УТВЕРЖДЕН

RU.17701729.11.04-01 ПЗ 01-1

Разработка программного комплекса для исследования влияния аномальных наблюдений на точность прогнозирования в регрессионных моделях

Пояснительная записка

RU.17701729.11.04-01 РП 01-1

Листов 29

Подп.идата	
Инв.№дубл.	
Взам.инв.№	
Подп.идата	
Инв.№подл	

Москва 2025

ГЛОССАРИЙ

1. Регрессионная модель – математический метод прогнозирования, устанавливающий зависимость между целевой переменной и одним или несколькими признаками.
2. Аномальные наблюдения – точки данных, которые значительно отклоняются от остальных наблюдений в наборе данных и могут негативно влиять на точность прогнозирования.
3. JSON – легкий формат обмена данными, используемый для хранения конфигураций моделей и параметров экспериментов.
4. Асинхронные вычисления - метод параллельного выполнения задач для повышения производительности, особенно при проведении множества экспериментов.
5. Целевая переменная - поле в наборе данных, значение которого модель стремится предсказать на основе признаков.
6. Признак - поле в наборе данных, которое используется как для предсказания величины-цели.
7. Метрики качества - показатели, используемые для оценки точности регрессионных моделей или обнаружения аномальных наблюдений.
8. Формат CSV – текстовый формат для представления табличных данных, где значения разделены специальным символом.
9. Шум – случайные отклонения в данных, которые не отражают истинную закономерность и могут возникать из-за ошибок измерения.
10. Хвост распределения – область распределения вероятностей, удаленная от его центральной части.
11. Гиперпараметр – параметр алгоритма машинного обучения, который устанавливается перед началом обучения и не изменяется в процессе его обучения.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Наименование программы

Наименование темы разработки: «Разработка программного комплекса для исследования влияния аномальных наблюдений на точность прогнозирования в регрессионных моделях».

Наименование темы разработки на английском языке: «Development of a Software Package to Study the Influence of Outliers on the Prediction Accuracy in Regression Models».

Наименование программы для пользователя – «MSnOutliers».

1.2. Документы, на основании которых ведется разработка

Реализация данного проекта обусловлена темой, которая была утверждена академическим руководителем программы по направлению 09.03.04 “Программная инженерия” в соответствии с учебным планом подготовки бакалавров. Такой выбор темы курсового проекта представляет собой необходимый этап в академической программе, что обуславливает проведение данной разработки.

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1. Функциональное назначение

Программный инструмент «MSnOutliers» предназначен для анализа и исследования робастности различных методов регрессионного анализа в условиях присутствия в данных аномальных наблюдений. Приложение предоставляет следующие возможности:

1) Моделирование набора данных с контролируемыми параметрами шума различных распределений.
2) Оценка эффективности различных статистических методов регрессии при наличии аномальных наблюдений разного количество и характера.
3) Применение алгоритмов машинного обучения для обнаружения аномальных наблюдений, последующее удаление аномальных наблюдений и оценка качества детектирования аномальных наблюдений.
4) Расчет метрик качества регрессионных моделей при их применении на очищенных моделью машинного обучения данных.
5) Итоговая визуализация полученных результатов в виде графиков зависимости ошибки от уровня шума.

2.2. Эксплуатационное назначение

Приложение «MSnOutliers» реализовано как проект для анализа и оценки надежности статистических методов регрессии при наличии шума различного характера. Инструмент работает как с пользовательскими данными, которые могут быть предоставлены в формате CSV, так и с данными, которые можно сгенерировать средствами самого продукта.

Благодаря модульной архитектуре и многопоточности, программа позволяет эффективно проводить серии экспериментов с различными параметрами методов и типов шума, а результаты экспериментов сохраняются и визуализируются. В совокупности это делает «MSnOutliers» мощным инструментом для анализа методов регрессии и машинного обучения.

2.3. Область применения программы

Аудитория продукта, который анализирует качество методов регрессии и машинного обучения, включает в себя студентов и преподавателей в рамках курсов по математической статистике, анализу данных и машинному обучению для детальной демонстрации положительных и отрицательных сторон методов.

Инструмент совместим с большинством основных операционных систем, таких как Windows, Linux и macOS и преимущественно реализован на C++ с использованием Python для визуализации результатов.

3. ТРЕБОВАНИЯ К ПРОГРАММЕ

3.1. Постановка задачи на разработку программы

Приложение должно быть разработано как инструмент для исследования эффективности методов регрессионого анализа с наличием аномальных наблюдений в исходных данных. Продукт должен поддерживать работу как с пользовательскими наборами данных, так и с генерируемыми выборками с выбранными параметрами. Программа должна предоставлять возможность контролировать добавление шума во входные данные перед применением методов регрессии. Также продукт должен включать возможность применения алгоритмов машинного обучения для предварительного обнаружения и исключение аномальных наблюдений. Результаты работы программы должны представляться в виде:

1) Визуализированных графиков зависимости ошибки модели от интенсивности шума
2) Метрик качества метода регрессии и обнаружения аномалий
3) Рассчитанных коэффициентов метода регрессии

3.2. Описание алгоритма программы

3.2.1. Теоретическое обоснование

В рамках данного проекта использованы различные методы машинного обучения для обнаружения аномальных наблюдений в данных. Выбранные алгоритмы представляют собой различные подходы к определению того, что является “нормальным” для заданного набора данных.

Существует два основных подхода к обнаружению аномалий: алгоритмы, основанные на плотности распределения данных, и методы, основанные на концепции близости наблюдений. К первым относятся методы Kernel Density Estimation и Isolation Forest, ко вторым - алгоритмы DBSCAN и K-Nearest Neighbors.

Методы, основанные на плотности, оценивают вероятностное распределение данных и классифицируют как аномальные те наблюдения, которые находятся в областях с низкой плотностью. Так, IForest строит композицию деревьев, которые изолируют наблюдения путем разделения пространства признаков, и

считает аномалиями те наблюдения, которые изолируются быстрее других. KDE оценивает плотность распределения с помощью ядерных функций и выявляет наблюдения с низкой оценкой плотности.

Методы, основанные на близости, рассматривают локальное окружение каждого наблюдения. DBSCAN группирует наблюдения, находящиеся в плотных областях, и классифицирует наблюдения, не принадлежащие ни одному кластеру, как аномалии. KNN определяет аномальные наблюдения на основе расстояния до k-го ближайшего соседа - наблюдения с большими расстояниями считаются выбросами.

Для каждого метода действует условие, что если этим методом все объекты были помечены как аномалии, то все объекты считаются неаномальными и исходные данные остаются неизмененными. Также до обучения методов к входным данным применяется нормализация всех признаков. Этот подход зачастую стабилизирует работу алгоритмов и упрощает подбор гиперпараметров.

Для генерации шума во входных данных используются различные типы статистических распределений: нормальное, Стьюдента, Коши и Лапласа, каждое из которых имеет свои характеристики, влияющие на свойства аномальных наблюдений. Нормальное распределение создает симметричный шум с быстро убывающими хвостами. Распределения Стьюдент и Коши, наоборот, порождают более тяжелые хвосты и более экстремальны аномальные наблюдения. Распределение Лапласа характеризуется экспоненциально убывающими хвостами. Такое разнообразие шумов в регрессии позволяет моделировать различные ситуации зашумления данных.

3.2.1.1. Характеристика использующихся в приложении методов машинного обучения

3.2.1.1.1. Метод изолирующего леса: Isolation Forest

Заключается в рекурсивном разделении данных случайным образом. Аномальные наблюдения требуют меньшего количества разделений для их изоляции, поскольку они обычно существенно

отличаются от нормальных данных. Метод использует композицию деревьев для каждого выбирается случайное подмножество признаков и случайная точка разделения. Аномальность наблюдения оценивается как среднее пути, необходимому для её изоляции (попадания в листовой узел дерева) в лесу. Чем меньше полученное среднее, тем более вероятнее, что наблюдение является аномальным.

Преимущество метода заключается в способности работать эффективно с данными любой размерности. Также гиперпараметры метода обычно задаются стандартными значениями (количество деревьев – 100, максимальная глубина дерева – 10-12), что исключает необходимость их подбора.

Недостатками алгоритма являются его вероятностный характер, из-за которого результаты алгоритма могут быть недетерминированными для одних и тех же данных. Существенным недостатком метода является сам результат метода на наблюдении – вероятность этого объекта быть аномальным. Делается неявное предположение о некотором пороге, при котором объект является аномальным, что зачастую заранее неизвестно. Также делается предположение, что аномальные наблюдения существенно отличаются от нормальных данных, что тоже заранее не подтверждено.

3.2.1.1.2. Метод кластеризации на основе плотности Density-Based Spatial Clustering of Applications with Noise: DBSCAN

Заключается в группировании наблюдений, находящихся в плотных областях и классификации наблюдений в областях с низкой плотностью как аномальные.

Использует два гиперпараметра:

1) r – определяет окрестность наблюдения, в которой другое наблюдение считается близкой к данной

2) minimumClusterSize – минимальное количество наблюдений, необходимое для формирования кластера

Наблюдения, которые не принадлежат ни к одному кластеру, рассматриваются как аномальные.

Преимущество метода заключается в его способности обнаруживать кластеры произвольной формы.

Недостатками метода является высокая чувствительность к выбору гиперпараметров метода даже при условии нормализации данных.

3.2.1.1.3. Метод оценки плотности ядра: KDE

Основан на непараметрической оценке функции плотности вероятности данных с использованием ядерных функций для вычисления гладкой аппроксимации распределения.

Gamma – гиперпараметр сглаживания. По умолчанию обратно пропорционален размерности входных данных.

Наблюдения с низкой оценкой плотности классифицируются аномальными. Пороговое значение плотности для определения аномальных наблюдений вычисляется на основе плотности распределения всех наблюдений.

Преимуществом алгоритма является гибкость и способность адаптироваться к многомодальным распределениям без предположений о форме данных.

Недостатком является необходимость точной настройки гиперпараметра сглаживания для достижения оптимальных результатов.

3.2.1.1.4. Метод k-ближайших соседей: KNN

Использует расстояние до k-го ближайшего соседа как меру локальной плотности. Наблюдения с большим расстоянием до своего k-ого ближайшего соседа классифицируются как

выбросы. Алгоритм применяет евклидово расстояние для определения близости наблюдений.

Использует два основных гиперпараметра: k - количество ближайших соседей для рассмотрения, contamination - ожидаемая доля аномальных наблюдений в наборе данных. Подход эффективен для обнаружения локальных аномалий и не требует предположений о глобальном распределении данных и особенно полезен, когда выбросы определяются относительно локальной структуры данных, а не глобальной. Недостатком подхода является сложность определения оптимального значения k, низкая эффективность в данных с переменной плотностью. Также одним из гиперпараметров метода является ожидаемая доля аномальных наблюдений, что зачастую заранее неизвестно.

3.2.1.1.5. Отсутствие предварительной обработки

Предусматривает использование исходных данных без предварительной обработки алгоритмами машинного обучения для удаления аномальных наблюдений.

3.2.1.2. Характеристика использующихся в приложении генераций шума

3.2.1.2.1. Гауссово распределение: Normal

Основано на нормальном распределении, характеризуемом двумя параметрами: средним значением и стандартным отклонением. Генерирует симметричный шум, значения которого фокусируются вокруг первого параметра, а вероятность аномальных значений экспоненциально уменьшается с увеличением расстояния от среднего.

3.2.1.2.2. Распределение Стьюдента: Student

Основано на распределении Стьюдента с n (параметр) степенями свободы. Создает шум с тяжелыми хвостами, увеличивая

вероятность появления экстремальных аномальных значений. Чем выше число степеней свободы n, тем больше распределение вырождается в нормальное.

3.2.1.2.3. Распределение Коши: Cauchy

Аналогично распределению Стьюдента, создает шум с тяжелыми хвостами, увеличивая вероятность появления экстремальных аномальных значений. Моделирует очень нестабильные данные из-за неопределенного среднего значения и среднеквадратического отклонения.

3.2.1.2.4. Распределение Лапласа: Laplace

Создает шум с более острым пиком в среднем значении (параметр) и экспоненциально убывающими хвостами (масштаб убывания задается вторым параметром). В сравнении с распределением Гаусса, имеет у экстремальных значений более высокую вероятность.

3.2.1.2.5. Масштабирование целевой переменной

Альтернативный метод создания аномальных наблюдений, заключается в умножении целевой переменной на константу. Подход моделирует ситуацию с резким выбросом, которые в реальной жизни могут возникать при внезапных ошибках устройств или измерений. Метод тем самым создает систематический шум, а не случайный.

3.2.2. Архитектура проекта

3.2.2.1. Пользовательский интерфейс

Представляет собой интерактивную среду для настройки параметров экспериментов. Пользователь получает возможность выбирать данные для анализа, параметры шума, методы регрессии и методы машинного обучения для обнаружения аномалий. Результат первичного взаимодействия (выбора всех доступных опций) является

конфигурационный JSON-файл, содержащий необходимую информацию о запуске экспериментов. Далее этот файл поступает на вход блоку с запуском экспериментов.

3.2.2.2. Блок с запуском экспериментов

Представляет собой главную часть, отвечающую за загрузку и генерацию данных, запуск и анализ экспериментов. Включает в себя:

1) Загрузку и обработку данных из CSV-файла.
2) К обработанным данным для каждого запущенного эксперимента добавляется шум при помощи статистических и систематических распределений. Статистические распределения реализованы в одном классе, систематическое распределение сделано как операция домножения на константу.
3) Далее применяется один из выбранных алгоритмов машинного обучения для обнаружения и удаления аномальных наблюдений. Каждый алгоритм реализован как отдельный класс.
4) На отфильтрованных после применения методов машинного обучения данных для определения коэффициентов регрессии используется один из выбранных методов регрессии.
5) Итоговым результатом является сохранение метрик качества регрессии и обнаружения аномальных наблюдений в JSON-файл.
6) Визуализация данных результатов в PNG-файлы. Создание изображений достигается при помощи оборачивания элементов изображения в классы, записи данных классов в JSON-файл и использования Python команд для обработки данных записей и последующего конструирования изображений.

Этапы 2-5 выполняются параллельно при помощи возможностей параллельного выполнения процессов стандарта C++17.

3.2.3. Описание структуры проекта

Программный инструмент «MSnOutliers» имеет разделение на компоненты:

1) Директория ui – отвечает за пользовательский интерфейс приложения. Компонента компилируется в отдельный исполняемый файл, являющийся основной точкой взаимодействия пользователя с системой.

2) Директория src – центр приложения, содержащий основную функцию main и логику использования средств директории lib и входных данных и опций, полученных от пользователя. В main выполняется все основные основные этапы, описанные в п.3.2.2.2. В поддиректории PAINTINGS содержатся классы-обертки вокруг различных геометрических образов для удобной записи этих образов в JSON-файл. Компонента компилируется в отдельный исполняемый файл, который будет запущен при использовании пользовательского интерфейса при отправке опций экспериментов, но также исполняемый файл может быть запущен вручную из командной строки.

3) Директория lib – отвечает за реализацию основного функционала и использования методов регрессии и машинного обучения. Включает в себя поддиректории COMMON, содержащую классы парсинга CSV-файлов, расчета метрик и предобработки данных, ML, содержащую реализации алгоритмов машинного обучения, MS, содержащую классы статистических методов регрессии и DISTIRIBUTIONS, содержащую класс с реализацией генерации шумов различных распределений. Компонента компилируется в статическую библиотеку, которая затем связывается с исполняемыми файлами из компонент ui и src.

3.2.4. Описание содержания исходного кода

3.2.4.1. Директория lib

3.2.4.1.1. Поддиректория COMMON

3.2.4.1.1.1. Файлы DataDeNoiser

Содержат класс DataDeNoiser, определяющий добавление шума в данные и последующее его удаление с помощью различных алгоритмов машинного обучения (3.2.5.1.1.1)

3.2.4.1.1.2. Файлы Metrics

Содержат класс Metrics, определяющий статистические методы для расчета различных метрик качества методов регрессии (3.2.5.1.1.2)

3.2.4.1.1.3. Файл Matrix

Определяющий типы данных и функции для работы с матрицами (3.2.5.1.1.3)

3.2.4.1.2. Поддиректория ML

3.2.4.1.2.1. Файлы KNN/KNN

Содержат класс KNN, определяющий реализацию алгоритма K-Nearest Neighbors для обнаружения аномальных наблюдений (3.2.5.1.2.1)

3.2.4.1.2.2. Файлы KDE/KDE

Содержат класс KDE, определяющий реализацию алгоритма Kernel Density Estimation для обнаружения аномальных наблюдений (3.2.5.1.2.2)

3.2.4.1.2.3. Файлы iForest/Node и iForest/iForest

Содержат классы Node и iForest, определяющие реализацию алгоритма Isolation Forest для обнаружения аномальных наблюдений (3.2.5.1.2.3)

3.2.4.1.2.4. Файлы DBSCAN/DBSCAN

Содержат класс DBSCAN, определяющий реализацию алгоритма Density-Based Spatial Clustering of Applications with Noise для выявления аномальных наблюдений (3.2.5.1.2.4)

3.2.4.1.3. Поддиректория DISTRIBUTIONS

3.2.4.1.3.1. Файлы ErrorDistributions

Содержат класс ErrorDistributions, определяющий реализацию генерации случайных чисел с различными типами распределений для создания шума во входных данных (3.2.5.1.3.1)

3.2.4.2. Директория src

3.2.4.2.1. Поддиректория PAINTINGS

3.2.4.2.1.1. Файлы Drawable

Содержат абстрактный базовый класс Drawable, определяющий общий интерфейс для всех графических объектов (3.2.5.2.1.1.1)

3.2.4.2.1.2. Файлы Graph

Содержат класс Graph, определяющий контейнер для графических объектов и управляет процессом их визуализации (3.2.5.2.1.2.1)

3.2.4.2.1.3. Файлы Scatter

Содержат класс Scatter, определяющий создание точечных графиков, используемых для визуализации зависимости ошибки от уровня шума (3.2.5.2.1.3.1)

3.2.4.2.1.4. Файлы Line

Содержат класс Line, определяющий отображения линейных функций (3.2.5.2.1.4.1)

3.2.4.2.1.5. Файлы HorizontalLine

Содержат класс HorizontalLine, определяющий создание горизонтальных линий на графиках (3.2.5.2.1.5.1)

3.2.4.2.1.6. Файлы VerticalLine

Содержат класс VerticalLine, определяющий создание вертикальных линий на графиках (3.2.5.2.1.6.1)

3.2.4.2.1.7. Файлы FunctionPlot

Содержат класс FunctionPlot, определяющий отображение произвольных функций (3.2.5.2.1.7.1)

3.2.4.2.1.8. Файл drawer.py

Содержит Python команды, определяющие чтение конфигурационного файла в формате JSON и создание визуализации с использованием библиотеки matplotlib (3.2.5.2.1.8.1)

3.2.4.2.2. Метод main()

Содержит основную точку входа программы и реализует логику проведения экспериментов с методами регрессии (3.2.5.2.2)

3.2.5. Описание реализации компонент приложения

3.2.5.1. Директория lib

3.2.5.1.1. Поддиректория COMMON

3.2.5.1.1.1. Класс DataDeNoiser

3.2.5.1.1.1.1. _data – исходные входные данные, предоставленные для эксперимента.

3.2.5.1.1.1.2. _dataNoised – исходные входные данные, в которые был внесен указанный шум.

3.2.5.1.1.1.3. _dataMatNoised – матричное представление зашумленных входных данных.

3.2.5.1.1.1.4. _precision – точность определения аномальных наблюдений.

3.2.5.1.1.1.5. _recall – полнота определения аномальных наблюдений.

3.2.5.1.1.1.6. _f1Score – F1-мера, среднее гармоническое точности и полноты.

3.2.5.1.1.1.7. _noisedIndices – вектор значений, указывающих, какие наблюдения стали аномальными.

3.2.5.1.1.1.8. _denoisedIndices – вектор значений, указывающих, какие наблюдения были определены методом машинного обучения как аномальные.

3.2.5.1.1.2. Класс Metrics

3.2.5.1.1.2.1. meanAbsoluteError() – метод для вычисления средней абсолютной ошибки между истинными и предсказанными значениями.

3.2.5.1.1.2.2. meanSquaredError() – метод для вычисления средней квадратической ошибки между истинными и предсказанными значениями.

3.2.5.1.1.2.3. rootMeanSquaredError() – метод для вычисления корня из средней квадратической ошибки между истинными и предсказанными значениями

3.2.5.1.1.2.4. meanAbsolutePercentageError() – метод для вычисления средней абсолютной процентной ошибки между истинными и предсказанными значениями.

3.2.5.1.1.2.5. symmetricMeanAbsolutePercentageError() – метод для вычисления средней симметричной абсолютной процентной ошибки между истинными и предсказанными значениями.

3.2.5.1.1.3. Заголовочгый файл Matrix

3.2.5.1.1.3.1. Shape() – метод, возвращающий размеры входных данных в виде пары из количества строк и количества столбцов.

3.2.5.1.1.3.2. EnsureSameShape() –метод, проверяющий равенство размеров двух матриц.

3.2.5.1.1.3.3. Apply() – шаблонный метод, применяющий переданную функцию к каждому элементу матрицы и возвращающая результат её применения.

3.2.5.1.1.3.4. L1Norm() – метод для вычисления суммы абсолютных разностей между двумя наборами данных.

3.2.5.1.1.3.5. L2Norm() – метод для вычисления суммы квадратов разностей между двумя наборами данных.

3.2.5.1.1.3.6. Print() – функция для вывода матрицы.

3.2.5.1.2. Поддиректория ML

3.2.5.1.2.1. Поддиректория KNN

3.2.5.1.2.1.1. Класс KNN

3.2.5.1.2.1.1.1. _k – количество ближайших соседей для рассмотрения

3.2.5.1.2.1.1.2. _contamination – ожидаемая доля аномальных наблюдения в данных

3.2.5.1.2.1.1.3. _sortedDistances – вектор пар расстояние-индекс, содержащий для каждого наблюдение расстояние до её k-го ближайшего соседа.

3.2.5.1.2.1.1.4. _data – матрица входных данных.

3.2.5.1.2.1.1.5. _threshold – пороговое значение расстояния для определения аномалий.

3.2.5.1.2.1.1.6. KNN() – конструктор, принимающий параметры k и contamination.

3.2.5.1.2.1.1.7. Fit() – метод обучения, который вычисляет расстояние до k-го ближайшего соседа для каждого наблюдения и определяет пороговое значение.

3.2.5.1.2.1.1.8. Predict() – метод, классифицирующий наблюдение на основе расстояния до k-го ближайшего соседа.

3.2.5.1.2.1.1.9. Threshold() – метод, вовзращающий пороговое значение расстояния.

3.2.5.1.2.1.1.10. PairDistances() – метод, возвращающий вектор пар расстояние-индекс.

3.2.5.1.2.2. Поддиректория KDE

3.2.5.1.2.2.1. Класс KDE

3.2.5.1.2.2.1.1. _gamma – параметр сглаживания ядра.

3.2.5.1.2.2.1.2. _rho – пороговое значение плотности для определения аномалий

3.2.5.1.2.2.1.3. _data – матрица входных данных.

3.2.5.1.2.2.1.4. _kernelMatrix – матрица значений ядерной функции между всеми парами наблюдений.

3.2.5.1.2.2.1.5. KernelDensityEstimator() – конструктор, принимающий параметр gamma.

3.2.5.1.2.2.1.6. Fit() – метод обучения, вычисляющий матрицу ядерных функций и пороговое значение.

3.2.5.1.2.2.1.7. Predict() – метод, классифицирующий наблюдения на основе плотности распределения.

3.2.5.1.2.2.1.8. Kernel() – метод, вычисляющий значений ядерной функции для пары наблюдений.

3.2.5.1.2.2.1.9. FindThreshol() – метод определения порового значения плотности.

3.2.5.1.2.3. Поддиректория iForest

3.2.5.1.2.3.1. Класс Node

3.2.5.1.2.3.1.1. _j – индекс признака во входных данных, предикат для которого был построен в узле.

3.2.5.1.2.3.1.2. _predicate – значение предиката в узле

3.2.5.1.2.3.1.3. _subL – указатель на левый дочерний узел.

3.2.5.1.2.3.1.4. _subR – указатель на правый дочерний узел.

3.2.5.1.2.3.1.5. Node() – конструктор, конструктор для создания узла.

3.2.5.1.2.3.1.6. FeatureIndex() – метод доступа к индексу признака во входных данных, предикат для которого был построен в узле.

3.2.5.1.2.3.1.7. Predicate() – методы доступа к значению предиката в узле.

3.2.5.1.2.3.1.8. Left() – метод доступа к левому дочернему узлу.

3.2.5.1.2.3.1.9. Right() – метод доступа к правому дочернему узлу.

3.2.5.1.2.3.1.10. ReplaceL() – метод для замены левого дочернего узла.

3.2.5.1.2.3.1.11. ReplaceR() – метод для замены правого дочернего узла.

3.2.5.1.2.3.2. Класс iForest

3.2.5.1.2.3.2.1. _data – матрица входных данных.

3.2.5.1.2.3.2.2. _dataRows – количество входных наблюдений.

3.2.5.1.2.3.2.3. _dataCols– количество признаков во входных данных.

3.2.5.1.2.3.2.4. _trees – вектор корней деревьев в композиции.

3.2.5.1.2.3.2.5. _nEstimators – количество деревьев в композиции

3.2.5.1.2.3.2.6. _depth– максимальная глубина одного дерева.

3.2.5.1.2.3.2.7. Объекты для генерации чисел: _device, _generator.

3.2.5.1.2.3.2.8. iForest() – конструктор, инициализирующий параметры алгоритма.

3.2.5.1.2.3.2.9. ~iForest() – деструктор, освобождающий память композиции деревьев.

3.2.5.1.2.3.2.10. Fit() – метод построения композиции деревьев изоляции.

3.2.5.1.2.3.2.11. Clear() – метод для очистки памяти деревьев.

3.2.5.1.2.3.2.12. PathLength() – метод определения длины пути до листового узла для наблюдения.

3.2.5.1.2.3.2.13. PredictProba() – метод, вычисляющий вероятность аномальности наблюдения, опираясь на среднею глубину изолированности.

3.2.5.1.2.3.2.14. Rand() – метод для генерации чисел.

3.2.5.1.2.4. Поддиректория DBSCAN

3.2.5.1.2.4.1. Класс DBSCAN

3.2.5.1.2.4.1.1. _r – радиус окрестности наблюдения.

3.2.5.1.2.4.1.2. _minimumClusterSize – минимальное количество наблюдений для формирования кластера

3.2.5.1.2.4.1.3. _idToCluster – вектор, сопоставляющий индексы наблюдений с идентификаторами кластеров. Значение 0 для наблюдения означает, что оно признано моделью аномальным

3.2.5.1.2.4.1.4. _data – матрица входных данных.

3.2.5.1.2.4.1.5. DBSCAN() – конструктор, принимающий параметры радиуса и минимального размера кластера.

3.2.5.1.2.4.1.6. GetNeighbors() – метод, находящий все наблюдения в радиусе _r.

3.2.5.1.2.4.1.7. ExpandCluster() – метод расширения кластера путем добавления.

3.2.5.1.2.4.1.8. getIdToCluster() – метод, вовзращающий сопоставление наблюдений с кластерами.

3.2.5.1.3. Поддиректория DISTRIBUTION

3.2.5.1.3.1. Класс ErrorDistributions

3.2.5.1.3.1.1. DistributionsType – перечисление, содержащее в себе типы распределений, доступных в проекте. В рамках приложения доступны распределения: нормальное, Стьюдента, Коши и Лапласа.

3.2.5.1.3.1.2. _type – поле типа DistributionsType, использующееся для хранения выбранного типа распределения.

3.2.5.1.3.1.3. _distribution – поле типа std::variant, хранящее один из выбранных объектов для генерации данных. Принимает разные значения в зависимости от переданного объекта типа DistributionsType.

3.2.5.1.3.1.4. ErrorDistributions() – конструктор, принимающий объект типа DistributionsType для определения типа распределения два параметра этих распределений.

3.2.5.1.3.1.5. generate() – метод, используутся для генерации значения из распределения _type.

3.2.5.1.3.1.6. laplaceGenerate() – метод для генерации значения из распределения Лапласа.

3.2.5.1.3.1.7. studentGenerate() – метод для генерации значения из распределения Стьюдента.

3.2.5.2. Директория src

3.2.5.2.1. Поддиректория PAINTINGS

3.2.5.2.1.1. Класс Drawable

3.2.5.2.1.1.1. toJson() - виртуальный метод, который должны реализовать все потомки. Преобразует параметры графического элемента в формат JSON для последующей визуализации с помощью скрипта на Python.

3.2.5.2.1.1.2. ~Drawable() - виртуальный деструктор для корректного освобождения ресурсов при удалении объектов с помощью указателя на базовый класс.

3.2.5.2.1.2. Класс Grap

3.2.5.2.1.2.1. _title – поле заголовка графика

3.2.5.2.1.2.2. _xLabel – поле подписи заголовка оси абсцисс

3.2.5.2.1.2.3. _yLabel – поле подписи заголовка оси ординат

3.2.5.2.1.2.4. _configPath – поле пути к конфигурационному файлу

3.2.5.2.1.2.5. _outputPath – поле пути к файлу вывода изображения.

3.2.5.2.1.2.6. _objects – вектор указателей на объекты Drawable.

3.2.5.2.1.2.7. Graph() – конструктор, принимающий информацию о графике и путях к файлам.

3.2.5.2.1.2.8. saveConfig() – метод, сохраняющий конфигурацию графика в JSON-файл.

3.2.5.2.1.2.9. addObject() – метод для добавления нового графического элемента в коллекцию.

3.2.5.2.1.2.10. draw() – метод, вызывающий Python-скрипт для отрисовки графика на основе сохраненной конфигурации.

3.2.5.2.1.3. Класс Scatter

3.2.5.2.1.3.1. _points – вектор пар координат абсцисс и ординат.

3.2.5.2.1.3.2. _color – цвет точек

3.2.5.2.1.3.3. _pointSize – размер точек.

3.2.5.2.1.3.4. _transparency – прозрачность точек.

3.2.5.2.1.3.5. Scatter() – конструктор, принимающий коллекцию точек и параметры отображения.

3.2.5.2.1.3.6. toJson() – реализация метода базового класса, преобразующая параметры точечного графика в JSON-формат.

3.2.5.2.1.4. Класс Line

3.2.5.2.1.4.1. Коэффициенты линейной функции: _k, _b.

3.2.5.2.1.4.2. _color – цвет линии

3.2.5.2.1.4.3. _thickness – толщина.

3.2.5.2.1.4.4. _transparency – прозрачность линии.

3.2.5.2.1.4.5. Line() – конструктор, принимающий коэффициенты линейной функции и параметры отображения.

3.2.5.2.1.4.6. toJson() – реализация метода базового класса, преобразующая параметры линии в JSON-формат.

3.2.5.2.1.5. Класс HorizontalLine

3.2.5.2.1.5.1. _y – ордината линии.

3.2.5.2.1.5.2. _min_x – левая граница линии

3.2.5.2.1.5.3. _max_x – правая граница линии

3.2.5.2.1.5.4. _color – цвет линии

3.2.5.2.1.5.5. _thickness – толщина.

3.2.5.2.1.5.6. _transparency – прозрачность линии.

3.2.5.2.1.5.7. HorizontalLine() – конструктор, принимающий координаты и параметры отображения.

3.2.5.2.1.5.8. toJson() – реализация метода базового класса, преобразующая параметры линии в JSON-формат.

3.2.5.2.1.6. Класс VerticalLine

3.2.5.2.1.6.1. _x: абсцисса линии.

3.2.5.2.1.6.2. _min_x – нижняя граница линии

3.2.5.2.1.6.3. _max_x – верхняя граница линии

3.2.5.2.1.6.4. _color – цвет линии

3.2.5.2.1.6.5. _thickness – толщина.

3.2.5.2.1.6.6. _transparency – прозрачность линии.

3.2.5.2.1.6.7. VerticalLine() – конструктор, принимающий координаты и параметры отображения.

3.2.5.2.1.6.8. toJson() – реализация метода базового класса, преобразующая параметры линии в JSON-формат.

3.2.5.2.1.7. Класс FunctionPlot

3.2.5.2.1.7.1. _points: вектор пар координат абсцисс и ординат – аргумента и значение функции от аргумента.

3.2.5.2.1.7.2. _color – цвет точек

3.2.5.2.1.7.3. _pointSize – размер точек.

3.2.5.2.1.7.4. _transparency – прозрачность точек.

3.2.5.2.1.7.5. FunctionPlot() – конструктор, принимающий функцию, диапозон аргументов, количество точек и параметры отображения.

3.2.5.2.1.7.6. toJson() – реализация метода базового класса, преобразующая функцию в JSON-формат.

3.2.5.2.1.8. Модуль drawer.py

3.2.5.2.1.8.1. Drawer – класс, содержащий методы для отрисовки графических элементов

3.2.5.2.1.8.2. draw_line() – метод для отрисовки линий

3.2.5.2.1.8.3. draw_points() – метод для отрисовки точечных графиков

3.2.5.2.1.8.4. draw_function() – метод для отрисовки функций

3.2.5.2.1.8.5. main() – основная функция, читает JSON-файл и создает графическое PNG представление графических элементов.

3.2.5.2.2. Метод main()

Метод инициализирует необходимые компоненты (FileParser и Metrics).

Далее загружается конфигурационный JSON-файл, соответствующий выходному файлу, появляющегося при загрузке пользователем опций экспериментов.

Полученная конфигурация проходит валидацию при помощи validateJson() из utils.h.

Далее для каждого метода регрессии main() проводит многопоточный сеанс с параметрами, введенными пользователем для конкретно этого метода, экспериментов при помощи стандартной библиотеки futures, состоящий из загрузки данных, добавления шума, использования метода обнаружения аномальных наблюдений и расчет параметров метода регрессии. Результат одного процесса, запущенного параллельно с остальными, является класс ExperimentResult из utils.h с метриками детекции и регрессии. Затем для каждого уровня шума для каждого метода регрессии вычисляются среднием показатели метрик с помощью calculateAverage() и сохраняются в формате, следующего пункту 3.4 и приложению 1.

Для визуализации изображений, которые в дальнейшем отправятся в пользовательский интерфейс, используется основная метрика – среднеквадратическая ошибка.

Для их создания полученная зависимость метрики от количества шума оборачивается вокруг Scatter объекта из PAINTINGS/Scatter.hpp и в график добавляется данный объект. Полученная конфигурация графика передается Python командам, сохраняющий требуемые изображения в формате, следующего пункту 3.4. Пример сохраненного изображения находится в Приложении 2.

3.3. Организация входных данных

Организация входных данных соответствует организации выходных JSON-файлов с описанием моделей, изложенной в п. 3.3. ««Разработка программного комплекса для исследования влияния аномальных наблюдений на точность прогнозирования в регрессионных моделях». Пояснительная записка» (RU.17701729.11.04-01 ПЗ 02-1).

3.4. Организация выходных данных

Организация выходных данных соответствует требованиям к организации выходных данных, изложенных в п. 4.1.2. ««Разработка программного комплекса для

исследования влияния аномальных наблюдений на точность прогнозирования в регрессионных моделях». Техническое задание» (RU.17701729.11.04-01 ТЗ 01-1).

3.5. Описание выбора технических и программных средств

Для разработки деталей обнаружения аномалий и визуализации результатов был выбран смешанный подход с использованием языков программирования C++ и Python. Подобное решение позволило совместить преимущества производительности языка C++ и большой функционал визуализации Python.

Анализ методов регрессии, добавление шума во входные данные, методы машинного обучения и расчет метрик для оценки качества методов регрессии и обнаружения аномальных наблюдений реализовано на C++. Выбор этого языка обусловлен потребностью эффективных вычислений и обработки больших объемов данных. Использующийся стандарт C++17 позволяет использовать механизм многопоточности, за счет чего процесс проведения и анализа экспериментов ускоряется в десятки раз.

Для построения системы сборки проекта использован CMake версии 3.14, который предоставляет кроссплатформенность и гибкую настройку процесса сборки. Это позволяет собирать проект на различных операционных системах без изменения исходного кода. Для работы с данными в формате JSON используется библиотека nlohmann/json как самая надежная и удобная библиотека для работы с этим форматом.

Визуализация результатов имплементирована при помощи языка программирования Python и библиотек matplotlib и numpy. Данная библиотека предоставляет удобный функционал для создания графиков зависимостей ошибки регрессии от уровня шума. Взаимодействие между модулями C++ и Python происходит по средствам файловой системы – программа на C++ генерирует конфигурации для изображения, а модуль Python их обрабатывает.

4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1. Ориентировочная экономическая эффективность

Приложение «MSnOutliers» представляет образовательную и исследовательскую ценность, особенно в сфере изучения методов статистического анализа и машинного обучения. Экономическая эффективность проекта определяется повышением эффективности образовательного процесса и исследовательской деятельности.

4.2. Предполагаемая потребность

Потребность в инструменте обусловлена интересом к алгоритмам машинного обучения, в том числе и методов обнаружения аномальных наблюдений, в различных областях, и статистическим методам регрессии. Особую ценность представляет возможность визуализировать и проанализировать эффективность методов регрессии и их чувствительность к искажению целевой переменной. Это делает приложение полезным в образовательном смысле.

Инструмент помогает пользователям тоньше понять принципы работы алгоритмов машинного обучения и статистических методов. С помощью полученных программой результатов пользователь сможет выбрать оптимальный метод для своей задачи.

4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

Существующие решения для анализа методов регрессии и влияния аномальных наблюдений на них такие как ELKI, auditor, outliers или scikit-learn обладают рядом ограничений и недостатков для поставленной задачи:

1) Данные инструменты обычно сфокусированы на решение только одной из задач обнаружения аномальных наблюдений и построения методов регрессии.
2) Не моделируются различные виды систематических и случайных аномальных наблюдений.
3) У большинства методов отсутствуют подробная визуализация и/или

подробные отчеты по каждому запущенному методу.

4) Не поддерживают многопоточные эксперименты с выбранными различными параметрами и гиперпараметрами.
5) Не интегрированы в единое решение.

Разработанный инструмент «MSnOutliers» предоставляет следующие возможности, демонстрирующие экономические преимущества разработки:

1) Комплексный подход к программе от генерации данных с шумом до метрик качества.
2) Возможность создания шума различного типа.
3) Эффективная многопоточная архитектура для запуска параллельных экспериментов.
4) Все метрики качества обнаружения аномальных наблюдений и методов регрессии, а также результаты визуализации методов укомплектованы в одной директории для удобного использования.

Таким образом, «MSnOutliers» дает более глубокое понимание влияния аномальных наблюдений на методы регрессии и принимать более обоснованные решения при выборе методов работы с реальными данными.

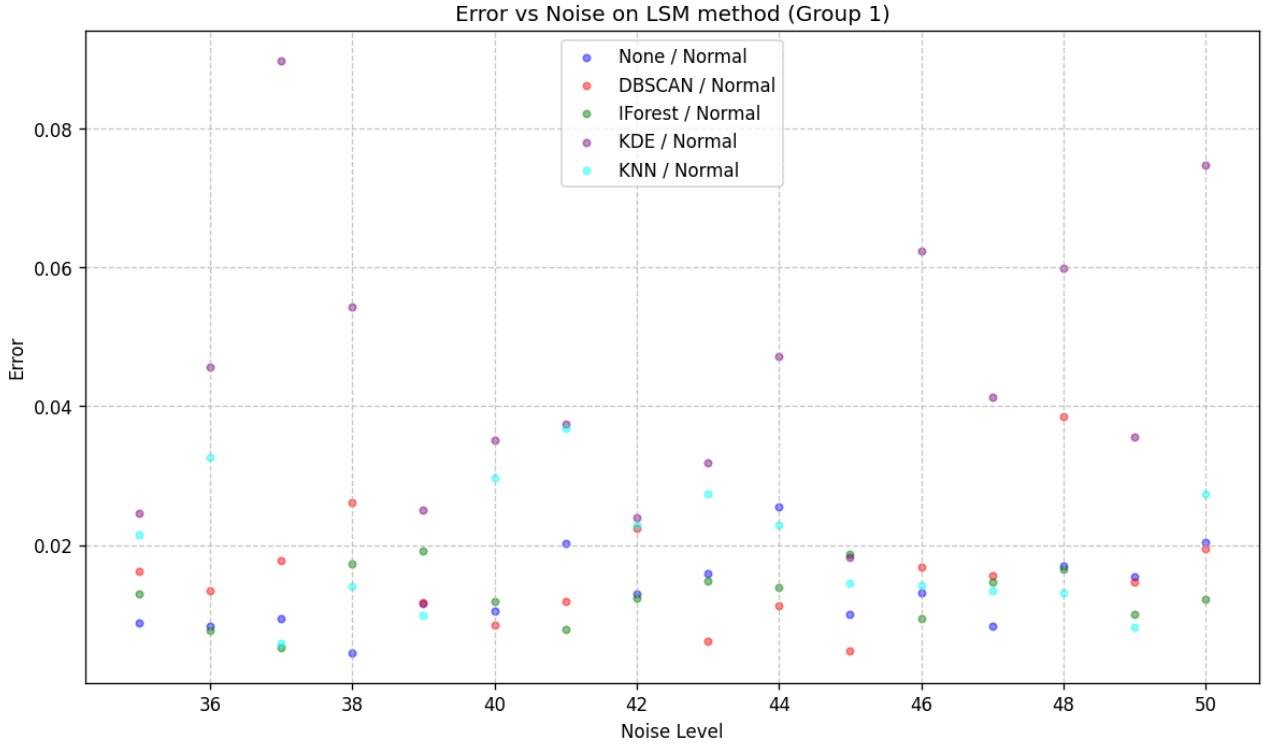
ПРИЛОЖЕНИЕ 1

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Библиотека “nlohmann/json” [Электронный ресурс] / Niels Lohmann – Режим доступа: https://github.com/nlohmann/json, свободный.
2. Библиотека “matplotlib” [Электронный ресурс] / Matplotlib Development Team – Режим доступа: https://matplotlib.org, свободный.
3. Библиотека “numpy” [Электронный ресурс] / NumPy Development Team – Режим доступа: https://numpy.org/, свободный.
4. Система сборки “CMake” [Электронный ресурс] / Kitware Inc. – Режим доступа: https://cmake.org/, свободный.
5. «Разработка программного комплекса для исследования влияния аномальных наблюдений на точность прогнозирования в регрессионных моделях». Техническое задание (ГОСТ 19.201-78).
6. «Разработка программного комплекса для исследования влияния аномальных наблюдений на точность прогнозирования в регрессионных моделях». Пояснительная записка (RU.17701729.11.04-01 ПЗ 02-1) (ГОСТ 19.404-79).
7. ГОСТ 19.201-78: Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

ПРИЛОЖЕНИЕ 2

Пример выходного файла: out_LSM_group0.png



ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					