Лабораторная работа №3

Деревья решений

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import tree
from sklearn import model_selection
from sklearn import metrics

voice_data = pd.read_csv('data/voice.csv')

features = voice_data.drop('label', axis=1).columns
X, y = voice_data[features], voice_data['label']

voice_data.head()
#voice_data.isnull().sum().sum()
```

```
   meanfreq        sd    median       Q25       Q75       IQR
skew
0  0.059781  0.064241  0.032027  0.015071  0.090193  0.075122
12.863462  \
1  0.066009  0.067310  0.040229  0.019414  0.092666  0.073252
22.423285
2  0.077316  0.083829  0.036718  0.008701  0.131908  0.123207
30.757155
3  0.151228  0.072111  0.158011  0.096582  0.207955  0.111374
1.232831
4  0.135120  0.079146  0.124656  0.078720  0.206045  0.127325
1.101174

          kurt    sp.ent       sfm  ...  centroid   meanfun    minfun

0   274.402906  0.893369  0.491918  ...  0.059781  0.084279  0.015702
\
1   634.613855  0.892193  0.513724  ...  0.066009  0.107937  0.015826

2  1024.927705  0.846389  0.478905  ...  0.077316  0.098706  0.015656

3     4.177296  0.963322  0.727232  ...  0.151228  0.088965  0.017798

4     4.333713  0.971955  0.783568  ...  0.135120  0.106398  0.016931


     maxfun   meandom    mindom    maxdom   dfrange   modindx  label
0  0.275862  0.007812  0.007812  0.007812  0.000000  0.000000   male
```

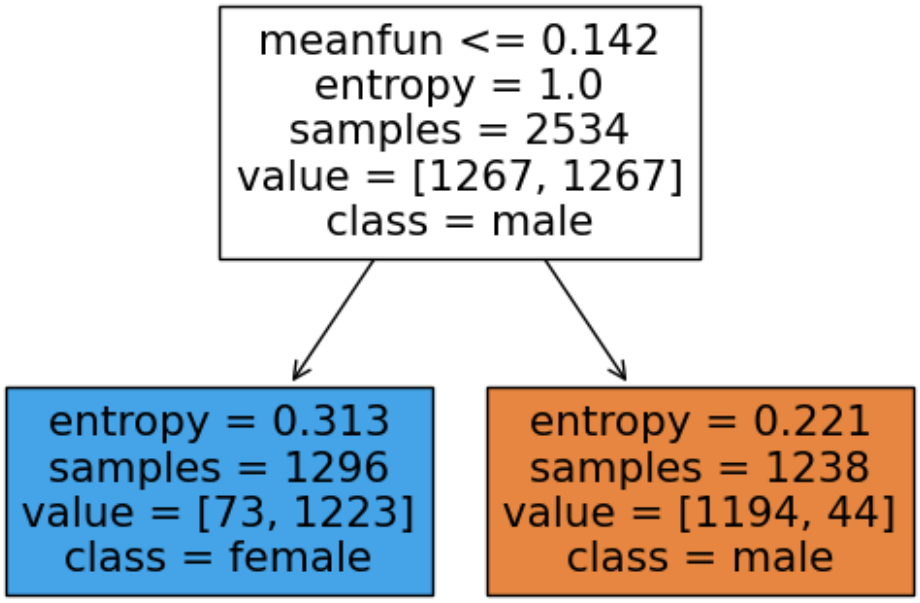
```
1  0.250000  0.009014  0.007812  0.054688  0.046875  0.052632   male
2  0.271186  0.007990  0.007812  0.015625  0.007812  0.046512   male
3  0.250000  0.201497  0.007812  0.562500  0.554688  0.247119   male
4  0.266667  0.712812  0.007812  5.484375  5.476562  0.208274   male

[5 rows x 21 columns]
```

```
# Формируем обучающую и тестовую выборки
X_train, X_test, y_train, y_test = model_selection.train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
    )
print('Train shape: {}'.format(X_train.shape))
print('Test shape: {}'.format(X_test.shape))
```

```
Train shape: (2534, 20)
Test shape: (634, 20)
```

Задание 1

```
# Инициализируем модель дерева решений с максимальной глубиной 1 и
обучаем ее
dt = tree.DecisionTreeClassifier(
    max_depth=1, #глубина
    criterion='entropy', #критерий информативности
    random_state=42 #генератор случайных чисел
)
dt.fit(X_train, y_train)
# Визуализируем дерево решений в виде графа
tree.plot_tree(
    decision_tree=dt, #дерево решений
    feature_names=X.columns, #имена факторов
    class_names=['male', 'female'],
    filled=True, #расцветка
    impurity=True, #отображать ли неоднородность в вершинах
);
```



1. На основе какого фактора будет построено решающее правило в корневой вершине? • meanfun (средняя основная частота в акустическом спектре)
2. Чему равно оптимальное пороговое значение для данного фактора? Ответ округлите до трёх знаков после точки-разделителя. 0.142
3. Сколько процентов наблюдений, для которых выполняется заданное в корневой вершине условие, содержится в обучающей выборке? Ответ округлите до одного знака после точкиразделителя. Не указывайте в ответе символ %. 0.5
4. Сделайте предсказание и рассчитайте значение метрики accuracy на тестовой выборке. Ответ округлите до трёх знаков после точки разделителя. 0.956

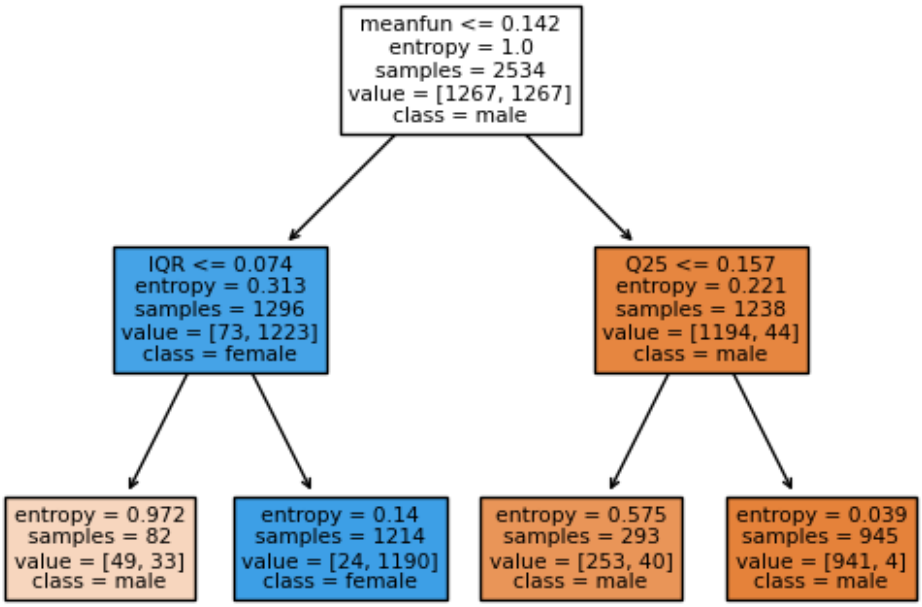
```
y_pred = dt.predict(X_test)
print("accuracy: {:.3f}".format(metrics.accuracy_score(y_test,
y_pred)))
```

```
accuracy: 0.956
```

Задание 2

```
# Инициализируем модель дерева решений с максимальной глубиной 1 и
обучаем ее
dt = tree.DecisionTreeClassifier(
    max_depth=2, #глубина
    criterion='entropy', #критерий информативности
    random_state=42 #генератор случайных чисел
)
dt.fit(X_train, y_train)
# Визуализируем дерево решений в виде графа
```

```
tree.plot_tree(
    decision_tree=dt, #дерево решений
    feature_names=X.columns, #имена факторов
    class_names=['male', 'female'],
    filled=True, #расцветка
    impurity=True, #отображать ли неоднородность в вершинах
);
```



1. Из приведённых ниже факторов выберите те, что используются при построении данного дерева решений:

- C IQR (межквартильный размах частот)
- D meanfun (средняя основная частота в акустическом спектре)
- F Q25 (первый квартиль частоты)

1. Сколько листьев в построенном дереве содержат в качестве предсказания класс female? Для того, чтобы отобразить имена классов при визуализации дерева решения с помощью функции plot_tree(), укажите параметр class_names=dt.classes_.

- 2

1. Сделайте предсказание и рассчитайте значение метрики accuracy на тестовой выборке. Ответ округлите до трёх знаков после точки-разделителя.

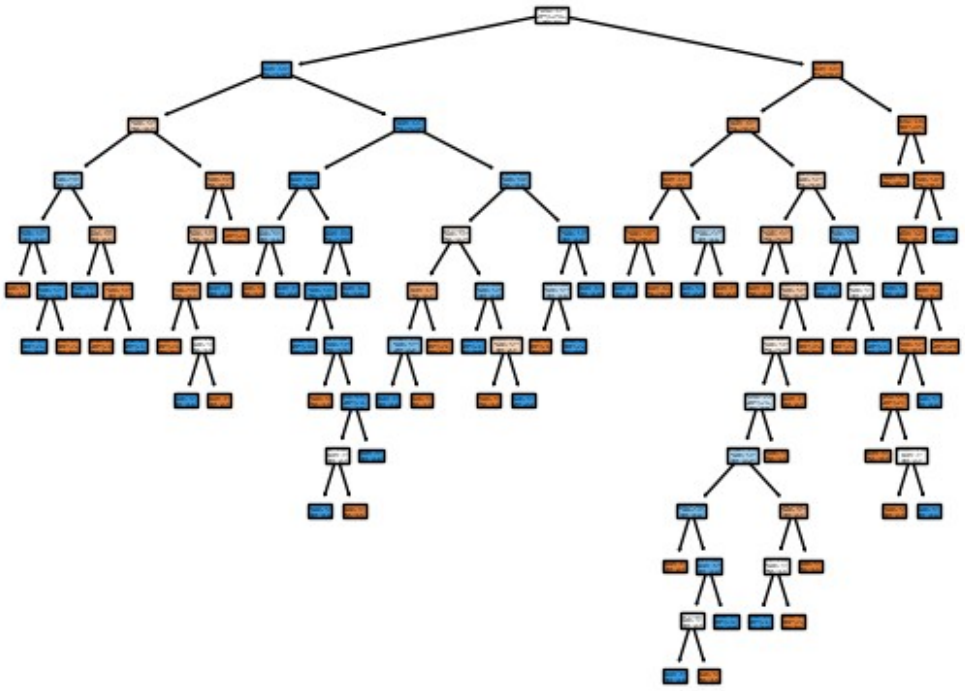
- 0.962

```
y_pred = dt.predict(X_test)
print("accuracy: {:.3f}".format(metrics.accuracy_score(y_test,
y_pred)))
```

```
accuracy: 0.962
```

Задание 3

```
# Инициализируем модель дерева решений с максимальной глубиной 1 и
обучаем ее
dt = tree.DecisionTreeClassifier(
    criterion='entropy', #критерий информативности
    random_state=0 #генератор случайных чисел
)
dt.fit(X_train, y_train)
# Визуализируем дерево решений в виде графа
tree.plot_tree(
    decision_tree=dt, #дерево решений
    feature_names=X.columns, #имена факторов
    class_names=['male', 'female'],
    filled=True, #расцветка
    impurity=True, #отображать ли неоднородность в вершинах
);
```



```
print('Глубина: ', dt.get_depth())
print('Количество листьев: ', dt.get_n_leaves())
```

```
Глубина:  12
Количество листьев:  54
```

```
y_pred1 = dt.predict(X_train)
print("accuracy: {:.3f}".format(metrics.accuracy_score(y_train,
y_pred1)))


y_pred2 = dt.predict(X_test)
print("accuracy: {:.3f}".format(metrics.accuracy_score(y_test,
y_pred2)))
```

```
accuracy: 1.000
accuracy: 0.973
```

Задание 4

```
# Задаём сетку параметров
param_grid = {
    'criterion': ['gini', 'entropy'], #критерий информативности
    'max_depth': [4, 5, 6, 7, 8, 9, 10], #максимальная глубина дерева
    'min_samples_split': [3, 4, 5, 10] #минимальное количество
объектов, необходимое для сплита
}

# Задаём метод кросс-валидации
cv = model_selection.StratifiedKFold(n_splits=5)

dt = tree.DecisionTreeClassifier(random_state=0)

# Инициализируем GridSearchCV
grid_search = model_selection.GridSearchCV(estimator=dt,
param_grid=param_grid, cv=cv, scoring='accuracy')

# Обучаем GridSearchCV на обучающей выборке
grid_search.fit(X_train, y_train)

# Выводим лучшие параметры
print("Best parameters:", grid_search.best_params_)

# Выводим лучшее значение метрики качества
print("Best accuracy:", round(grid_search.best_score_, 3))
```

```
Best parameters: {'criterion': 'gini', 'max_depth': 7,
'min_samples_split': 3}
Best accuracy: 0.966
```

1. Какой критерий информативности использует наилучшая модель?

- Критерий Джини

1. Чему равна оптимальная найденная автоматически (с помощью GridSearchCV) максимальная глубина?

- 7

1. Чему равно оптимальное минимальное количество объектов, необходимое для разбиения?

- 3

1. С помощью наилучшей модели сделайте предсказание отдельно для обучающей и тестовой выборок. Рассчитайте значение метрики accuracy на каждой из выборок. Ответы округлите до трёх знаков после точкиразделителя.

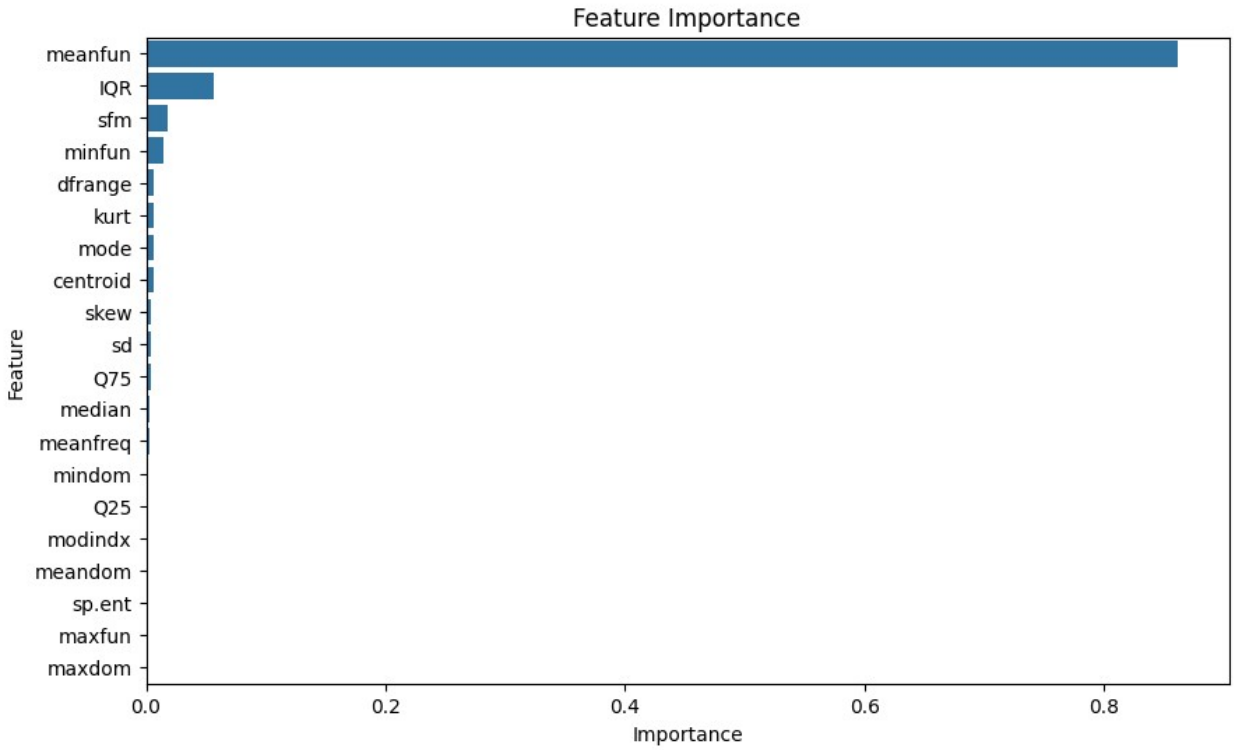
- 0.996 и 0.970

```
best_model = grid_search.best_estimator_

y_pred_train_best = best_model.predict(X_train)
accuracy_train_best = metrics.accuracy_score(y_train,
y_pred_train_best)
print("Accuracy on training set with best model:
{:.3f}".format(accuracy_train_best))

y_pred_test_best = best_model.predict(X_test)
accuracy_test_best = metrics.accuracy_score(y_test, y_pred_test_best)
print("Accuracy on test set with best model:
{:.3f}".format(accuracy_test_best))
```

```
Accuracy on training set with best model: 0.996
Accuracy on test set with best model: 0.970
```

Задание 5

```
# Получаем наилучшую модель из GridSearchCV
best_model = grid_search.best_estimator_

# Получаем важности каждого признака
feature_importances = best_model.feature_importances_

# Создаем DataFrame для удобства визуализации
importance_df = pd.DataFrame({'Feature': X_train.columns,
'Importance': feature_importances})

# Сортируем по убыванию важности
importance_df = importance_df.sort_values(by='Importance',
ascending=False)

# Визуализируем важность признаков
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



1. D meanfun (средняя основная частота в акустическом спектре)
2. C IQR (межквартильный размах частот)
3. F sfm (спектральная равномерность)