MDM-2025 Homework 2

Mert Dagli (100830277)
Tuomas Mäkinen (103382573)
Maja Sellmer (103396682)

October 13, 2025

1 Methods

The task was to discover interesting patterns in a dataset describing birds. To this end, we extracted features from the given dataset, searched for associations using the Kingfisher algorithm and evaluated the most significant rules. We completed the assignment in Python, using the following libraries:

- `pandas` to handle the data as a dataframe
- `numpy` for calculations

Furthermore, we relied on the Kingfisher program provided for the assignment. We ran the program with the parameters `-k167 -M-1 -q30`. This was chosen to get more versatile rules, not just pertaining to one aspect of the data.

2 Feature extraction

The data had to be preprocessed first to extract useful features.

To start with, for numerical features given as a range we consider the midpoint instead. As seen in the previous assignment, we introduce the indices $BMI = \frac{weight}{length^2}$ and $WSI = \frac{wingspan}{length}$. We then drop the features *weight*, *length*, and *wingspan*.

For the remaining numerical features, i.e. $F \in \{BMI, WSI, AR, wload, eggs\}$, we calculate the mean μ and standard deviation σ and introduce new attributes low-F (true iff $F \leq \mu - \sigma$) and high-F (true iff $F \geq \mu + \sigma$).

For most binary features, we use only the Yes-values, except for the one describing whether male or female birds of this species look similar or different, where both values are interesting, so we get two attributes *similar* and *nonsimilar*.

For the migration month ranges, we manually chose the following categories: early-arrive (those who arrive before April), late-arrive (those who arrive only after April), early-leave (those who leave before August), late-leave (those who leave only after August).

3 Results

Table 3 shows some of the associations we discovered. Note that these are not all the associations, but merely a selection meant to highlight some interesting points.

Rule	**fr**	**cf**	γ	δ	**M**
fish C-ftype → diver	0.14	1.000	5.556	0.115	-14.84
B-ftype webbed-feet → plunge-dives	0.10	1.000	8.333	0.088	-12.77
high-AR B-ftype → plunge-dives	0.10	1.000	8.333	0.088	-12.77
plunge-dives → B-ftype	0.10	0.833	5.952	0.083	-9.77
F-ccare → F-incub	0.24	1.000	2.778	0.154	-15.69
similar both-ccare → both-incub	0.56	0.875	1.411	0.163	-13.83
nonsimilar → F-ccare	0.16	1.000	2.778	0.128	-12.37
both-incub → similar	0.62	1.000	1.250	0.124	-11.62
both-incub → both-ccare	0.56	0.903	1.328	0.138	-10.56
high-eggs → F-ccare	0.16	0.727	3.030	0.107	-9.062
F-ccare plants → high-eggs	0.16	1.000	4.545	0.125	-15.00
F-ccare high-eggs → plants	0.16	1.000	3.846	0.118	-12.94
F-incub high-eggs → plants	0.16	1.000	3.846	0.118	-12.94
F-incub plants → high-eggs	0.16	0.889	4.040	0.120	-12.86
low-eggs fish → high-AR	0.12	0.857	5.357	0.098	-11.34
low-eggs lakes → fish	0.12	0.857	5.357	0.098	-11.34
lakes fish → high-AR	0.14	0.700	4.375	0.108	-11.62
lakes → webbed-feet	0.30	0.882	2.005	0.150	-11.90
long-legs → wading-bird	0.18	1.000	2.381	0.104	-9.051
both-incub fish webbed-feet → high-AR	0.16	0.727	4.545	0.125	-15.00
both-ccare fish webbed-feet → high-AR	0.16	0.727	4.545	0.125	-15.00
similar fish webbed-feet → high-AR	0.16	0.727	4.545	0.125	-15.00
accipitridae → A-ftype	0.16	1.000	5.556	0.131	-17.90
scolopacidae → invertebrates	0.16	1.000	2.778	0.102	-9.415
invertebrates long-billed → scolopacidae	0.10	1.000	6.250	0.084	-10.54
C-ftype wading-bird → scolopacidae	0.16	0.444	2.778	0.102	-9.415
invertebrates wading-bird → scolopacidae	0.16	0.667	4.167	0.122	-13.90

Table 1: Association rules with frequency, confidence, lift, leverage and mutual information

Some observations from the data:

- Birds with flying type C (continuous flapping) are likely to be good divers, and birds with flying type B (flapping and gliding) to be plunge-divers.
- There is a very interesting correlation between the appearance and roles of the sexes. If females and males look different, it is likely the female who incubates and takes care of the chicks. However, if they look similar, it is more likely that both will incubate and care for the chicks. Also, if the bird lays many eggs, it is more likely the female will care for them.
- Interestingly, there seems to also be some correlation between eating plants and laying many eggs, and eating fish and laying fewer eggs.
- There are some obvious relations between morphology and type of bird or habitat, for example birds living in lakes are likely to have webbed feet and birds with long legs tend to be wading birds.
- There are also some significant rules related to specific families, for example long-billed wading birds eating invertebrates are likely to be scolopacidae.

We did not include colours in searching for rules as they were too specific to be helpful. We did include diet and habitat, but many items did not appear in the dataset frequently enough to be part of any rules. The migration times also did not appear in any rules, indicating that either our categories were not helpful or these have little connection with the other attributes in the dataset.

4 Appendix – Code

```
import pandas as pd
import numpy as np

def month_to_num(month_str):
    months = {
        "january": 1, "february": 2, "march": 3, "april": 4, "may": 5, "june": 6,
        "july": 7, "august": 8, "september": 9, "october": 10, "november": 11, "
            december": 12}

    if not isinstance(month_str, str) or month_str.strip() == '':
        return None

    parts = [m.strip().lower() for m in month_str.split('-') if m.strip()]
    if len(parts) == 1:
        return months.get(parts[0])
    elif len(parts) == 2:
        a, b = months.get(parts[0]), months.get(parts[1])
        if a and b:
            return (a + b) / 2
    return None


def migration_times(df):
    df = df.copy()

    def arrival_feature(month_num):
        if month_num is None:
            return None
        if month_num <= 3.5:
            return "early-arrival"
        elif 5<= month_num :
            return "late-arrival"
        else:
            return None
    def leaving_feature(month_num):
        if month_num is None:
            return None
        if month_num <= 8:
            return "early-leave"
        elif 10 <= month_num:
            return "late-leave"
        else:
            return None

    df["arrives"] = df["arrives"].apply(month_to_num).apply(arrival_feature)
    df["leaves"] = df["leaves"].apply(month_to_num).apply(leaving_feature)

    return df

def midpoint(value):
        if isinstance(value, str) and '-' in value:
            try:
                a, b = map(float, value.split('-'))
                return (a + b) / 2
            except ValueError:
                return value
        try:
            return float(value)
        except ValueError:
            return value

def midpoint_column(df, features=['length', 'wspan', 'weight','eggs']):
    df_mid = df.copy()
    for feature in features:
        df_mid[feature] = df_mid[feature].apply(midpoint)
    return df_mid

def add_bmi(df):
    """
    Adds a new column 'bmi' = weight / length^2
    """
    new_df = df.copy()
    new_df['bmi'] = new_df["weight"] / (new_df["length"] ** 2)
    return new_df


def add_wsi(df):
    """
    Adds a new column 'wsi' = wspan / length
    """
    new_df = df.copy()
    new_df['wsi'] = new_df["wspan"] / new_df["length"]
    return new_df

def binary_features(df, features=['diver', 'long-billed', 'webbed-feet', 'long-
    legs', 'wading-bird', 'plunge-dives', 'sim']):
    df = df.copy()
    for feature in features:
        if feature in df.columns:
            if feature=='sim':
                 df[feature] = df[feature].str.strip().str.lower().replace({
                'yes': 'similar',
                'no': 'nonsimilar'
            })

            df[feature] = df[feature].str.strip().str.lower().replace({
                'yes': feature,
                'no': None
            })
    return df

def numerical_features_std(df, features=['bmi', 'wsi', 'AR', 'wload', 'eggs']):
    df = df.copy()

    for feature in features:
        if feature in df.columns:
            mean = df[feature].mean()
            std = df[feature].std()
            high = mean + std
            low = mean - std

            df[feature] = df[feature].apply(
                lambda x: f'high-{feature}' if x >= high
                          else (f'low-{feature}' if x <= low
                                else None)
            )
    return df

def multivalue_features(df, features=[]):
    df = df.copy()
    for feature in features:
        if feature in df.columns:
            df[feature] = df[feature].str.replace(" ", "", regex=False)
            df[feature] = df[feature].apply(lambda x: f"{x}-{feature}" if x and x.
                lower() != 'nan' else None)
    return df

def feature_extraction(df):
    "This function applies all the other functions to compute the feature
        extraction."
    df = midpoint_column(df)
    df = add_bmi(df)
    df = add_wsi(df)
    df = df.drop(columns=['weight', 'length', 'wspan','species'])
    df = binary_features(df)
    df = numerical_features_std(df)
    df = multivalue_features(df, features=['belly','back', 'ftype', 'billcol', '
        legcol', 'incub', 'ccare'])
    df = migration_times(df)
    return df

df = pd.read_csv("birds2025ext.csv", sep=';')
df = feature_extraction(df)

def create_transaction_file(df, filename="features.txt"):
    with open(filename, "w") as f:
        for _, row in df.iterrows():
            transaction = []
            for value in row:
                if value != None:
                    value = str(value).replace(" ", "")
                    parts = [v.strip() for v in str(value).split(',') if v.strip()
                        ]
                    transaction.extend(parts)
            if transaction:
                f.write(" ".join(transaction) + "\n")

create_transaction_file(df, 'features.txt')
```

5 Bibliography

References

[1] Charu C. Aggarwal. *Data Mining - The Textbook*. Springer, 2015.