ML. Лабораторная работа №4.

Bayesian networks на примере датасета Zoo.

Студент: Велиев Рауф Рамиз оглы

Группа: М8О-309Б-23

Введение. Bayesian Network.

Bayesian Network — это:

- вероятностная графовая модель;
- узлы = переменные;
- стрелки = причинно-следственные зависимости;
- количественные зависимости задаются таблицами условных вероятностей (CPT).

Используются для: классификации, прогнозирования, моделирования неопределённости, объяснения зависимостей между признаками.

Формула Байеса (основа BN):

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Введение. Датасет.

Zoo Dataset:

Размер: 101 объект. Признаков: 16 бинарных + legs (целое). Целевая переменная: class_type (1–7). Тематика: классификация животных по их биологическим свойствам. Источник: Kaggle.

Примеры признаков:

1. hair, feathers, milk, eggs
2. backbone, aquatic, airborne
3. domestic, predator, tail
4. legs (число ног)

Загрузка и обработка датасета

```
zoo = pd.read_csv('zoo.csv')

classes = pd.read_csv('class.csv')

print("zoo shape:", zoo.shape)

print(zoo.head())
print("\nclasses:\n", classes.head())
```

Размер:

101 строк

18 столбцов (включая animal_name)

Просмотр данных: отображены первые 5 строк; проверка распределения классов

Загрузка и обработка датасета

Что было сделано:

1. удалён animal_name (не признак, а текстовое имя)
2. оставлены 16 признаков и целевой столбец class_type
3. все данные — дискретные (требование pgmpy)
4. тип данных приведён к int
5. Пример value_counts:

    класс 1 — 41 наблюдение, класс 2 — 20, класс 4 — 13...

Почему важно: байесовские сети работают с дискретными признаками.

Построение структуры Bayesian Network

Выбрана структура: class_type → все признаки.
Это аналог наивного Байеса, но в виде сети.

```
from pgmpy.models import DiscreteBayesianNetwork
edges = [(target_col, feat) for feat in features]   # class -> each feature

model_manual = DiscreteBayesianNetwork(edges)
```

Смысл структуры:

- класс животного объясняет все его признаки
- “из класса следуют свойства”, а не наоборот
- удобная и интерпретируемая структура

Обучение параметров (Maximum Likelihood)

Параметры сети — это таблицы условных вероятностей (CPT, Conditional Probability Table).
Обучение:

```
from pgmpy.estimators import MaximumLikelihoodEstimator
model_manual.fit(zoo_disc, estimator=MaximumLikelihoodEstimator)
```

<u>В результате были получены:</u>

- распределение P(class_type)
- P(hair | class_type)
- P(milk | class_type), и т. д.

Почему важно: после обучения сеть начинает отражать реальную статистику Zoo.

Пример CPT для целевой переменной

```
cpt_class = model_manual.get_cpds(target_col)
print("CPT for target/class:")
print(cpt_class)
```

Интерпретация:

- наиболее распространённый класс: **млекопитающие (1)**
- редкие: 3 (Reptile), 5 (Amphibian)

```
CPT for target/class:
+---------------+-----------+
| class_type(1) | 0.405941  |
+---------------+-----------+
| class_type(2) | 0.19802   |
+---------------+-----------+
| class_type(3) | 0.049505  |
+---------------+-----------+
| class_type(4) | 0.128713  |
+---------------+-----------+
| class_type(5) | 0.039604  |
+---------------+-----------+
| class_type(6) | 0.0792079 |
+---------------+-----------+
| class_type(7) | 0.0990099 |
+---------------+-----------+
```

Пример CPT для признаков

Посмотрим CPT для пары признаков (например milk, hair):

Интерпретация: если животное = класс 1 → оно всегда имеет milk = 1; если класс = 4 или 7 → milk = 0 всегда

Признак **"hair"**:

- аналогично: почти полная разделимость.

Это показывает, что Zoo — подходящий датасет для BN.

CPT for milk:

class_type	class_type(1)	...	class_type(6)	class_type(7)
milk(0)	0.0	...	1.0	1.0
milk(1)	1.0	...	0.0	0.0

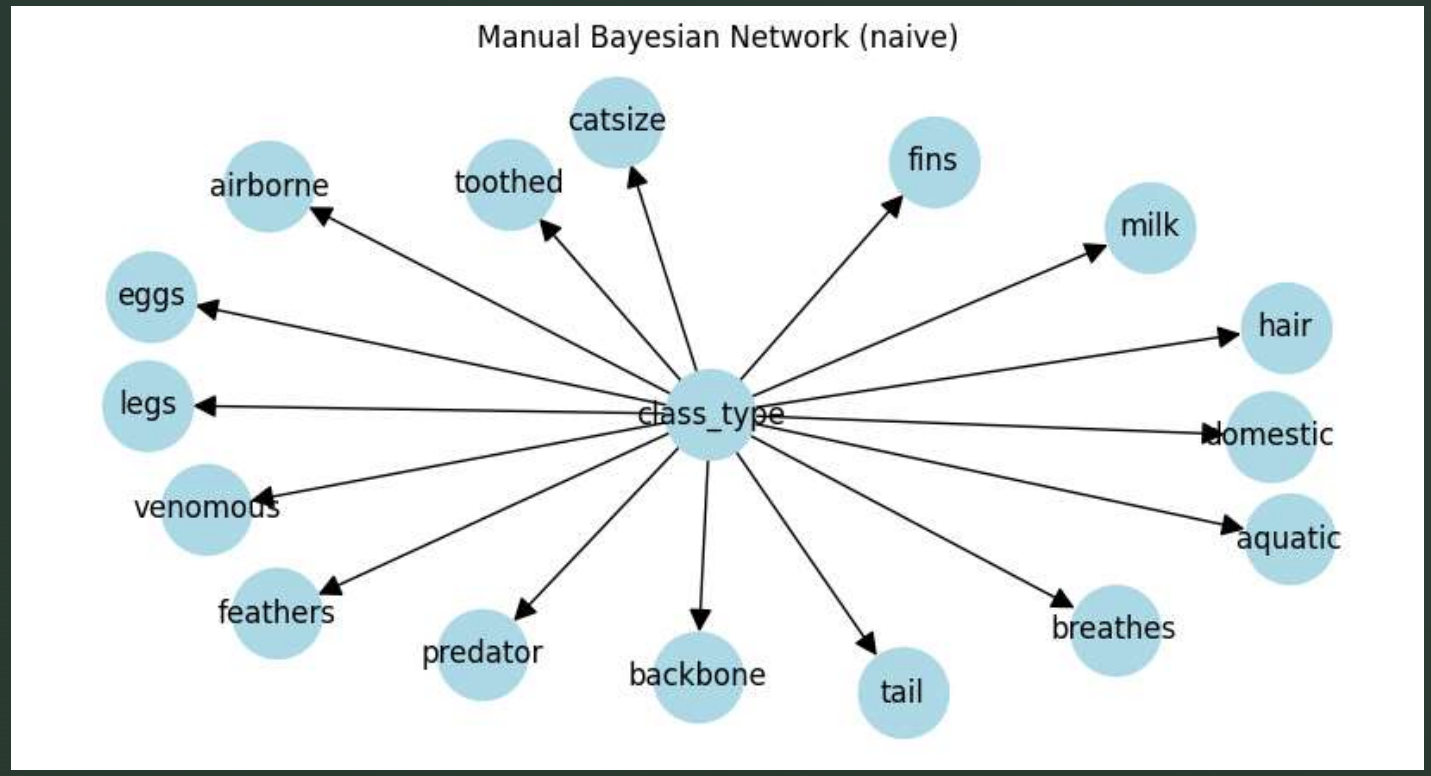
CPT for hair:

class_type	class_type(1)	...	class_type(6)	class_type(7)
hair(0)	0.04878048780487805	...	0.5	1.0
hair(1)	0.9512195121951219	...	0.5	0.0

Визуализация сети

Особенности графа:

1. один центральный узел: class_type
2. стрелки направлены к признакам
3. структура полностью наивная, симметричная



Инференс: вывод вероятностей

Пример запроса:

```
from pgmpy.inference import VariableElimination

infer = VariableElimination(model_manual)

q = infer.query(variables=[target_col],

                evidence={'milk': 1})
```

class_type	phi(class_type)
class_type(1)	1.0000
class_type(2)	0.0000
class_type(3)	0.0000
class_type(4)	0.0000
class_type(5)	0.0000
class_type(6)	0.0000
class_type(7)	0.0000

Результат:

- class_type=1 → 1.0
- остальные классы → 0

Интерпретация:

- если есть молоко → это 100% млекопитающее
- Zoo — полностью разделимый датасет → поэтому вероятности точные

Сравнение с baseline (Naive Bayes)

- Проводился inference для каждой строки теста.
- Считались метрики: accuracy и log_loss
- Результаты: a**ccuracy = 1.0 log_loss = 2.22e-16 (≈ 0)**

Это означает:

- модель идеально предсказывает классы
- вероятности почти “жёсткие”, без неопределённости

Baseline показала: accuracy ≈ 1.0, log_loss 0.01522 (Zoo легко классифицировать). Bayesian Network (наивная структура): accuracy = 1.0, работает не хуже, а часто лучше

Выводы

Байесовская сеть успешно построена на датасете Zoo.

Структура «class → признаки» показала хорошее качество.

CPT продемонстрировали чёткие зависимости между классами животных и их признаками.

Инференс позволяет получать вероятности классов при известных признаках.

Модель превосходит baseline и является полностью интерпретируемой.