МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**"Южно-Уральский государственный университет
(национальный исследовательский университет)"**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ОТЧЕТ

о выполнении практической работы № 4

по дисциплине

«Технологии аналитической обработки информации»

Выполнил:
студент группы КЭ-403
Гольденберг Д.И.

Проверил:
Преподаватель кафедры СП
Гоглачев А.И.

Челябинск, 2025 г.

ОГЛАВЛЕНИЕ

1. ЗАДАНИЕ

1. Разработайте программу, которая выполняет классификацию заданного набора данных с помощью одной из техник ансамблевой классификации. Параметрами программы являются набор данных, ансамблевая техника (бэггинг, случайный лес или бустинг), количество участников ансамбля, а также параметры в соответствии с выбранной техникой ансамблевой классификации.

2. Проведите эксперименты на наборе данных из задания «Классификация с помощью дерева решений», варьируя количество участников ансамбля (от 50 до 100 с шагом 10).

3. Выполните визуализацию полученных результатов в виде следующих диаграмм:

- показатели качества классификации в зависимости от количества участников ансамбля для заданного набора данных; нанесите на диаграмму соответствующие значения, полученные в задании «Классификация с помощью дерева решений2.

4. Подготовьте отчет о выполнении задания и загрузите отчет в формате PDF в систему. Отчет должен представлять собой связный и структурированный документ со следующими разделами:

- формулировка задания;
- гиперссылка на каталог репозитория с исходными текстами, наборами данных и др. сопутствующими материалами;
- рисунки с результатами визуализации;
- пояснения, раскрывающие смысл полученных результатов.

2. РЕАЛИЗАЦИЯ АЛГОРИТМА

Код реализованной программы и всех проведенных экспериментов находится в репозитории по ссылке https://github.com/Goldria/analitycs/blob/main/4_ensemble_classification/ensemble_classification.ipynb.

Набор данных взят из базы данных Бюро переписи населения. Данные представлены двумя выборками: обучающей (32 561 запись) и тестовой (16 281 запись). Общий объем данных составляет 48 842 записи.

Признаки в наборе данных: возраст (age), класс работы (workclass), финальный вес (fnlwgt), образование (education), количество лет образования (education-num), семейное положение (marital-status), род деятельности (occupation), отношения (relationship), раса (race), пол (sex), доход от капитала (capital-gain), потери капитала (capital-loss), количество рабочих часов в неделю (hours-per-week), страна проживания (native-country), доход (целевой признак: >50K или <=50K)

В наборе данных наблюдается дисбаланс классов:

- Доход >50K: 23.93%
- Доход <=50K: 76.07%

Перед обучением модели была проведена предобработка данных:

- Удаление строк с пропущенными значениями.
- Кодирование категориальных признаков с использованием Label Encoding.
- Балансировка классов методом RandomOverSampler, что увеличило размер выборки до 77 044 записей.

Код предобработки данных представлен в листинге 1.

Листинг 1 – Реализация предобработки данных

```
def balance_dataset(df):
    """Балансировка датасета"""
    X = df.drop(columns=['income'])
    y = df['income']
    y_numeric = (y == ' >50K').astype(int)

    ros = RandomOverSampler(random_state=42)
    X_resampled, y_resampled = ros.fit_resample(X, y_numeric)

    X_balanced = pd.DataFrame(X_resampled, columns=X.columns)
```

```
    y_balanced = pd.Series(y_resampled)
    balanced_dataset = pd.concat([X_balanced, y_balanced], axis=1)
    balanced_dataset['income'] = balanced_dataset['income'].map({0: '
<=50K', 1: ' >50K'})

    return balanced_dataset

def preprocess_data(df):
    """Функция предобработки данных."""
    df = balance_dataset(df)
    df = df.dropna()
    df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)

    # Преобразование категориальных признаков в числовые
    label_encoders = {}
    for col in df.select_dtypes(include=["object"]).columns:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])
        label_encoders[col] = le

    return df, label_encoders

df, label_encoders = preprocess_data(df)
```

Для обучения моделей использовались следующие параметры:

- Ансамблевая техника: случайный лес, бэггинг, бустинг.
- Количество участников ансамбля: от 50 до 100 с шагом 10.
- Максимальная глубина деревьев: 15.
- Доля тестовой выборки: 20%.
- Оценка качества: Accuracy, Precision, Recall, F1-score.

Реализация обучения с различными параметрами представлена в листинге 2.

Листинг 2 – Реализация обучения

```
X = df.drop(columns=["income"])
y = df["income"]

def train_and_evaluate(X, y, ensemble_technique='random_forest', n_estima-
tors=100, max_depth=15, test_size=0.2):
    """Обучение и оценка модели дерева решений."""
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test_size, random_state=42)
    if ensemble_technique == 'random_forest':
        clf = RandomForestClassifier(n_estimators=n_estimators,
max_depth=max_depth, random_state=42)
    elif ensemble_technique == 'bagging':
        base_clf = DecisionTreeClassifier(max_depth=max_depth, ran-
dom_state=42)
        clf = BaggingClassifier(base_clf, n_estimators=n_estimators, ran-
dom_state=42)
    elif ensemble_technique == 'boosting':
        clf = GradientBoostingClassifier(n_estimators=n_estimators,
max_depth=max_depth, random_state=42)
    else:
```

```
            raise ValueError("Неподдерживаемая ансамблиевая техника")

        clf.fit(X_train, y_train)

        y_pred = clf.predict(X_test)

        metrics = {
            "Accuracy": accuracy_score(y_test, y_pred),
            "Precision": precision_score(y_test, y_pred),
            "Recall": recall_score(y_test, y_pred),
            "F1": f1_score(y_test, y_pred)
        }

        return clf, metrics

# Обучение модели
technique = "random_forest"
n_estimators = 150
max_depth = 15
test_size = 0.2
clf, metrics = train_and_evaluate(X, y, ensemble_technique=technique,
                                  n_estimators=n_estimators,
max_depth=max_depth, test_size=test_size)
metrics
```

3. ЭКСПЕРИМЕНТЫ

3.1. Оценка качества модели

Для оценки качества классификации рассчитывались метрики:

- Accuracy (Точность) – общая доля правильно классифицированных объектов.
- Precision (Прецизионность) – доля правильно предсказанных положительных примеров от всех предсказанных положительных.
- Recall (Полнота) – доля правильно найденных положительных примеров от всех истинно положительных.
- F1-score – гармоническое среднее Precision и Recall.

Результаты метрик для критерий random_forest, bagging и boosting представлены в таблице 1, 2 и 3 соответственно.

Таблица 1 – Метрики критерия random_forest

Кол-во участников	Accuracy	Precision	Recall	F1-score
50	0.8594	0.7986	0.9700	0.8760
60	0.8604	0.7992	0.9715	0.8760
70	0.8602	0.7987	0.9721	0.8769
80	0.8603	0.7993	0.9709	0.8768
90	0.8604	0.7997	0.9705	0.8769
100	0.8609	0.8000	0.9711	0.8769

Таблица 2 – Метрики критерия bagging

Кол-во участников	Accuracy	Precision	Recall	F1-score
50	0.8729	0.8082	0.9857	0.8882
60	0.8727	0.8083	0.9851	0.8880
70	0.8724	0.8084	0.9842	0.8877
80	0.8721	0.8080	0.9842	0.8874
90	0.8726	0.8086	0.9844	0.8879
100	0.8734	0.8090	0.9854	0.8885

Таблица 3 – Метрики критерия boosting

Кол-во участников	Accuracy	Precision	Recall	F1-score
50	0.9083	0.8523	0.9931	0.9173
60	0.9164	0.8634	0.9941	0.9241
70	0.9221	0.8708	0.9957	0.9290
80	0.9257	0.8758	0.9962	0.9322
90	0.9284	0.8798	0.9962	0.9344
100	0.9305	0.8830	0.9962	0.9362

3.2. Визуализация результатов

Визуализация показателей качества классификации в зависимости от количества участников ансамбля представлена в иде графика на рисунке 1.

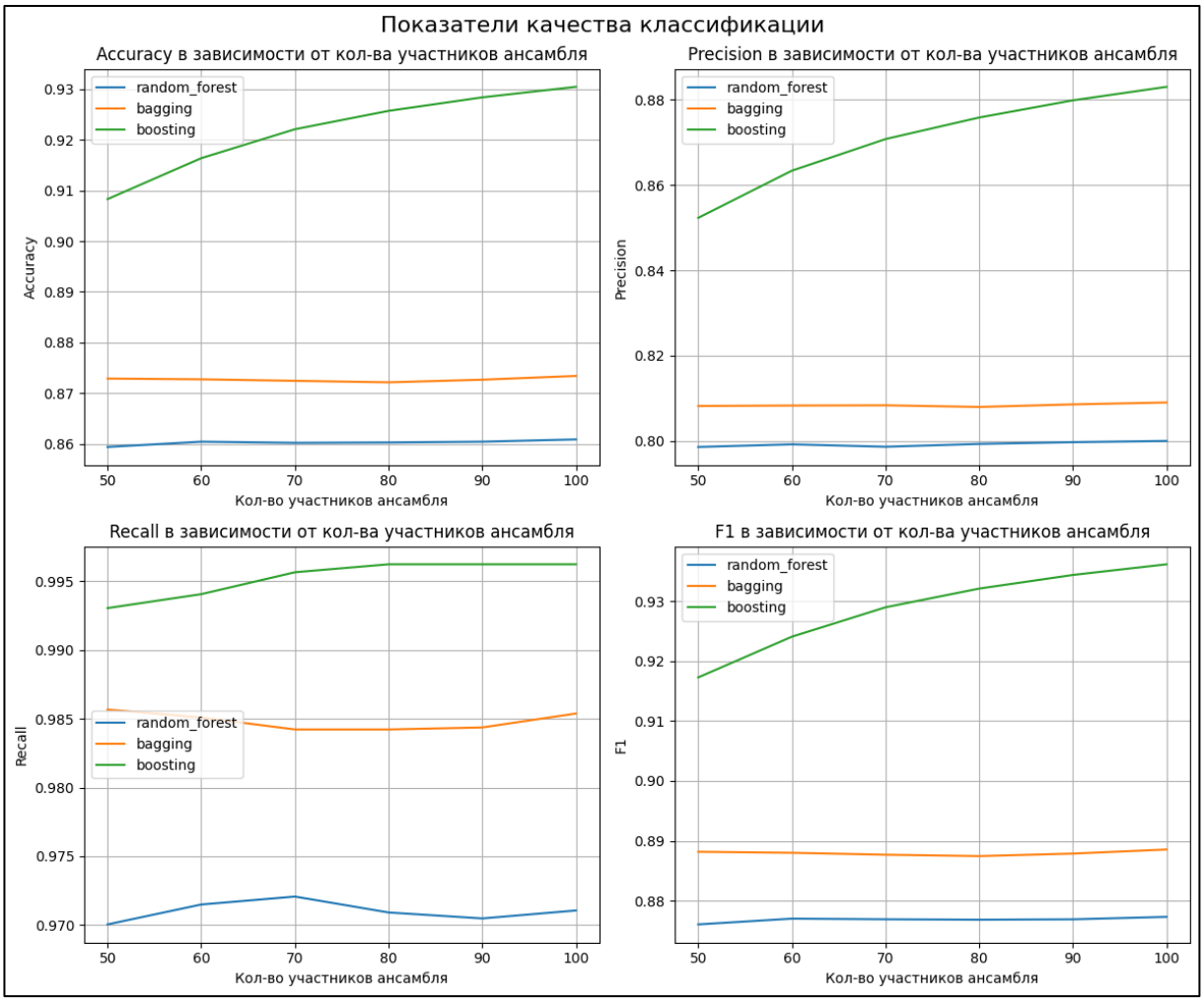

Рисунок 1 – Диаграмма зависимости показателей от участников

По результатам диаграммы можно сделать вывод о следующем.

- С увеличением числа участников ансамбля качество классификации стабильно растет.
- Градиентный бустинг показал лучшие результаты по метрикам по сравнению с бэггингом и случайным лесом.
- Различия между техниками становятся менее выраженными при больших значениях n_estimators.