



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

**Институт информационных технологий (ИИТ)
Кафедра прикладной математики (ПМ)**

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ
по дисциплине «Технологии и инструментарий анализа больших данных»

Практическая работа № 7

Студент группы *ИКБО-01-22 Прокопчук Роман Олегович*

(подпись)

Ассистент *Тетерин Николай Николаевич*

(подпись)

Отчёт представлен «__» декабря 2025 г.

Москва, 2025 г.

ЦЕЛЬ

Ознакомиться с инструментами работы с ансамблевым обучением на Python.

ХОД РАБОТЫ

Задание

- 1) Найти данные для задачи классификации или для задачи регрессии (данные не должны повторяться в группе).
- 2) Реализовать баггинг.
- 3) Реализовать бустинг на тех же данных, что использовались для баггинга.
- 4) Сравнить результаты работы алгоритмов (время работы и качество моделей). Сделать выводы.

Ход работы

В качестве данных был выбран датасет с результатами гонок Формулы 1 – <https://www.kaggle.com/datasets/ignale/formula-one-races-results-1950-2022>.

Код для подготовки данных представлен на рисунке 1.

```
data = pd.read_csv('data/results.csv')

features = ['grid', 'constructorId', 'driverId']
target_col = 'positionOrder'

X = data[features]

y = (data[target_col] <= 10).astype(int)

print(f"Признаки: {features}")
print(f"Целевое событие: Финиш в очках (positionOrder <= 10)")

X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.2, random_state=42)
print(f"Размер обучающей выборки: {X_train.shape[0]}")
print(f"Размер тестовой выборки: {X_test.shape[0]}\n")
```

Рисунок 1 – Код для подготовки данных

Код для реализации баггинга представлен на рисунке 2.

```

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

start_time = time.time()
rf_model.fit(X_train, y_train)
rf_time = time.time() - start_time

rf_pred = rf_model.predict(X_test)

rf_f1 = f1_score(y_test, rf_pred)
rf_acc = accuracy_score(y_test, rf_pred)

print(f"Время обучения Random Forest: {rf_time:.4f} сек.")
print(f"F1-score: {rf_f1:.4f}")
print(f"Accuracy: {rf_acc:.4f}\n")

```

Рисунок 2 – Код для реализации баггинга

Код для реализации бустинга представлен на рисунке 3.

```

cat_features = ['constructorId', 'driverId']

cb_model = CatBoostClassifier(iterations=100, random_state=42, verbose=False)

start_time = time.time()
cb_model.fit(X_train, y_train, cat_features=cat_features)
cb_time = time.time() - start_time

cb_pred = cb_model.predict(X_test)

cb_f1 = f1_score(y_test, cb_pred)
cb_acc = accuracy_score(y_test, cb_pred)

print(f"Время обучения CatBoost: {cb_time:.4f} сек.")
print(f"F1-score: {cb_f1:.4f}")
print(f"Accuracy: {cb_acc:.4f}\n")

```

Рисунок 3 – Код для реализации бустинга

Код для сравнения результатов представлен на рисунке 4. Вывод всей программы представлен на рисунке 5.

```

print(f"{'Алгоритм':<20} | {'F1-score':<10} | {'Время (сек)':<10}")
print("-" * 46)
print(f"{'Random Forest':<20} | {rf_f1:.4f} | {rf_time:.4f}")
print(f"{'CatBoost':<20} | {cb_f1:.4f} | {cb_time:.4f}")

print("\n--- Краткий вывод ---")
if cb_f1 > rf_f1:
    print("CatBoost показал лучшее качество (F1-score).")
else:
    print("Random Forest показал лучшее качество (F1-score).")

if cb_time < rf_time:
    print("CatBoost обучился быстрее.")
else:
    print("Random Forest обучился быстрее.")

```

Рисунок 4 – Код для сравнения результатов

Run 1-4_bagging_vs_boosting (1) x

↺

↻

⋮

↑

↓

≡

≡

🖨

🗑

C:\Users\Remsely\dev\mirea\python\big-data\.venv\Scripts

Признаки: ['grid', 'constructorId', 'driverId']

Целевое событие: Финиш в очках (positionOrder <= 10)

Размер обучающей выборки: 21407

Размер тестовой выборки: 5352

Время обучения Random Forest: 0.8802 сек.
F1-score: 0.6061
Accuracy: 0.6620

Время обучения CatBoost: 1.4432 сек.
F1-score: 0.6371
Accuracy: 0.7031

Алгоритм	F1-score	Время (сек)
Random Forest	0.6061	0.8802
CatBoost	0.6371	1.4432

--- Краткий вывод ---

CatBoost показал лучшее качество (F1-score).

Random Forest обучился быстрее.

Process finished with exit code 0

Рисунок 5 – Результат работы программы

Сравнение алгоритмов показало, что CatBoost превзошел Random Forest по качеству классификации. Точность (Accuracy) модели CatBoost составила 70.3% против 66.2% у Random Forest, а метрика F1-score оказалась выше (0.637 против 0.606).

Это преимущество объясняется способностью CatBoost эффективно обрабатывать категориальные признаки, такие как идентификаторы команд и пилотов. Хотя Random Forest обучился быстрее (0.88 с против 1.45 с), разница во времени не столь значительна, поэтому для данной задачи предпочтительнее использовать CatBoost.

ВЫВОД

В ходе выполненной практической работы проведено ознакомление с инструментами работы с ансамблевым обучением на Python.