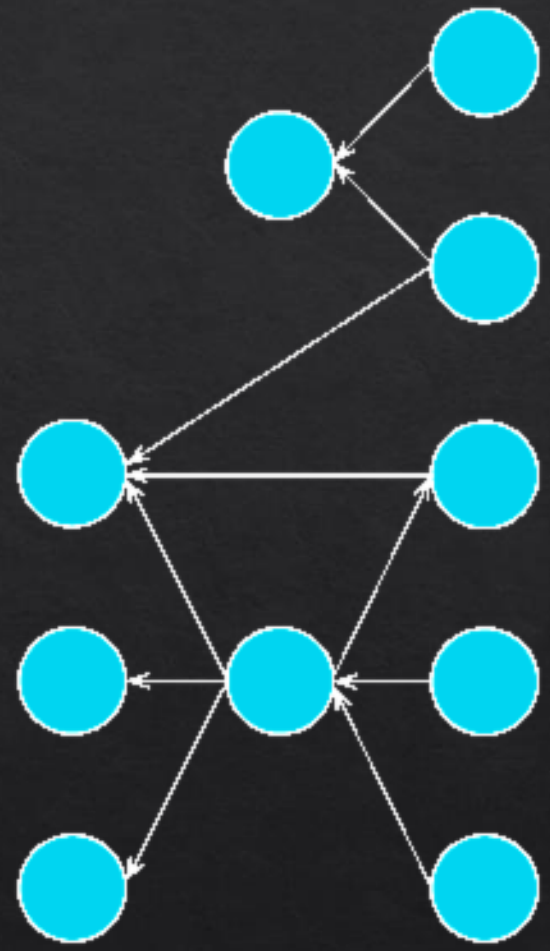
Баессовские сети на примере датасета mushrooms.csv

Дорофеев Демид Сергеевич

Введение

- **Байесовская сеть** (также известна как сеть Байеса или сеть доверия) — **графовая вероятностная модель**, которая представляет собой множество переменных и их вероятностные зависимости по Байесу.



Датасет mushrooms.csv

Атрибуты:

- ядовитость, форма шляпки, поверхность шляпки, цвет шляпки,
- синяки, запах, прикрепление жабр, расстояние между жабрами,
- размер жабр, цвет жабр, форма стебля, стебель-корень,
- поверхность стебля над кольцом, поверхность стебля под кольцом,
- цвет стебля над кольцом, цвет стебля под кольцом, тип завесы,
- цвет завесы, кольцо- количество, тип кольца, цвет отпечатка спор,
- популяция, среда обитания

Всего 23 атрибутов

Загружаем датасет на прямую с KuggleHub

```python
import kagglehub
import pandas as pd
from pathlib import Path

path = Path(kagglehub.dataset_download("uciml/mushroom-classification"))
print("Path to dataset files:", path)

file_dir = path / 'mushrooms.csv'
data = pd.read_csv(file_dir, header=None,
                   names=['class', 'cap-shape', 'cap-surface',
                          'cap-color', 'bruises','odor',
                          'gill-attachment', 'gill-spacing','gill-size',
                          'gill-color','stalk-shape','stalk-root',
                          'stalk-surface-above-ring','stalk-surface-below-ring',
                          'stalk-color-above-ring','stalk-color-below-ring',
                          'veil-type','veil-color','ring-number','ring-type',
                          'spore-print-color','population','habitat'])
data = data.drop(0)
RND_SEED=7123654
data.shape, data.size
```

Label Encoding

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

label_encoders = {}
mappings = {}

for col in data.columns:
    le = LabelEncoder()
    original_values = data[col].unique()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le
    mappings[col] = dict(zip(le.classes_, le.transform(le.classes_)))

for col, mapping in mappings.items():
    print(f"Столбец '{col}':")
    for value, encoded in mapping.items():
        print(f"  {value} -> {encoded}")
    print()

data.head(5)
```

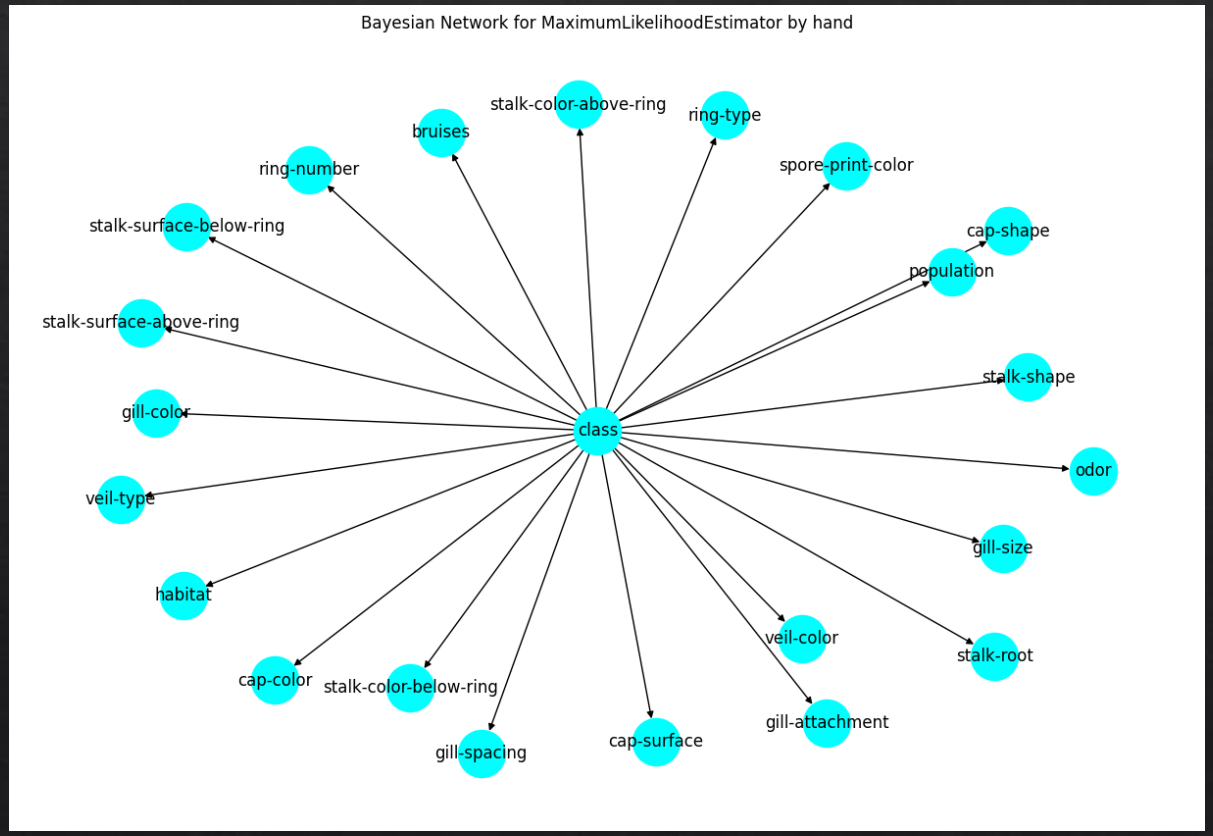
Построение Баессовской сети

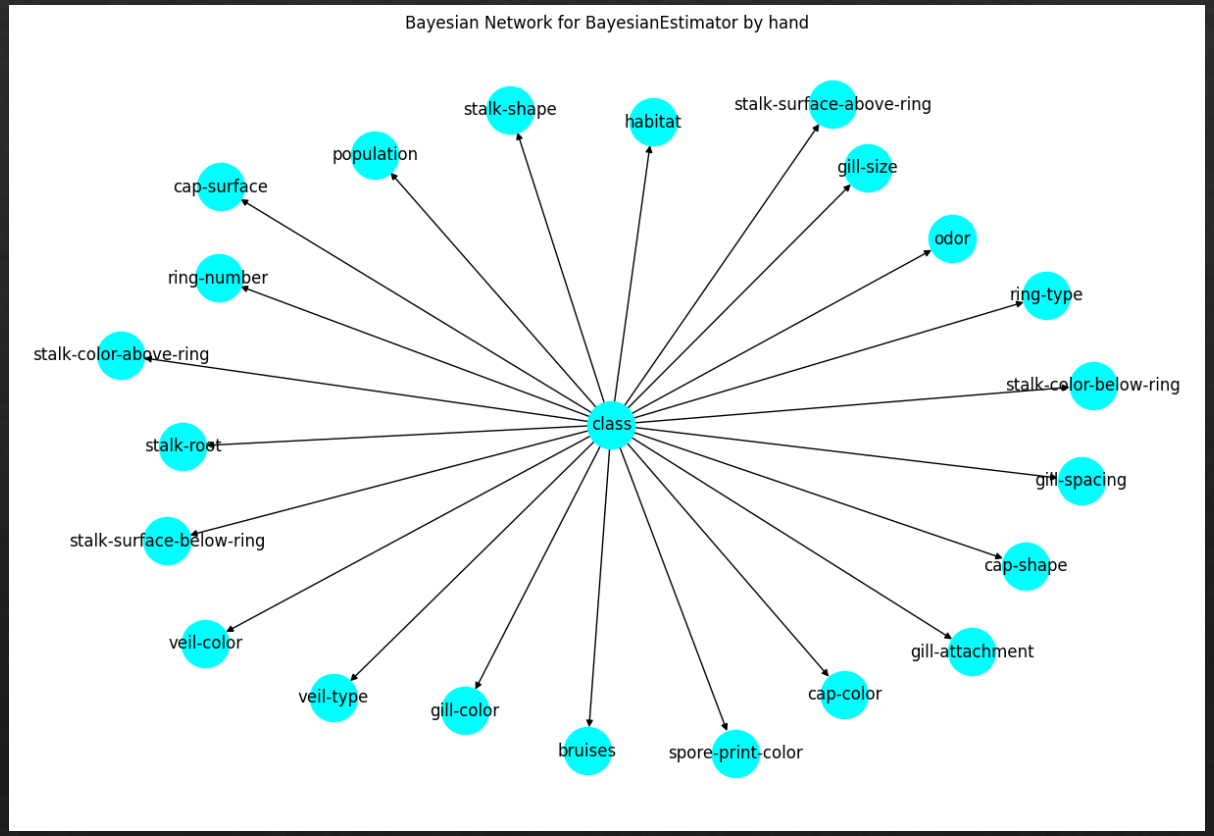
```python
network = [
    ('class', 'cap-shape'),
    ('class', 'cap-surface'),
    ('class', 'cap-color'),
    ('class', 'bruises'),
    ('class', 'odor'),
    ('class', 'gill-attachment'),
    ('class', 'gill-spacing'),
    ('class', 'gill-size'),
    ('class', 'gill-color'),
    ('class', 'stalk-shape'),
    ('class', 'stalk-root'),
    ('class', 'stalk-surface-above-ring'),
    ('class', 'stalk-surface-below-ring'),
    ('class', 'stalk-color-above-ring'),
    ('class', 'stalk-color-below-ring'),
    ('class', 'veil-type'),
    ('class', 'veil-color'),
    ('class', 'ring-number'),
    ('class', 'ring-type'),
    ('class', 'spore-print-color'),
    ('class', 'population'),
    ('class', 'habitat'),
]
```

```python
# Строим Дискретную Байесовскую сеть
model_h_m = DiscreteBayesianNetwork(network)
model_h_b = DiscreteBayesianNetwork(network)
model_h_m.edges()  # Просмотр ребер

from sklearn.model_selection import train_test_split

X = data.drop(columns=['class'])
y = data['class']

# Разделение выборки на test/train (20/80)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=RND_SEED
)

hc = HillClimbSearch(data_train)
best_model = hc.estimate(scoring_method=BIC(data_train))
model_a_m = DiscreteBayesianNetwork(best_model.edges())
model_a_b = DiscreteBayesianNetwork(best_model.edges())
model_a_m.edges()  # Автоматическая структура

# Метод Максимального Правдоподобия
from pgmpy.estimators import MaximumLikelihoodEstimator

model_h_m.fit(data_train, estimator=MaximumLikelihoodEstimator)
model_a_m.fit(data_train, estimator=MaximumLikelihoodEstimator)
```

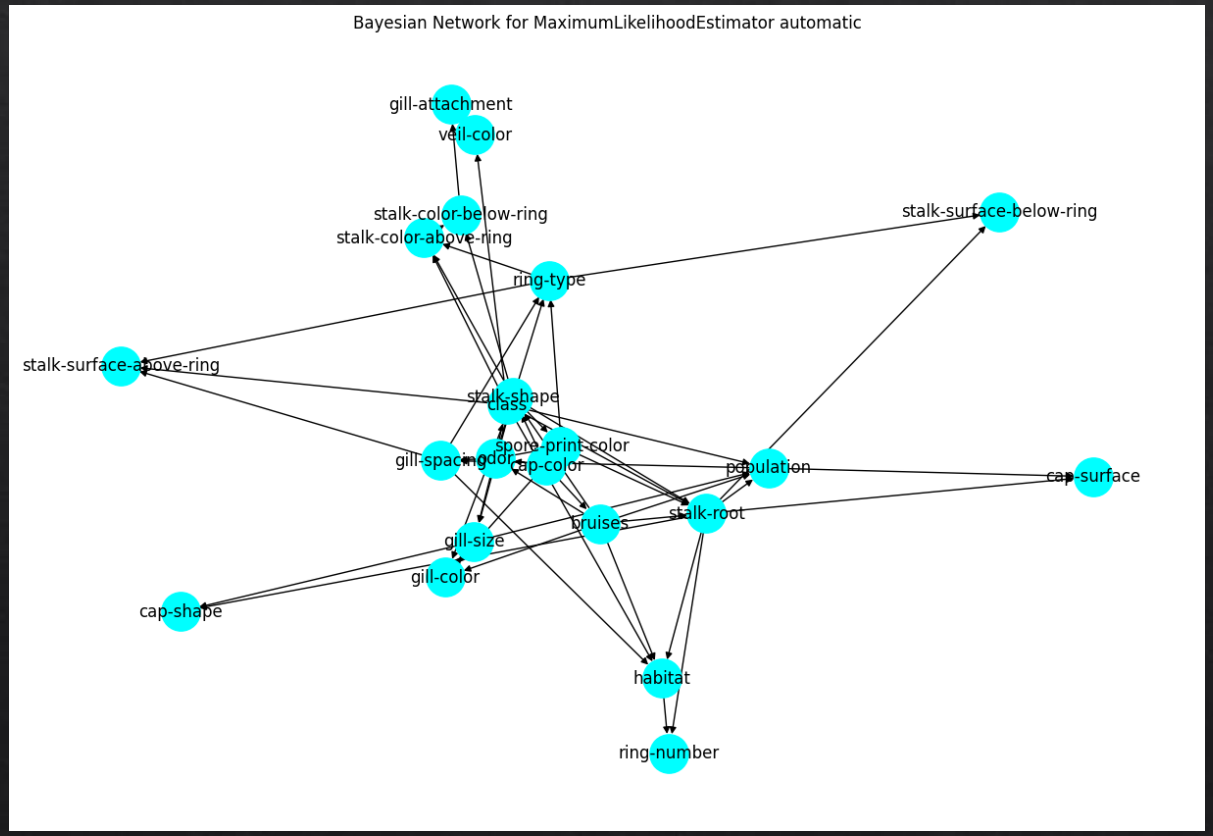
```python
# Байесовский оценщик
from pgmpy.estimators import BayesianEstimator

model_h_b.fit(data_train, estimator=BayesianEstimator,
              prior_type='BDeu', equivalent_sample_size=10)
model_a_b.fit(data_train, estimator=BayesianEstimator,
              prior_type='BDeu', equivalent_sample_size=10)
```

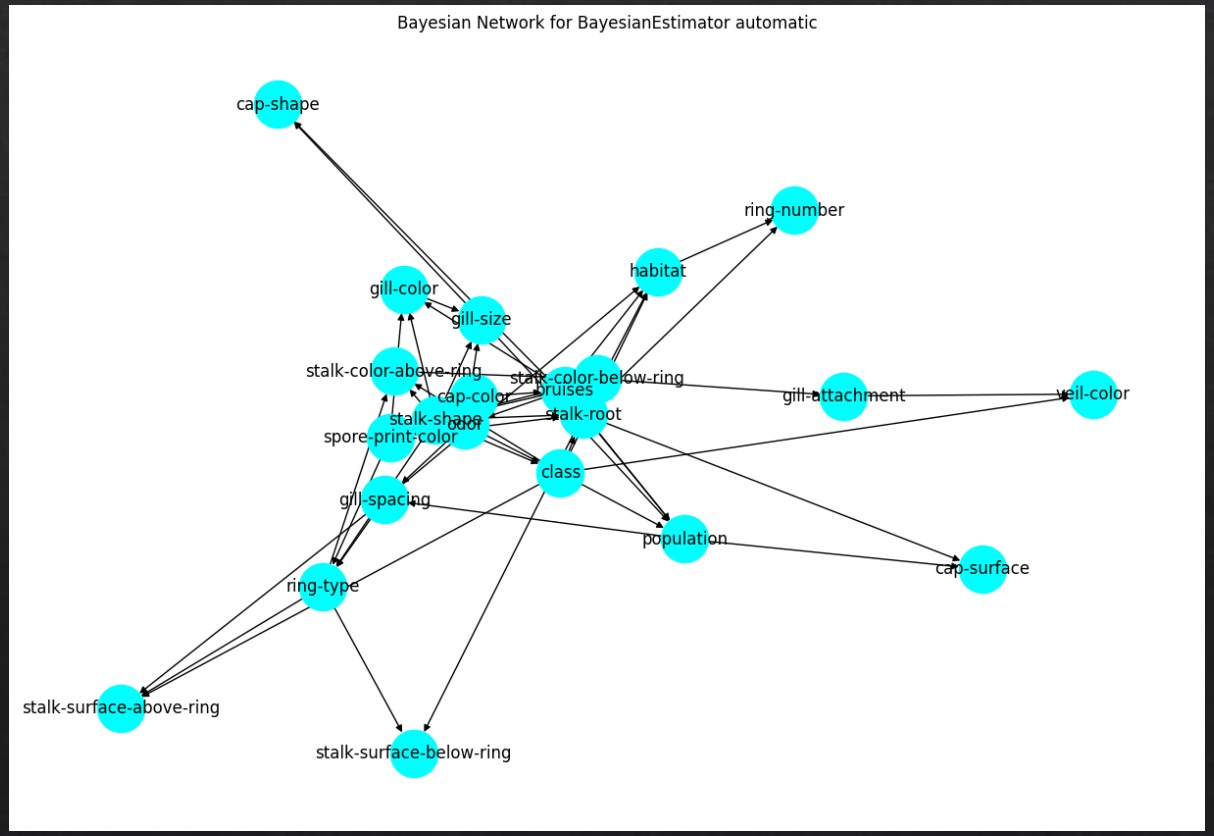
Визуализация (модели с ручными графами)





Визуализация (модели с автоматическими графами)





Инференс (разные значения одного параметра)

```python
from pgmpy.inference import VariableElimination

# black,brown,gray,pink,white,chocolate,purple,red,buff,green,yellow,orange,
# ['k' , 'n' , 'g', 'p', 'w' ,    'h'  ,  'u' ,'e', 'b', 'r' ,  'y' ,  'o' ]
# [ 4  ,  5  ,  2 ,  7 , 10  ,     3   ,   9  , 1 ,  0 ,  8  ,  11  ,   6  ]

infer = VariableElimination(model_h_m)
query = infer.query(variables=['class'], evidence={'gill-color': 0})
print("buff gill-color")
print(query)
query = infer.query(variables=['class'], evidence={'gill-color': 1})
print("red gill-color")
print(query)
query = infer.query(variables=['class'], evidence={'gill-color': 2})
print("gray gill-color")
print(query)
query = infer.query(variables=['class'], evidence={'gill-color': 3})
print("chocolate gill-color")
print(query)
```

```
buff gill-color
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       0.0000 |
+----------+--------------+
| class(1) |       1.0000 |
+----------+--------------+
red gill-color
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       1.0000 |
+----------+--------------+
| class(1) |       0.0000 |
+----------+--------------+
gray gill-color
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       0.3234 |
+----------+--------------+
| class(1) |       0.6766 |
+----------+--------------+
chocolate gill-color
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       0.2774 |
+----------+--------------+
| class(1) |       0.7226 |
+----------+--------------+
```

Инференс (один параметр в разных моделях)

```python
from pgmpy.inference import VariableElimination

# black,brown,gray,pink,white,chocolate,purple,red,buff,green,yellow,orange,
# ['k' , 'n' , 'g', 'p', 'w' ,    'h'  ,  'u' ,'e', 'b', 'r' ,  'y' ,  'o' ]
# [ 4  ,  5  ,  2 ,  7 , 10  ,     3   ,   9  , 1 ,  0 ,  8  ,  11  ,   6  ]

infer = VariableElimination(model_h_m)
query = infer.query(variables=['class'], evidence={'gill-color': 2})
print("model_h_m")
print(query)
infer = VariableElimination(model_h_b)
query = infer.query(variables=['class'], evidence={'gill-color': 2})
print("model_h_b")
print(query)
infer = VariableElimination(model_a_m)
query = infer.query(variables=['class'], evidence={'gill-color': 2})
print("model_a_m")
print(query)
infer = VariableElimination(model_a_b)
query = infer.query(variables=['class'], evidence={'gill-color': 2})
print("model_a_b")
print(query)
```

```
model_h_m
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       0.3234 |
+----------+--------------+
| class(1) |       0.6766 |
+----------+--------------+
model_h_b
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       0.3236 |
+----------+--------------+
| class(1) |       0.6764 |
+----------+--------------+
model_a_m
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       0.2880 |
+----------+--------------+
| class(1) |       0.7120 |
+----------+--------------+
model_a_b
+----------+--------------+
| class    |   phi(class) |
+==========+==============+
| class(0) |       0.2887 |
+----------+--------------+
| class(1) |       0.7113 |
+----------+--------------+
```

Сравнение результатов с baseline моделью

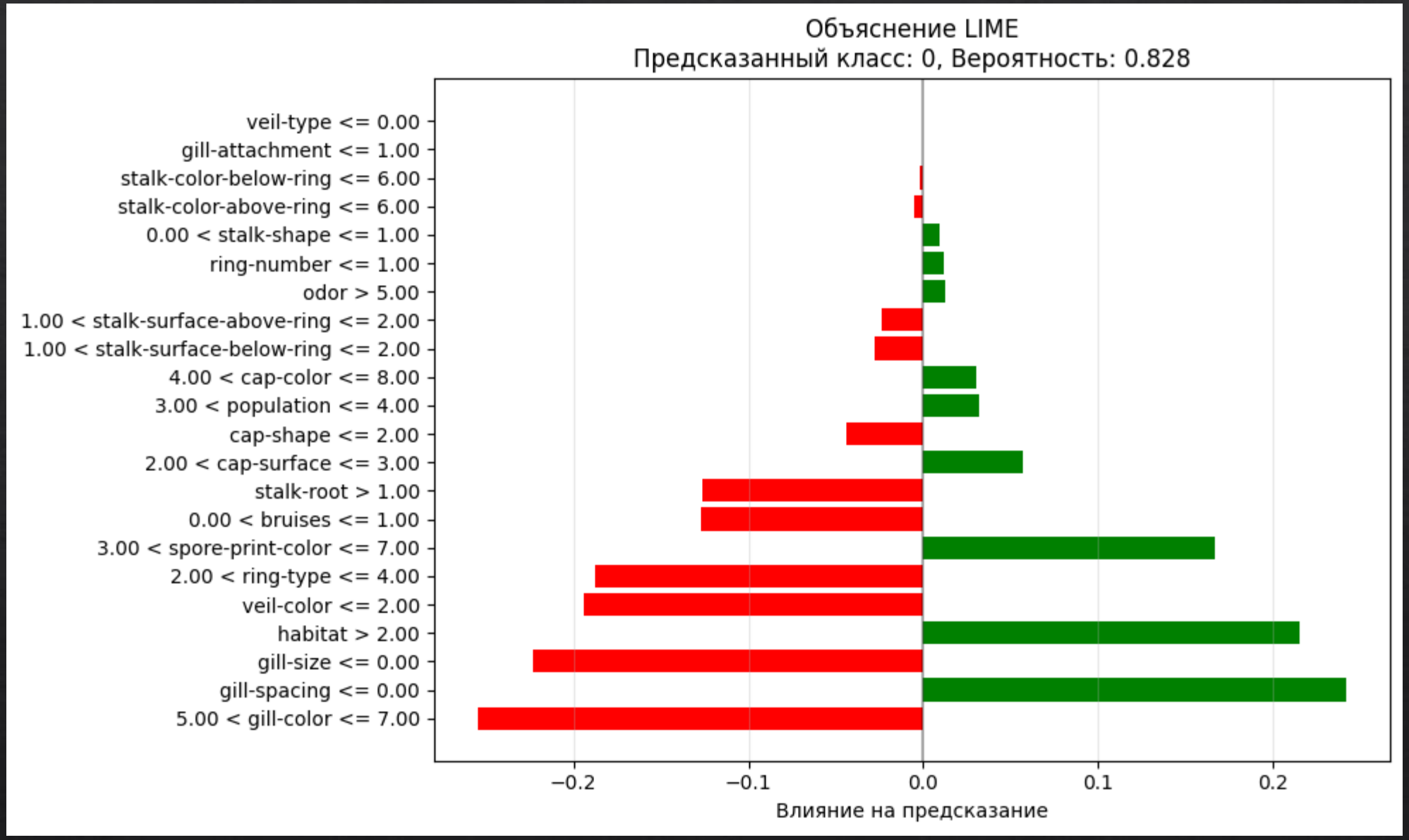
```python
from sklearn.metrics import accuracy_score

y_pred_mnb = mnb.predict(X_test)
accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
print(f"MultinomialNB: {accuracy_mnb:.4f}")

y_pred_comnb = comnb.predict(X_test)
accuracy_comnb = accuracy_score(y_test, y_pred_comnb)
print(f"ComplementNB: {accuracy_comnb:.4f}")

y_pred_catnb = catnb.predict(X_test)
accuracy_catnb = accuracy_score(y_test, y_pred_mnb)
print(f"CategoricalNB: {accuracy_catnb:.4f}")

# 4.2. Предсказание для Байесовской сети
# Для предсказания в БС нужен inference engine.
inference = VariableElimination(model_h_m)
y_pred_bn = []

for i, test_case in X_test.iterrows():
    evidence_dict = test_case.to_dict()
    query = inference.map_query(variables=['class'], evidence=evidence_dict, show_progress=False)
    y_pred_bn.append(query['class'])

# y_pred_bn = pd.Series(y_pred_bn)
accuracy_bn = accuracy_score(y_test, y_pred_bn)
print(f"BayesianNetwork: {accuracy_bn:.4f}")
```

```
MultinomialNB: 0.8154
ComplementNB: 0.8160
CategoricalNB: 0.8154
BayesianNetwork: 0.9975
```

Калькулятор Limme

```python
import lime
import lime.lime_tabular
import pandas as pd
import numpy as np


explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    feature_names=X_train.columns,
    mode='classification',
    discretize_continuous=True,
    sample_around_instance=True,
    random_state=RND_SEED
)

x_test = pd.DataFrame(np.zeros((1, len(X.columns))), columns=X_train.columns)

flag = 1
for col, mapping in mappings.items():
    if flag:
        flag = 0
        continue
    print(f"Столбец '{col}':")
    for value, encoded in mapping.items():
        print(f"  {value} -> {encoded}")
    print("Выберите значение данного признака (число): ", end="")
    number = input()
    if number == '':
        number = 0
    else:
        number = int(number)
    print(number)
    x_test.loc[0, col] = number

print(x_test.head(5))
exp = explainer.explain_instance(x_test.values[0], mnb.predict_proba, num_features=22)
```

```
Столбец 'cap-shape':
  b -> 0
  c -> 1
  f -> 2
  k -> 3
  s -> 4
  x -> 5
Выберите значение данного признака (число): 0
Столбец 'cap-surface':
  f -> 0
  g -> 1
  s -> 2
  y -> 3
Выберите значение данного признака (число): 3
Столбец 'cap-color':
  b -> 0
  c -> 1
  e -> 2
  g -> 3
  n -> 4
  p -> 5
  r -> 6
  u -> 7
  w -> 8
  y -> 9
Выберите значение данного признака (число): 7
Столбец 'bruises':
  f -> 0
  t -> 1
```


Объяснение LIME
Предсказанный класс: 0, Вероятность: 0.828
veil-type <= 0.00
gill-attachment <= 1.00
stalk-color-below-ring <= 6.00
stalk-color-above-ring <= 6.00
0.00 < stalk-shape <= 1.00
ring-number <= 1.00
odor > 5.00
1.00 < stalk-surface-above-ring <= 2.00
1.00 < stalk-surface-below-ring <= 2.00
4.00 < cap-color <= 8.00
3.00 < population <= 4.00
cap-shape <= 2.00
2.00 < cap-surface <= 3.00
stalk-root > 1.00
0.00 < bruises <= 1.00
3.00 < spore-print-color <= 7.00
2.00 < ring-type <= 4.00
veil-color <= 2.00
habitat > 2.00
gill-size <= 0.00
gill-spacing <= 0.00
5.00 < gill-color <= 7.00
−0.2
−0.1
0.0
0.1
0.2
Влияние на предсказание