s, which takes values approximately between -1 and 1.

We compute the silhouette for each example $x^{(i)}$ from the average distance to all examples in its group, a_i, and the smallest average distance to examples in any other group, b_i. The silhouette of an example is then defined as

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}.$$

Values close to 1 mean that the example is well placed in its group, values around 0 indicate overlap between groups, and negative values indicate incorrect assignment.

The silhouette of a partition is the average of the silhouettes of all examples.

When choosing the number of groups, we compute the silhouette for different values of K and choose the one that gives the largest value of s. We can use it both with hierarchical methods and with the k-means method. The approach does not optimize the partition directly, but serves as an external estimator of quality. Its advantage is intuitive interpretation and the consideration of two key aspects of clustering, while its disadvantages are sensitivity to the chosen distance measure and lower reliability in cases where differences

between possible partitions are small or where clusters are not clearly separated.

Density-Based Spatial Clustering of Applications with Noise

DBSCAN is a clustering method that looks for dense regions in the data. Unlike *k*-means, it does not require us to specify the number of clusters in advance. Instead, we specify two parameters: the radius ε, which defines a neighborhood around an example, and $minPts$, the smallest number of examples that must be found in this neighborhood for the region to be considered dense.

For an example x, let $N_\varepsilon(x)$ denote the set of examples whose distance from x is at most ε. DBSCAN distinguishes between three kinds of examples:

- a *core point*, for which $|N_\varepsilon(x)| \geq minPts$;
- a *border point*, which is not a core point, but lies in the neighborhood of a core point;
- an *outlier*, or a noise point, which is neither a core point nor a border point.

The algorithm can be described as follows:

1. For every example x, compute its ε-neighborhood $N_\varepsilon(x)$.
2. Mark x as a *core point* if $|N_\varepsilon(x)| \geq minPts$.
3. Build clusters from the core points. Start with one core point and put it into a new cluster. Then add all core points that are within distance ε of any core point already in the cluster. Repeat this until no more core points can be added. Then start a new cluster from another unused core point.
4. Add each non-core point to a cluster if it lies within distance ε of a core point from that cluster. Such points are called *border points*.
5. Points that are neither core points nor border points are labeled as *noise*.

The main advantage of DBSCAN is that it can find clusters of irregular shape and can explicitly identify outliers. Its main weakness is the choice of ε and $minPts$: if ε is too small, many points become noise; if it is too large, distinct clusters may be merged. DBSCAN also works less well when different parts of the data have very different densities.

Clustering on Networks

In clustering on networks, we first represent the data as a graph. The nodes of the graph are examples, and edges connect examples that are similar or related. If the data are already a network, this graph is given. If the data are described by attributes, we can first construct a graph, for example by connecting each example to its nearest neighbors or by connecting examples whose similarity is above some threshold.

One simple method for finding groups in such a graph is *label propagation*. The algorithm is as follows:

1. Assign a different label to each node.
2. Visit the nodes in some order.
3. For each node, inspect the labels of its neighbors.
4. Replace the node's label with the label that is most common among its neighbors.
5. Repeat the procedure until the labels no longer change substantially.

Nodes with the same final label form a cluster, or community. The method is very fast and does not require us to choose the number of clusters in advance. Its weakness is that the result can depend on the order in which nodes are visited and on how ties between labels are resolved.

A more frequently used method is the *Louvain* method. It searches for communities by increasing *modularity*, a score that is high when there are many edges within communities and relatively few edges between them. The algorithm alternates between two steps:

1. Move individual nodes between neighboring communities whenever this increases modularity.
2. Replace each discovered community by a single new node and build a smaller graph whose nodes are these communities.

These two steps are repeated on the smaller and smaller graphs until modularity no longer improves.

Network clustering is therefore useful when similarity between examples is best expressed through relations. Its advantages are that it can discover groups of irregular shape and works well on large graphs. Its disadvantages are that the result depends on how the graph is constructed and, in methods such as Louvain, on parameters that influence the size and number of communities.

The Louvain method was proposed by Blondel et al. (2008) in *Journal of Statistical Mechanics: Theory and Experiment*. It is named after the University of Louvain. Its best-known modern successor is the Leiden algorithm, proposed by Traag et al. (2019) in *Scientific Reports*, which improves Louvain by refining communities and guaranteeing better-connected clusters. Other variants adapt these ideas to weighted, directed, multilayer, temporal, or resolution-parameterized networks.

Probabilistic Clustering

Most clustering methods we have mentioned so far assign each example to exactly one cluster. Such clustering is called *crisp* or *hard* clustering. In probabilistic clustering, the result is different: for each example, we estimate the probability that it belongs to each cluster. An example can therefore belong partly to several clusters.

A typical probabilistic clustering method is the *Gaussian mixture model*. It assumes that the data were generated from a mixture of several Gaussian distributions. Each Gaussian distribution corresponds to one cluster. The model therefore has, for each cluster, a mean vector, a covariance matrix, and a weight that tells us how common this cluster is in the data.

The algorithm most often used to fit such a model is the *expectation–maximization* algorithm, or EM. If the number of clusters is K, the algorithm proceeds as follows:

1. Initialize the parameters of the K Gaussian distributions.
2. For each example, estimate the probability that it belongs to each Gaussian component. This is the expectation step.
3. Using these probabilities as weights, update the means, covariances, and mixture weights of the Gaussian components. This is the maximization step.
4. Repeat the expectation and maximization steps until the parameters no longer change substantially.

The output is not only a partition of the data, but also a probability distribution over clusters for each example. If a crisp clustering is needed, we can assign each example to the cluster with the largest probability.

The advantage of Gaussian mixture models is that they provide a softer and more informative clustering than methods such as k-means. They can also model elliptical clusters through the covariance matrices. Their disadvantages are that the number of components K must be chosen in advance, the result can depend on initialization, and the method assumes that clusters are reasonably well described by Gaussian distributions.

From Clusters to Explanations

Clustering is useful only when the discovered groups can be interpreted. A cluster should therefore be treated as a hypothesis: it suggests that some examples belong together, but we still need to explain why.

In this chapter we saw several ways to do this. We can test whether known groups are enriched in a cluster, rank continuous features by how strongly they distinguish the cluster from the rest, examine categorical features with contingency tables, and summarize explanations through groups of related variables. In all these cases, domain knowledge is essential: it turns numerical patterns into meaningful descriptions.

Different clustering methods define groups in different ways. Some produce compact partitions, some find dense regions, some work on networks, and some assign examples to clusters probabilistically. For this reason, clustering should not end with a list of labels or a visualization. It should end with an explanation of what the groups represent. Such explanations are best supported within interactive environments, where we can select groups, inspect features, drill down to examples, and immediately see how the explanation changes. This kind of exploration, however, can lead to cherry-picking, overfitting, and overly convenient interpretations. But exploration is also where good questions often begin. Data science should remain interesting and fun, but it should also be careful, transparent, and statistically rigorous.