МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УТВЕРЖДАЮ
Директор Ассоциации «Искусственный интеллект в промышленности»

_____ Т. М. Супатаев
___ _____ 2022 г.

УТВЕРЖДАЮ
Научный руководитель ИЦ СИИП Университета ИТМО

_____ А. В. Бухановский
___ _____ 2022 г.

ПРЕДОБРАБОТКА, АВТОНОМНОЕ ОЦЕНИВАНИЕ И ПРОГНОЗИРОВАНИЕ СОСТОЯНИЯ СЛОЖНЫХ ОБЪЕКТОВ И ПРОЦЕССОВ НА ОСНОВЕ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ СОБЫТИЙ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ И НЕДОСТОВЕРНОСТИ ДАННЫХ

РУКОВОДСТВО ПРОГРАММИСТА. ОПИСАНИЕ ПРИМЕНЕНИЯ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.00837-01 13 YY

Представители
Организации-разработчика

Руководитель разработки

_____ И.В. Котенко
____ _____ 2022 г.

Нормоконтролер

_____ А. В. Киреева
____ _____ 2022 г.

2022

Подп. и дата | Инв.№ | Взам.и | Подп. и дата | Инв.№

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УТВЕРЖДЕН
RU.СНАБ.00837-01 13 YY

ПРЕДОБРАБОТКА, АВТОНОМНОЕ ОЦЕНИВАНИЕ И ПРОГНОЗИРОВАНИЕ СОСТОЯНИЯ СЛОЖНЫХ ОБЪЕКТОВ И ПРОЦЕССОВ НА ОСНОВЕ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ СОБЫТИЙ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ И НЕДОСТОВЕРНОСТИ ДАННЫХ

РУКОВОДСТВО ПРОГРАММИСТА

RU.СНАБ.00837-01 13 YY

ЛИСТОВ 55

2022

Подп. и дата | Инв.№ | Взам.и | Подп. и дата | Инв.№

АННОТАЦИЯ

Документ содержит руководство программиста программы «Предобработка, автономное оценивание и прогнозирование состояния сложных объектов и процессов на основе интеллектуальной обработки событий в условиях неопределенности и недостоверности данных» RU.СНАБ.00837-01 13 YY. Программное обеспечение для предобработки, автономного оценивания и прогнозирования состояния сложных объектов и процессов входит в состав инструментального ПО, разрабатываемого в рамках плана Исследовательского центра в сфере искусственного интеллекта «Сильный ИИ в промышленности» (ИЦ ИИ) в рамках соглашения с АНО «Аналитический центр при Правительстве Российской Федерации» (ИГК 000000D730321P5Q0002), № 70–2021–00141, с целью осуществления предобработки, автономного оценивания и прогнозирования состояния сложных объектов и процессов на основе интеллектуальной обработки событий в условиях неопределенности и недостоверности данных.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММ

1.1 Назначение программного компонента

В состав компонента входят 4 алгоритма, реализующие функции сильного ИИ, а именно следующие:

1) Алгоритм предварительной обработки сырых и нечетких данных (сокр. ПОСНД) – в части улучшения качества предоставляемых данных и может быть задействован на подготовительной стадии оценивания и прогнозирования состояния;

2) Алгоритм извлечения знаний из собираемых наборов данных (сокр. ИЗСНД) – в части построения модели сложных объектов и процессов (сокр. СлОП), основанной на знаниях. Алгоритм позволяет извлечь множество знаний, описывающих связи между понятиями и действиями в предметной области оценивания и прогнозирования состояния СлОП, а также множество действий (выводов), вытекающих из этих знаний;

3) Алгоритм автономного оценивания текущего состояния сложных объектов и процессов (сокр. АОТССОП) – в части наличия возможности автоматического извлечения высокоуровневых представлений исходных данных и взаимосвязей между ними. Наличие большого числа гиперпараметров и настраиваемых весов, свойственных для глубоких НС, позволяет с достаточно высокой точностью выявлять закономерности между признаками обрабатываемого объекта и оцениваемой меткой его класса;

4) Алгоритм автономного прогнозирования состояний сложных объектов и процессов (сокр. АПССОП) – в части выполнения автономного прогнозирования состояний СлОП. Наличие большого числа гиперпараметров и настраиваемых весов, свойственных для глубоких нейронных сетей, позволяет с достаточно высоким качеством выявлять закономерности между признаками состояния системы и прогнозировать последующие состояния за заданный отрезок времени.

1.2 Область применения

Областью применения являются компьютерные системы и сети большой размерности (включая Интернет вещей, киберфизические системы), автономные робототехнические комплексы (беспилотные летательные аппараты, беспилотные автомобили и др.) и т.п.

К направлениям применения интеллектуальных средств оценки и прогнозирования сложных технических объектов и процессов можно отнести различные программы модернизации инфраструктуры предприятия, совершенствования промышленных установок, повышения срока их службы и снижения вероятности возникновения инцидентов на них.

Представленные интеллектуальные автономные алгоритмы позволят на основе имеющихся исторических и статистических данных бизнес-процессов выявлять их некорректные состояния и переходы, связанные в том числе со следующими событиями:

- неправильная настройка оборудования;
- износ отдельных деталей;
- возможные ошибки и неправильное использование различного технического оборудования персоналом;

- злонамеренными воздействиями со стороны внешних или внутренних нарушителей информационной безопасности.

Кроме того, использованием таких алгоритмов будет способствовать определению основных закономерностей и тенденций в дальнейшем ходе индустриальных процессов и прогнозирования дальнейших состояний технических объектов и особенностей, а также характеристик проистекающих в них процессов.

1.3 Функциональные условия применения

Экспериментально обоснованные функциональные ограничения на применение алгоритмов компонента сильного ИИ являются следующими:

1) для алгоритма ПОСНД устанавливаются:

- порог уникальности значений признаков, устанавливаемый пользователем;
- максимальный размер семантического словаря (встроенного или пользовательского) для анализа признаков;
- максимальная длина последовательностей значений признака;
- максимальное количество кластеров, на которые предполагается разбивать данные;
- максимальное количество узлов, реализующих параллельную обработку кластеров;
- минимальное/максимальное значение порога информативности признаков;

2) для алгоритма ИЗСНД устанавливаются:

- максимальный размер / объем обучающих данных, характеризующих СлО, заданных множеством описаний;
- максимальное количество агрегатов, которые объединяют отдельные значения каждого признака;
- максимальное количество ассоциативных правил;
- минимальное/максимальное пороговое значение метрики информативности;

3) для алгоритма АОТССОП вводятся ограничения:

- экспериментальный набор данных должен быть разбит на две части, одна из которых используется для обучения модели, а другая – для ее тестирования;
- набор данных должен представлять собой набор записей, каждая из которых описывает состояние исследуемого объекта в определенный момент времени;
- каждая запись из набора данных должна представлять собой последовательность числовых признаков фиксированной размерности, при этом величина этой размерности должна быть постоянной для всех записей;
- оценка состояния должна выполняться на основе метки класса, которая должна быть присвоена каждой записи из набора данных;

4) для алгоритма АПССОП вводятся ограничения:

- прогнозируются только числовые параметры состояний СлОП; работа с категориальными характеристиками не поддерживается, если они не были предварительно закодированы в числовые значения;

- для обучения модели используется фиксированный вектор характеристик для каждого состояния СлОП, длина и последовательность характеристик должна быть неизменной для состояния СЛоП в каждый момент времени;
- прогнозирование с использованием обученной модели осуществляется только для фиксированного вектора характеристик, заложенного в обученную модель;
- прогнозирование с использованием обученной модели осуществляется только на основе текущей или смоделированной последовательности состояний СлОП за промежуток времени, равный длине исторической последовательности, заложенной в обученную модель.

1.4 Технические условия применения

Техническими средствами являются электронно-вычислительные машины и устройства, которые используются при работе программы, должны иметь минимально необходимые характеристики, представленные в Табл. 1.1.

Таблица 1.1 – Минимально необходимые характеристики электронно-вычислительных машин и устройств для выполнения программы

Тип компьютера	**Кол-во CPU x кол-во ядер**	**Тактовая частота CPU, ГГц**	**Кол-во GPU x кол-во ядер**	**Тактовая частота GPU, ГГц**	**Оперативная память, Гб**	**Дисковая память, Гб**
Рабочая станция	1 x 8	3.8	1 x 3584	5.505	32	2000

2. ОПИСАНИЕ ПРИКЛАДНЫХ ЗАДАЧ

2.1 Классы решаемых задач

Компонент может быть использован для решения следующих задач (для соответствующих областей):

- для нефтегазовой отрасли: производит оценивание и прогнозирование состояния оборудования для разведки запасов углеводородов, их добычи, очистки и переработки, логистики и транспортировки нефтепродуктов с использованием магистральных трубопроводов и др.;
- для энергетического сектора: оценивание показателей энергогенерации, стоимостных, аварийности и пр.;
- для транспорта: оценивание логистических процессов с учетом больших объемов накопленных статистических данных и моделирования;
- для промышленного производства снижение сбоев конвейера и вовлеченности персонала в производство;
- для торговли: оценивание и прогнозирование спроса и оборота товаров и услуг;
- и др.

1.3 Примеры решения задач

Примеры применения конкретных задач с помощью компонента являются следующие.

Пример задачи № 1. Нарастающее прогнозирование тестовой выборки.

Постановка задачи:

В рамках данного примера решается задача прогнозирования на основе данных от системы паровых турбин и гидроаккумулирующих электростанций (набор данных HAI), а также для данных, описывающих функционирование сервера мостового крана при движении по L-образному пути с различными нагрузками (набор данных DSC). Для прогнозирования берется последний пакет обучающей выборки. Каждый прогнозируемый образец становится элементом нового пакета для последующего прогнозирования. Графическое представление отражено на рисунке 2.2.1. Спрогнозированные образцы обозначаются апострофом (‘), последовательный процесс прогнозирования образцов – нисходящей зеленой стрелкой. Добавление нового образца к пакету для прогнозирования обозначается серой стрелкой. Прогнозируемый на каждом этапе образец выделен зеленым. В результате полученная спрогнозированная выборка сравнивается с исходной тестовой выборкой.

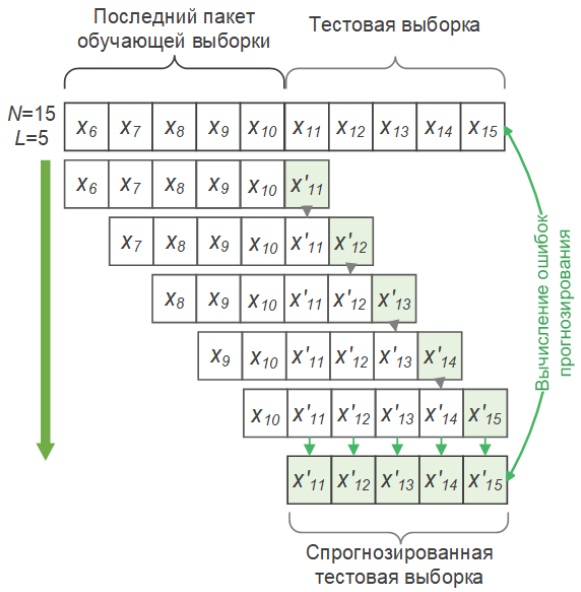

Рисунок 2.2.1 – Нарастающее прогнозирование тестовой выборки

Показатели эффективности прогнозирования основаны на вычислении разницы между реальными значениями образца (X) длиной N и прогнозируемыми значениями образца (X’). Значение ошибок прогнозирования должно стремится к 0. Используются следующие показатели – среднеквадратичная ошибка (mean squared error, MSE) и средняя абсолютная ошибка (mean absolute error. MAE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(X_i - X_i')^2, \quad (2.2.1)$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}(X_i - X_i'). \quad (2.2.2)$$

Показатель точности прогнозирования определяется как:

$$P = 1 - MAE. \quad (2.2.3)$$

Исходные данные:

Для оценки предложенной модели прогнозирования состояний использовались наборы данных:

- DSC (Driving Smart Crane with Various Loads) – набор данных содержит данные, собранные с сервера мостового крана при движении по L-образному пути с различными нагрузками (0 кг, 120 кг, 500 кг и 1000 кг). Прикладная задача: оценка работоспособности и надежности системы, выявление неисправностей. Размер: 3,66 МБ. Количество экземпляров: 31 304. Количество признаков: 12.
- HAI (HIL-based Augmented ICS) Security (2022 г.) – набор собран на испытательном стенде реалистичной промышленной системы управления, дополненной симулятором аппаратного обеспечения в контуре, который имитирует выработку электроэнергии с помощью паровых турбин и гидроаккумулирующих электростанций. Прикладная задача: оценка кибербезопасности, выявление атак и аномалий. Размер: 730 МБ. Количество экземпляров: 1 365 602. Количество признаков: 86.

Решение задачи:

Для каждого из экспериментальных наборов данных проводилась следующая предобработка:

1) разделении набора на тренировочную и тестовую выборки с соотношением 9:1;
2) нормализации выборок с использованием масштабирования значений признаков на отрезок [0, 1];
3) формировании генератора временных рядов.

Модель прогнозирования представляет собой глубокую нейронную сеть с тремя слоями LSTM. Функция активации скрытых слоев – ReLU. Функция активации выходного слоя – сигмоида (sigmoid). Функция оптимизации – adam. Длина исторической последовательности для всех наборов данных и экспериментов составляет 10 образцов.

Результаты решения и их интерпретация:

Для набора данных DSC прогнозирование осуществляет для каждого рабочего цикла отдельно. В таблице 2.2.1 представлены значения показателей эффективности для прогнозирования параметров состояний каждого цикла в результате эксперимента. На рисунке 2.2.2 отображен график прогнозирования значений признаков для цикла 1. Зеленым цветом обозначены реальные значения признаков, оранжевым – прогнозируемые.

Таблица 2.2.1 – Показатели эффективности прогнозирования для каждого цикла DSC (нарастающее прогнозирование)

№ цикла	**Число образцов**	**Размер обучающих данных**	**Размер тестовых данных**	***MSE***	***MAE***	***P***
1	2490	2241	249	0,035	0,109	0,891
2	3709	3338	371	0,104	0,211	0,789
3	2754	2479	275	0,171	0,255	0,745
4	3671	3304	367	0,035	0,102	0,898

5	4115	3704	411	0,131	0,208	0,792
6	7189	6470	719	0,129	0,204	0,796
7	3213	2892	321	0,148	0,236	0,764
8	4162	3746	416	0,160	0,274	0,726
Средняя точность прогнозирования по всем циклам						**0,800**

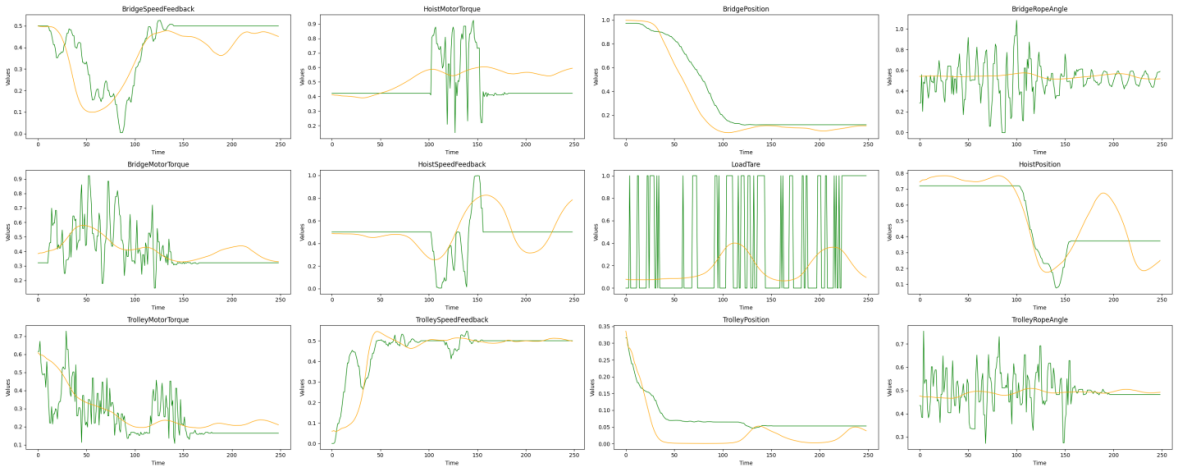

Рисунок 2.2.2 – Признаки тестовой выборки для цикла 1 DSC (нарастающее прогнозирование)

Средняя точность прогнозирования по всем циклам составляет 80%. Так как в экспериментах используется масштабирование признаков на отрезок [0, 1], то и значение ошибок прогнозирования лежит в этом же отрезке, где минимальное значение ошибки 0, а максимальное – 1. Эксперимент показывает более высокие значения ошибок, так как из-за характера эксперимента ошибка прогнозирования накапливается с каждым новым этапом из-за зависимости прогнозирования нового образца от прогнозирования предыдущих. При этом самую высокую ошибку прогнозирования имеют признаки BridgePosition, TrolleyPosition, LoadTare и HoistPosition.

Размер обучающей выборки для набора данных HAI составляет 903 962 образца, тестовой выборки – 100 440 образца. Результаты общей оценки эффективности на тестовой выборке составили: *MSE* = 0,228, *MAE* = 0,339; *P* = 0,66. По сравнению с предыдущим, набор данных HAI имеет большее количество признаков, что влияет на общую оценку точность прогнозирования. При этом большая длина прогнозируемых данных увеличивает накопление ошибки прогнозирования.

Пример задачи № 2. Поэтапное прогнозирование тестовой выборки.

Постановка задачи:

В рамках данного примера решается задача поэтапного прогнозирования на основе данных от системы паровых турбин и гидроаккумулирующих электростанций (набор данных HAI), а также для данных, описывающих функционирование сервера мостового крана при движении по L-образному пути с различными нагрузками (набор данных DSC). Для прогнозирования первого образца берется последний пакет обучающей выборки. Далее на каждом этапе к пакету добавляется образец тестовой выборки. Графическое представление отражено на рисунке 2.2.3. Обозначения аналогичны рисунку 2.2.1.

Ошибки прогнозирования определяются как для каждого признака отдельно, так и в целом для всей последовательности образцов.

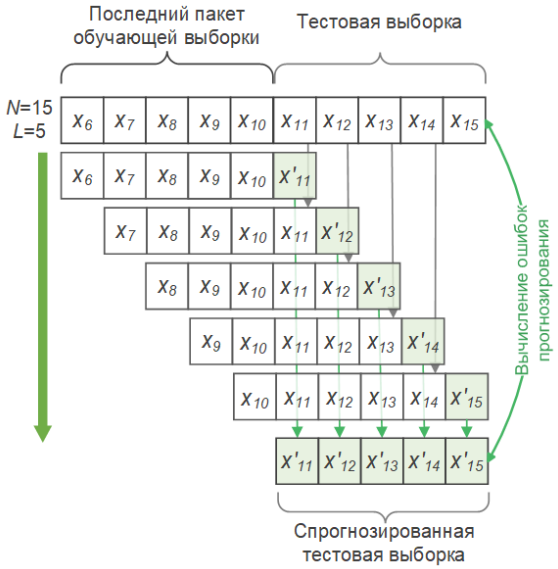

Рисунок 2.2.3 – Поэтапное прогнозирование тестовой выборки

Исходные данные:

Аналогичны данным в примере № 1.

Решение задачи:

Параметры модели прогнозирования аналогичны модели в примере №1. В эксперименте образцы прогнозируются независимо друг от друга.

Результаты решения и их интерпретация:

Для набора данных DSC прогнозирование осуществляет для каждого рабочего цикла отдельно. В таблице 2.2.2 представлены значения показателей эффективности для прогнозирования параметров состояний каждого цикла в результате эксперимента. В таблице 2.2.3 представлены значения показателей для прогнозирования каждого признака. На рисунке 2.2.4 отображен график прогнозирования значений признаков для цикла 1. Зеленым цветом обозначены реальные значения признаков, оранжевым – прогнозируемые.

Средняя точность прогнозирования по всем циклам составляет 93,7%. Также можно отметить, что признак LoadTare имеет самую низкую точность прогнозирования. Остальные 11 признаков прогнозируются с точностью выше 90%. Можно отметить, что большое количество значений ошибок прогнозирования близких к 0 и высокая точность прогнозирования, говорит об эффективности алгоритма прогнозирования на наборе данных DSC.

Таблица 2.2.2 – Показатели эффективности прогнозирования для каждого цикла DSC (поэтапное прогнозирование)

№ цикла	**Число образцов**	**Размер обучающих данных**	**Размер тестовых данных**	***MSE***	***MAE***	***P***
1	2490	2241	249	0,027	0,091	0,909

2	3709	3338	371	0,012	0,076	0,924
3	2754	2479	275	0,006	0,050	0,95
4	3671	3304	367	0,005	0,046	0,954
5	4115	3704	411	0,005	0,048	0,952
6	7189	6470	719	0,026	0,086	0,914
7	3213	2892	321	0,007	0,058	0,942
8	4162	3746	416	0,007	0,052	0,948
Средняя точность прогнозирования по всем циклам						**0,937**

Таблица 2.2.3 – Показатели эффективности прогнозирования каждого признака DSC (поэтапное прогнозирование)

№ цикла	**Показатель**	**Название признака**											
		BridgeSpeedFeedback	HoistMotorTorque	BridgePosition	BridgeRopeAngle	BridgeMotorTorque	HoistSpeedFeedback	LoadTare	HoistPosition	TrolleyMotorTorque	TrolleySpeedFeedback	TrolleyPosition	TrolleyRopeAngle
1	MSE	0,003	0,018	0,001	0,027	0,015	0,013	0,213	0,004	0,011	0,006	0,002	0,006
	MAE	0,041	0,098	0,022	0,113	0,096	0,078	0,371	0,046	0,088	0,043	0,038	0,052
	P	0,959	0,902	0,978	0,887	0,904	0,922	0,629	0,954	0,912	0,957	0,962	0,948
2	MSE	0,002	0,011	0,002	0,013	0,015	0,004	0,043	0,011	0,008	0,001	0	0,036
	MAE	0,025	0,092	0,036	0,102	0,073	0,058	0,178	0,092	0,062	0,018	0,01	0,164
	P	0,975	0,908	0,964	0,898	0,927	0,942	0,822	0,908	0,938	0,982	0,99	0,836
3	MSE	0,006	0,004	0,003	0,01	0,012	0,004	0,004	0,002	0,008	0,004	0,002	0,012
	MAE	0,068	0,035	0,051	0,065	0,074	0,037	0,042	0,034	0,054	0,032	0,037	0,074
	P	0,932	0,965	0,949	0,935	0,926	0,963	0,958	0,966	0,946	0,968	0,963	0,926
4	MSE	0,001	0,009	0,001	0,004	0,005	0,003	0,031	0,002	0,004	0,0004	0,0002	0,005
	MAE	0,022	0,086	0,02	0,035	0,057	0,039	0,152	0,025	0,042	0,016	0,014	0,039
	P	0,978	0,914	0,98	0,965	0,943	0,961	0,848	0,975	0,958	0,984	0,986	0,961
5	MSE	0,003	0,006	0,002	0,008	0,012	0,001	0,006	0,003	0,013	0,002	0,001	0,009
	MAE	0,044	0,045	0,037	0,062	0,082	0,02	0,046	0,038	0,079	0,031	0,02	0,068
	P	0,956	0,955	0,963	0,938	0,918	0,98	0,954	0,962	0,921	0,969	0,98	0,932
6	MSE	0,004	0,011	0,003	0,016	0,012	0,022	0,006	0,199	0,013	0,014	0,006	0,009
	MAE	0,041	0,067	0,045	0,104	0,06	0,069	0,047	0,307	0,089	0,08	0,055	0,069
	P	0,959	0,933	0,955	0,896	0,94	0,931	0,953	0,693	0,911	0,92	0,945	0,931
7	MSE	0,008	0,006	0,004	0,017	0,014	0,003	0,006	0,006	0,005	0,001	0,003	0,014
	MAE	0,054	0,06	0,042	0,094	0,072	0,043	0,058	0,07	0,042	0,026	0,033	0,096
	P	0,946	0,94	0,958	0,906	0,928	0,957	0,942	0,93	0,958	0,974	0,967	0,904
8	MSE	0,002	0,006	0,001	0,017	0,011	0,002	0,013	0,001	0,008	0,004	0,001	0,021
	MAE	0,033	0,049	0,024	0,092	0,06	0,027	0,084	0,025	0,048	0,054	0,023	0,104
	P	0,967	0,951	0,976	0,908	0,94	0,973	0,916	0,975	0,952	0,946	0,977	0,896
Средняя точность по всем циклам		**0,959**	**0,933**	**0,965**	**0,916**	**0,928**	**0,953**	**0,877**	**0,920**	**0,937**	**0,962**	**0,971**	**0,917**

Размер обучающей выборки для набора данных HAI составляет 903 962 образца, тестовой выборки – 100 440 образца. Результаты общей оценки эффективности на тестовой выборке: *MSE* = 0,147, *MAE* = 0,238, *P* = 0,762. В таблице 3.2.4 можно отметить, что значение точности прогнозирования для разных признаков лежит в диапазоне от 55% до 100%.

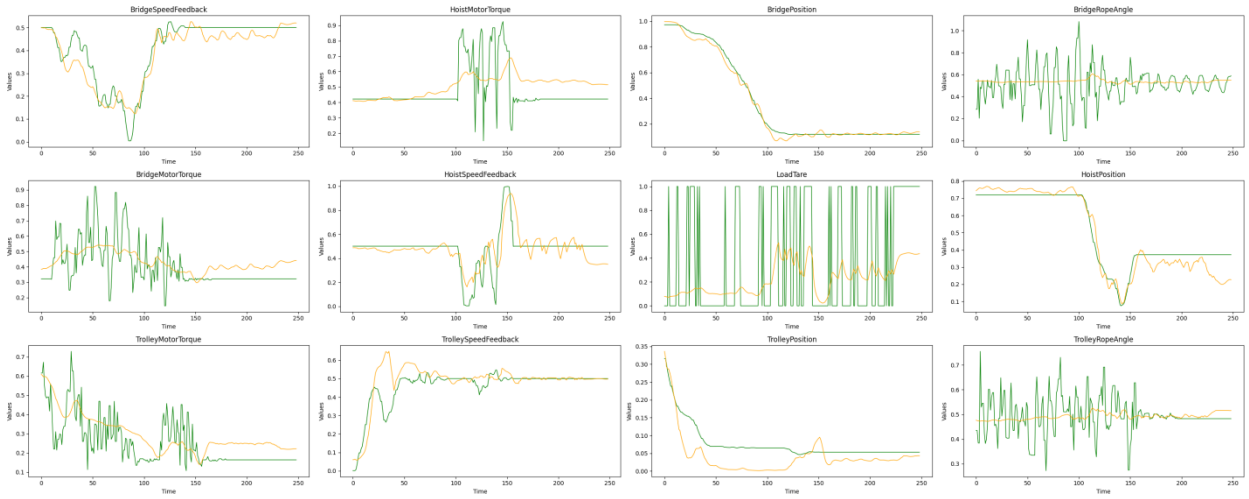

Рисунок 2.2.4 – Признаки тестовой выборки для цикла 1 DSC (поэтапное прогнозирование)

Таблица 2.2.4 – Показатели эффективности прогнозирования каждого признака HAI (поэтапное прогнозирование)

Признак	***MSE***	***MAE***	***P***	**Признак**	***MSE***	***MAE***	***P***
P1_B2004	0,353	0,448	0,552	P1_TIT03	0,105	0,254	0,746
P1_B2016	0,114	0,268	0,732	P2_24Vdc	0,133	0,292	0,708
P1_B3004	0,172	0,313	0,687	P2_ATSW_Lamp	0,015	0,024	0,976
P1_B3005	0,272	0,411	0,589	P2_AutoGO	0,006	0,014	0,986
P1_B4002	0,276	0,404	0,596	P2_AutoSD	0,137	0,269	0,731
P1_B4005	0,379	0,459	0,541	P2_Emerg	0,000001	0,0005	1,000
P1_B400B	0,359	0,437	0,563	P2_MASW	0,014	0,023	0,977
P1_B4022	0,172	0,315	0,685	P2_MASW_Lamp	0,011	0,019	0,981
P1_FCV01D	0,284	0,419	0,581	P2_ManualGO	0,012	0,021	0,979
P1_FCV01Z	0,282	0,417	0,583	P2_ManualSD	0,221	0,439	0,561
P1_FCV02D	0,191	0,227	0,773	P2_OnOff	0,000001	0,00047	1,000
P1_FCV02Z	0,355	0,413	0,587	P2_RTR	0,000001	0,00048	1,000
P1_FCV03D	0,151	0,293	0,707	P2_SCO	0,423	0,482	0,518
P1_FCV03Z	0,150	0,291	0,709	P2_SCST	0,099	0,250	0,750
P1_FT01	0,150	0,276	0,724	P2_SIT01	0,231	0,320	0,680
P1_FT01Z	0,146	0,285	0,715	P2_TripEx	0,000001	0,00048	1,000
P1_FT02	0,364	0,434	0,566	P2_VIBTR01	0,154	0,319	0,681
P1_FT02Z	0,362	0,440	0,560	P2_VIBTR02	0,143	0,306	0,694
P1_FT03	0,231	0,381	0,619	P2_VIBTR03	0,153	0,317	0,683
P1_FT03Z	0,229	0,373	0,627	P2_VIBTR04	0,128	0,289	0,711
P1_LCV01D	0,128	0,278	0,722	P2_VT01	0,152	0,317	0,683
P1_LCV01Z	0,126	0,277	0,723	P2_VTR01	0,000001	0,000482	1,000
P1_LIT01	0,147	0,289	0,711	P2_VTR02	0,000001	0,000483	1,000
P1_PCV01D	0,132	0,313	0,687	P2_VTR03	0,000001	0,000472	1,000
P1_PCV01Z	0,133	0,319	0,681	P2_VTR04	0,000001	0,000482	1,000
P1_PCV02D	0,000001	0,0005	1,000	P3_FIT01	0,035	0,096	0,904
P1_PCV02Z	0,112	0,147	0,853	P3_LCP01D	0,115	0,219	0,781
P1_PIT01	0,126	0,269	0,731	P3_LCV01D	0,212	0,361	0,639
P1_PIT01_HH	0,000	0,000	1,000	P3_LH01	0,456	0,498	0,502
P1_PIT02	0,109	0,181	0,819	P3_LIT01	0,182	0,360	0,640
P1_PP01AD	0,000001	0,00048	1,000	P3_LL01	0,457	0,498	0,502
P1_PP01AR	0,000001	0,00048	1,000	P3_PIT01	0,058	0,107	0,893
P1_PP01BD	0,000001	0,00048	1,000	P4_HT_FD	0,131	0,272	0,728
P1_PP01BR	0,000001	0,00048	1,000	P4_HT_PO	0,218	0,368	0,632
P1_PP02D	0,000001	0,00047	1,000	P4_HT_PS	0,141	0,217	0,783
P1_PP02R	0,000001	0,00048	1,000	P4_LD	0,138	0,293	0,707
P1_PP04	0,442	0,493	0,507	P4_ST_FD	0,161	0,303	0,697
P1_PP04SP	0,294	0,438	0,562	P4_ST_GOV	0,197	0,344	0,656

P1_SOL01D	0,000001	0,000486	1,000	P4_ST_LD	0,198	0,349	0,651
P1_SOL03D	0,000001	0,000484	1,000	P4_ST_PO	0,188	0,335	0,665
P1_STSP	0,000001	0,000486	1,000	P4_ST_PS	0,308	0,464	0,536
P1_TIT01	0,120	0,265	0,735	P4_ST_PT01	0,124	0,266	0,734
P1_TIT02	0,203	0,355	0,645	P4_ST_TT01	0,120	0,265	0,735

Пример задачи № 3. Оценивание тестовой выборки.

В рамках данного примера решается задача оценивания текущего состояния на основе данных от системы паровых турбин и гидроаккумулирующих электростанций (набор данных HAI), а также для данных, описывающих функционирование сервера мостового крана при движении по L-образному пути с различными нагрузками (набор данных DSC).

Постановка задачи:

Для оценки данного алгоритма использовались два набора данных в системе управления паровыми турбинами и движение мостового крана при различных нагрузках.

Исходные данные.

Для оценки предложенной модели прогнозирования состояний использовались наборы данных:

- HAI (HIL-based Augmented ICS) Security (2022 г.) – набор собран на испытательном стенде реалистичной промышленной системы управления, дополненной симулятором аппаратного обеспечения в контуре, который имитирует выработку электроэнергии с помощью паровых турбин и гидроаккумулирующих электростанций.
- DSC (Driving Smart Crane with Various Loads) – набор данных содержит данные, собранные с сервера мостового крана при движении по L-образному пути с различными нагрузками (0 кг, 120 кг, 500 кг и 1000 кг).

Решение задачи.

Для каждого из экспериментальных наборов данных проводилась следующая предобработка:

- разделении набора на тренировочную и тестовую выборки с соотношением 8:2;
- нормализации выборок с использованием min-max-преобразования.

Результаты решения и их интерпретация.

В таблице 2.2.5 представлены показатели точности для алгоритма АОТССОП. Отметим, что значение этого показателя, вычисленное для обучающей выборки, превосходит значение этого же показателя для тестовой выборки.

Таблица 2.2.5 – Показатели точности для алгоритма АОТССОП

Набор данных (файл)	Количество классов	Выборка	Значение точности (accuracy)
HAI (hai-20.07/test1.csv.gz)	2	Обучающая (80%)	99.36%
HAI (hai-20.07/test1.csv.gz)	2	Тестовая (20%)	99.33%
HAI (hai-21.03/test1.csv.gz)	2	Обучающая (80%)	99.57%
HAI (hai-21.03/test1.csv.gz)	2	Тестовая (20%)	99.49%
DSC (combined.csv)	8	Обучающая (80%)	81.96%
DSC (combined.csv)	8	Тестовая (20%)	81.28%

На рисунках 2.2.5 - 2.2.7 представлены графики, отражающие зависимость точности определения класса объекта от номера эпохи обучения для разных наборов данных.

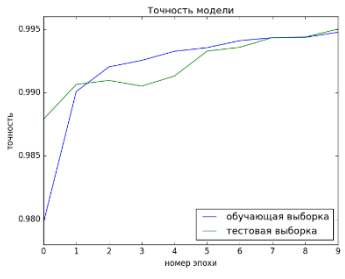

Рис. 2.2.5 – Точность обучения НС на наборе данных HAI (hai-20.07/test1.csv.gz)

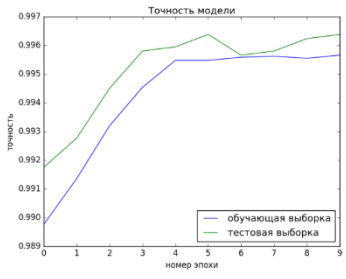

Рис. 2.2.6 – Точность обучения НС на наборе данных HAI (hai-21.03/test1.csv.gz)

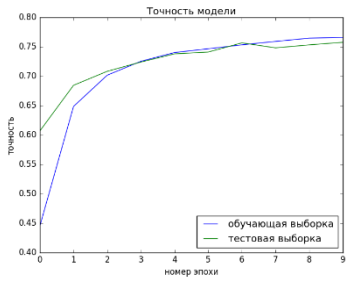

Рис. 2.2.7 – Точность обучения НС на наборе данных DSC (combined.csv)

Пример задачи № 4. Прогнозирование с предобработкой.

Постановка задачи:

В рамках данного примера решается задача прогнозирования на основе предобработанных алгоритмом ИЗСНД данных от системы паровых турбин и гидроаккумулирующих электростанций (набор данных HAI), а также для данных, описывающих функционирование сервера мостового крана при движении по L-образному пути с различными нагрузками (набор данных DSC). Для прогнозирования берется последний пакет обучающей выборки. Далее на каждом этапе к пакету добавляется образец тестовой выборки.

Исходные данные:

Аналогичны данным в примерах № 1 и № 2.

Решение задачи:

Перед обучением модели и перед прогнозированием данные проходят дополнительную предобработку с использованием алгоритма ИЗСНД. Параметры модели прогнозирования аналогичны модели в примерах №1 и №2. В эксперименте образцы прогнозируются как независимо друг от друга (априорная оценка), так и путем нарастающего прогнозирования (апостериорная оценка). Производится оценка потерь обучения модели MSE (формула 2.2.1), оценка качества прогнозирования путем вычисления MSE, MAE, P (формулы 2.2.1-2.2.3), а также оценка ресурсопотребления в виде вычисления временных затрат (T), средней доли загрузки процессора (CPU) и средняя загрузка памяти (RAM). Оценка ресурсопотребления проводилась при следующих параметрах системы: процессор – Intel(R) Core(TM) i5-8250U, CPU 1.60GHz, 4 ядра; оперативная память – 8,00 ГБ.

Результаты решения и их интерпретация:

Для набора данных DSC прогнозирование осуществляет для каждого рабочего цикла отдельно. В таблице 2.2.6 представлено сравнение результатов оценки качества прогнозирования параметров состояний каждого цикла DSC в результате эксперимента

без использования алгоритма ИЗСНД и с использованием данного алгоритма. Обозначения для алгоритма ИЗСНД: красный шрифт – оценка параметра ухудшилась, зеленый шрифт – оценка параметра улучшилась.

Таблица 2.2.6 – Показатели оценки комбинированного применения алгоритмов ИЗСНД и АППСОП для каждого цикла DSC

Эксперимент	№ цикла	Размер данных	Число признаков	ИЗСНД	Оценка качества			Оценка ресурсопотребления		
					MSE	MAE	1-MAE	Время (сек)	CPU ср., %	RAM ср., Мб
Обучение модели прогнозирования	1	2241	12	–	0,0203	–	–	71,054	28,448	397,895
			10	+	0,0227	–	–	59,336	27,601	430,416
	2	3338	12	–	0,0145	–	–	90,645	28,053	464,089
			10	+	0,0113	–	–	124,66	25,976	468,629
	3	3304	12	–	0,0087	–	–	72,69178	26,836	476,184
			10	+	0,005	–	–	66,251	26,864	472,858
	4	3671	12	–	0,0104	–	–	87,022	27,387	433,419
			10	+	0,0753	–	–	87,407	26,48	438,342
	5	3704	12	–	0,0075	–	–	96,970	27,148	442,113
			10	+	0,0028	–	–	96,484	26,939	468,105
	6	6470	12	–	0,0059	–	–	169,7033	27,779	470,072
			10	+	0,0054	–	–	167,98	27,377	478,016
	7	2892	12	–	0,0079	–	–	76,362	27,48	473,832
			10	+	0,0082	–	–	75,537	27,608	506,011
	8	3746	12	–	0,009	–	–	98,522	28,055	500,94
			10	+	0,0007	–	–	100,87	27,486	505,997
Априорная оценка модели прогнозирования на тестовой выборке	1	249	12	–	0,0266	0,0933	0,9067	11,273	13,58	368,36
			10	+	0,038	0,038	0,962	10,962	13,278	434,453
	2	371	12	–	0,0146	0,0896	0,9104	33,345	11,892	405,854
			10	+	0,0081	0,0081	0,9919	25,426	11,693	415,93
	3	275	12	–	0,0059	0,0490	0,951	11,604	13,283	421,201
			10	+	0,0046	0,0046	0,9954	11,717	13,263	416,677
	4	367	12	–	0,0052	0,0486	0,9513	16,006	13,318	435,27
			10	+	0,1771	0,1771	0,8229	15,366	13,284	440,149
	5	411	12	–	0,0059	0,0491	0,9509	17,361	13,067	438,092
			10	+	0,0022	0,0022	0,9978	17,896	13,272	427,209
	6	719	12	–	0,0295	0,0918	0,9082	30,759	13,198	441,597
			10	+	0,0240	0,0240	0,976	30,146	13,024	461,722
	7	321	12	–	0,0071	0,0580	0,9419	13,485	13,193	472,854
			10	+	0,0057	0,0057	0,9943	13,679	13,08	471,726
	8	416	12	–	0,0084	0,0592	0,9407	19,348	13,453	484,581
			10	+	0,0006	0,0006	0,9993	17,598	13,411	485,061
Апостериорная оценка модели прогнозирования на тестовой выборке	1	249	12	–	0,0412	0,13	0,87	10,280	12,995	381,935
			10	+	0,038	0,038	0,9620	10,354	13,188	404,376
	2	371	12	–	0,1080	0,2043	0,7957	39,794	12,065	419,013
			10	+	0,0081	0,0081	0,9919	15,96	13,076	426,117
	3	275	12	–	0,1213	0,2224	0,7775	11,296	13,029	430,552
			10	+	0,0046	0,0046	0,9954	11,171	13,46	399,583
	4	367	12	–	0,124	0,1980	0,8019	15,495	13,661	401,438
			10	+	0,1771	0,1771	0,8229	15,185	13,544	421,591
	5	411	12	–	0,1414	0,2339	0,7661	17,004	13,521	427,549
			10	+	0,0022	0,0022	0,9978	17,045	13,585	429,621
	6	719	12	–	0,1239	0,1900	0,81	29,154	13,345	444,546
			10	+	0,0240	0,0240	0,976	29,120	13,269	450,244
	7	321	12	–	0,0637	0,1413	0,8587	13,233	13,206	480,621
			10	+	0,0057	0,0057	0,9943	13,173	13,244	473,754
	8	416	12	–	0,0563	0,1366	0,8634	17,093	13,3	483,439
			10	+	0,0006	0,0006	0,9993	17,406	13,396	487,892

В таблице 2.2.7 представлено сравнение результатов оценки качества прогнозирования параметров состояний на наборе данных HAI в результате эксперимента без использования алгоритма ИЗСНД и с использованием данного алгоритма.

Таблица 2.2.7 – Показатели оценки комбинированного применения алгоритмов ИЗСНД и АППСОП для HAI

Эксперимент	Размер данных	Число признаков	ИЗСНД	Оценка качества			Оценка ресурсопотребления		
				MSE	MAE	1-MAE	Время (сек)	CPU ср., %	RAM ср., Мб
Обучение модели прогнозирования	74520	86	–	0,0035	–	–	2602,34	33,808	166,525
		82	+	0,0021	–	–	2727,34	34,912	213,998
Априорная оценка модели прогнозирования на тестовой выборке	1000	86	–	0,0036	0,0352	0,9648	61,73	13,74	264,44
		82	+	0,0122	0,0122	0,9878	60,81	13,824	446,71
Апостериорная оценка модели прогнозирования на тестовой выборке	1000	86	–	0,0626	0,129	0,871	66,91	13,343	314,685
		82	+	0,0122	0,0122	0,9878	55,93	14,046	470,343

Точность прогнозирования в большинстве случаев возрастает, в особенности для апостериорной оценки прогнозирования. Также использование алгоритма ИЗСНД часто приводит к уменьшению времени на обучение/прогнозирование, но при этом затрачиваемые ресурсы процессора и памяти могут возрастать.

3. ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1 Режимы работы ключевых алгоритмов

Эксперименты по проверке работоспособности и устойчивости результатов работы алгоритмов компонента сильного ИИ приведены далее.

3.1.1 Алгоритм ПОСНД

Для оценки работоспособности и устойчивости алгоритма ПОСНД использовался набор данных – Titanic. Это стандартный набор данных для применения алгоритмов машинного обучения и искусственного интеллекта, позволяющий предсказать выживание пассажира Титаника. Формат данных – csv. Размер файла – 93.08 Кб. Всего 890 строк данных.

В ходе эксперимента выполнялись следующие шаги:

1) данные передавались в формате, понятном алгоритму ПОСНД (признаки и метки отдельно, разбиение на значения, названия и типы данных, передача известной информации о типах данных);

2) подключался словарь наименований признаков, для которых известен тип данных – категориальный или численный;

3) выставлялся вес выводов о типах данных признаков в зависимости от задействованного метода;

4) выставлялись настройки кластеризации: метод, количество кластеров и итераций;

5) выставлялись пороговые значения при анализе информативности;

6) выставлялись пороговые значения при анализе мультиколлинеарности;

7) выставлялись настройки журналирования данных;

8) запускался алгоритм коррекции типов данных;

9) запускался алгоритм устранения неполноты данных;

10) запускался алгоритм анализа информативности признаков;

11) запускался алгоритм устранения мультиколлинеарности данных.

При анализе типов данных было установлено:

- тип данных признака «PassengerId» был указан некорректно, изменен с категориального на численный;
- тип данных признака «SibSp» не был указан, изменен на категориальный;
- тип данных признака «Parch» не был указан, изменен на категориальный;
- тип данных признака «LABEL_type» не был указана, изменен на численный.

При устранении неполноты данных было заполнено 177 пустых значений. На это ушло 2 итерации процесса кластеризации. Неинформативных признаков не было обнаружено.

В результате анализа мультиколлинеарности было обнаружено:

- признак «SibSp» коррелирует с признаком «PassengerId»;
- признак «Age» коррелирует с признаком «Pclass».

Рекомендовано удалить признаки «SibSp» и «Age».

3.1.2 Алгоритм ИЗСНД

Для оценки работоспособности и устойчивости алгоритма ИЗСНД использовались наборы данных:

- DSC (Driving Smart Crane with Various Loads) – набор данных содержит данные, собранные с сервера мостового крана при движении по L-образному пути с различными нагрузками (0 кг, 120 кг, 500 кг и 1000 кг). Прикладная задача: оценка работоспособности и надежности системы, выявление неисправностей. Размер: 3,66 МБ. Количество экземпляров: 31 304. Количество признаков: 12.
- HAI (HIL-based Augmented ICS) Security (2022 г.) – набор собран на испытательном стенде реалистичной промышленной системы управления, дополненной симулятором аппаратного обеспечения в контуре, который имитирует выработку электроэнергии с помощью паровых турбин и гидроаккумулирующих электростанций. Прикладная задача: оценка кибербезопасности, выявление атак и аномалий. Размер: 730 МБ. Количество экземпляров: 1 365 602. Количество признаков: 86.

В ходе эксперимента для каждого из наборов данных выполнялись следующие шаги:

1) определение размера набора данных в оперативной памяти до применения алгоритма ИЗСНД;

2) предобработка набора с помощью алгоритма ИЗСНД;

3) определение размера набора данных в оперативной памяти после применения алгоритма ИЗСНД;

4) разбиение предобработанных и не предобработанных версий набора на обучающие и тренировочные подвыборки в соотношении 70 к 30;

5) обучение классификатора на основе случайного леса на тренировочной подвыборке не предобработанной версии набора;

6) расчет времени обучения классификатора на основе случайного леса на тренировочной подвыборке не предобработанной версии набора;

7) обучение классификатора на основе случайного леса на тренировочной подвыборке предобработанной версии набора;

8) расчет времени обучения классификатора на основе случайного леса на тренировочной подвыборке предобработанной версии набора.

Результаты экспериментов представлены в таблицах 3.1.2.1 и 3.1.2.2.

Таблица 3.1.2.1 – Оценка работоспособности и устойчивости алгоритма ИЗСНД с помощью случайного леса на наборе данных DSC

	Размер набора в оперативной памяти, MB	Время обучения классификатора, мс
RandomForestClassifier	3.3	5.96
RandomForestClassifier + IZDAP preprocessing	1.6	5.01

Таблица 3.1.2.2 – Оценка работоспособности и устойчивости алгоритма ИЗСНД с помощью случайного леса на наборе данных HAI

	Размер набора в оперативной памяти, MB	Время обучения классификатора, мс
RandomForestClassifier	55.6+	13.6
RandomForestClassifier + IZDAP preprocessing	7.1+	1.2

Согласно результатам, представленным в Таблицах 3.1.2.1 и 3.2.3.2, практические эксперименты на типовом ПК для рассмотренных примеров данных показывают оптимизацию объема вычислительных ресурсов, необходимых для обучения классификаторов при применении ИЗСНД. Таким образом, данный алгоритм может иметь практическое применение и для реальных задач.

3.1.3 Алгоритм АОТССОП

Эксперимент № 1.

Исходные данные:

Для оценки предложенной модели прогнозирования состояний использовались наборы данных:

- HAI (HIL-based Augmented ICS) Security (2022 г.) – набор собран на испытательном стенде реалистичной промышленной системы управления, дополненной

симулятором аппаратного обеспечения в контуре, который имитирует выработку электроэнергии с помощью паровых турбин и гидроаккумулирующих электростанций.

- DSC (Driving Smart Crane with Various Loads) – набор данных содержит данные, собранные с сервера мостового крана при движении по L-образному пути с различными нагрузками (0 кг, 120 кг, 500 кг и 1000 кг).

Оценка устойчивости данного алгоритма выполняется следующим образом. К признакам набора данных случайным образом добавляется шум, величина которого не превышает 30% от величины размаха в значениях признаков. Сгенерированной записи присваивается метка класса, которая соответствует исходной записи (до добавления искажений).

Результаты эксперимента:

Результаты оценки устойчивости алгоритма оценки состояния на наборе HAI составляют 96.56%, а на наборе данных DSC 96.84%. При этом максимальное потребление памяти составляет 208 МБ (оценка этого показателя ресурсопотребления выполнялась при помощи пакета tracemalloc).

Результаты экспериментальных исследований алгоритма сильного ИИ АОТССОП для наборы данных HAI и DSC, характеризующие его ресурсопотребление, представлены в Таблице 3.1.4.1. Эксперименты проводились на CPU Intel(R) Core(TM) i5-3210M (2.50GHz) с 4 ядрами и 4 Гб памяти.

Согласно результатам эксперимента (см. Таблицу 3.1.3) практические эксперименты на типовом ПК для рассмотренных примеров данных показывают удовлетворительные значения ресурсопотребления. Таким образом, данный алгоритм может иметь практическое применение и для реальных задач.

Таблица 3.1.3 – Ресурсопотребление алгоритма АОТССОП

Набор данных	**Эксперимент**	**Размер данных**	**Процесс**	**Время (сек)**	**CPU min, %**	**CPU mean, %**	**CPU max, %**	**RAM min, Mb**	**RAM mean, Mb**	**RAM max, Mb**
HAI	Обучение	233280	Загрузка набора данных	21.580719	0.0	20.87	242.68	389.26	1158.82	2170.45
			Подготовка модели	0.13265	0.0	6.28	12.55	481.18	491.21	501.24
			Обучение	125.365428	0.0	42.97	722.73	639.61	854.14	875.72
			Сериализация	0.0	0.0	0.0	0.0	771.37	771.37	771.37
			Всего	**147.896896**	**0.0**	**41.79**	**722.73**	**389.26**	**879.63**	**2170.45**
	Априорная оценка на обучающей выборке		Загрузка набора данных	20.362066	0.0	26.6	217.03	388.74	1177.99	2133.68
			Подготовка модели	0.101993	47.12	98.14	149.15	528.68	532.88	537.09
			Десериализация	0.0	13.32	13.32	13.32	727.07	727.07	727.07
			Тестирование	12.850455	0.0	31.45	432.3	727.08	905.57	987.61
			Всего	**34.905482**	**0.0**	**36.12**	**432.3**	**388.74**	**1012.83**	**2133.68**
	Апостериорная оценка на тестовой выборке	58320	Загрузка набора данных	20.807359	0.0	26.91	265.98	382.59	1170.25	2216.57
			Подготовка модели	0.219451	10.88	12.95	15.97	491.64	501.63	509.36
			Десериализация	0.0	14.28	14.28	14.28	704.24	704.24	704.24
			Тестирование	3.956152	0.0	30.07	181.32	704.25	757.79	769.44
			Всего	**26.417068**	**0.0**	**34.47**	**265.98**	**382.59**	**1030.34**	**2216.57**
DSC	Обучение	25042	Загрузка набора данных	0.475079	0.0	17.92	48.77	388.63	398.35	412.95
			Подготовка модели	0.0	0.0	0.0	0.0	396.78	396.78	396.78

			Обучение	14.754938	0.0	57.48	783.12	417.95	457.76	463.24
			Сериализация	0.0	35.55	35.55	35.55	460.42	460.42	460.42
			Всего	**15.741302**	**0.0**	**33.84**	**783.12**	**283.5**	**455.58**	**463.24**
	Априорная оценка на обучающей выборке		Загрузка набора данных	0.472925	20.3	59.55	175.15	393.5	406.18	427.61
			Подготовка модели	0.0	10.07	10.07	10.07	402.12	402.12	402.12
			Десериализация	0.0	44.35	44.35	44.35	436.11	436.11	436.11
			Тестирование	1.674084	0.0	51.15	347.35	436.11	447.74	459.6
			Всего	**2.780305**	**0.0**	**15.56**	**347.35**	**393.5**	**436.82**	**459.6**
	Апостериорная оценка на тестовой выборке	6261	Загрузка набора данных	0.458326	0.0	7.36	16.7	392.38	403.69	422.37
			Подготовка модели	0.0	0.0	0.0	0.0	401.2	401.2	401.2
			Десериализация	0.0	16.85	16.85	16.85	435.0	435.0	435.0
			Тестирование	0.528282	0.0	22.99	28.43	435.01	441.85	446.99
			Всего	**1.532124**	**0.0**	**23.23**	**38.33**	**392.38**	**426.82**	**446.99**

3.1.4 Алгоритм АПССОП

Эксперимент № 1.

Исходные данные:

Для оценки предложенной модели прогнозирования состояний использовались наборы данных:

- DSC (Driving Smart Crane with Various Loads) – набор данных содержит данные, собранные с сервера мостового крана при движении по L-образному пути с различными нагрузками (0 кг, 120 кг, 500 кг и 1000 кг). Размер: 3,66 МБ. Количество экземпляров: 31 304. Количество признаков: 12.
- HAI (HIL-based Augmented ICS) Security (2022 г.) – набор собран на испытательном стенде реалистичной промышленной системы управления, дополненной симулятором аппаратного обеспечения в контуре, который имитирует выработку электроэнергии с помощью паровых турбин и гидроаккумулирующих электростанций. Размер: 730 МБ. Количество экземпляров: 1 365 602. Количество признаков: 86.

Оценка устойчивости алгоритма прогнозирования. К исходным данным добавляется случайный шум, не превосходящий 30% от величины размаха в значениях признаков. Полученную последовательность обозначим Z. Для прогнозирования первого образца берется последний пакет обучающей выборки. Далее на каждом этапе к пакету добавляется образец тестовой выборки. Образцы прогнозируются независимо друг от друга.

Пусть результат прогнозирования входной последовательности – X', а результат прогнозирования зашумленной последовательности – Z'. Показатель устойчивости определяется следующим образом:

$$S = 1 - \frac{1}{N}\sum_{i=1}^{N}(X_i' - Z_i'). \qquad (4.1.4.1)$$

Результаты эксперимента:

Результаты оценки устойчивости алгоритма прогнозирования на наборе DSC представлены в таблице 3.1.4.1.

Оценка устойчивости алгоритма прогнозирования на HAI продемонстрировала показатель устойчивости в 89,6%. На рисунке 3.1.4.1 показана разница между средней

абсолютной ошибкой прогнозирования (MAE) для признаков набора данных HAI для входных данных (initial, синий) и зашумленных данных (noised, оранжевый).

Интерпретация результатов:

На наборе данных DSC получены значения показателя устойчивости выше 95,9%. На наборе данных HAI устойчивость составила 89,6%. Соответственно, алгоритм прогнозирования демонстрирует высокую устойчивость к зашумлению данных

Таблица 3.1.4.1 – Оценка устойчивости алгоритма прогнозирования на DSC

Признак	**№ цикла**							
	1	2	3	4	5	6	7	8
BridgeSpeedFeedback	0,990	0,989	0,982	0,984	0,977	0,978	0,976	0,995
HoistMotorTorque	0,996	0,991	0,996	0,984	0,966	0,945	0,974	0,993
BridgePosition	0,986	0,984	0,973	0,995	0,974	0,970	0,968	0,997
BridgeRopeAngle	0,998	0,990	0,996	0,984	0,991	0,961	0,996	0,997
BridgeMotorTorque	0,992	0,985	0,988	0,986	0,984	0,981	0,984	0,998
HoistSpeedFeedback	0,989	0,983	0,994	0,992	0,987	0,983	0,986	0,998
LoadTare	0,986	0,990	0,988	0,872	0,977	0,913	0,970	0,992
HoistPosition	0,983	0,978	0,987	0,950	0,980	0,904	0,956	0,995
TrolleyMotorTorque	0,994	0,993	0,987	0,993	0,973	0,973	0,980	0,997
TrolleySpeedFeedback	0,994	0,994	0,987	0,977	0,981	0,973	0,982	0,996
TrolleyPosition	0,997	0,997	0,994	0,994	0,987	0,973	0,988	0,998
TrolleyRopeAngle	0,999	0,997	0,996	0,981	0,995	0,956	0,991	0,997
Общая оценка	**0,992**	**0,989**	**0,989**	**0,974**	**0,981**	**0,959**	**0,979**	**0,996**

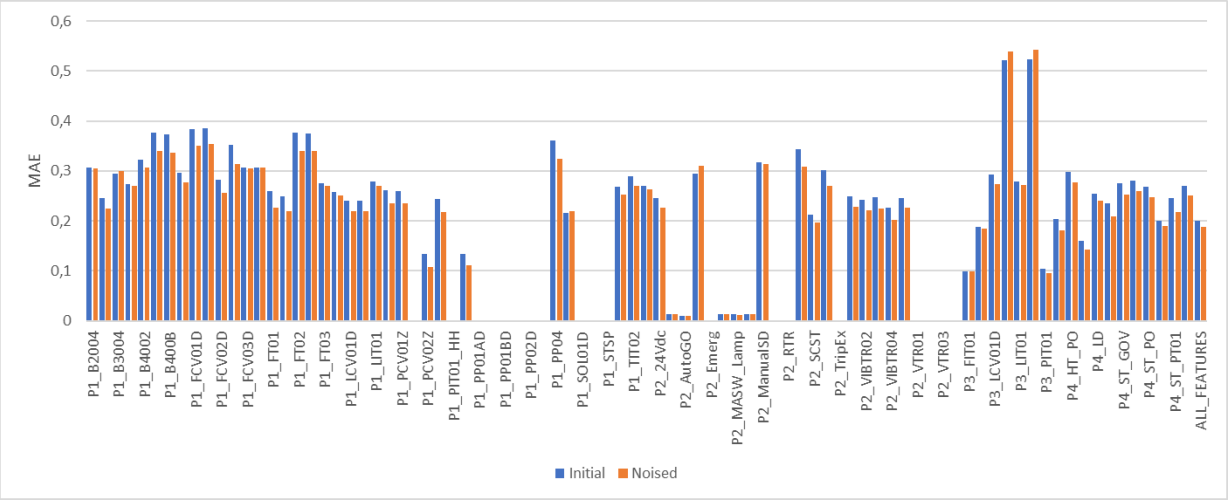

Рисунок 3.1.4.1 – Ошибка прогнозирования для входных и зашумленных данных

Эксперимент № 2.

Исходные данные:

HAI (HIL-based Augmented ICS) Security (2022 г.) – набор собран на испытательном стенде реалистичной промышленной системы управления, дополненной симулятором аппаратного обеспечения в контуре, который имитирует выработку электроэнергии с

помощью паровых турбин и гидроаккумулирующих электростанций. Размер: 730 МБ. Количество экземпляров: 1 365 602. Количество признаков: 86.

В эксперименте рассчитывается оценка влияния длины исторической последовательности (L) на качество прогнозирования. Значение длины варьировалось в промежутке от 10 до 60 секунд с шагом в 1 секунду.

Результаты эксперимента:

На рисунке 3.1.4.2 показан график зависимости между длины исторической последовательности и среднеквадратической ошибкой при обучении модели прогнозирования (loss = MSE).

Рисунок 3.1.4.2 – Зависимость длины исторической последовательности на прогнозирование

Интерпретация результатов:

Можно отметить, что с ростом длины исторической последовательности возрастает значение ошибки прогнозирования, следовательно, уменьшается точность. Оптимальное значение данного параметра может быть выбрано как экспериментально, так и экспертно, при этом отдавая предпочтение малым значениям.

Результаты экспериментальных исследований алгоритма сильного ИИ АПССОП для наборы данных HAI и DSC, характеризующие его ресурсопотребление, представлены в Таблице 4.1.4.1. Эксперименты проводились на CPU Intel(R) Core(TM) i5-8250U (1.60GHz) с 4 ядрами и 8 Гб памяти.

Согласно результатам эксперимента (см. Таблицу 3.1.4.2) практические эксперименты на типовом ПК для рассмотренных примеров данных показывают удовлетворительные значения ресурсопотребления. Таким образом, данный алгоритм может иметь практическое применение и для реальных задач.

Таблица 3.1.4.2 – Ресурсопотребление алгоритма АПССОП

Набор	Эксперимент	Размер данных	Процесс	Время (сек)	CPU min, %	CPU mean, %	CPU max, %	RAM min, Mb	RAM mean, Mb	RAM max, Mb
HAI	Обучение	903 962	Предобработка	16.71951	0.0	4.538	36.62	430.83	843.49	1632.2
			Нормализация	2.54774	0.0	6.99	36.62	770.95	1201.9	1713.7
			Подготовка модели	0.886559	0.0	3.271	14.55	966.88	990.60	1011.5
			Обучение	17160.40	0.0	19.408	130.21	789.75	1151.8	1429.1
			Всего	**17180.82**	**0.0**	**19.392**	**130.21**	**430.83**	**1151.5**	**1713.7**
	Априорная оценка на обучающей выборке		Предобработка	18.44830	0.0	8.303	18.9	463.22	954.10	1611.0
			Нормализация	3.807075	0.0	7.344	26.18	320.22	798.48	1505.4
			Подготовка модели	1.22841	0.0	2.051	12.43	1047.46	1074.2	1096.5
			Прогнозирование	41999.65	0.0	8.212	104.16	397.42	986.19	3324.3
			Всего	**42029.95**	**0.0**	**8.211**	**104.16**	**320.22**	**986.28**	**3324.3**
	Априорная оценка на тестовой выборке	100 440	Предобработка	18.34764	8.88	12.342	14.34	310.29	804.83	1979.2
			Нормализация	1.502499	12.43	12.499	12.54	1188.11	1723.5	2051.3
			Подготовка модели	206.7369	4.78	15.177	50.35	1454.59	1776.8	3466.7
			Прогнозирование	5024.117	0.0	13.39	37.29	25.69	308.00	818.32
			Всего	**5344.648**	**0.0**	**13.386**	**50.35**	**25.69**	**310.44**	**3466.7**
	Апостериорная оценка на тестовой выборке		Предобработка	18.33556	0.0	12.289	24.41	376.83	870.61	1991.7
			Нормализация	1.108996	0.0	6.028	14.06	1045.7	1687.9	2226.1
			Подготовка модели	297.4781	0.0	5.619	12.54	52.07	1960.5	5029.4
			Прогнозирование	4299.914	0.0	7.847	104.16	58.65	377.94	851.34
			Всего	**4617.821**	**0.0**	**7.864**	**104.16**	**52.07**	**380.63**	**5029.4**
DSC (цикл 1)	Обучение	2241	Подготовка модели	0.231863	5.36	10.097	12.5	293.9	299.91	303.19
			Обучение	72.04426	6.24	28.319	51.96	310.91	391.27	396.6
			Всего	**73.15062**	**5.36**	**28.21**	**51.96**	**283.5**	**390.68**	**396.6**
	Априорная оценка на обучающей выборке		Подготовка модели	0.222437	8.96	11.31	12.54	400.63	400.72	400.92
			Прогнозирование	111.5242	0.0	13.13	26.88	371.39	380.88	412.22
			Всего	**112.1837**	**0.0**	**13.11**	**26.88**	**371.39**	**380.96**	**412.22**
	Априорная оценка на тестовой выборке	249	Подготовка модели	0.351068	0.0	10.71	12.6	380.11	380.62	381.72
			Прогнозирование	12.49861	0.0	7.337	36.62	381.95	385.72	387.05
			Всего	**13.07310**	**0.0**	**7.482**	**36.62**	**371.95**	**385.48**	**387.05**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.218415	0.0	4.168	12.54	388.34	388.37	388.39
			Прогнозирование	11.07139	0.0	4.818	26.88	382.91	387.39	393.14
			Всего	**11.51429**	**0.0**	**4.793**	**26.88**	**382.91**	**387.41**	**393.14**
DSC (цикл 2)	Обучение	3338	Подготовка модели	0.221013	0.0	6.947	12.6	388.36	388.39	388.43
			Обучение	107.4279	0.0	19.425	104.16	388.43	440.97	446.48
			Всего	**107.8108**	**0.0**	**19.382**	**104.16**	**382.98**	**440.80**	**446.48**
	Априорная оценка на обучающей выборке		Подготовка модели	0.110705	8.96	10.73	12.5	445.82	445.82	445.83
			Прогнозирование	196.0278	0.82	13.319	25.09	336.01	365.99	450.42
			Всего	**196.4661**	**0.82**	**13.312**	**25.09**	**336.01**	**366.18**	**458.97**
	Априорная оценка на тестовой выборке	371	Подготовка модели	0.353088	12.4	12.481	12.6	346.08	347.28	350.21
			Прогнозирование	16.37650	0.0	6.783	25.09	350.31	351.37	352.09
			Всего	**16.93600**	**0.0**	**6.96**	**25.09**	**339.16**	**351.25**	**355.49**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.296208	0.0	7.973	12.53	352.34	352.64	353.3
			Прогнозирование	16.04704	0.0	4.586	23.3	334.46	340.01	358.46
			Всего	**16.56868**	**0.0**	**4.679**	**23.3**	**334.46**	**340.40**	**365.05**
DSC (цикл 3)	Обучение	2479	Подготовка модели	0.110738	0.0	9.996	13.03	342.67	342.75	342.77
			Обучение	67.63732	0.0	20.1	104.16	342.77	390.43	392.74
			Всего	**67.94235**	**0.0**	**20.057**	**104.16**	**334.61**	**390.25**	**392.74**
	Априорная оценка на обучающей выборке		Подготовка модели	0.219449	9.3	12.211	13.03	392.1	392.11	392.17
			Прогнозирование	111.2226	0.0	7.401	73.24	361.56	372.60	403.53
			Всего	111.5688	0.0	7.413	73.24	361.56	372.67	404.71
	Априорная оценка на тестовой выборке	275	Подготовка модели	0.293217	0.0	8.061	13.03	367.39	368.14	370.7
			Прогнозирование	12.98918	0.0	5.691	39.06	349.85	351.86	372.29
			Всего	**13.39007**	**0.0**	**5.76**	**39.06**	**349.85**	**352.43**	**377.53**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.24734	0.0	6.704	12.6	357.06	357.40	361.18
			Прогнозирование	11.97028	0.0	4.919	48.83	361.64	364.22	365.66
			Всего	**12.36646**	**0.0**	**4.975**	**48.83**	**350.0**	**364.07**	**366.62**
DSC (цикл 4)	Обучение	3304	Подготовка модели	0.170579	0.0	7.181	13.03	367.39	367.44	367.46
			Обучение	89.95096	0.0	8.076	104.16	344.01	398.34	399.67
			Всего	**90.27637**	**0.0**	**8.071**	**104.16**	**344.01**	**398.24**	**399.67**
	Априорная		Подготовка модели	0.115688	12.43	12.485	12.54	400.0	400.0	400.0

	оценка на обучающей выборке		Прогнозирование	156.3130	6.25	13.509	26.88	361.07	364.84	402.31
			Всего	**156.6455**	**6.25**	**13.511**	**26.88**	**361.07**	**364.92**	**402.31**
	Априорная оценка на тестовой выборке	367	Подготовка модели	0.273271	12.4	12.487	12.6	371.64	372.35	374.99
			Прогнозирование	16.12880	0.0	13.41	52.09	375.01	376.42	376.58
			Всего	**16.56467**	**0.0**	**13.385**	**52.09**	**364.04**	**376.30**	**379.8**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.273269	0.0	9.763	13.03	377.44	377.48	377.56
			Прогнозирование	16.08127	0.0	8.29	37.8	377.34	377.62	377.93
			Всего	**16.57685**	**0.0**	**8.349**	**37.8**	**376.42**	**377.65**	**386.86**
DSC (цикл 5)	Обучение	3704	Подготовка модели	0.141615	0.0	6.282	12.6	378.93	378.93	378.93
			Обучение	98.79186	0.0	14.142	81.9	379.17	413.78	415.4
			Всего	**99.12301**	**0.0**	**14.118**	**81.9**	**377.77**	**413.66**	**415.4**
	Априорная оценка на обучающей выборке		Подготовка модели	0.110668	10.94	11.74	12.54	415.66	415.66	415.66
			Прогнозирование	157.5025	8.88	13.394	33.74	385.54	386.61	416.22
			Всего	**157.8440**	**1.77**	**13.383**	**33.74**	**385.54**	**386.67**	**416.22**
	Априорная оценка на тестовой выборке	411	Подготовка модели	0.262154	0.0	10.452	13.03	393.07	394.60	397.58
			Прогнозирование	17.92918	0.0	6.755	23.3	396.96	397.29	397.43
			Всего	**18.46160**	**0.0**	**6.876**	**23.3**	**387.03**	**397.21**	**398.2**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.341088	0.0	8.775	13.03	383.02	384.61	388.64
			Прогнозирование	17.39524	0.0	4.649	26.04	386.45	387.19	387.34
			Всего	**18.0757**	**0.0**	**4.76**	**26.04**	**373.5**	**387.09**	**388.64**
DSC (цикл 6)	Обучение	6470	Подготовка модели	0.109741	0.0	3.121	12.54	388.27	388.27	388.27
			Обучение	170.0644	0.0	11.658	104.16	388.51	426.96	427.55
			Всего	**170.3906**	**0.0**	**11.645**	**104.16**	**387.22**	**426.90**	**427.55**
	Априорная оценка на обучающей выборке		Подготовка модели	0.221448	0.0	7.091	14.34	427.75	428.19	428.25
			Прогнозирование	286.3668	0.0	6.985	78.12	396.91	397.64	428.66
			Всего	**286.7430**	**0.0**	**6.986**	**78.12**	**396.91**	**397.68**	**433.89**
	Априорная оценка на тестовой выборке	719	Подготовка модели	0.577899	0.0	8.88	13.03	392.99	403.00	409.07
			Прогнозирование	32.93858	0.0	4.735	61.04	390.62	391.65	391.86
			Всего	**33.73001**	**0.0**	**4.813**	**61.04**	**390.62**	**391.85**	**415.96**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.337103	0.0	4.162	12.6	397.29	397.43	397.62
			Прогнозирование	31.30825	0.0	3.709	39.06	397.61	397.94	398.36
			Всего	**31.84880**	**0.0**	**3.714**	**39.06**	**391.94**	**397.96**	**407.27**
DSC (цикл 7)	Обучение	2892	Подготовка модели	0.112697	0.0	6.251	12.6	399.48	399.48	399.48
			Обучение	79.50204	0.0	12.079	104.16	399.48	429.73	430.66
			Всего	**79.78733**	**0.0**	**12.062**	**104.16**	**398.09**	**429.65**	**430.66**
	Априорная оценка на обучающей выборке		Подготовка модели	0.234375	0.0	5.274	13.03	429.58	429.626	429.8
			Прогнозирование	133.9086	0.0	6.092	85.45	418.84	421.56	436.18
			Всего	**134.3038**	**0.0**	**6.09**	**85.45**	**418.84**	**421.59**	**439.76**
	Априорная оценка на тестовой выборке	321	Подготовка модели	0.237734	0.0	4.912	26.04	424.62	424.9	425.03
			Прогнозирование	15.56620	0.0	4.858	73.24	426.77	427.54	427.71
			Всего	**16.00095**	**0.0**	**4.856**	**73.24**	**405.05**	**427.41**	**436.14**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.290435	0.0	4.514	13.03	411.82	412.44	415.27
			Прогнозирование	14.29012	0.0	4.678	48.83	415.84	416.89	417.05
			Всего	**14.73918**	**0.0**	**4.675**	**48.83**	**405.07**	**416.78**	**423.13**
DSC (цикл 8)	Обучение	3746	Подготовка модели	0.174565	0.0	3.754	13.03	416.57	416.66	416.75
			Обучение	102.9804	0.0	6.042	104.16	416.8	438.66	439.21
			Всего	**103.2907**	**0.0**	**6.036**	**104.16**	**416.57**	**438.61**	**439.21**
	Априорная оценка на обучающей выборке		Подготовка модели	0.224365	12.43	12.503	12.54	441.02	441.02	441.02
			Прогнозирование	159.1642	8.88	13.376	26.88	422.15	425.75	441.56
			Всего	**159.6080**	**8.88**	**13.375**	**26.88**	**422.15**	**425.79**	**441.56**
	Априорная оценка на тестовой выборке	416	Подготовка модели	0.295175	12.4	15.07	27.9	429.89	430.84	432.87
			Прогнозирование	18.15190	0.0	6.779	25.09	430.08	430.29	430.42
			Всего	**18.67742**	**0.0**	**6.961**	**27.9**	**425.5**	**430.32**	**441.12**
	Апостериорная оценка на тестовой выборке		Подготовка модели	0.261376	0.0	7.833	13.03	431.57	431.79	432.42
			Прогнозирование	17.60833	0.0	9.473	61.04	430.75	430.84	430.98
			Всего	**18.02432**	**0.0**	**9.438**	**61.04**	**430.28**	**430.87**	**434.55**

3.2 Порядок оценки качества алгоритмов

Эксперименты по априорной и апостериорной оценке качества работы алгоритмов компонента сильного ИИ приведены далее.

1) Алгоритм ПОСНД

Для оценки работоспособности и устойчивости алгоритма ПОСНД использовался набор данных – Titanic. Это стандартный набор данных для применения алгоритмов машинного обучения и искусственного интеллекта, позволяющий предсказать выживание пассажира Титаника. Формат данных – csv. Размер файла – 93.08 Кб. Всего 890 строк данных.

В ходе эксперимента выполнялись следующие шаги:

1) данные передавались в формате, понятном алгоритму ПОСНД (признаки и метки отдельно, разбиение на значения, названия и типы данных, передача известной информации о типах данных);
2) подключался словарь наименований признаков, для которых известен тип данных – категориальный или численный;
3) выставлялся вес выводов о типах данных признаков в зависимости от задействованного метода;
4) выставлялись настройки кластеризации: метод, количество кластеров и итераций;
5) выставлялись пороговые значения при анализе информативности;
6) выставлялись пороговые значения при анализе мультиколлинеарности;
7) выставлялись настройки журналирования данных;
8) запускался алгоритм коррекции типов данных;
9) запускался алгоритм устранения неполноты данных;
10) запускался алгоритм анализа информативности признаков;
11) запускался алгоритм устранения мультиколлинеарности данных.

При анализе типов данных было установлено:

- тип данных признака «PassengerId» был указан некорректно, изменен с категориального на численный;
- тип данных признака «SibSp» не был указан, изменен на категориальный;
- тип данных признака «Parch» не был указан, изменен на категориальный;
- тип данных признака «LABEL_type» не был указана, изменен на численный.

При устранении неполноты данных было заполнено 177 пустых значений. На это ушло 2 итерации процесса кластеризации.

Неинформативных признаков не было обнаружено.

В результате анализа мультиколлинеарности было обнаружено:

- признак «SibSp» коррелирует с признаком «PassengerId»;
- признак «Age» коррелирует с признаком «Pclass».

Рекомендовано удалить признаки «SibSp» и «Age».

2) Алгоритм ИЗСНД

Для оценки работоспособности и устойчивости алгоритма ИЗСНД использовались наборы данных:

- DSC (Driving Smart Crane with Various Loads) – набор данных содержит данные, собранные с сервера мостового крана при движении по L-образному пути с различными нагрузками (0 кг, 120 кг, 500 кг и 1000 кг). Прикладная задача: оценка работоспособности и надежности системы, выявление неисправностей. Размер: 3,66 МБ. Количество экземпляров: 31 304. Количество признаков: 12.
- HAI (HIL-based Augmented ICS) Security (2022 г.) – набор собран на испытательном стенде реалистичной промышленной системы управления, дополненной симулятором аппаратного обеспечения в контуре, который имитирует выработку электроэнергии с помощью паровых турбин и гидроаккумулирующих электростанций. Прикладная задача: оценка кибербезопасности, выявление атак и аномалий. Размер: 730 МБ. Количество экземпляров: 1 365 602. Количество признаков: 86.

В ходе эксперимента для каждого из наборов данных выполнялись следующие шаги:

1) предобработка набора с помощью алгоритма ИЗСНД;
2) разбиение набора на обучающие и тренировочные подвыборки в соотношении 70 к 30.;
3) обучение классификатора на основе случайного леса на тренировочной подвыборке;
4) расчет метрик качества (аккуратность, точность, полнота, F-мера) на тренировочной подвыборке.

Результаты экспериментов представлены в таблицах 3.2.2.1 и 3.2.2.2.

Таблица 3.2.2.1 – Априорная и апостериорная оценки алгоритма ИЗСНД с помощью случайного леса на наборе данных DSC

	Точность	Полнота	F-мера	Аккуратность	ROC-AUC
RandomForestClassifier	0.476583	0.497822	0.486971	0.949207	0.439478
RandomForestClassifier + IZDAP preprocessing	0.476680	0.50	0.488061	0.953360	0.5

Таблица 3.2.2.2 – Априорная и апостериорная оценки алгоритма ИЗСНД с помощью случайного леса на наборе данных HAI

	Точность	Полнота	F-мера	Аккуратность	ROC-AUC
RandomForestClassifier	0.552254	0.504592	0.498865	0.949799	0.433413
RandomForestClassifier + IZDAP preprocessing	0.546849	0.507378	0.504924	0.947424	0.477919

Согласно результатам, представленным в таблицах 3.2.2.1 и 3.2.2.2, практические эксперименты на типовом ПК для рассмотренных примеров данных показывают прирост точности классификатора на описанных наборах данных при использовании алгоритма ИЗСНД. Таким образом, данный алгоритм может иметь практическое применение и для реальных задач.

3) Алгоритм АОТССОП

В таблице 3.2.3.1 представлены результаты оценки алгоритма АОТССОП. Априорная оценка выполнялась на обучающей выборке, что представляло собой 80% выборку от исходного набора данных. Апостериорная оценка выполнялась на тестовой выборке, включающей оставшуюся выборку.

Таблица 3.2.3.1 – Априорная и апостериорная оценки алгоритма АОТССОП

Набор данных (файл)	**Количество классов**	**Оценка**	**Значение точности (accuracy)**
HAI (hai-20.07/test1.csv.gz)	2	Априорная	99.36%
HAI (hai-20.07/test1.csv.gz)	2	Апостериорная	99.33%
HAI (hai-21.03/test1.csv.gz)	2	Априорная	99.57%
HAI (hai-21.03/test1.csv.gz)	2	Апостериорная	99.49%
DSC (combined.csv)	8	Априорная	81.96%
DSC (combined.csv)	8	Апостериорная	81.28%

4) Алгоритм АПССОП

Эксперимент по априорной оценке № 1.

Используемые метрики оценки и их обоснование:

В процессе априорной оценки не оказывается значительное внешнее воздействие на процесс прогнозирования, а лишь осуществляется наблюдение за ним. Все вектора значений признаков прогнозируются независимо друг от друга. После окончания прогнозируемого периода сопоставляются значения спрогнозированных показателей и параметров с полученными в действительности.

Для оценки алгоритма АПССОП используются следующие показатели качества: (1) среднеквадратичная ошибка прогнозирования *MSE* (формула 3.2.1), (2) средняя абсолютная ошибка прогнозирования *MAE* (формула 3.2.2), (3) точность прогнозирования *P* (формула 3.2.3). Выбор данных показателей обоснован тем, что они позволяют оценить отклонение прогнозируемых значений от реальных. Так как в экспериментах используется масштабирование признаков на отрезок [0, 1], то и значение ошибок прогнозирования лежит в этом же отрезке, где минимальное значение ошибки 0, а максимальное – 1.

Шаги методики оценки:

1) загрузка обученных моделей прогнозирования и нормализации данных;
2) разделение исходного набора данных на тренировочную и тестовую выборки с соотношением 9:1;
3) нормализации выборок с использованием масштабирования значений признаков на отрезок [0, 1];
4) формирование генератора временных рядов на основе тренировочной выборки;
5) извлечение последнего пакета данных из генератора временных рядов;
6) прогнозирование векторов признаков последовательности, равной длине тестовой выборки, при этом для каждого последующего прогнозируемого вектора к пакету добавляется вектор тестовой выборки;
7) вычисление показателей качества алгоритма.

Пример подобной оценки и ее результаты представлены в разделе 2.1, пример задачи №2 алгоритма АПССОП.

Программный код оценки:

```
from apssop import *
from data_classes import AopssopData as PDATA

data = DataLoaderAndPreprocessor(dataset_name, mode=mode, suf=suf)
PDATA.features_names = data.features_names

PDATA.forecasting_model_path = data.forecasting_model_path
PDATA.normalization_model_path = data.normalization_model_path

train, test = data.train_test_split(train_size=train_size)

normalization_model = DataScaler(scaler_path=PDATA.normalization_model_path,
                    open=True)
scaled_train = normalization_model.transform(train)
scaled_test = normalization_model.transform(test)

forecasting_model = AIForecaster(model_path=PDATA.forecasting_model_path,
                    open=True)

train_generator = forecasting_model.data_to_generator(scaled_train)

PDATA.forecasting_time_window = len(scaled_test)
current_batch = forecasting_model.get_batch(train_generator, -1)

predictions = []
for i in range(PDATA.forecasting_time_window):
  sys.stdout.write('\r\x1b[K' + 'Forecasting: {0}/{1}'.format(i,
            PDATA.forecasting_time_window - 1))
  sys.stdout.flush()

  current_pred = forecasting_model.forecasting(current_batch,
                                forecasting_data_length=1,
                                verbose=False)
  predictions.append(current_pred[0])
  new_event = scaled_test[i]
  current_batch = np.append(current_batch[:, 1:, :], [[new_event]], axis=1)

predictions = pd.DataFrame(predictions).values
PDATA.forecasting_results = normalization_model.inverse(predictions)

PDATA.forecasting_quality = estimator.estimate(true=scaled_test,
                              pred=predictions,
                            feature_names=PDATA.features_names)
estimator.save(file_name=dataset_name + suf + '_test_independently')
print(PDATA.forecasting_quality)
print('Done')
```

Эксперимент по апостериорной оценке № 2.

Используемые метрики оценки и их обоснование:

Примером влияния на ход развития событий может являться, в частности, корректировка на основании ожидаемых спрогнозированных значений. В процессе апостериорной оценки вектора значений признаков прогнозируются в зависимости от ранее спрогнозированных значений. После окончания прогнозируемого периода сопоставляются значения спрогнозированных показателей и параметров с полученными в действительности.

Для оценки алгоритма АПССОП используются следующие показатели качества: (1) среднеквадратичная ошибка прогнозирования *MSE* (формула 3.2.1), (2) средняя абсолютная ошибка прогнозирования *MAE* (формула 3.2.2), (3) точность прогнозирования *P* (формула 3.2.3). Выбор данных показателей обоснован тем, что они позволяют оценить отклонение прогнозируемых значений от реальных. Так как в экспериментах используется масштабирование признаков на отрезок [0, 1], то и значение ошибок прогнозирования лежит в этом же отрезке, где минимальное значение ошибки 0, а максимальное – 1.

Шаги методики оценки:

1) загрузка обученных моделей прогнозирования и нормализации данных;
2) разделение исходного набора данных на тренировочную и тестовую выборки с соотношением 9:1;
3) нормализации выборок с использованием масштабирования значений признаков на отрезок [0, 1];
4) формирование генератора временных рядов на основе тренировочной выборки;
5) извлечение последнего пакета данных из генератора временных рядов;
6) прогнозирование векторов признаков последовательности, равной длине тестовой выборки, при этом для каждого последующего прогнозируемого вектора к пакету добавляется ранее спрогнозированный вектор признаков;
7) вычисление показателей качества алгоритма.

Пример подобной оценки и ее результаты представлены в разделе 2.1, пример задачи №1 алгоритма АПССОП.

Программный код оценки:

```
from apssop import *
from data_classes import AopssopData as PDATA

data = DataLoaderAndPreprocessor(dataset_name, mode=mode, suf=suf)
PDATA.features_names = data.features_names

PDATA.forecasting_model_path = data.forecasting_model_path
PDATA.normalization_model_path = data.normalization_model_path

train, test = data.train_test_split(train_size=train_size)

normalization_model = DataScaler(scaler_path=PDATA.normalization_model_path,
                    open=True)
```

```
scaled_train = normalization_model.transform(train)
scaled_test = normalization_model.transform(test)

forecasting_model = AIForecaster(model_path=PDATA.forecasting_model_path,
                    open=True)

train_generator = forecasting_model.data_to_generator(scaled_train)

PDATA.forecasting_time_window = len(scaled_test)
last_batch = forecasting_model.get_batch(train_generator, -1)

pred = forecasting_model.forecasting(last_batch,
                      forecasting_data_length=PDATA.forecasting_time_window)
PDATA.forecasting_results = normalization_model.inverse(pred)

estimator = ForecastEstimator()
PDATA.forecasting_quality = estimator.estimate(true=scaled_test, pred=pred,
feature_names=PDATA.features_names)

estimator.save(file_name=dataset_name + suf + 'test_dependently')

print(PDATA.forecasting_quality)
print('Done')
```

4. ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 Точки входа в программу

Обращение к программе происходит путем создания объектов классов и вызова их них необходимых функций.

Алгоритм ПОСНД

Основными обращениям к программе через вызовы функций модуля ПОСНД являются следующие:

а) из состава класса *CheckDataTypes*:

1) *__determite_type_by_substring__* – определение типа данных по названию признака;

2) *__determite_type_by_unique__* – определение типа данных на основе анализа количества уникальных значений признака;

3) *__determite_type_by_float__* – определение типа данных на основе анализа наличия float значений у признака;

4) *__calculate_by_priority__* – принятие решение о типе данных признака на основе веса решений отдельных алгоритмов;

5) *correct_types* – алгоритм, объединяющий предыдущие в единый процесс работы с признаками данных;

б) из состава класса *ClusterFilling*:

1) *__fill_with_centroids__* – кластеризация на основе центройдов;

2) *__fill_with_stats__* – кластеризация на основе расчета статистик;

3) *fill* – алгоритм, применяющий один из предыдущих на основе решения пользователя или значения по умолчанию;

в) из состава класса *Informativity*:

1) *calculate_informativity* – алгоритм анализа информативности признаков данных;

г) из состава класса *MultiCollinear*:

1) *remove_uninformative_features* – алгоритм устранения мультиколлинеарности данных.

Алгоритм ИЗСНД

Основными обращениям к программе через вызовы функций модуля ИЗСНД являются следующие:

а) из состава класса *IzdapAlgo*:

1) конструктор – инициализация модели алгоритма ИЗСНД путем указания порога для построения предикатов;

2) *fit()* – обучение модели алогоритма;

3) *get_rules()* – вывод построенных правил в консоль в строковом виде;

4) *transform()* – преобразование набора данных с использованием построенных правил к бинарный формат.

Алгоритм АОТССОП

Обращение к программе происходит путем создания объектов классов и вызова их них необходимых функций. Основными обращениям к программе через вызовы функций модуля АОТССОП являются следующие:

б) из состава класса *SAIClassifier*:

5) конструктор – инициализация модели классификатора путем указания типа классификаторов, конфигурации искусственной нейронной сети и количества эпох обучения;

6) *fit()* – обучение модели классификатора;

7) *predict()* – определение класса объектов при помощи модели классификатора;

8) *load()* – десериализация модели классификатора;

9) *save()* – сериализация модели классификатора;

в) из состава класса *FormatDetector*:

1) конструктор – инициализация класса для определения типа файла с обрабатываемым набором данных путем указания типа пути к этому файлу;

г) из состава класса *DataLoader*:

1) конструктор – инициализация класса путем указания пути к файлу с набором данных и формате данных;

д) из состава класса *ClsEstimator*:

2) конструктор – инициализация класса для оценки параметров эффективности классификаторов путем указания признаков, метод классов и классификаторов;

3) *estimate()* – оценка параметров эффективности классификаторов;

Алгоритм АПССОП

Обращение к программе происходит путем создания объектов классов и вызова их них необходимых функций. Основными обращениям к программе через вызовы функций модуля АОТССОП являются следующие:

1) класс *AIForecaster* – элемент для прогнозирования СлОП:

Атрибуты класса:

а) *model* – нейросетевая модель прогнозирования (объект keras.models.Sequential),

б) *time_window_length* – длина исторической последовательности (целое число),

в) *n_features* – количество признаков (целое число),

г) *model_path* – путь для внешнего сохранения модели прогнозирования (текстовая строка),

д) *epochs* – количество эпох для обучения модели прогнозирования (целое число),

е) *batch_size* – размер пакетов для обучения (целое число),

ж) *early_stop* – объект EarlyStopping (библиотека keras).

Методы класса:

а) *def __init__(self, time_window_length=0, n_features=0, model_path='', n_epochs=1, open=False)* – конструктор класса для базовой инициализации параметров, где *open* – указатель на необходимость загрузить существующую модель (булевая переменная);

б) *def train(self, train_generator, validation_generator=None, save=True)* – обучение и валидация модели нейронной сети на данных, *train_generator* – генератор временных рядов обучающих данных (объект keras TimeseriesGenerator), *validation_generator* – генератор временных рядов данных валидации (объект keras TimeseriesGenerator); *save* – указатель на необходимость сохранить модель во внешний файл (булевая переменная);

в) *def forecasting(self, current_batch, forecasting_data_length, verbose=True)* – прогнозирование значений, где *current_batch* – исходный пакет данных для прогнозирования (многомерный массив), *forecasting_data_length* – длина прогнозируемой последовательности (целое число), *verbose* – указать на необходимость отображать процесс прогнозирования в командной строке (булевая переменная);

г) *def save_model(self)* – сохранение модели прогнозирования во внешний файл;

д) *def open_model(self)* – загрузка модели прогнозирования из внешнего файла;

е) *def data_to_generator(self, data)* – преобразование массива данных в генератор временных рядов, где *data* – исходные данные (многомерный массив);

ж) *def get_batch(self, generator, current_batch_id)* – извлечение отдельного пакета из генератора временных рядов, где *generator* – генератор временных рядов (объект keras TimeseriesGenerator), *current_batch_id* – порядковый номер извлекаемого пакета (целое число).

2) класс *DataScaler* – элемент для нормализации данных:

Атрибуты класса:

а) *scaler* – объект модели нормализации (объект MinMaxScaler),

б) *scaler_path* – путь для внешнего сохранения модели нормализации (текстовая строка).

Методы класса:

а) *def fit(self, data, save=True)* – обучение модели нормализации, где *data* – исходные данные (многомерный массив), *save* – указатель на необходимость сохранить модель нормализации во внешний файл (булевая переменная);

б) *def save(self)* – сохранение модели нормализации во внешний файл;

в) *def open(self)* – загрузка модели нормализации из внешнего файла;

г) *def transform(self, data)* – нормализация данных,

д) *def inverse(self, data)* – обратное преобразование нормализованных данных.

3) класс *ForecastEstimator* – элемент для оценки качества модели прогнозирования:

Атрибут класса: *quality* – матрица результатов оценки качества (объект pandas DataFrame).

Методы класса:

а) *def estimate(self, true, pred, feature_names=[])* – оценка качества модели прогнозирования, где *true* – фактические значения (многомерный массив), *pred* – прогнозируемые значения (многомерный массив), *feature_names* – имена признаков (список);

б) *def save(self, file_name)* – сохранение результатов оценки во внешний файл, где *file_name* – путь к внешнему файлу (текстовая строка).

4.2 Базовые функции

Модуль ПОСНД

Начало работы с набором данных

```
data = Data()
titanic_path = '../datasets/titanic.csv'
titanic = pd.read_csv(titanic_path)
data.features_names = ["PassengerId", "Pclass", "Age", "SibSp", "Parch"]
data.labels_names = ["Survived", "Fare"]
data.features_matrix = np.array(titanic[data.features_names])
data.labels_matrix = np.array(titanic[data.labels_names])
data.features_types = ["cat", "cat", "num", None, None]
data.labels_types = ["cat", None]
```

Подключение журналирования

```
__Verbose__.PrintLog.instance().set_print_mode(True)
__Verbose__.PrintLog.instance().set_severity_level("status")
```

Запуск алгоритмов модуля

```
CheckDataTypes.CheckDataTypes.correct_types(data)
ClusterFilling.ClusterFilling.fill(data)
Informativity.Informativity.calculate_informativity(data)
Multicolinear.MultiCollinear.remove_uninformative_features(data)
```

Модуль ИЗСНД

Создание объекта IzdapAlgo

```
algo = IzdapAlgo(0.1)
```

Обучение модели
algo.fit(test_data, class_column = class_column, positive_class_label = positive_class_label)

Модуль АОТССОП

Создание объекта SAIClassifier
SAIClassifier('neural_network', 10, 1)

Создание объекта FormatDetector
FormatDetector('../hai/hai-20.07/test1.csv.gz')

Создание объекта DataLoader
DataLoader('../hai/hai-20.07/test1.csv.gz', 64, ‘,’)

Создание объекта ClsEstimator
classifiers = [SAIClassifier(c, in_size, out_size, plot=False) for c in classifier_types]
xor_ds = {'features': np.array([[0, 0], [0, 1], [1, 0], [1, 1]]), 'labels': np.array([[0], [1], [1], [0]])}
ClsEstimator(xor_ds['features'], xor_ds['labels'], xor_ds['labels'], [classifier]

Модуль АПССОП

Создание объекта AIForecaster
forecasting_model = AIForecaster(n_epochs=3,
time_window_length = 10,
n_features = len(features_names),
model_path = ”…/forecasting_model)

Создание существующей модели для объекта AIForecaster
forecasting_model = AIForecaster (model_path = ”…/forecasting_model, open = True)

Формирование генератора временных рядов
generatorX = forecasting_model.data_to_generator(trainX)

Обучение модели объекта AIForecaster
forecasting_model.train(generatorX)

Прогнозирование с использованием модели объекта AIForecaster
prediction = forecasting_model.forecasting(current_batch = batch, forecasting_data_length = 1000)

Создание объекта DataScaler
scaler = DataScaler(scaler_path = ”scaler.pkl”)

Открытие существующей модели для объекта DataScaler
scaler = DataScaler(scaler_path = “scaler.pkl”, open=True)

Обучение модели объекта DataScaler
scaler.fit(data = trainX, save = True)

Нормализация данных с использованием DataScaler
```
scaled_trainX = scaler.transform(trainX)
```

Создание объекта ForecastEstimator
```
estimator = ForecastEstimator()
```

Оценка эффективности прогнозирования
```
quality = estimator.estimate(true = scaled_testX, pred = prediction)
```

5. ПРОВЕРКА ПРОГРАММЫ

5.1 Модульные и интеграционные тесты

Модульные и интеграционные тесты всех модулей компонента приведены далее.

5.1.1 Модуль ПОСНД

Модульный тест №1. test_check_data_types

Задача модуля: проверка корректности исправления типов данных

Исходный код:
```
expected_features_types = ['cat', 'num', 'cat', 'cat', 'num']
CheckDataTypes.CheckDataTypes.correct_types(data)
self.assertEqual(data.features_types, expected_features_types)
```

Модульный тест №2. test_cluster_filling

Задача модуля: проверка корректности устранения неполноты данных

Исходный код:
```
expected_features_matrix = np.array(list(zip(*[
[1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
 [4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
 [4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
 [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 1, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2],
[4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
])))
ClusterFilling.ClusterFilling.fill(data)
self.assertTrue(np.array_equal(data.features_matrix, expected_features_matrix))
```

Модульный тест №3. test_informativity

Задача модуля: проверка корректности анализа информативности признаков данных

Исходный код:
```
expected_features_matrix = np.array(list(zip(*[
[1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
[4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 1, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2]
])))
Informativity.Informativity.calculate_informativity(data)
self.assertTrue(np.array_equal(data.features_matrix, expected_features_matrix))
```

Модульный тест №4. test_multicolinearity

Задача модуля: проверка корректности устранения мультиколлинеарности признаков данных

Исходный код:

```
expected_features_matrix = np.array(list(zip(*[
[1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
 [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 1, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2]
])))
Multicolinear.MultiCollinear.remove_uninformative_features(data)
self.assertTrue(np.array_equal(data.features_matrix, expected_features_matrix))
```

5.1.2 Модуль ИЗСНД

Задача модуля: проверка корректности подсчета коэффициента регрессии

Исходный код:

```
print(inspect.stack()[0][3])
message = 'The regression coefficient is calculated incorrectly'
self.assertEqual(round(IzdapAlgo().calculate_regression_coefficient(nA=200, nB=200, nAB=100, N=1000), 3), 0.375, message)
```

Модульный тест №2: test_klosgen_measure

Задача модуля: проверка корректности подсчета меры Клозгена

Исходный код:

```
print(inspect.stack()[0][3])
message = 'The klosgen measure is calculated incorrectly'
self.assertEqual(round(IzdapAlgo().calculate_klosgen_measure(nA=200, nB=200, nAB=100, N=1000), 3), 0.134, message)
```

Модульный тест №3: test_string_column

Задача модуля: проверка корректности определения алгоритмом строковых признаком в данных

Исходный код:

```
print(inspect.stack()[0][3])
message = 'String columns were identified incorrectly'

data = make_classification(n_samples=200, n_features=4,
                           n_informative=2, n_classes=2,
                           random_state=42)
df = pd.DataFrame(data[0], columns = ['col1','col2','col3','col4',])
df['class'] = pd.Series(data[1])
algo = IzdapAlgo(0.2)
algo.fit(df, class_column = "class", positive_class_label = '1')

self.assertEqual(len(algo.string_columns), 0, message)
```

Модульный тест №4: test_number_column

Задача модуля: проверка корректности определения алгоритмом числовых признаков в данных

Исходный код:

```
print(inspect.stack()[0][3])
```

```
message = 'String columns were identified incorrectly'

data = make_classification(n_samples=200, n_features=4,
                           n_informative=2, n_classes=2,
                           random_state=42)
df = pd.DataFrame(data[0], columns = ['col1','col2','col3','col4',])
df['class'] = pd.Series(data[1])
algo = IzdapAlgo(0.2)
algo.fit(df, class_column = "class", positive_class_label = '1')

self.assertEqual(len(algo.number_columns), 4, message)
```

Модульный тест №5: test_class_stats

Задача модуля: проверка корректности расчета статистик класса

Исходный код:

```
print(inspect.stack()[0][3])
message = 'Class stats was calculated incorrectly'

data = make_classification(n_samples=200, n_features=4,
                           n_informative=2, n_classes=2,
                           random_state=42)
df = pd.DataFrame(data[0], columns = ['col1','col2','col3','col4',])
    df['class'] = pd.Series(data[1])
    algo = IzdapAlgo(0.2)
    algo.fit(df, class_column = "class", positive_class_label = 1)

self.assertEqual(algo.class_stats[1], df['class'].value_counts()[1] , message)
```

Модульный тест №6: test_data_stats

Задача модуля: проверка корректности расчета статистик значений признаков

Исходный код:

```
print(inspect.stack()[0][3])
message = 'Data stats was calculated incorrectly'

data = make_classification(n_samples=200, n_features=4,
                           n_informative=2, n_classes=2,
                           random_state=42)
df = pd.DataFrame(data[0], columns = ['col1','col2','col3','col4',])
df['class'] = pd.Series(data[1])
algo = IzdapAlgo(0.2)
algo.fit(df, class_column = "class", positive_class_label = 1)

self.assertEqual(len(algo.data_stats), 4, message)
```

Модульный тест №7: test_rules

Задача модуля: проверка корректности извлечения правил из данных

Исходный код:

```
print(inspect.stack()[0][3])
message = 'Rules were extracted incorrectly incorrectly'
```

```
data = make_classification(n_samples=200, n_features=4,
                           n_informative=2, n_classes=2,
                           random_state=42)
df = pd.DataFrame(data[0], columns = ['col1','col2','col3','col4',])
df['class'] = pd.Series(data[1])
algo = IzdapAlgo(0.2)
algo.fit(df, class_column = "class", positive_class_label = 1)
rule = "if col1 in [Interval(1.032, 1.661, closed='right'), Interval(0.404, 1.032, closed='right'), Interval(-
0.224, 0.404, closed='right'), Interval(1.661, 2.289, closed='right'), Interval(2.289, 2.917, closed='right')]
then 1 (regression_coef=0.759)"
self.assertEqual(algo.get_rules()[0], rule, message)
```

Модульный тест №8: test_predicates

Задача модуля: проверка корректности построения предикатов

Исходный код:

```
print(inspect.stack()[0][3])
message = 'Predicates is not chacking data correctly'

data = make_classification(n_samples=200, n_features=4,
                           n_informative=2, n_classes=2,
                           random_state=42)
df = pd.DataFrame(data[0], columns = ['col1','col2','col3','col4',])
df['class'] = pd.Series(data[1])
algo = IzdapAlgo(0.2)
algo.fit(df, class_column = "class", positive_class_label = 1)

predicate = algo.predicates[0]
row = df[0]
        self.assertEqual(predicate.is_true(df[:1]).iloc[0], 1, message)
```

5.1.3 Модуль АОТССОП

Модульный тест №1. test_sc_is_not_none

Задача модуля: Проверка того, что модель оценивания создана корректно.

Исходный код:

```
self.assertFalse(SAIClassifier('neural_network', 10, 1) is None, 'The object of class SAIClassifier could not be
created')
```

Модульный тест №2. test_sc_type_is_correct

Задача модуля: Проверка того, что тип модели оценивания корректно проинициализирована.

Исходный код:

```
classifier_types = ['decision_tree', 'naive_bayes', 'logistic_regression', 'neural_network']
for c in classifier_types:
    classifier = SAIClassifier(c, 10, 1, plot=False)
    self.assertTrue(classifier.cls_type in classifier_types, 'The classifier type is not correct')
```

Модульный тест №3. test_sc_fit

Задача модуля: Проверка того, что модель оценивания обучена корректно.

Исходный код:

```
classifier_types = ['decision_tree', 'naive_bayes', 'logistic_regression', 'neural_network']
in_size = np.shape(self.xor_ds['features'])[1]
out_size = np.shape(self.xor_ds['labels'])[1]
for c in classifier_types:
    classifier = SAIClassifier(c, in_size, out_size, plot=False)
    try:
      classifier.fit(self.xor_ds['features'], self.xor_ds['labels'])
    except:
      self.assertTrue(False, 'Error while calling fit')
```

Модульный тест №4. test_sc_predict

Задача модуля: Проверка того, что модель оценивания корректно выполнила оценку тестового набора данных.

Исходный код:

```
classifier_types = ['decision_tree', 'naive_bayes', 'logistic_regression', 'neural_network']
in_size = np.shape(self.xor_ds['features'])[1]
out_size = np.shape(self.xor_ds['labels'])[1]
for c in classifier_types:
    classifier = SAIClassifier(c, in_size, out_size, plot=False)
    classifier.fit(self.xor_ds['features'], self.xor_ds['labels'])
    try:
      classifier.predict(self.xor_ds['features'])
    except:
      self.assertTrue(False, 'Error while calling predict')
```

Модульный тест №5. test_sc_save

Задача модуля: Проверка того, что модель оценивания корректно сохранилась в файл.

Исходный код:

```
classifier_types = ['decision_tree', 'naive_bayes', 'logistic_regression', 'neural_network']
in_size = np.shape(self.xor_ds['features'])[1]
out_size = np.shape(self.xor_ds['labels'])[1]
for c in classifier_types:
    classifier = SAIClassifier(c, in_size, out_size, plot=False)
    classifier.fit(self.xor_ds['features'], self.xor_ds['labels'])
    f = './' + c + '.bin'
if os.path.isfile(f):
os.remove(f)
classifier.save(f)
self.assertTrue(os.path.isfile(f) or os.path.isfile(f + '.index'), 'The classifier could not be saved into the file')
```

Модульный тест №6. test_sc_load

Задача модуля: Проверка того, что модель оценивания корректно создана из файла.

Исходный код:

```
classifier_types = ['decision_tree', 'naive_bayes', 'logistic_regression', 'neural_network']
in_size = np.shape(self.xor_ds['features'])[1]
out_size = np.shape(self.xor_ds['labels'])[1]
for c in classifier_types:
    classifier = SAIClassifier(c, in_size, out_size, plot=False)
    f = './' + c + '.bin'
    classifier.save(f)
```

```
    try:
      classifier.load(f)
    except:
      self.assertTrue(False, 'Error while calling load')
```

Модульный тест №7. test_fd_is_not_none

Задача модуля: Проверка того, что объект определителя разделителя полей данных корректно проинициализирован.

Исходный код:

```
f = '../hai/hai-20.07/test1.csv.gz'
if os.path.isfile(f):
    self.assertFalse(FormatDetector(f) is None, 'The object of class FormatDetector could not be created')
```

Модульный тест №8. test_fd_file_exists

Задача модуля: Проверка того, что загружаемый файл существует.

Исходный код:

```
tmp_file = tempfile.NamedTemporaryFile()
try:
    FormatDetector(tmp_file.name)
  except:
    self.assertTrue(False, 'Error: file "{}" must exist'.format(tmp_file))
```

Модульный тест №9. test_fd_delimiter_is_correct

Задача модуля: Проверка корректности разделителя в наборе данных.

Исходный код:

```
f = '../hai/hai-20.07/test1.csv.gz'
if os.path.isfile(f):
    fd = FormatDetector(f)
    self.assertTrue(fd.d in [';', ','], 'The delimiter is not correct')
```

Модульный тест №10. test_dl_is_not_none

Задача модуля: Проверка того, что объект загрузки данных корректно проинициализирован.

Исходный код:

```
f = '../hai/hai-20.07/test1.csv.gz'
class DataLoaderExample(DataLoader):
    def load(self, file):
      pass
  if os.path.isfile(f):
    self.assertFalse(DataLoaderExample(f, 0, 0) is None, 'The object of class DataLoaderExample could not be
created')
```

Модульный тест №11. test_ce_is_not_none

Задача модуля: Проверка того, что объект вычисления показателей эффективности классификации данных корректно проинициализирован.

Исходный код:

```
classifier_types = ['decision_tree', 'naive_bayes', 'logistic_regression', 'neural_network']
```

```
in_size = np.shape(self.xor_ds['features'])[1]
out_size = np.shape(self.xor_ds['labels'])[1]
classifiers = [SAIClassifier(c, in_size, out_size, plot=False) for c in classifier_types]
for classifier in classifiers:
    self.assertFalse(ClsEstimator(self.xor_ds['features'], self.xor_ds['labels'], self.xor_ds['labels'], [classifier])
is None,
                'The object of class ClsEstimator could not be created')
```

Модульный тест №12. test_ce_estimate

Задача модуля: Проверка того, что вычисление показателей эффективности классификации данных выполнено корректно.

Исходный код:

```
classifier_types = ['decision_tree', 'naive_bayes', 'logistic_regression', 'neural_network']
in_size = np.shape(self.xor_ds['features'])[1]
out_size = np.shape(self.xor_ds['labels'])[1]
classifiers = [SAIClassifier(c, in_size, out_size, plot=False) for c in classifier_types]
for classifier in classifiers:
    classifier.fit(self.xor_ds['features'], self.xor_ds['labels'])
  try:
    ClsEstimator(self.xor_ds['features'], self.xor_ds['labels'], self.xor_ds['labels'], classifiers).estimate()
  except:
    self.assertTrue(False, 'Error while calling estimate')
```

5.1.4 Модуль АПССОП

Модульный тест №1. test_forecasting_model_is_not_none

Задача модуля: Проверить, что модель прогнозирования создана.

Исходный код:

```
message = 'The object of class AIForecaster could not be created'
self.assertIsNotNone(AIForecaster(10, 10), message)
```

Модульный тест №2. test_forecasting_model_save

Задача модуля: Проверить, что модель прогнозирования сохранена во внешний файл.

Исходный код:

```
AIForecaster(10, 10).save_model('model.h5')
message = 'The forecasting model could not be saved into the file'
self.assertTrue(os.path.isfile('model.h5'), message)
```

Модульный тест №3. test_forecasting_model_open

Задача модуля: Проверка, что модель прогнозирования загружена из внешнего файла.

Исходный код:

```
message = 'File with forecasting model does not exist'
self.assertIsNotNone(AIForecaster(10, 10).open_model('model.h5'), message)
```

Модульный тест №4. test_generator_create

Задача модуля: Проверка, что генератор временных рядов создан.

Исходный код:

```
data = range(0, 1000)
message = 'The object of class TimeseriesGenerator could not be created'
```

```
self.assertIsNotNone(AIForecaster(10, 10).data_to_generator(data), message)
```

Модульный тест №5. test_data_load

Задача модуля: Проверка, что набор данных загружен.

Исходный код:

```
message = 'The dataset name is not correct'
for dataset_name in ['hai', 'alarms', 'edge-iiotset', 'test']:
self.assertIsNotNone(DataLoaderAndPreprocessor(dataset_name).data, message)
```

Модульный тест №6. test_dataframe_split

Задача модуля: Проверка, что набор данных разделен на обучающую и тестовую выборки.

Исходный код:

```
message = 'Data split failed'
for i in [0.1, 0.25, 0.5, 0.8, 0.99]:
    self.assertIsNot(DataLoaderAndPreprocessor('test').train_test_split(i), (None, None), message)
```

Модульный тест №7. test_normalization_model_is_not_none

Задача модуля: Проверить, что модель нормализации создана.

Исходный код:

```
message = 'The object of class DataScaler could not be created'
self.assertIsNotNone(DataScaler(scaler_path='scaler.pkl'), message)
```

Модульный тест №8. test_normalization_model_save

Задача модуля: Проверить, что модель нормализации сохранена во внешний файл.

Исходный код:

```
message = 'The normalization model could not be saved into the file'
DataScaler(scaler_path='scaler.pkl').save()
self.assertTrue(os.path.isfile('scaler.pkl'), message)
```

Модульный тест №9. test_normalization_model_open

Задача модуля: Проверка, что модель нормализации загружена из внешнего файла.

Исходный код:

```
message = 'File with normalization model does not exist'
self.assertIsNone(DataScaler('scaler.pkl').open(), message)
```

Модульный тест №10. test_estimator_mae_results

Задача модуля: Проверка, что средняя абсолютная ошибка считается корректно.

Исходный код:

```
message = 'Forecasting quality evaluation failed in MAE'
true = [0, 0, 1, 0, 2, 1, 1, 0, 0, 1]
pred = [0, 1, 1, 0, 0, 0, 1, 0, 1, 1]
result = ForecastEstimator().estimate(true, pred)
self.assertEqual(result.loc['ALL_FEATURES', 'MAE'], 0.5, message)
```

Модульный тест №11. test_estimator_mse_results

Задача модуля: Проверка, что среднеквадратичная ошибка считается корректно.

Исходный код:

```
message = 'Forecasting quality evaluation failed in MSE'
true = [0, 0, 1, 0, 2, 1, 1, 0, 0, 1]
pred = [0, 1, 1, 0, 0, 0, 1, 0, 1, 1]
result = ForecastEstimator().estimate(true, pred)
self.assertEqual(result.loc['ALL_FEATURES', 'MSE'], 0.7, message)
```

Модульный тест №12. test_estimator

Задача модуля: Проверка, что входные данные для оценки качества имею одинаковую длину.

Исходный код:

```
message = 'Forecasting quality evaluation failed in MSE'
true = [0, 0, 1, 0, 2, 1, 1, 0, 0, 1]
pred = [0, 1, 1, 0, 0, 0]
result = ForecastEstimator().estimate(true, pred)
self.assertTrue(result.empty, message)
```

5.1.5 Интеграционный тест

Описание теста:

В данном тесте представлена интеграция алгоритмов ИЗСНД и АПССОП. Перед обучением модели и для прогнозирования используются данные, предобработанные алгоритмом ИЗСНД на основе правил.

Исходный код:

```
from aopssop import DataScaler, AIForecaster, ForecastEstimator
from aopssop import IzdapAlgo
from aopssop import AopssopData as PDATA

# loading data
data = PDATA.features_matrix

algo = IzdapAlgo(probability_threshold)
algo.fit(data, class_column='Attack', positive_class_label=1)

transformed_data = algo.transform(data)

train, test = train_test_split(transformed_data, train_size=0.9)

forecasting_model = AIForecaster(n_epochs=3,
                    time_window_length=PDATA.time_window_length,
                    n_features= transformed_data.shape[1],
                    model_path=PDATA.forecasting_model_path,
                    )

train_generator = forecasting_model.data_to_generator(train)

loss = forecasting_model.train(train_generator)
```

```
current_batch = forecasting_model.get_batch(train_generator, -1)

predictions = []
        for i in range(PDATA.forecasting_time_window):
                current_pred = forecasting_model.forecasting(current_batch,
                        forecasting_data_length=1,
                        verbose=False)
        predictions.append(current_pred[0])
        new_event = test[i]
        current_batch = np.append(current_batch[:, 1:, :], [[new_event]], axis=1)

predictions = pd.DataFrame(predictions).values
PDATA.forecasting_results = normalization_model.inverse(predictions)
```

5.2 Контрольные примеры

Контрольные примеры для демонстрации работы алгоритмов компонента сильного ИИ приведены далее.

5.2.1 Алгоритм ПОСНД

Контрольный пример №1.

Описание: Обработка набора данных Titanic.

Входные данные

```
data = Data()
titanic_path = '../datasets/titanic.csv'
titanic = pd.read_csv(titanic_path)
data.features_names = ["PassengerId", "Pclass", "Age", "SibSp", "Parch"]
data.labels_names = ["Survived", "Fare"]
data.features_matrix = np.array(titanic[data.features_names])
data.labels_matrix = np.array(titanic[data.labels_names])
data.features_types = ["cat", "cat", "num", None, None]
data.labels_types = ["cat", None]
```

Настройка алгоритма

```
data.n_jobs = 2
data.feature_names_substrings = {
    "num": ["age", "id"],
    "cat": ["surv", "tick", "cabin"]
}
data.feature_max_cat = 10
data.types_priority = {
    "manual": 0.5,
    "substring": 1,
    "unique": 1,
    "float": 0.3
}
data.fill_method = "mean_mode"
data.n_clusters = 10
data.cluster_max_iter = 5
data.thresholds_correlation_with_label = {
```

```
    "num_num": [0.2] * len(data.labels_names),
    "cat_cat": [0.1] * len(data.labels_names),
    "num_cat": [0.2] * len(data.labels_names)
}
data.thresholds_min_number_of_predicted_labels = [1] * len(data.features_names)
data.thresholds_multicolinear = {
    "num_num": 0.9,
    "cat_cat": 0.7,
    "num_cat": 0.8
}
```

Обработка

```
CheckDataTypes.CheckDataTypes.correct_types(data)
ClusterFilling.ClusterFilling.fill(data)
Informativity.Informativity.calculate_informativity(data)
Multicolinear.MultiCollinear.remove_uninformative_features(data)
```

Выходные данные

```
START ANALYSIS OF DATA TYPES (N of substr[num,cat]=[2,3], max_cat=10,
priority[manual,substring,unique,float]=[0.5,1,1,0.3])
   PassengerId with FEATURE_type=cat is incorrect. Changing to num (num=1.0,cat=0.8).
   SibSp with FEATURE_type=None is incorrect. Changing to cat (num=0.0,cat=1.3).
   Parch with FEATURE_type=None is incorrect. Changing to cat (num=0.0,cat=1.3).
   Fare with LABEL_type=None is incorrect. Changing to num(num=1.0,cat=0.3).
ANALYSIS OF DATA TYPES IS COMPLETE
START CLUSTER FILLING (fill_method=mean_mode, n_clusters=10, max_iter=5)
   iteration 0, fill 177 NaNs
   iteration 1, fill 177 NaNs
   iteration 2, fill 177 NaNs
   iteration 3, fill 177 NaNs
   => conversged on iteration 3
DONE CLUSTER FILLING
START INFORMATIVITY ANALYSIS
NO UNINFORMATIVE FEATURES
START MULTICOLINEARITY ANALYSIS
   delete feature=SibSp as it correlates with feature=PassengerId
   delete feature=Age as it correlates with feature=Pclass
REMOVE MILTICOLINEAR FEATURES: [Age, SibSp]
```

5.2.2 Алгоритм ИЗСНД

Контрольный пример №1.

Описание: применение алгоритма ИЗСНД совместно с алгоритмом Random Forest на наборе данных IEEE Smart Crane.

Входные данные

```
DATA_PATH = "../datasets/IEEE_smart_crane.csv"
ieee_data = pd.read_csv(DATA_PATH)
```

Обработка Random Forest

```
X_train, X_test, y_train, y_test = train_test_split (ieee_data.drop(columns=['Alarm']), ieee_data.Alarm,
test_size = 0.3, shuffle = False)
clf = RandomForestClassifier(random_state=5)
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_test_proba = clf.predict_proba(X_test)
```

Обработка с ИЗСНД + Random Forest

```
algo = IzdapAlgo(0.1)
algo.fit(ieee_data, class_column = "Alarm", positive_class_label = 1)
transformed_data = algo.transform(ieee_data)
X_train, X_test, y_train, y_test = train_test_split (transformed_data.drop(columns=['Alarm']),
transformed_data.Alarm, test_size = 0.3, shuffle = False)
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_test_proba = clf.predict_proba(X_test)
```

Выходные данные

```
Random forest without IZSND preproccesing
memory usage: 3.3 MB
ROC-AUC: 0.439477698605656
Accuracy: 0.9492066872537536

Random forest with IZSND preproccesing
memory usage: 2.9+ MB
ROC-AUC: 0.5
Accuracy: 0.9533595996166543
```

5.2.3. Алгоритм АОТССОП

Контрольный пример №1.

Описание: Загрузка набора данных. Для загрузки данных используются вспомогательные классы FormatDetector и DataLoader, которые осуществляют определение типа разделителя полей и загрузку набора данных.

```
import numpy as np
import pandas as pd
from collections import OrderedDict
from sklearn.utils import shuffle
from aossop import SAIClassifier, FormatDetector, DataLoader, ClsEstimator

class DataLoaderHai(DataLoader):
        def __init__(self, file, n, d):
                DataLoader.__init__(self, file, n, d)

        def load(self, file):
                n, d = self.n, self.d
                if n in [64, 84]:
        self.features = np.genfromtxt(file, delimiter=d, dtype=float, skip_header=1, usecols=range(1,n-4))
        attacks = np.genfromtxt(file, delimiter=d, dtype=float, skip_header=1, usecols=range(n-4,n))
        self.labels = np.array([[0] if sum(a) == 0 else [1] for a in attacks])
```

```
        self.num_labels = self.labels
    elif n == 88:
        self.features = np.genfromtxt(file, delimiter=d, dtype=float, skip_header=1, usecols=range(1,n-1))
        self.labels = np.genfromtxt(file, delimiter=d, dtype=float, skip_header=1, usecols=range(n-1,n))
        if len(np.shape(self.labels)) == 1:
            self.labels = np.array([[e] for e in self.labels])
        self.num_labels = self.labels

class DataLoaderEdgeIIoTSet(DataLoader):
  def __init__(self, file, n, d):
    DataLoader.__init__(self, file, n, d)

  def load(self, file):
    df = pd.read_csv(file, low_memory=False, sep=self.d)
    drop_columns = ["frame.time", "ip.src_host", "ip.dst_host", "arp.src.proto_ipv4", "arp.dst.proto_ipv4",
            "http.request.method", "http.file_data", "http.referer", "http.request.full_uri",
"http.request.version",
            "icmp.transmit_timestamp", "http.request.uri.query", "tcp.options", "tcp.payload", "tcp.srcport",
            "tcp.dstport", "udp.port", "dns.qry.name", "dns.qry.name.len", "dns.qry.qu",
            "mqtt.conack.flags", "mqtt.msg", "mqtt.protoname", "mqtt.topic", "Attack_label"]
    df.drop(drop_columns, axis=1, inplace=True)
    df.dropna(axis=0, how='any', inplace=True)
    df.drop_duplicates(subset=None, keep="first", inplace=True)
    df = shuffle(df)
    str_labels = df.iloc[:,-1].tolist()
    self.features = np.array(df.iloc[:,:-1].values.tolist())
    unique_labels = list(OrderedDict.fromkeys(str_labels))
    labels = list(map(lambda x: unique_labels.index(x), str_labels))
    self.labels = np.array([np.zeros(len(unique_labels))] * len(labels))
    for i in range(len(labels)): # one-hot encoding
      self.labels[i][labels[i]] = 1
    self.num_labels = labels

class DataLoaderDataPort(DataLoader):
  def __init__(self, file, n, d):
    DataLoader.__init__(self, file, n, d)

  def load(self, file):
    self.features = np.genfromtxt(file, delimiter=self.d, dtype=float, skip_header=1, usecols=range(1,self.n-
1))
    labels = np.genfromtxt(file, delimiter=self.d, dtype=float, skip_header=1, usecols=range(self.n-1,self.n))
    self.labels = np.array([np.zeros(len(set(labels)))] * len(labels))
    for i in range(len(labels)): # one-hot encoding
      self.labels[i][int(labels[i]) - 1] = 1
    self.num_labels = labels

fd = FormatDetector(file)
dl = None
if dataset == 'hai':
  dl = DataLoaderHai(file, fd.n, fd.d)
elif dataset == 'edge-iiotset':
  dl = DataLoaderEdgeIIoTSet(file, fd.n, fd.d)
```

```
elif dataset == 'dataport':
   dl = DataLoaderDataPort(file, fd.n, fd.d)
else:
   assert(False)
```

Входные данные: *file* – название набора данных (текстовая строка)

Выходные данные: *dl* – объект, содержащий загруженный набор данных.

Контрольный пример №2.

Описание: Обучение и тестирование модели оценивания. Предварительно должны быть созданы объекты fd и dl классов FormatDetector и DataLoader.

```
import numpy as np
import pandas as pd
from collections import OrderedDict
from sklearn.utils import shuffle
from aossop import SAIClassifier, FormatDetector, DataLoader, ClsEstimator
classifiers = [SAIClassifier(cls_type=c, in_size=np.shape(dl.features)[1], out_size=np.shape(dl.labels)[1]) \
        for c in ['neural_network']]
ce = ClsEstimator(dl.features, dl.labels, dl.num_labels, classifiers)
r = ce.estimate(print_metrics=False)
print(r)
```

Входные данные: *fd* и *dl* – объекты классов FormatDetector и DataLoader

Выходные данные: *r* – объект, содержащий результаты оценивания экземпляров обучающей и тестовой выборок.

5.2.4 Алгоритм АПССОП

Контрольный пример №1.

Описание: Обучение модели прогнозирования. Для загрузки данных используется вспомогательный класс DataLoaderAndPreprocessor, который осуществляет загрузку и предобработку набора данных.

```
from apssop import *
from data_classes import AopssopData as PDATA


data = DataLoaderAndPreprocessor(dataset_name, mode=mode, suf=suf)
PDATA.features_names = data.features_names

PDATA.forecasting_model_path = data.forecasting_model_path
PDATA.normalization_model_path = data.normalization_model_path

train, test = data.train_test_split(train_size=0.9)

normalization_model = DataScaler(scaler_path=PDATA.normalization_model_path)
normalization_model.fit(train)
scaled_train = normalization_model.transform(train)
```

```
forecasting_model = AIForecaster(n_epochs=3,
                    time_window_length=PDATA.time_window_length,
                    n_features=len(PDATA.features_names),
                    model_path=PDATA.forecasting_model_path,
                    )

train_generator = forecasting_model.data_to_generator(scaled_train)
loss = forecasting_model.train(train_generator)
```

Входные данные: *dataset_name* – название набора данных (текстовая строка), *suf* – суффикс-метка для сохранения внешний файлов (текстовая строка), *mode* – вспомогательная переменная, режим загрузки данных (целое число).

Выходные данные: *loss* – значение потерь (среднеквадратическая ошибка) на всех эпохах обучения (список).

Контрольный пример №2.

Описание: Прогнозирование данных с использованием обученной модели прогнозирования. Прогнозируется последовательность длиной 100. Для загрузки данных используется вспомогательный класс DataLoaderAndPreprocessor, который осуществляет загрузку и предобработку набора данных.

```
from apssop import *
from data_classes import AopssopData as PDATA


data = DataLoaderAndPreprocessor(dataset_name, mode=mode, suf=suf)
PDATA.features_names = data.features_names

PDATA.forecasting_model_path = data.forecasting_model_path
PDATA.normalization_model_path = data.normalization_model_path

normalization_model = DataScaler(scaler_path=PDATA.normalization_model_path,
                    open=True)
scaled_data = normalization_model.transform(data.data)

forecasting_model = AIForecaster(model_path=PDATA.forecasting_model_path,
                    open=True)

generator = forecasting_model.data_to_generator(scaled_data)

PDATA.forecasting_time_window = 100
last_batch = forecasting_model.get_batch(generator, -1)

forecasting_results = forecasting_model.forecasting(last_batch,
                forecasting_data_length=PDATA.forecasting_time_window)
PDATA.forecasting_results = normalization_model.inverse(pred)
print('Done')
```

Входные данные: *dataset_name* – название набора данных (текстовая строка), *suf* – суффикс-метка для сохранения внешний файлов (текстовая строка), *mode* – вспомогательная переменная, режим загрузки данных (целое число).

Выходные данные: *forecasting_results* – последовательность прогнозируемых значений (многомерный массив).

6. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

6.1 Состав и структура входных данных

1) Входными данными для модуля ПОСНД являются следующие (имеющие определенный формат):

- *features_matrix (numpy.array)* – двумерный массив (только численные значения), состоящий из объектов и их признаков;
- *features_types (numpy.array)* – одномерный массив (возможные значения элементов *["num", "cat", None]*), длина которого соответствует количеству признаков;
- *features_names (numpy.array)*: одномерный массив (только строковые значения), длина которого соответствует количеству признаков;
- *labels_matrix (numpy.array)* – двумерный массив (только численные значения), состоящий из объектов и их меток;
- *labels_types (numpy.array)* – одномерный массив (возможные значения элементов ["num", "cat", None]), длина которого соответствует количеству меток;
- *labels_names (numpy.array)* – одномерный массив (только строковые значения), длина которого соответствует количеству меток.

2) Входными данными для модуля ИЗСНД являются следующие (имеющие определенный формат):

- *data* – признаки и метки обучающей выборки (многомерный массив);
- *class_column* – имя признака, содержащего метки обучающей выборки (строка);
- *positive_class_label* – метка положительного класса;
- *rule_metric* – метрика оценки правил (название в строковом формате);
- *probability_threshold* – вероятность, задающая порог отсечения значений для построения предикатов.

3) Входными данными для модуля АОТССОП являются следующие (имеющие определенный формат):

- *x_train* – признаки обучающей выборки (многомерный массив);
- *y_train* – метки обучающей выборки (многомерный массив);
- *x_test* – признаки тестовой выборки (многомерный массив);
- *y_test* – метки тестовой выборки (многомерный массив);
- *x* – объект для определения класса с помощью глубокой нейронной сети (многомерный массив);
- *saved_file* – путь к файлу для сериализации модели (текстовая строка);
- *loaded_file* – путь к файлу для десериализации модели (текстовая строка);
- *file* – путь к файлу для определения его типа и загрузки данных (текстовая строка);

- *features* – признаки данных сложного объекта (массив с данными);
- *labels(num_labels)* – метки данных сложного объекта (массив с данными);
- *classifiers* – выбранные ранее классификаторы (программный объект).

4) Входными данными для модуля АПССОП являются следующие:

- *model_path* – путь к модели прогнозирования (текстовая строка);
- *scaler_path* – путь к модели нормализации (текстовая строка);
- *time_window_length* – размер временного окна при обучении, длина источеской последовательности (число);
- *forecast_len* – размер временного окна для прогнозирования (число),
- *data* – матрица признаков исходного набора данных (многомерный массив).

6.2 Подготовка входных данных

Дополнительной предобработки входные данные не требуют, однако модули компонента загружают часть данных их внешних файлов следующим образом.

1) Модуль ПОСНД: данные для работы модуля ПОСНД передаются в виде пути к файлу, содержащему данные, нуждающиеся в предобработке. При этом необходимо отделить признаки данных от их меток.

2) Модуль ИЗСНД: данные для работы модуля ИЗСНД передаются в виде пути к файлу, содержащему обучающие данные.

3) Модуль АОТССОП: модель десериализуется из внещнего файла – loader_file; файл имеет бинарный формат, реализованный в библиотеке pickle. Входной набор данных для определения типа и загрузки загружается из файла по пути к нему – file; файл имеет текстовый формат в виде набора строк, элементы которых разделяются символами ‘,’ или ‘;’ (также, он может быть запакован с помощью gzip).

4) Модуль АПССОП: Модель прогнозирования загружается из файла по пути к нему – model_path; файл имеет бинарный формат, реализованный в библиотеке keras.models. Модель нормализации загружается из файла по пути к нему – scaler_path; файл имеет бинарный формат, реализованный в библиотеке pickle.

6.3 Состав и структура выходных данных

1) Выходными данными для модуля ПОСНД являются следующие (имеющие определенный формат):

- вывод в консоль результатов анализа корректности типов данных.
- вывод в консоль результатов устранения неполноты данных.
- вывод в консоль результатов анализа информативности признаков.
- вывод в консоль результатов анализа мультиколлинеарности.

2) Выходными данными для модуля ИЗСНД являются следующие (имеющие определенный формат):

- вывод в консоль построенных ассоциативных правил класса.
- возврат преобразованных данных в формате pandas.DataFrame (функция transform).

3) Выходными данными для модуля АОТССОП являются следующие (имеющие определенный формат):

- оцениваемые значения (числовые значения);
- показатели качества оценивания (числовые значения);
- вывод обученной модели по некоторому датасету (программный формат модели).

4) Выходными данными для модуля АПССОП являются следующие:

- прогнозируемые значения (числовые значения);
- показатели качества прогнозирования (числовые значения);
- вывод обученной модели прогнозирования по некоторому датасету (программный формат модели);
- вывод обученной модели нормализации по некоторому датасету (программный формат модели).

6.4 Интерпретация выходных данных

Дополнительной постобработки выходные данные не требуют, однако модули компонента сохраняют часть данных во внешние файлы следующим образом.

1) Модуль ПОСНД не сохраняет выходные данные во внешние файлы.

2) Модуль ИЗСНД не сохраняет выходные данные во внешние файлы.

3) Модуль АОТССОП: модель сериализуется во внешний файл – saved_file; файл имеет бинарный формат, реализованный в библиотеке pickle.

4) Модуль АПССОП: Модель прогнозирования сериализуется во внешний файл – model_path; файл имеет бинарный формат, реализованный в библиотеке keras. Модель нормализации сериализуется во внешний файл – scaler_path; файл имеет бинарный формат, реализованный в библиотеке pickle. Результаты оценки прогнозирования сохраняются во внешний файл формата CSV – file_name.

7. СООБЩЕНИЯ

Компонент выводит следующие сообщения.

№	Текст сообщение	Значение сообщения
1	PassengerId with FEATURE_type=cat is incorrect. Changing to num (num=1.0,cat=0.8).	Сообщение содержит информацию о проверке корректности типа данных признака
2	iteration 2, fill 177 NaNs	Сообщение содержит информацию о процессе устранения неполноты данных, а именно – об итерации процесса кластеризации и количестве заполненных пустых значений
3	price is uninformative, can predict 1, while required 3	Сообщение содержит информацию о процессе анализа информативности признаков, неинформативные признаки рекомендуется удалить
4	delete feature=SibSp as it correlates with feature=PassengerId	Сообщение содержит информацию о процессе устранения мультиколлинеарности данных, а

		именно рекомендацию об удалении признака, если он коррелирует с другим признаком
5	“Test accuracy: N”, где N – дробное число до 3-го знака	Сообщение содержит информацию о точности обучения модели оценивания состояния сложного объекта.
6	“{MN} of {CN} on {N} sample ({L} instances): {M}”, где MN – имя метрики (precision/accuracy – точность, recall – полнота, fscore – F-метрика); CN – имя классификатора; N – тип процесса для вычисления метод (training – обучение, testing – тестирование); L – число меток; M – значение метрики	Сообщение содержит информацию о точности обучения и тестирования модели оценивания состояния сложного объекта.
7	“File with forecasting model does not exist”	Сообщение сигнализирует об отсутствии модели предсказания по заданному пути.
8	“Too many values for categorical feature '{F}'. Delete feature from data”, где F – число признаков	Сообщение сигнализирует о превышении допустимого числа признаков.
9	“The proportion of the training sample is not in the interval (0, 1)”	Сообщение сигнализирует о том, что доля обучающей выборки не входит в интервал от 0 до 1.
10	“File with normalization model does not exist”	Сообщение сигнализирует об отсутствии модели нормализации по заданному пути.
11	“The length of the samples is not equal”	Сообщение сигнализирует о том, что длина реальных и спрогнозированных данных не совпадают.
12	“{FC}”, где FC – показатели качества предсказания	Сообщение содержит показатели качества проведенного предсказания.
13	“Done”	Сообщение сигнализирует о завершении процесса тестирования.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
	измененных	замененных	новых	Аннулированных					