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-213
Дмитриев Антон Александрович

Работу проверил: Преподаватель
Сороковых Дарья Анатольевна

Новосибирск 2025 г.

Введение (задание)

Задание
1. Подготовка данных
1.1. Выберите любой датасет для бинарной или многоклассовой классификации с сайта Kaggle.com.
Рекомендуемые датасеты: Titanic, Iris, Wine Quality, Heart Disease Prediction и др.
1.2. Загрузите данные и выполните предобработку:
Обработайте пропущенные значения.
Закодируйте категориальные признаки (One-Hot Encoding или Label Encoding).
Разделите данные на обучающую и тестовую выборки (70/30 или 80/20).
2. Базовое дерево
2.1. Обучите решающее дерево с параметрами по умолчанию.
2.2. Вычислите accuracy на тестовой выборке.
2.3. Определите 3 наиболее важных признака для классификации.
3. Подбор гиперпараметров
3.1. Проведите поиск оптимальных параметров методом перебора max_depth и max_leaf_nodes в интервалах, подходящих вашему набору данных.
3.2. Для каждой комбинации параметров:
Обучите модель на тренировочных данных;
Вычислите accuracy на тестовых данных;
Зафиксируйте результаты**4. Анализ результатов**
4.1. Определите комбинацию параметров, дающую наивысшую точность.
4.2. Визуализируйте зависимость accuracy от глубины дерева и количества листьев
4.3. Сравните accuracy лучшей модели с базовой (параметры по умолчанию)
5. Визуализация
5.1. Обучите финальную модель с оптимальными параметрами.
5.2. Визуализируйте дерево с помощью plot_tree или export_graphviz.

Основная часть

1. Подготовка данных

Размерность данных: (3547, 11)

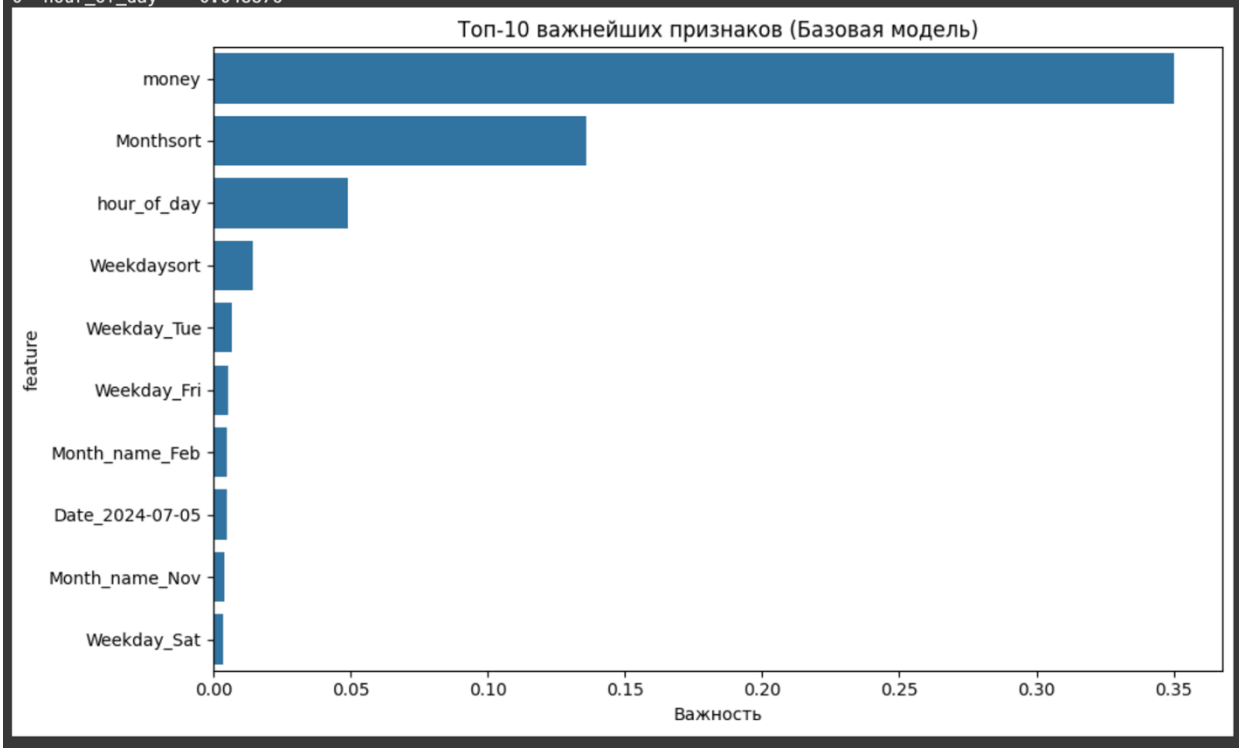
Первые 5 строк:

	hour_of_day	cash_type	money	coffee_name	Time_of_Day	Weekday	Month_name	Weekdaysort	Monthsort	Date	Time
0	10	card	38.7	Latte	Morning	Fri	Mar	5	3	2024-03-01	10:15:50.520000
1	12	card	38.7	Hot Chocolate	Afternoon	Fri	Mar	5	3	2024-03-01	12:19:22.539000
2	12	card	38.7	Hot Chocolate	Afternoon	Fri	Mar	5	3	2024-03-01	12:20:18.089000
3	13	card	28.9	Americano	Afternoon	Fri	Mar	5	3	2024-03-01	13:46:33.006000
4	13	card	38.7	Latte	Afternoon	Fri	Mar	5	3	2024-03-01	13:48:14.626000

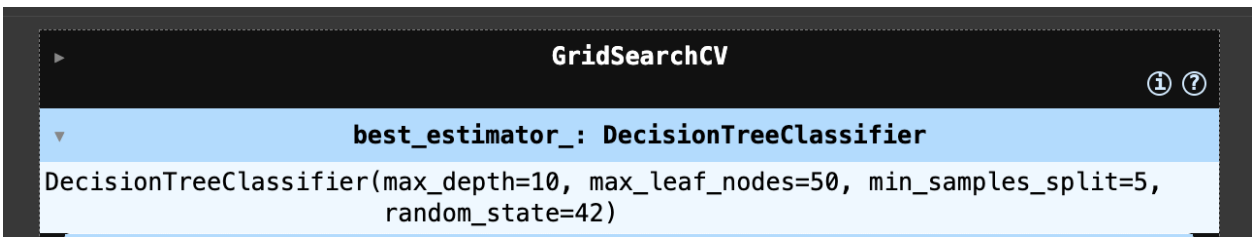
```
Пропущенные значения:
hour_of_day    0
cash_type      0
money          0
coffee_name    0
Time_of_Day    0
Weekday        0
Month_name     0
Weekdaysort    0
Monthsort      0
Date           0
Time           0
dtype: int64

Данные после предобработки:
Размерность: (3547, 3956)
```

2. Базовое дерево

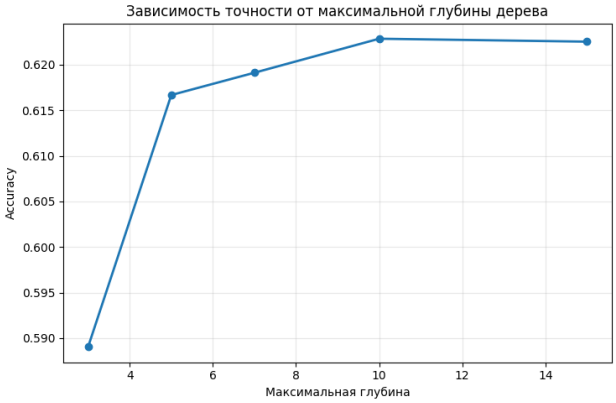
```
Accuracy базовой модели: 0.6282

Топ-3 наиболее важных признака:
       feature  importance
1        money    0.350247
3    Monthsort    0.135931
0  hour_of_day    0.048876
```

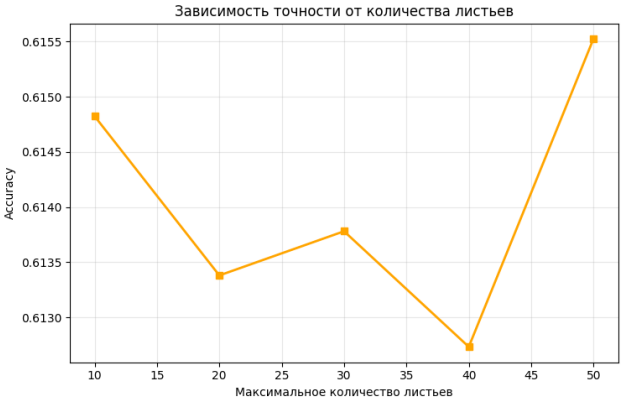


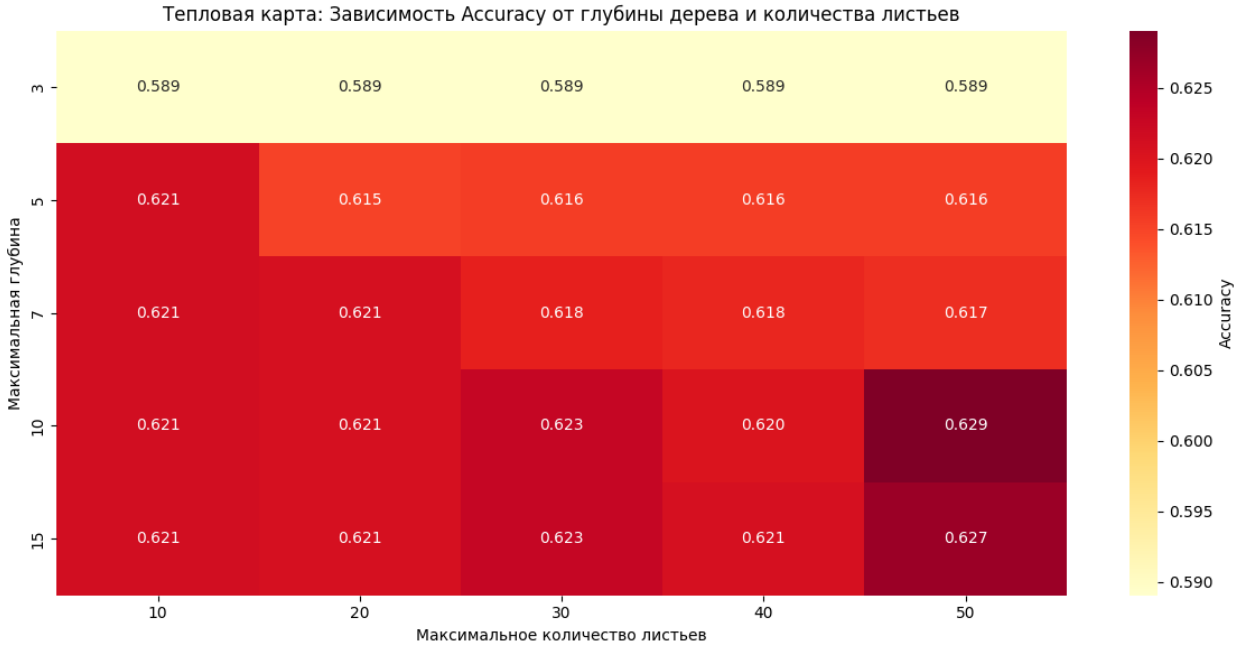
3. Подбор гиперпараметров



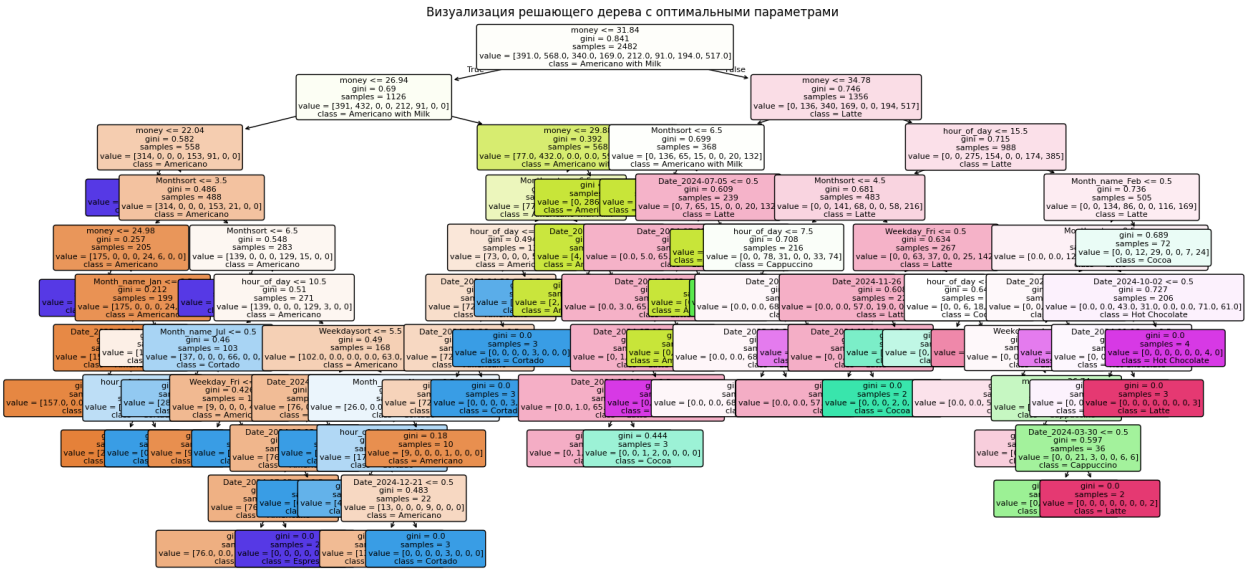
4. Анализ результатов







5. Визуализация



Заключение

Ссылка на colab:

https://colab.research.google.com/drive/1mUxTBUnTwlapyDvW3wwD4L9bqfieeyTf?usp=sharing

Код программы

```
# Импорт необходимых библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer

# Для подбора гиперпараметров
from sklearn.model_selection import GridSearchCV

# 1.1. Загрузка данных
df = pd.read_csv('/content/drive/MyDrive/Coffe_sales.csv')
print("Размерность данных:", df.shape)
print("\nПервые 5 строк:")
display(df.head())

# 1.2. Предобработка данных
# Анализ пропущенных значений
print("\nПропущенные значения:")
print(df.isnull().sum())

# Обработка пропущенных значений
numerical_cols = df.select_dtypes(include=[np.number]).columns
imputer = SimpleImputer(strategy='median')
df[numerical_cols] = imputer.fit_transform(df[numerical_cols])

# Кодирование категориальных признаков
categorical_cols = df.select_dtypes(include=['object']).columns

# Label Encoding для целевой переменной
target_column = 'coffee_name'
le = LabelEncoder()
df[target_column] = le.fit_transform(df[target_column])

# One-Hot Encoding для остальных категориальных признаков
df = pd.get_dummies(df, columns=categorical_cols.drop(target_column,
errors='ignore'))

print("\nДанные после предобработки:")
print("Размерность:", df.shape)
display(df.head())

# Разделение на признаки (X) и целевую переменную (y)
X = df.drop(target_column, axis=1)
y = df[target_column]
```

```
# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42) # 70/30
print(f"\nРазмеры выборок: X_train: {X_train.shape}, X_test:
{X_test.shape}")
# 2.1. Обучение дерева с параметрами по умолчанию
base_model = DecisionTreeClassifier(random_state=42)
base_model.fit(X_train, y_train)

# 2.2. Вычисление accuracy на тестовой выборке
y_pred_base = base_model.predict(X_test)
base_accuracy = accuracy_score(y_test, y_pred_base)
print(f"Accuracy базовой модели: {base_accuracy:.4f}")

# 2.3. Определение 3 наиболее важных признаков
feature_importances = base_model.feature_importances_
feature_names = X.columns

# Создание DataFrame для удобства
importance_df = pd.DataFrame({
    'feature': feature_names,
    'importance': feature_importances
}).sort_values('importance', ascending=False)

print("\nТоп-3 наиболее важных признака:")
print(importance_df.head(3))

# Визуализация важности признаков
plt.figure(figsize=(10, 6))
sns.barplot(data=importance_df.head(10), x='importance', y='feature')
plt.title('Топ-10 важнейших признаков (Базовая модель)')
plt.xlabel('Важность')
plt.tight_layout()
plt.show()
# 3.1. Определение сетки параметров для перебора
param_grid = {
    'max_depth': [3, 5, 7, 10, 15, None],
    'max_leaf_nodes': [10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10]
}

# 3.2. Поиск по сетке
grid_search = GridSearchCV(
    DecisionTreeClassifier(random_state=42),
    param_grid,
    cv=5,  # 5-кратная кросс-валидация
    scoring='accuracy',
    n_jobs=-1
)
# 4.1. Лучшая комбинация параметров
```

```
print("Лучшие параметры:", grid_search.best_params_)
print("Лучшая точность на кросс-валидации:
{:.4f}".format(grid_search.best_score_))

# Оценка лучшей модели на тестовой выборке
best_model = grid_search.best_estimator_
y_pred_best = best_model.predict(X_test)
best_accuracy = accuracy_score(y_test, y_pred_best)
print(f"Accuracy лучшей модели на тестовой выборке: {best_accuracy:.4f}")

# 4.2. Визуализация зависимости accuracy от параметров
results_df = pd.DataFrame(grid_search.cv_results_)

# Создаем subplot для двух графиков
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

# График 1: Зависимость точности от максимальной глубины дерева
depth_data =
results_df[results_df['param_max_leaf_nodes'].notna()].groupby('param_max_
depth')['mean_test_score'].mean()
ax1.plot(depth_data.index, depth_data.values, marker='o', linewidth=2,
markersize=6)
ax1.set_title('Зависимость точности от максимальной глубины дерева')
ax1.set_xlabel('Максимальная глубина')
ax1.set_ylabel('Accuracy')
ax1.grid(True, alpha=0.3)

# График 2: Зависимость точности от количества листьев
# Группируем по max_leaf_nodes, усредняем результаты
leaf_data =
results_df[results_df['param_max_depth'].notna()].groupby('param_max_leaf_
nodes')['mean_test_score'].mean()
ax2.plot(leaf_data.index, leaf_data.values, marker='s', color='orange',
linewidth=2, markersize=6)
ax2.set_title('Зависимость точности от количества листьев')
ax2.set_xlabel('Максимальное количество листьев')
ax2.set_ylabel('Accuracy')
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# Дополнительная визуализация: тепловая карта зависимости accuracy от
глубины и количества листьев
# Создаем сводную таблицу для тепловой карты
pivot_table = results_df.pivot_table(
    values='mean_test_score',
    index='param_max_depth',
    columns='param_max_leaf_nodes',
    aggfunc='mean'
)
```

```
plt.figure(figsize=(12, 6))
sns.heatmap(pivot_table, annot=True, fmt='.3f', cmap='YlOrRd',
cbar_kws={'label': 'Accuracy'})
plt.title('Тепловая карта: Зависимость Accuracy от глубины дерева и
количества листьев')
plt.xlabel('Максимальное количество листьев')
plt.ylabel('Максимальная глубина')
plt.tight_layout()
plt.show()

# 4.3. Сравнение с базовой моделью
print(f"\nСравнение моделей:")
print(f"Базовая модель (по умолчанию): {base_accuracy:.4f}")
print(f"Лучшая модель (с подбором параметров): {best_accuracy:.4f}")
print(f"Улучшение: {best_accuracy - base_accuracy:.4f}")

grid_search.fit(X_train, y_train)



# 5.1. & 5.2. Обучение и визуализация финального дерева с оптимальными
параметрами
final_model = DecisionTreeClassifier(**grid_search.best_params_,
random_state=42)
final_model.fit(X_train, y_train)

plt.figure(figsize=(20, 10))
plot_tree(
    final_model,
    feature_names=X.columns,
    class_names=[str(c) for c in le.classes_],  # Если использовали
LabelEncoder для target
    filled=True,
    rounded=True,
    fontsize=8
)
plt.title('Визуализация решающего дерева с оптимальными параметрами')
plt.show()

# 5.3. Сохранение изображения (раскомментируйте, если нужно)
plt.savefig('optimal_decision_tree.png', dpi=300, bbox_inches='tight')
print("Изображение дерева сохранено как 'optimal_decision_tree.png'")
```