Санкт-Петербургский государственный электротехнический университет им. В.И. Ульянова (Ленина)

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ВОССТАНОВЛЕНИЯ ПРОПУЩЕННЫХ ЗНАЧЕНИЙ В НАБОРЕ ДАННЫХ

Выполнил: Вологдин Максим Дмитриевич, гр. 7381
Руководитель: Жукова Наталия Александровна, к.т.н., доцент
Консультант: Жангиров Тимур Рафаилович, ассистент

Санкт-Петербург, 2021

Актуальность

В реальных наборах данных бывают пропущенные значения. Это может произойти по следующим причинам:

- Ошибки при записи
- Ошибки при измерении
- Невозможность сбора данных

Не все алгоритмы способны работать с неполными данными, поэтому необходимо уметь их восстанавливать наиболее точным образом

Цель и задачи

Цель: Проведение сравнительного анализа методов восстановления пропущенных значений в наборе данных

Задачи:

1. Определение списка методов для исследования
2. Проведение экспериментов по сравнению методов при различных условиях
3. Классификация методов

Методы для исследования

Для сравнительного анализа были выбраны одни из наиболее популярных методов машинного обучения:

1. Заполнение средним и медианой
2. Алгоритм K–средних
3. Алгоритм K–ближайших соседей
4. EM-алгоритм
5. Итеративное вменение

- Линейная регрессия
- Деревья решений
- Случайный лес
- Дополнительные деревья

6. Искусственные нейронные сети

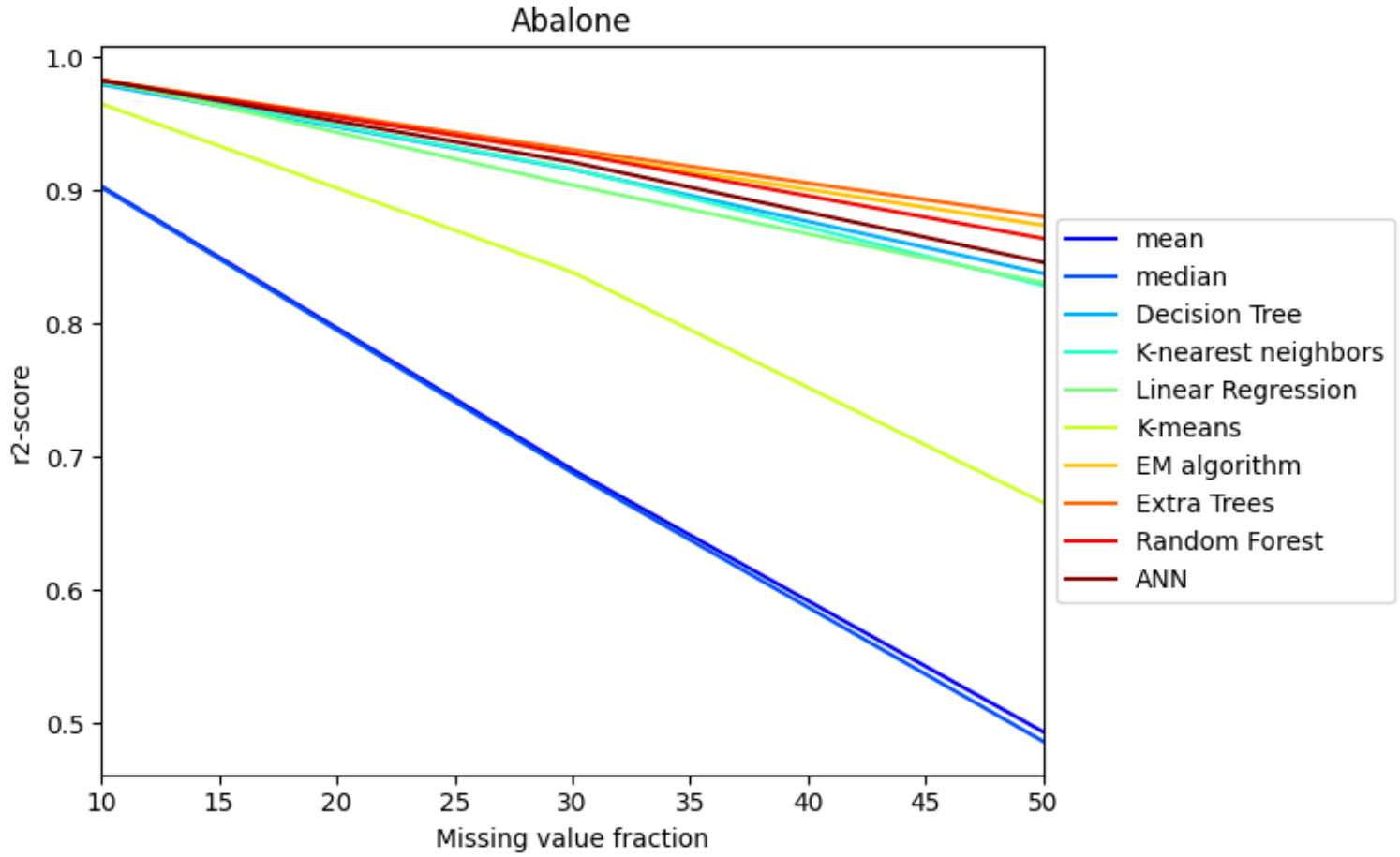
Эксперименты по сравнению методов.

Используемые наборы данных:

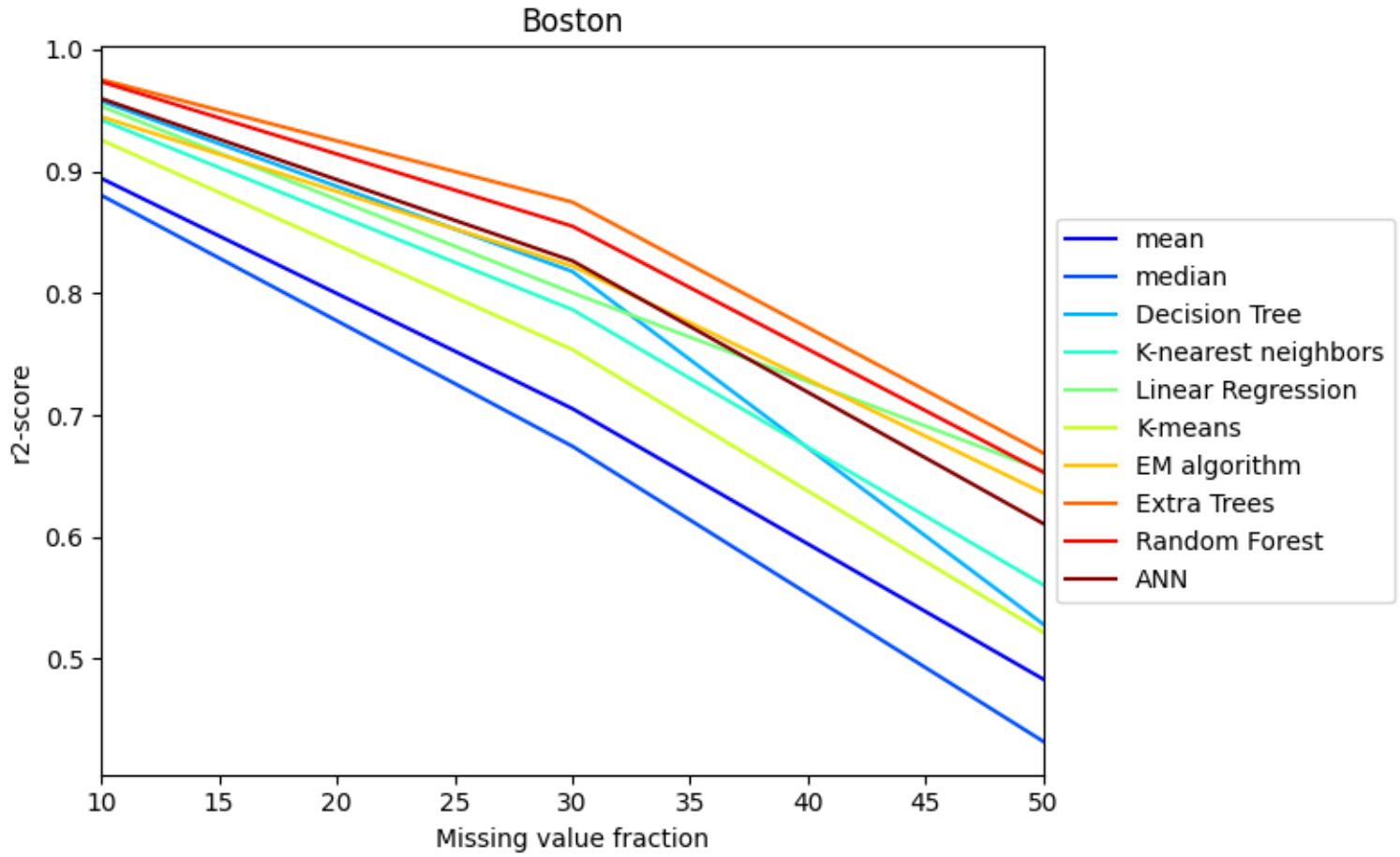
1. **Abalone Dataset**: 4177 объектов и 8 признаков
2. **Boston House Prices**: 506 объектов и 13 признаков
3. **California Housing Prices**: 20640 объектов и 8 признаков
4. **Breast Cancer Wisconsin (Diagnostic) Data Set**: 569 объектов и 30 признаков

Эксперименты проводились на 10%, 30% и 50% пропусков

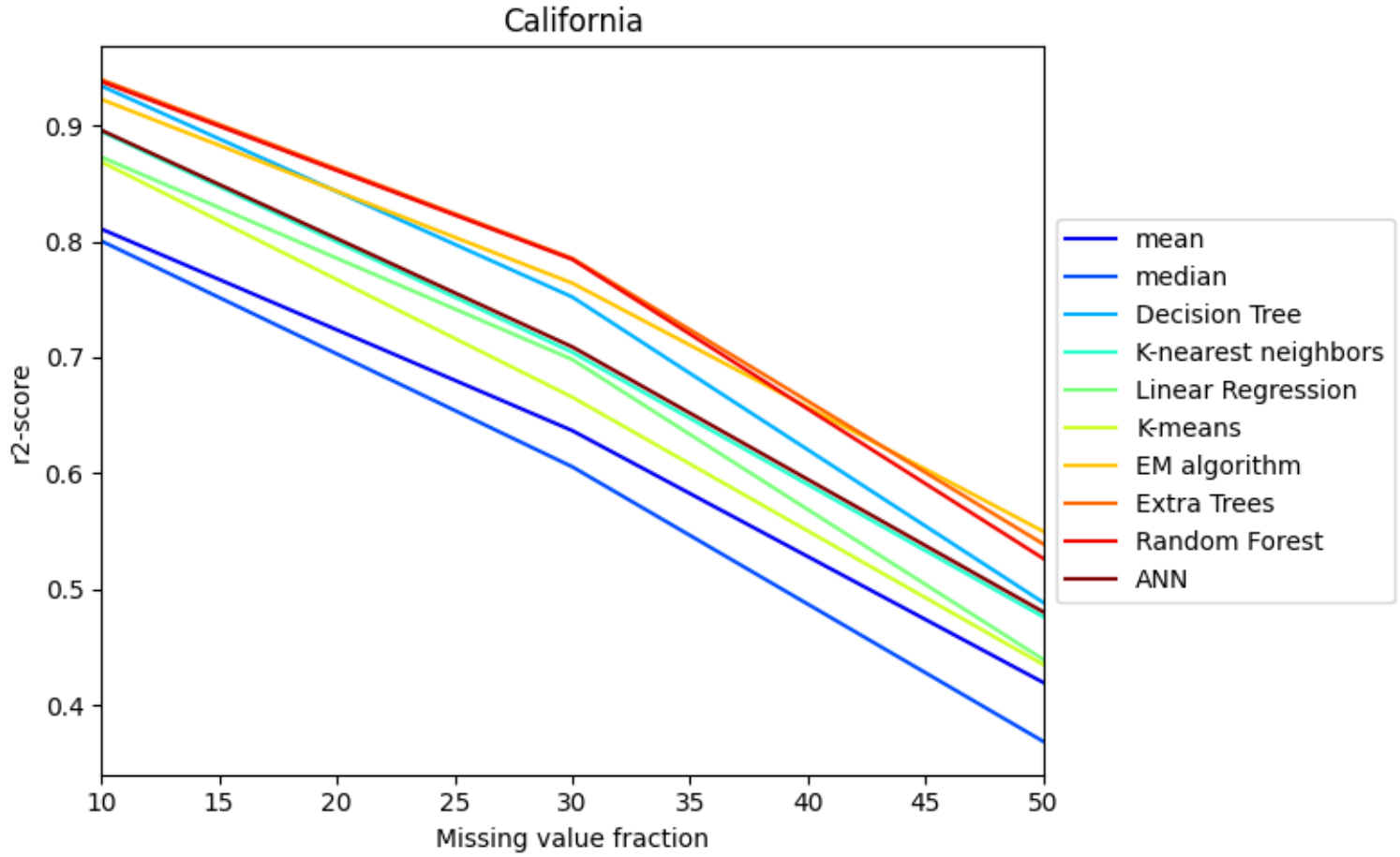
Эксперименты по сравнению методов. Набор Abalone



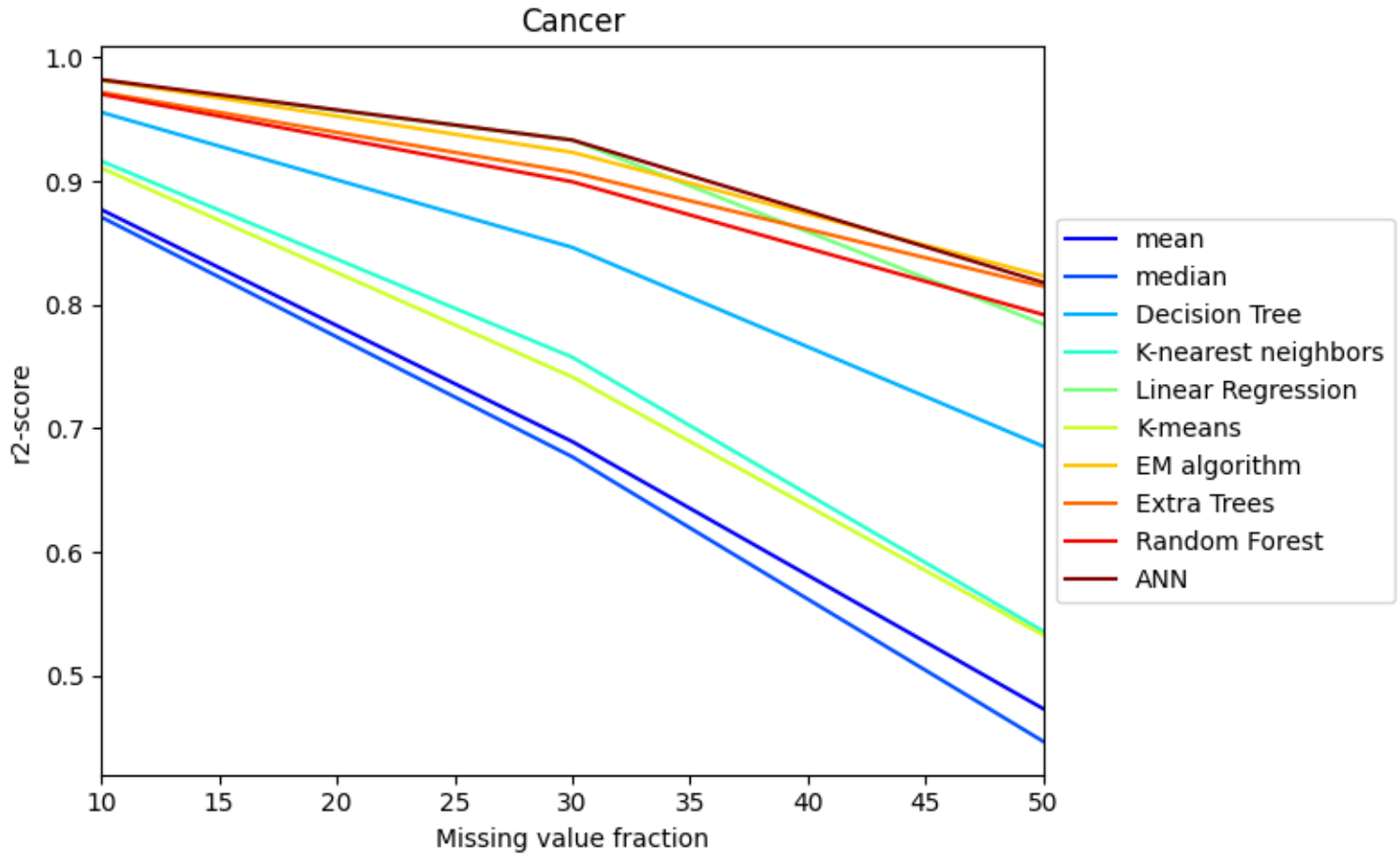
Эксперименты по сравнению методов. Набор Boston



Эксперименты по сравнению методов. Набор California



Эксперименты по сравнению методов. Набор Cancer



Классификация методов

Точность / Скорость	**Низкая (ниже чем у 70% методов)**	**Средняя**	**Высокая (выше чем у 70% методов)**
Высокая (Выше чем у 70% методов)	Заполнение средним и медианой	Итеративное заполнение Линейной регрессией	–
Средняя	Алгоритм K-средних	Алгоритм K–ближайших соседей и Итеративное заполнение Деревьями решений	EM-алгоритм
Низкая (ниже чем у 70% методов)	–	Искусственные нейронные сети	Итеративное заполнение Случайным лесом и Дополнительными деревьями

Заключение

- Был выполнен обзор предметной области и выделены следующие методы для изучения: Метод заполнение средним и медианой, Линейная регрессия, Алгоритм k-средних, Алгоритм k–ближайших соседей, EM-алгоритм, Деревья решений, Случайный лес, Дополнительные деревья, Искусственные нейронные сети
- Были проведены эксперименты по сравнению методов на 10%, 30% и 50% пропусков и по их результатам построены результирующие графики для каждого набора
- Методы были классифицированы по их точности и скорости работы в задаче восстановления пропущенных значений. Наиболее высокие результаты в этих параметрах показали EM–алгоритм и итеративное вменение с помощью Линейной регрессии.

Дальнейшие направления исследований включают в себя рассмотрение методов, не вошедших в данную работу, а также более глубокое изучение нейронных сетей в контексте поставленной задачи.

Апробация работы

- Репозиторий проекта
  https://github.com/Makkksx/VKR2021

Запасные слайды

Коэффициент детерминации

Коэффициент детерминации, или R^2 показывает, насколько условная дисперсия полученной модели отличается от дисперсии реальных значений, то есть насколько хорошо модель описывает данные.

$$R^2\left(y, y_i\right) = 1 - \frac{\sum_{i=0}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=0}^{n}\left(y_i - \bar{y}\right)^2}$$

Где $\hat{y}_i$ – предсказанное значение, y_i – соответствующее истинное значение, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

Постановка задачи

Имеется матрица объектов-признаков $X^{n \times d}$, где n – количество объектов, d – количество признаков. Часть значений матрицы пропущена. Необходимо получить матрицу объектов-признаков без пропущенных значений, наиболее близкую к оригинальной.

Реализация методов

Все методы были реализованы на языке программирования Python 3.9 с помощью библиотеки scikit-learn 0.24.1 для основной части методов и библиотеки datawig 0.2.0 для нейронных сетей.

Время и точность методов на примере набора Abalone с 30% пропусков

Метод	R2	Время выполнения, мс
Среднее	0.507	~15
Медиана	0.504	~15
Линейная регрессия	0.849	~300
K-средних	0.649	~200
K–ближайших соседей	0.755	~3к
EM–алгоритм	0.886	~1.5к
Деревья решений	0.847	~100
Случайный лес	0.870	~30к
Дополнительные деревья	0.889	~14к
ИНС	0.853	~200к

Сложность методов

n – количество объектов, d – количество признаков, i – количество итераций

1. Заполнение средним и медианой – O(n×d)
2. Алгоритм K–средних – O(i×k×d×n), где k – количество кластеров
3. Алгоритм K–ближайших соседей – O(k×d×n), где k – количество соседей
4. EM-алгоритм – O(i×n×d)
5. Итеративное вменение

- Линейная регрессия – O(i×d×n)
- Деревья решений – O(i×d×n×log(n)),
- Случайный лес и Дополнительные деревья – O(m×i×d×n×log(n)),

где m – количество деревьев

Типы пропущенных значений

- Пропуски называют полностью случайными (MCAR), если условная вероятность P (X_j пропущено/прочие X) не зависит ни от X_j, ни от X.
- Пропуски называют случайными (MAR), если условная вероятность P (X_j пропущено/прочие X) не зависит от X_j, но может зависеть от других X. В этом и предыдущем случае механизм пропусков несущественен и к данным применимы методы восстановления.
- Пропуски называют неслучайными (MNAR), если условная вероятность P (X_j пропущено/прочие X) зависит от X_j. В этом случае механизм пропусков существенен и для корректного анализа данных необходимо знать этот механизм.

Случайный лес и Дополнительные деревья

- В случайных лесах каждое дерево строится независимо друг от друга на случайной выборке из начального набора данных. Кроме того, в процессе построения деревьев выбирается оптимальная точка при разбиении каждого узла.
- Алгоритм дополнительных деревьев также построен на множестве решающих деревьев, но в отличии от случайного леса, дополнительные деревья выбирают случайную точку при разбиении узлов, что позволяет алгоритму работать быстрее.