|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА** – **Российский технологический университет»**  **РТУ МИРЭА** | | | | |
|  | Институт информационных технологий (ИТ) |
|  | Кафедра прикладной математики |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 7** | |
| **по дисциплине «Технологии и инструментарий анализа больших данных»** | |
|  | |
| Выполнил студент группы ИКБО-20-21 | Сидоров С.Д. |
| Проверил ассистент кафедры ПМ ИИТ | Тетерин Н.Н. |

Москва 2024

**Практическая работа**

1. Найти данные для задачи классификации или для задачи регрессии (данные не должны повторятся в группе).

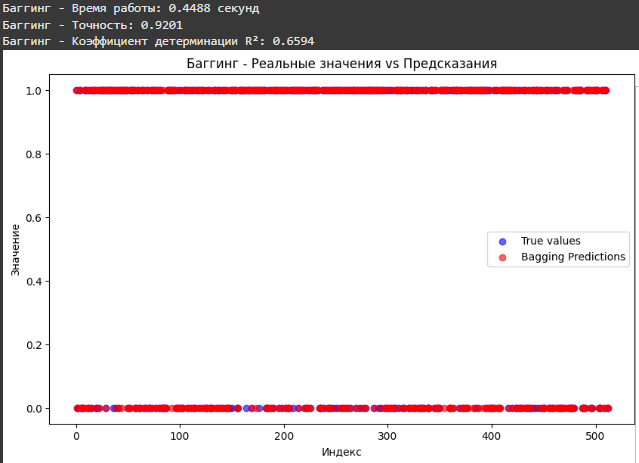
Листинг 1:

|  |
| --- |
| # Загрузка датасета breast cancer из sklearn  cancer\_data = datasets.load\_breast\_cancer()  # Преобразование в DataFrame  data = pd.DataFrame(data=cancer\_data.data, columns=cancer\_data.feature\_names)  data['target'] = cancer\_data.target  # Проверка структуры данных  print(data.head())  # Выделение признаков и целевой переменной  X = data.drop(columns=['target'])  y = data['target']  # Разделение на обучающую и тестовую выборки  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.9, random\_state=42) |

1. Реализовать баггинг.

Листинг 2:

|  |
| --- |
| # Инициализация модели баггинга  bagging\_clf = BaggingClassifier(estimator=DecisionTreeClassifier(), n\_estimators=50, random\_state=42)  # Засекаем время обучения  start\_time = time.time()  # Обучение модели  bagging\_clf.fit(X\_train, y\_train)  # Время работы модели  bagging\_time = time.time() - start\_time  # Предсказания на тестовой выборке  y\_pred\_bagging = bagging\_clf.predict(X\_test)  # Расчет точности (accuracy) и коэффициента детерминации (R²)  bagging\_accuracy = accuracy\_score(y\_test, y\_pred\_bagging)  bagging\_r2 = r2\_score(y\_test, y\_pred\_bagging)  # Вывод результатов  print(f"Баггинг - Время работы: {bagging\_time:.4f} секунд")  print(f"Баггинг - Точность: {bagging\_accuracy:.4f}")  print(f"Баггинг - Коэффициент детерминации R²: {bagging\_r2:.4f}")  # Построение графика для баггинга  plt.figure(figsize=(10, 6))  plt.scatter(range(len(y\_test)), y\_test, color='blue', label='True values', alpha=0.6)  plt.scatter(range(len(y\_pred\_bagging)), y\_pred\_bagging, color='red', label='Bagging Predictions', alpha=0.6)  plt.title('Баггинг - Реальные значения vs Предсказания')  plt.xlabel('Индекс')  plt.ylabel('Значение')  plt.legend()  plt.show() |



1. Реализовать бустинг на тех же данных, что использовались для баггинга.

Листинг 3:

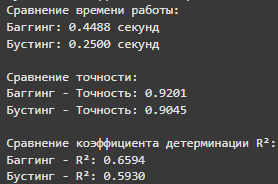
|  |
| --- |
| # Инициализация модели градиентного бустинга  boosting\_clf = GradientBoostingClassifier(n\_estimators=50, random\_state=42)  # Засекаем время обучения  start\_time = time.time()  # Обучение модели  boosting\_clf.fit(X\_train, y\_train)  # Время работы модели  boosting\_time = time.time() - start\_time  # Предсказания на тестовой выборке  y\_pred\_boosting = boosting\_clf.predict(X\_test)  # Расчет точности (accuracy) и коэффициента детерминации (R²)  boosting\_accuracy = accuracy\_score(y\_test, y\_pred\_boosting)  boosting\_r2 = r2\_score(y\_test, y\_pred\_boosting) |



1. Сравнить результаты работы алгоритмов (время работы и качество моделей). Сделать выводы.

Листинг 4:

|  |
| --- |
| # Сравнение времени работы и качества моделей  print(f"Сравнение времени работы:")  print(f"Баггинг: {bagging\_time:.4f} секунд")  print(f"Бустинг: {boosting\_time:.4f} секунд")  print("\nСравнение точности:")  print(f"Баггинг - Точность: {bagging\_accuracy:.4f}")  print(f"Бустинг - Точность: {boosting\_accuracy:.4f}")  print("\nСравнение коэффициента детерминации R²:")  print(f"Баггинг - R²: {bagging\_r2:.4f}")  print(f"Бустинг - R²: {boosting\_r2:.4f}") |



**Выводы:**

**Время работы**: Баггинг оказался медленее

**Качество предсказаний**: на наборе данных **BrestCancer** обе модели показали одинаково высокие результаты. Однако бустинг на более сложных данных может быть точнее, так как его механизмы направлены на улучшение ошибок, тогда как баггинг создает независимые модели, не исправляя ошибки друг друга.