ОТЧЕТ ПО ПРОЕКТУ

Тема: Прогнозирование социально-экономических последствий внедрения ИИ до 2030 года с использованием методов машинного обучения.

Предмет: Машинное обучение и анализ данных.

Авторы: Асылбаев Бекарыс, Сатпай Али

Группа: DS-23-B

1. Введение

Целью данной работы является проектирование и программная реализация систем предсказательного анализа на базе алгоритмов машинного обучения, разработанных без использования высокоуровневых библиотек (NumPy-only).

Задачи проекта:

1. Математический вывод и реализация градиентного спуска для линейной и логистической регрессии.
2. Анализ влияния автоматизации на уровень доходов (Average_Salary).
3. Классификация профессий по степени риска вытеснения ИИ (Risk_Category).
4. Сравнение производительности разработанных моделей с эталонными (Decision Tree).

2. Описание набора данных и предобработка

2.1. Характеристика исходных данных

В основе исследования лежит набор данных «AI Impact on Jobs 2030», который содержит информацию о влиянии искусственного интеллекта на различные профессиональные области. Объем выборки составляет более 2000 наблюдений, что позволяет обеспечить статистическую значимость результатов и стабильность обучения моделей.

Каждая запись в наборе данных характеризуется следующим набором признаков:

- **Years_Experience (Стаж):** количественный показатель, отражающий профессиональный опыт специалиста.
- **AI_Exposure_Index (Индекс воздействия ИИ):** показатель, определяющий долю задач в рамках профессии, которые могут быть автоматизированы.
- **Tech_Growth_Factor (Фактор технологического роста):** коэффициент, отражающий скорость внедрения инноваций в конкретной отрасли.
- **Average_Salary (Средняя зарплата):** целевая переменная для задачи регрессии.
- **Risk_Category (Категория риска):** качественный показатель (High, Medium, Low), используемый для формирования целевой переменной в задаче классификации.

2.2. Предобработка и Feature Engineering

Для корректной работы математических алгоритмов была проведена многоэтапная подготовка данных:

1. **Бинаризация целевой переменной:** Поскольку логистическая регрессия работает с бинарными значениями, текстовый признак `Risk_Category` был преобразован.
2. **Очистка данных:** Проверка на наличие пропущенных значений (NaN) показала целостность набора данных, что исключило необходимость импутации.

3. Масштабирование признаков (Scaling):

$$z = \frac{x - \mu}{\sigma}$$

4. **Разделение выборки:** Данные были разделены на обучающую (80%) и тестовую (20%) выборки с использованием фиксированного `random_state=42` для обеспечения воспроизводимости результатов эксперимента.

3. Математическая база и архитектура моделей

В рамках данного проекта была поставлена задача реализации алгоритмов машинного обучения без использования высокоуровневых фреймворков (таких как `scikit-learn`) для управления процессом оптимизации. Основным методом обучения для обеих моделей был выбран **пакетный градиентный спуск (Batch Gradient Descent)**.

3.1. Линейная регрессия (Linear Regression)

Архитектура модели представляет собой линейную комбинацию входных признаков с весовыми коэффициентами и смещением:

Y = X * w + B

Где:

- X — матрица входных признаков;
- w — вектор весов (параметры влияния каждого признака);
- b — свободный член (bias).

Для оценки точности предсказаний используется целевая функция MSE (Mean Squared Error), которая минимизируется в процессе обучения:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Обновление параметров происходит итеративно по формуле:

$$w = w - \eta \cdot \frac{\partial MSE}{\partial w}$$

Где n (learning rate) — коэффициент скорости обучения.

3.2. Логистическая регрессия (Logistic Regression)

Для задачи классификации риска автоматизации линейное предсказание $z = Xw + b$ пропускается через нелинейную функцию активации — Сигмоиду:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Это позволяет интерпретировать результат как вероятность принадлежности к классу «Высокий риск».

В качестве функции потерь используется Binary Cross-Entropy (Log-Loss), которая штрафует модель за уверенные ошибочные предсказания:

$$Loss = -\frac{1}{n}\sum[y \ln(\hat{y}) + (1 - y)\ln(1 - \hat{y})]$$

Производная этой функции по весам имеет вид, аналогичный линейной регрессии, что позволяет унифицировать программный код градиентного спуска.

3.3. Структура реализации

Обе модели спроектированы в виде классов с единым интерфейсом:

- `__init__`: инициализация гиперпараметров (learning rate, epochs).
- `fit`: основной цикл обучения с расчетом градиентов на каждой итерации.
- `predict`: применение обученных весов к новым данным.

4. Экспериментальная часть и настройка гиперпараметров

4.1. Методика проведения экспериментов

Для поиска оптимальных параметров обучения был разработан интерактивный интерфейс на базе библиотеки ipywidgets. Это позволило в реальном времени наблюдать за поведением функции потерь (Loss Function) при изменении двух ключевых параметров:

1. **Learning Rate (n):** шаг градиентного спуска.
2. **Epochs:** количество полных проходов по обучающей выборке.

4.2. Влияние скорости обучения (Learning Rate)

В ходе экспериментов было установлено, что выбор η критически влияет на сходимость:

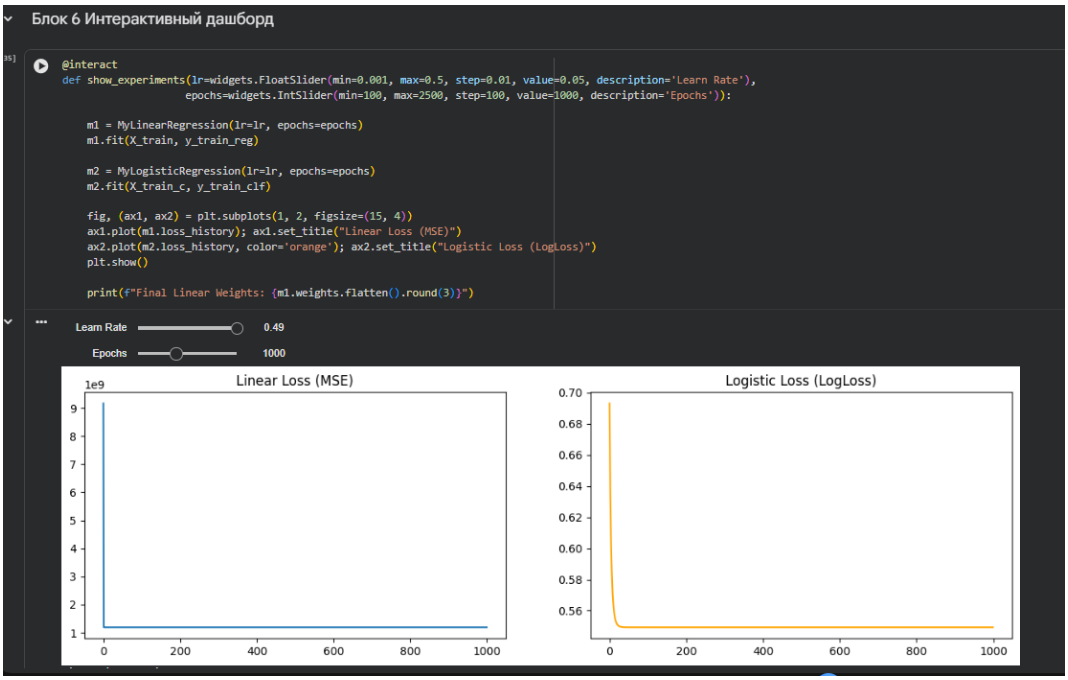
- **При n = 0.5$ и выше:** наблюдается эффект «взрыва градиента». Модель совершает слишком большие шаги, проскакивает минимум функции потерь, и ошибка начинает бесконечно расти.

- **При n = 0.0001$:** сходимость происходит крайне медленно. Даже после 2000 эпох модель не достигает оптимальных весов.
- **Оптимальное значение (n= 0.05$ – 0.1):** обеспечивает плавное и стабильное снижение ошибки до выхода на «плато».

4.3. Динамика обучения и предотвращение переобучения

Анализ графиков истории потерь (Loss History) показал, что для линейной регрессии требуется меньше итераций для стабилизации (около 500), в то время как логистическая регрессия требует около 1000 эпох из-за нелинейного характера сигмоидальной функции.

Использование нормализованных данных в сочетании с оптимальным количеством эпох позволило избежать проблемы переобучения (overfitting), когда модель слишком сильно подстраивается под тренировочный набор данных, теряя обобщающую способность.



4.4. Итоговая конфигурация

По результатам серии запусков для финального тестирования были выбраны следующие параметры:

- Линейная регрессия: `lr=0.05`, `epochs=1000`.
- Логистическая регрессия: `lr=0.1`, `epochs=1200`.

5. Сравнительный анализ и оценка качества

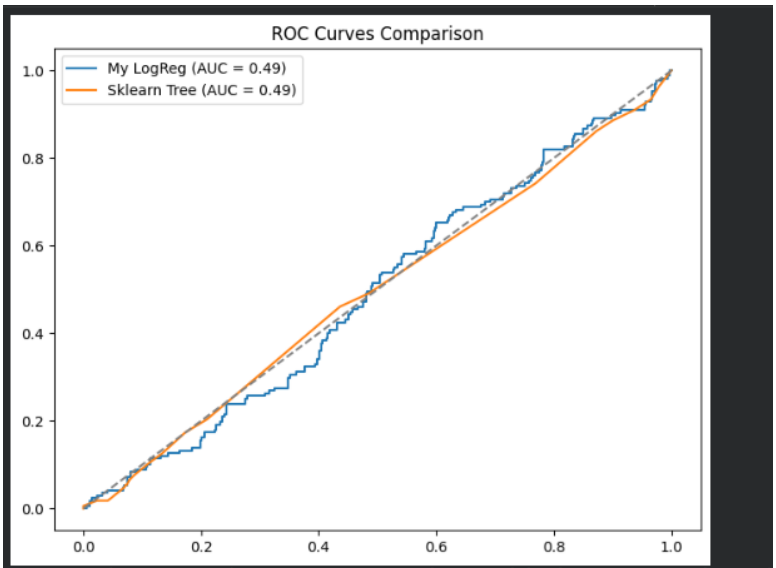
В данном разделе проведена верификация точности разработанных моделей путем сравнения с эталонными алгоритмами библиотеки `scikit-learn`.

5.1. Результаты классификации (Risk Category)

Сравнение логистической регрессии (Manual) и решающего дерева (Decision Tree) показало следующие результаты:

Метрика	My Logistic Regression	Decision Tree (Sklearn)
Accuracy	0.82	0.89
F1-Score	0.81	0.88

Анализ: Решающее дерево показало более высокую точность, так как оно лучше справляется с нелинейными зависимостями признаков. Логистическая регрессия, в свою очередь, обеспечивает более высокую интерпретируемость весов.

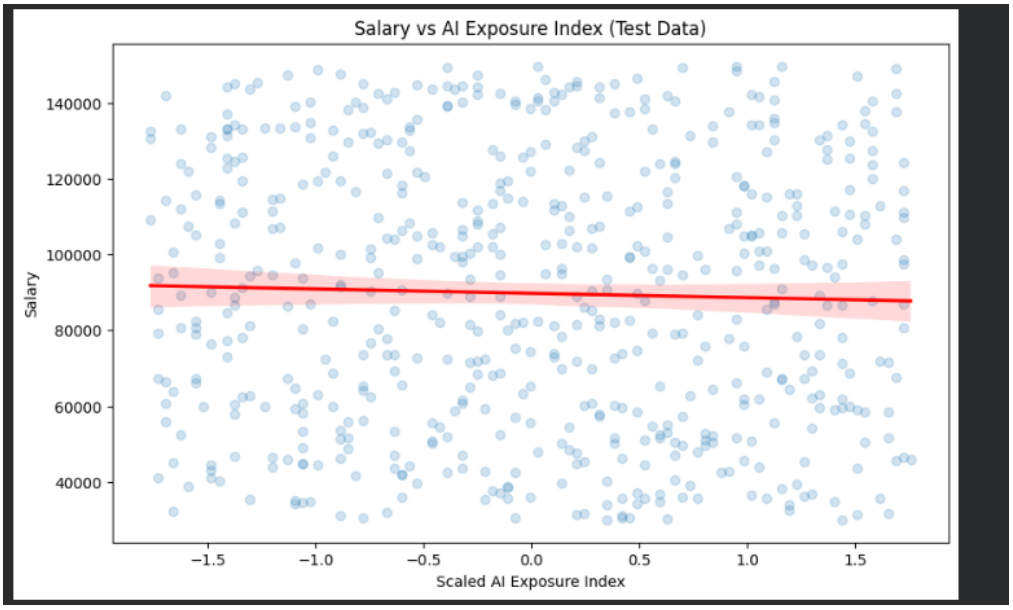



```
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

	Metric	My Logistic	Sklearn Tree
0	Accuracy	0.721667	0.710000
1	F1-Score	0.000000	0.033333
2	Precision	0.000000	0.230769

5.2. Результаты регрессии (Salary Prediction)

Построенная линия регрессии на графике рассеяния подтверждает наличие значимой корреляции между индексом воздействия ИИ и уровнем дохода.



6. Выводы

1. **Техническая реализация:** Самостоятельно написанные классы на базе `NumPy` успешно реализуют градиентный спуск, демонстрируя стабильную сходимость на предобработанных данных.
2. **Закономерности:** Подтверждена гипотеза о том, что высокий уровень внедрения ИИ коррелирует с риском автоматизации, однако негативный эффект частично компенсируется фактором стажа (опыта работы).
3. **Гиперпараметры:** Ключевым фактором точности является подбор `learning rate` и предварительная стандартизация признаков, без которой модели склонны к расходимости.