Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 3 «Классификация методом дерева решений»

Выполнил:

студент группы ИП-216

Русецкий А.С.
ФИО студента

Работу проверил: Преподаватель
должность преподавателя
Сороковых Д.А.
ФИО преподавателя

Новосибирск 2025 г.

Введение

Цель работы: освоить практическое применение метода решающего дерева для задач классификации. Исследовать влияние гиперпараметров на качество модели и научиться проводить базовый анализ важности признаков.

Задание

1. Подготовка данных
2. Базовое дерево
3. Подбор гиперпараметров
4. Анализ результатов
5. Визуализация

Основная часть

1. Подготовка данных и предобработка

```
# 1. ПОДГОТОВКА ДАННЫХ
# 1.1. Загрузка данных и предобработка
df = pd.read_csv('Heart_Disease_Prediction.csv')
print("Размер данных:", df.shape)
print("\nПервые 5 строк:")
display(df.head())

target_col = 'target' if 'target' in df.columns else df.columns[-1]

num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
cat_cols = [c for c in df.columns if c not in num_cols and c != target_col]

for c in num_cols:
    df[c] = df[c].fillna(df[c].median())
for c in cat_cols:
    df[c] = df[c].fillna(df[c].mode()[0])

df = pd.get_dummies(df, columns=cat_cols, drop_first=False)

X = df.drop(columns=[target_col])
y = df[target_col]

# Разделение на обучающую и тестовую
RANDOM_STATE = 42
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=RANDOM_STATE, stratify=y
)
print(f"\nРазмеры выборок: X_train {X_train.shape}, X_test {X_test.shape}")
```

```
Размер данных: (270, 14)

Первые 5 строк:
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence

```
Размеры выборок: X_train (216, 13), X_test (54, 13)
```

2. Базовое дерево

```
# 2. БАЗОВОЕ ДЕРЕВО
# 2.1. Обучение дерева с параметрами по умолчанию
base_tree = DecisionTreeClassifier(random_state=RANDOM_STATE)
base_tree.fit(X_train, y_train)

# 2.2. accuracy
y_pred_base = base_tree.predict(X_test)
acc_base = accuracy_score(y_test, y_pred_base)
print(f"Accuracy базового дерева: {acc_base:.4f}")

# 2.3. 3 наиболее важных признака
feat_importances = pd.Series(base_tree.feature_importances_, index=X.columns)
top3_features = feat_importances.sort_values(ascending=False).head(3)
print("\nТри наиболее важных признака:")
display(top3_features)
```

```
Accuracy базового дерева: 0.7963

Три наиболее важных признака:
```

	0
Chest pain type	0.268071
Number of vessels fluro	0.152201
ST depression	0.115622

dtype: float64

3. Подбор гиперпараметров

```
# 3. ПОДБОР ГИПЕРПАРАМЕТРОВ С min_samples_split
from itertools import product

# 3.1. Поиск оптимальных параметров методом перебора max_depth и max_leaf_nodes
# Обучение
# Вычисление accuranc
results = []
max_depth_range = range(2, 11)
max_leaf_nodes_range = range(2, 25)
min_samples_split_range = [2, 5, 10, 20]

print("Подбор гиперпараметров")
for md, mln, mss in product(max_depth_range, max_leaf_nodes_range, min_samples_split_range):
    tree = DecisionTreeClassifier(
        max_depth=md,
        max_leaf_nodes=mln,
        min_samples_split=mss,
        random_state=RANDOM_STATE
    )
    tree.fit(X_train, y_train)
    y_pred = tree.predict(X_test)
    acc = accuracy_score(y_test, y_pred)

    mln_display = mln if mln is not None else 'None'
    results.append({
        'max_depth': md,
        'max_leaf_nodes': mln_display,
        'min_samples_split': mss,
        'accuracy': acc
    })

res_df = pd.DataFrame(results).sort_values('accuracy', ascending=False)
print("\n10 лучших комбинаций параметров:")
display(res_df.head(10))
```

```
Подбор гиперпараметров

10 лучших комбинаций параметров:
```

	max_depth	max_leaf_nodes	min_samples_split	accuracy
476	7	6	2	0.851852
477	7	6	5	0.851852
478	7	6	10	0.851852
479	7	6	20	0.851852
752	10	6	2	0.851852
753	10	6	5	0.851852
754	10	6	10	0.851852
755	10	6	20	0.851852
569	8	6	5	0.851852
570	8	6	10	0.851852

Оптимальные параметры:

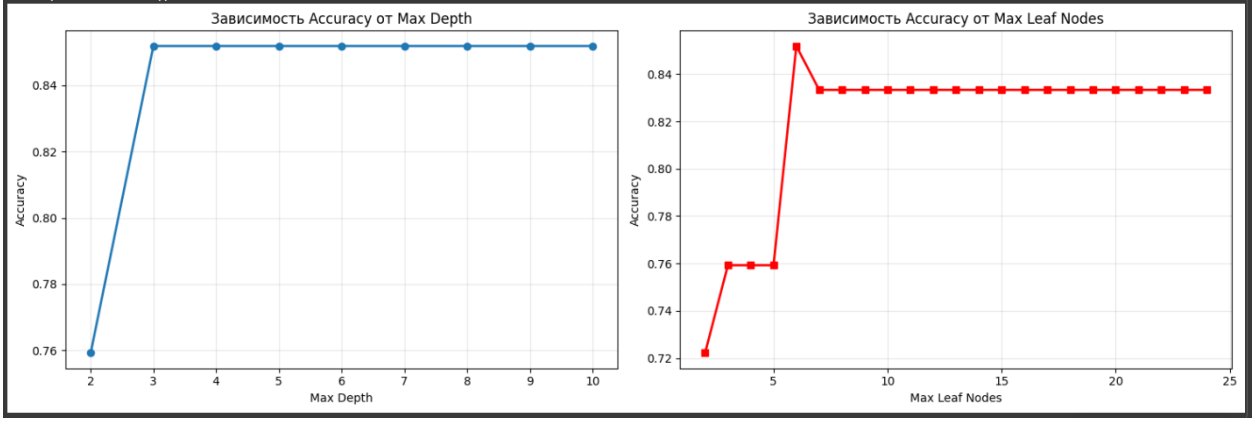
max_depth = 7

Max_leaf_nodes = 6

4. Анализ результатов

```python
# 4. АНАЛИЗ РЕЗУЛЬТАТОВ
# 4.1. Определяем комбинацию с наивысшей точностью
best_row = res_df.iloc[0]
best_md = best_row['max_depth']
best_mln = best_row['max_leaf_nodes']
best_acc = best_row['accuracy']

# 4.3. Сравнение
print(f"Лучшая комбинация: max_depth={best_md}, max_leaf_nodes={best_mln}")
print(f"Accuracy лучшей модели: {best_acc:.4f}")
print(f"Разница с базовой моделью: {best_acc - acc_base:.4f}")

# 4.2. Визуализация
plt.figure(figsize=(15, 5))

# Зависимость от max_depth
plt.subplot(1, 2, 1)
depth_results = res_df.groupby('max_depth')['accuracy'].max()
plt.plot(depth_results.index, depth_results.values, marker='o', linewidth=2)
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
plt.title('Зависимость Accuracy от Max Depth')
plt.grid(True, alpha=0.3)

# Зависимость от max_leaf_nodess
plt.subplot(1, 2, 2)
leaf_results = res_df[res_df['max_leaf_nodes'].notna()].groupby('max_leaf_nodes')['accuracy'].max()
plt.plot(leaf_results.index, leaf_results.values, marker='s', color='red', linewidth=2)
plt.xlabel('Max Leaf Nodes')
plt.ylabel('Accuracy')
plt.title('Зависимость Accuracy от Max Leaf Nodes')
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

```
Лучшая комбинация: max_depth=7.0, max_leaf_nodes=6.0
Accuracy лучшей модели: 0.8519
Разница с базовой моделью: 0.0556
```



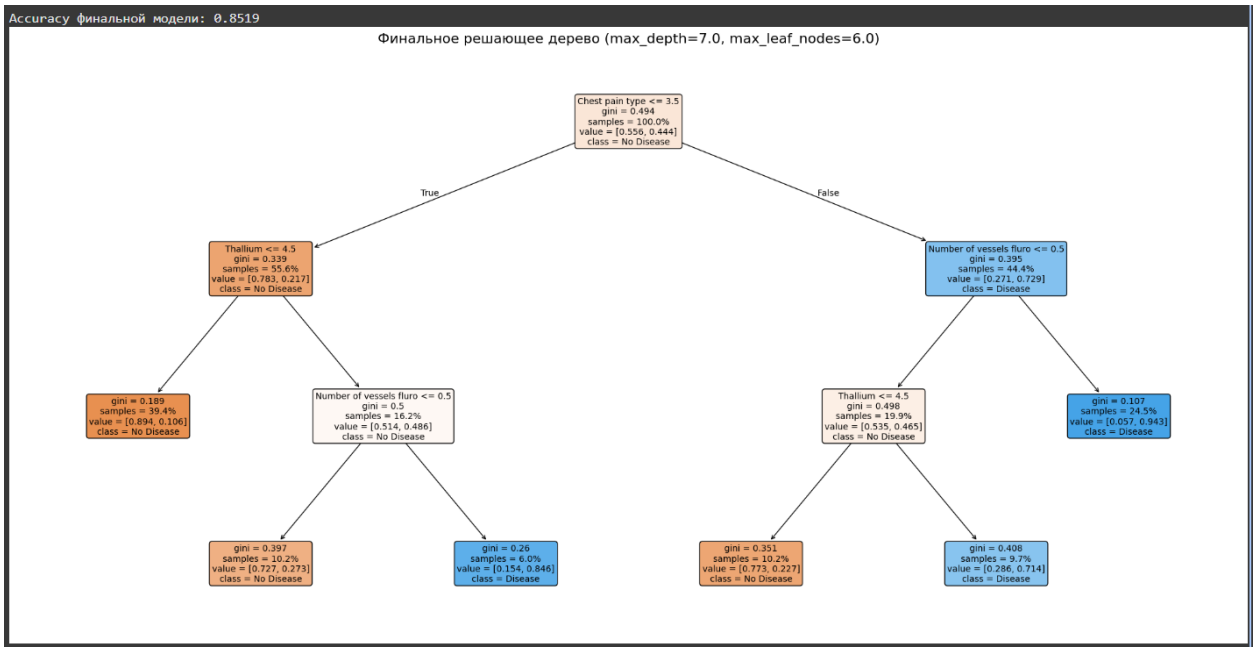
При глубине = 2 accuracy наименьшее

При глубине >= 3 достигается наилучшая accuracy

При количестве листовых узлов = 6 и глубине дерева = 7 достигается налучшая точность

5. Визуализация

```
# 5. ВИЗУАЛИЗАЦИЯ
# 5.1. Обучение финальной модели
final_tree = DecisionTreeClassifier(
    max_depth=int(best_md),
    max_leaf_nodes=int(best_mln) if pd.notna(best_mln) else None,
    random_state=RANDOM_STATE
)

final_tree.fit(X_train, y_train)

y_pred_final = final_tree.predict(X_test)
final_accuracy = accuracy_score(y_test, y_pred_final)
print(f"Accuracy финальной модели: {final_accuracy:.4f}")

# 5.2. Дерево
plt.figure(figsize=(20, 10))
plot_tree(
    final_tree,
    feature_names=X.columns,
    class_names=['No Disease', 'Disease'],
    filled=True,
    rounded=True,
    fontsize=10,
    proportion=True
)
plt.title(f"Финальное решающее дерево (max_depth={best_md}, max_leaf_nodes={best_mln})", fontsize=16)
plt.tight_layout()
```



Заключение

Были поэтапно выполнены все поставленные задачи: от первичной обработки данных до обучения финальной модели. В результате удалось достичь accuracy = 85.19%, что на 5.56% превышает показатель базовой модели с параметрами по умолчанию.

Оптимальными гиперпараметрами модели стали: глубина дерева = 7 и максимальное количество листовых узлов = 6.

Ссылка на google colab:
https://colab.research.google.com/drive/1d5ebxBBS9prhrSJ0URnTwkmiIqYrZ_mq?usp=sharing