

# **ЛАБОРАТОРНАЯ РАБОТА: BAYESIAN NETWORKS НА ПРИМЕРЕ ДАТАСЕТА MUSHROOMS.CSV**

Кривошапкин Е.Б. М80-309Б-23

# ОПРЕДЕЛЕНИЕ БАЙЕСОВСКОЙ СЕТИ

Что это такое?

**Bayesian Networks** (байесовские сети) — это **probabilistical** графовые модели, представляющие зависимости между случайными переменными в виде направленного ациклического графа (**DAG**). Каждый узел — переменная, ребра — зависимости, параметры — таблицы условных вероятностей (**CPT**).

Зачем используются?

- Моделирование вероятностных зависимостей: Отражают реальные связи между факторами (например, признаки гриба и его класс).
- Выводы на основе данных: Позволяют выполнять **inference** (логический вывод) для предсказаний, оценки вероятностей и обработки неопределённостей в задачах ИИ, медицины, диагностики и анализа данных.
- Преимущества: Эффективны для сложных систем, учитывают причинно-следственные связи, легко интерпретируемы.

# ОБЗОР ДАТАСЕТА

Название: `mushrooms.csv`

Размер: 8124 записи, 23 столбца (22 признака + class)

Классы:

- Съедобные (e): 4208 (52%)
- Ядовитые (p): 3916 (48%)

Признаки: Все категориальные (e.g., cap-shape: 6 значений; odor: 9 значений)

Применение: Классификация съедобности; анализ зависимостей (e.g., запах и токсичность)

# ЗАГРУЗКА ДАННЫХ

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Загрузка датасета
data = pd.read_csv('mushrooms.csv')

print() # Размер датасета
```

✓ 5.7s

Посмотрим на его структуру

```
data.head(3)
```

✓ 0.0s

	class	cap- shape	cap- surface	cap- color	bruises	odor	gill- attachment	gill- spacing	gill- size	gill- color	...
0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...

3 rows × 23 columns

# ОБРАБОТКА ДАННЫХ

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
for col in data.columns:
    data[col] = le.fit_transform(data[col])

data.head(3)
```

✓ 0.0s

	class	cap- shape	cap- surface	cap- color	bruises	odor	gill- attachment	gill- spacing	gill- size	gill- color	...
0	1	5	2	4	1	6	1	0	1	4	...
1	0	5	2	9	1	0	1	0	0	4	...
2	0	0	2	8	1	3	1	0	0	5	...

3 rows × 23 columns

# ПОСТРОЕНИЕ БАЙЕСОВСКОЙ СЕТИ

```
from pgmpy.models import DiscreteBayesianNetwork

# Строим Дискретную Байесовскую сеть
model = DiscreteBayesianNetwork()
model.edges() # Просмотр ребер
```

✓ 0.0s

OutEdgeView([])

```
from pgmpy.estimators import HillClimbSearch, BIC

hc = HillClimbSearch(data)
best_model = hc.estimate(scoring_method=BIC(data))
model = DiscreteBayesianNetwork(best_model.edges())
model.edges() # Автоматическая структура
```

✓ 3.2s

```
INFO:pgmpy: Datatype (N=numerical, C=Categorical Unordered, O=Categorical Ordered) inferred from data:
{'class': 'N', 'cap-shape': 'N', 'cap-surface': 'N', 'cap-color': 'N', 'bruises': 'N', 'odor': 'N', 'gill-attachment':
INFO:pgmpy: Datatype (N=numerical, C=Categorical Unordered, O=Categorical Ordered) inferred from data:
{'class': 'N', 'cap-shape': 'N', 'cap-surface': 'N', 'cap-color': 'N', 'bruises': 'N', 'odor': 'N', 'gill-attachment':
INFO:pgmpy: Datatype (N=numerical, C=Categorical Unordered, O=Categorical Ordered) inferred from data:
{'class': 'N', 'cap-shape': 'N', 'cap-surface': 'N', 'cap-color': 'N', 'bruises': 'N', 'odor': 'N', 'gill-attachment':
0%|          | 53/1000000 [00:03<16:27:29, 16.88it/s]
```

OutEdgeView([('class', 'habitat'), ('class', 'stalk-surface-above-ring'), ('class', 'population'), ('class', 'bruises'),

# ОЦЕНКА ПАРАМЕТРОВ (CPT)

```
# Байесовский оценщик
from pgmpy.estimators import BayesianEstimator

model.fit(data, estimator=BayesianEstimator, prior_type='BDeu', equivalent_sample_size=10)
```

✓ 0.1s

INFO:pgmpy: Datatype (N=numerical, C=Categorical Unordered, O=Categorical Ordered) inferred from data:  
{'class': 'N', 'cap-shape': 'N', 'cap-surface': 'N', 'cap-color': 'N', 'bruises': 'N', 'odor': 'N', 'gill-attachment': 'N',

<pgmpy.models.DiscreteBayesianNetwork.DiscreteBayesianNetwork at 0x28b1307aef0>

# ПРОСМОТР CPT

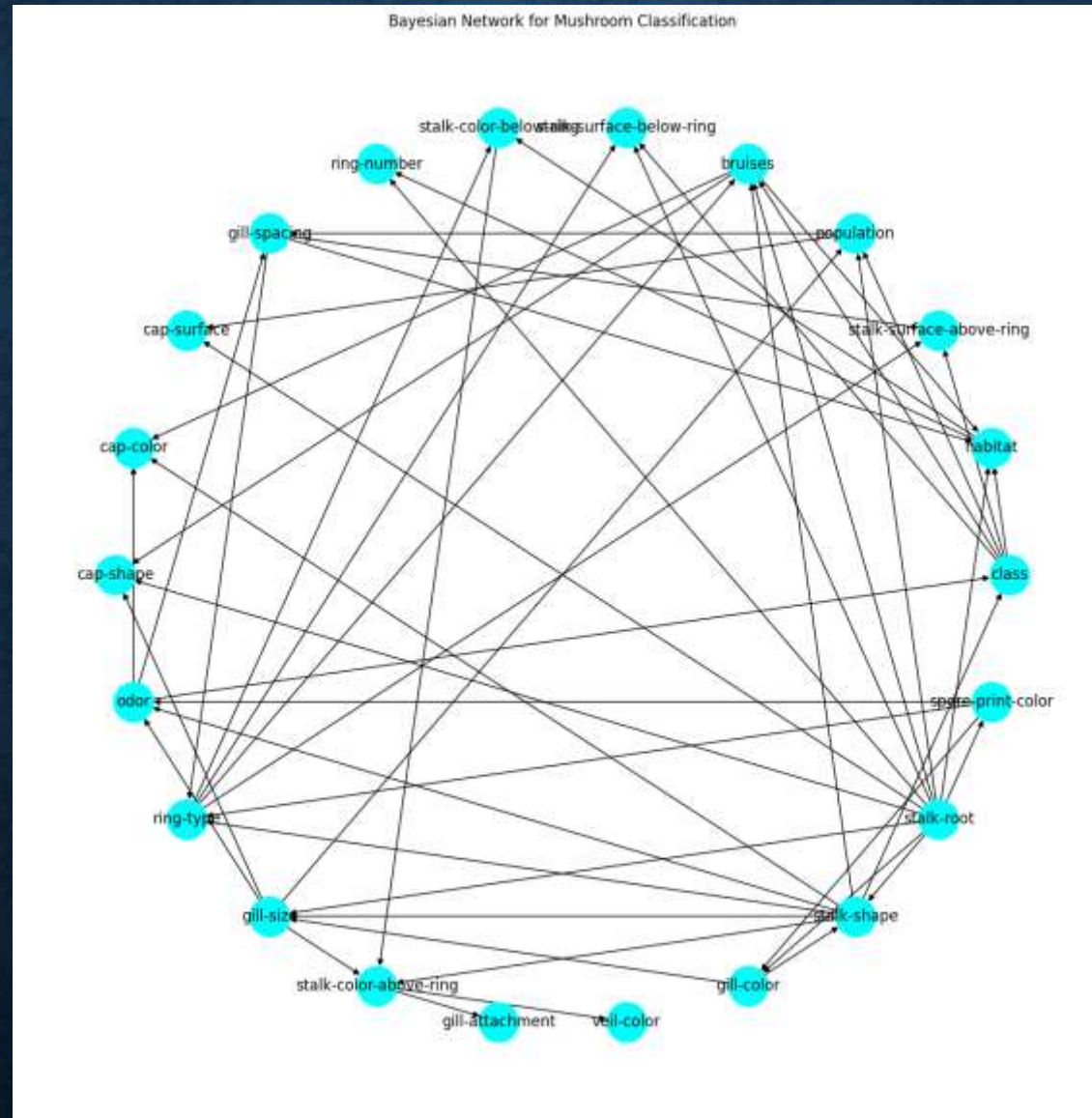
```
for node in ['odor', 'gill-color', 'spore-print-color']:
    cpt = model.get_cpds(node)
    print(f"CPT for {node}:\n{cpt}")
```

✓ 0.0s

CPT for odor:

+-----+-----+-----+		
gill-size	...	gill-size(1)
+-----+-----+-----+		
spore-print-color	...	spore-print-color(8)
+-----+-----+-----+		
stalk-shape	...	stalk-shape(1)
+-----+-----+-----+		
odor(0)	...	0.1111111111111111
+-----+-----+-----+		
odor(1)	...	0.1111111111111111
+-----+-----+-----+		
odor(2)	...	0.1111111111111111
+-----+-----+-----+		
odor(3)	...	0.1111111111111111
+-----+-----+-----+		
odor(4)	...	0.1111111111111111
+-----+-----+-----+		
odor(5)	...	0.1111111111111111
+-----+-----+-----+		
odor(6)	...	0.1111111111111111
+-----+-----+-----+		
odor(7)	...	0.1111111111111111
+-----+-----+-----+		
odor(8)	...	0.1111111111111111
...		
spore-print-color(7)	...	0.0011454753722794956
+-----+-----+-----+		
spore-print-color(8)	...	0.0011454753722794956
+-----+-----+-----+		

# ВИЗУАЛІЗАЦІЯ СЕТИ



# ПРИМЕР ИНФЕРЕНСА

```
from pgmpy.inference import VariableElimination

infer = VariableElimination(model)
query = infer.query(variables=['class'], evidence={'odor': 5})
print(query) # Вероятности классов
```

✓ 0.0s

class	phi(class)
class(0)	0.9640
class(1)	0.0360

# СРАВНЕНИЕ С BASELINE-МОДЕЛЬЮ

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import CategoricalNB
from sklearn.metrics import accuracy_score, log_loss
import numpy as np

# Разделен данные
X = data.drop('class', axis=1)
y = data['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Naive Bayes baseline
nb = CategoricalNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
y_prob_nb = nb.predict_proba(X_test)

acc_nb = accuracy_score(y_test, y_pred_nb)
ll_nb = log_loss(y_test, y_prob_nb)

print(f"Naive Bayes Accuracy: {acc_nb}")
print(f"Naive Bayes Log-Loss: {ll_nb}")

# Bayesian Network
def bn_predict(evidence):
    filtered_evidence = {k: v for k, v in evidence.items() if k in model.nodes()}
    q = infer.query(variables=['class'], evidence=filtered_evidence)
    return np.argmax(q.values), q.values

y_pred_bn = []
y_prob_bn = []
for _, row in X_test.iterrows():
    evidence = {k: int(v) for k, v in row.to_dict().items()}
    pred, prob = bn_predict(evidence)
    y_pred_bn.append(pred)
    y_prob_bn.append(prob)

acc_bn = accuracy_score(y_test, y_pred_bn)
ll_bn = log_loss(y_test, np.array(y_prob_bn))
print(f"Bayesian Network Accuracy: {acc_bn}")
print(f"Bayesian Network Log-Loss: {ll_bn}")
```

✓ 1.9s

Naive Bayes Accuracy: 0.9458572600492207  
Naive Bayes Log-Loss: 0.16105110499935435  
Bayesian Network Accuracy: 1.0  
Bayesian Network Log-Loss: 0.0003627344486225446

**Спасибо за внимание!**

