МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

# Отчет по лабораторной работе №5

Специальность ИИ(з)

Выполнил
А. Ю. Кураш,
студент группы ИИ-24

Проверил
Андренко К.В,
Преподаватель-стажер кафедры ИИТ,
«___k _____2025 г.

Брест 2025

**Цель:** На практике сравнить работу нескольких алгоритмов одиночного дерева решений, случайного леса и бустинга для деревьев решений.

**Задачи:**

1. Загрузить датасет по варианту;
2. Разделить данные на обучающую и тестовую выборки;
3. Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
4. Оценить точность каждой модели на тестовой выборке;
5. Сравнить результаты, сделать выводы о применимости каждого метода для данного набора данных.

## Вариант 8

- `Seeds`
- Классифицировать семена на три сорта пшеницы (Kama, Rosa, Canadian) на основе их геометрических параметров
- **Задания:**
    1. Загрузите и стандартизируйте данные;
    2. Разделите выборку на обучающую и тестовую;
    3. Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
    4. Сравните общую точность (accuracy) всех трех моделей;
    5. Визуализируйте данные в 2D (например, с помощью PCA), раскрасив точки в соответствии с предсказаниями лучшей модели.

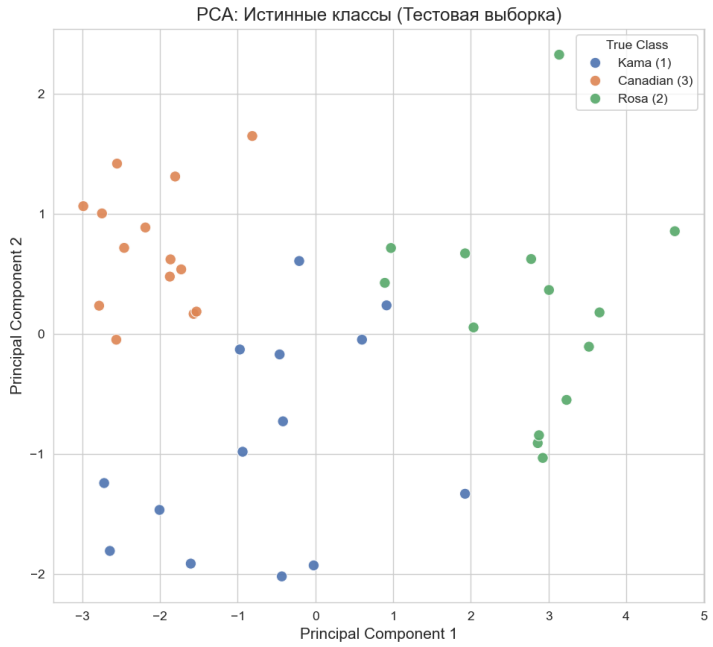
**Код программы:**

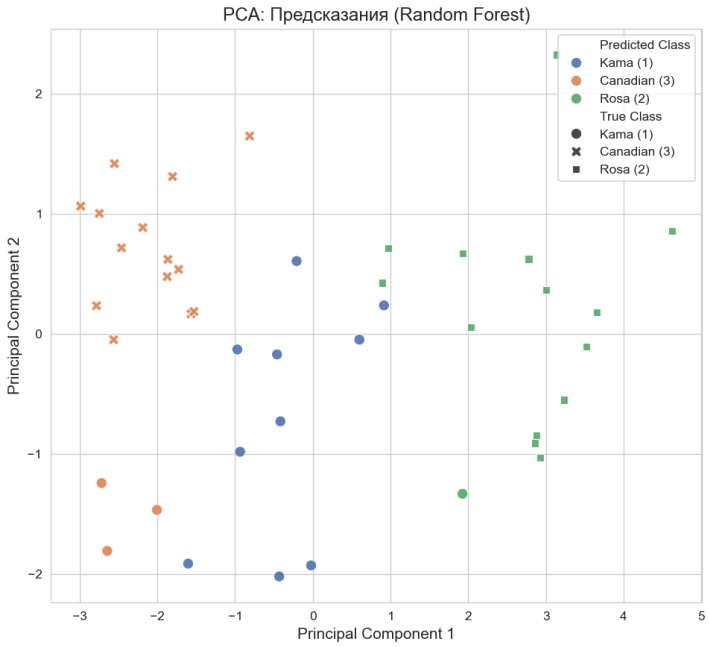
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
from xgboost import XGBClassifier
from catboost import CatBoostClassifier

# --- 1. Загрузка и подготовка данных ---

# URL датасета Seeds (UCI)
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00236/seeds_dataset.txt"

# Названия колонок (7 признаков + 1 класс)
column_names = [
    'Area', 'Perimeter', 'Compactness', 'Length_of_Kernel',
```

```
    'Width_of_Kernel', 'Asymmetry_Coefficient', 'Length_of_Kernel_Groove', 'Class'
]

# Загружаем данные, используя '\s+' в качестве разделителя (табуляция/пробелы)
try:
    data = pd.read_csv(url, sep='\s+', header=None, names=column_names)
    print("Датасет успешно загружен.")
    print(f"Форма данных: {data.shape}")
    print("\nРаспределение классов (1=Kama, 2=Rosa, 3=Canadian):")
    print(data['Class'].value_counts())
except Exception as e:
    print(f"Ошибка при загрузке данных: {e}")
    exit()

# Преобразуем классы из 1, 2, 3 в 0, 1, 2 (стандарт для scikit-learn)
data['Class'] = data['Class'] - 1

# --- 2. Разделение и стандартизация ---

# Выделяем признаки (X) и целевую переменную (y)
X = data.drop('Class', axis=1)
y = data['Class']

# Разделяем на обучающую и тестовую выборки (80% / 20%)
# stratify=y гарантирует, что пропорции классов в train и test будут одинаковыми
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Стандартизируем данные
# (Обучаем scaler на X_train, применяем к X_train и X_test)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(f"\nРазмер обучающей выборки: {X_train_scaled.shape}")
print(f"Размер тестовой выборки: {X_test_scaled.shape}")

# --- 3. Обучение моделей ---

# Задаем random_state для воспроизводимости
rs = 42

# Инициализируем модели
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=rs),
    "Random Forest": RandomForestClassifier(random_state=rs),
    "AdaBoost": AdaBoostClassifier(random_state=rs),
    "XGBoost": XGBClassifier(random_state=rs, use_label_encoder=False,
eval_metric='mlogloss'),
```

```
    "CatBoost": CatBoostClassifier(random_state=rs, verbose=0) # verbose=0 отключает вывод логов обучения
}

# Словарь для хранения результатов
results = {}

print("\n--- 4. Обучение и оценка моделей ---")

# Обучаем и оцениваем каждую модель
for name, model in models.items():
    # Обучение
    model.fit(X_train_scaled, y_train)

    # Предсказание на тестовой выборке
    y_pred = model.predict(X_test_scaled)

    # Оценка точности
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy

    print(f"Модель: {name:15} | Accuracy на тесте: {accuracy:.4f}")

# --- 5. Сравнение и выбор лучшей модели ---

best_model_name = max(results, key=results.get)
best_model = models[best_model_name]
best_accuracy = results[best_model_name]

print(f"\n----------------------------------------------")
print(f"□ Лучшая модель: {best_model_name} (Accuracy: {best_accuracy:.4f})")
print(f"----------------------------------------------")

# --- 6. Визуализация (PCA) ---

print("\nПодготовка визуализации PCA...")

# Снижаем размерность тестовых данных до 2D
pca = PCA(n_components=2)
X_test_pca = pca.fit_transform(X_test_scaled)

# Получаем предсказания лучшей модели на тестовых данных
y_pred_best = best_model.predict(X_test_scaled)

# Создаем DataFrame для удобства отрисовки
pca_df = pd.DataFrame(data=X_test_pca, columns=['PC1', 'PC2'])
# Добавляем истинные метки (сбрасываем индекс, чтобы корректно соединить)
pca_df['True Class'] = y_test.reset_index(drop=True)
# Добавляем предсказанные метки
pca_df['Predicted Class'] = y_pred_best
```

```
# Меняем 0, 1, 2 обратно на 1, 2, 3 для наглядности в легенде
pca_df['True Class'] = pca_df['True Class'].map({0: 'Kama (1)', 1: 'Rosa (2)', 2: 'Canadian (3)'})
pca_df['Predicted Class'] = pca_df['Predicted Class'].map({0: 'Kama (1)', 1: 'Rosa (2)', 2:
'Canadian (3)'})

# Отрисовка
plt.figure(figsize=(16, 7))
sns.set_style("whitegrid")

# График 1: Истинные классы
plt.subplot(1, 2, 1)
sns.scatterplot(
    data=pca_df,
    x='PC1',
    y='PC2',
    hue='True Class',
    palette='deep',
    s=70,
    alpha=0.9
)
plt.title('PCA: Истинные классы (Тестовая выборка)', fontsize=14)
plt.xlabel('Principal Component 1', fontsize=12)
plt.ylabel('Principal Component 2', fontsize=12)

# График 2: Предсказания лучшей модели
plt.subplot(1, 2, 2)
sns.scatterplot(
    data=pca_df,
    x='PC1',
    y='PC2',
    hue='Predicted Class', # Раскрашиваем по предсказаниям
    style='True Class', # Используем разные маркеры для истинных классов
    palette='deep',
    s=70,
    alpha=0.9
)
plt.title(f'PCA: Предсказания ({best_model_name})', fontsize=14)
plt.xlabel('Principal Component 1', fontsize=12)
plt.ylabel('Principal Component 2', fontsize=12)

plt.suptitle('Сравнение Истинных и Предсказанных классов (Метод PCA)', fontsize=18,
y=1.03)
plt.tight_layout()
plt.show()

print("Визуализация отображена.")
```




```
Распределение классов (1=Kama, 2=Rosa, 3=Canadian):
Class
1    70
2    70
3    70
Name: count, dtype: int64

Размер обучающей выборки: (168, 7)
Размер тестовой выборки: (42, 7)

--- 4. Обучение и оценка моделей ---
Модель: Decision Tree    | Accuracy на тесте: 0.8810
Модель: Random Forest    | Accuracy на тесте: 0.9048
Модель: AdaBoost         | Accuracy на тесте: 0.8810
C:\Users\User\AppData\Local\Programs\Python\Python313\Lib\site-
xgboost\xgboost\src\learner.cc:790:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
Модель: XGBoost          | Accuracy на тесте: 0.9048
Модель: CatBoost         | Accuracy на тесте: 0.8810

----------------------------------------------
🏆 Лучшая модель: Random Forest (Accuracy: 0.9048)
----------------------------------------------
```

**Вывод:** Изучил деревья решений и методы бустинга нейронных сетей.