МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УТВЕРЖДАЮ
Научный руководитель ИЦ СИИП Университета ИТМО

_____ А. В. Бухановский
_____ 2022 г.

УТВЕРЖДАЮ
Директор СПб ФИЦ РАН

_____ А. Л. Ронжин
_____ 2022 г.

КОМПОНЕНТ БИБЛИОТЕКИ АЛГОРИТМОВ СИЛЬНОГО ИИ В ЧАСТИ АЛГОРИТМОВ ПРЕДОБРАБОТКИ, АВТОНОМНОГО ОЦЕНИВАНИЯ И ПРОГНОЗИРОВАНИЯ СОСТОЯНИЯ СЛОЖНЫХ ОБЪЕКТОВ И ПРОЦЕССОВ НА ОСНОВЕ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ СОБЫТИЙ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ И НЕДОСТОВЕРНОСТИ ДАННЫХ

ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.00853-01 13 12

Представители
Организации-разработчика

Руководитель разработки

_____ И.В. Котенко
_____ 2022 г.

Нормоконтролер

_____Н. А. Александрова
_____ 2022 г.

2022

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

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УТВЕРЖДЕН
RU.СНАБ.00853-01 13 12

КОМПОНЕНТ БИБЛИОТЕКИ АЛГОРИТМОВ СИЛЬНОГО ИИ В ЧАСТИ АЛГОРИТМОВ ПРЕДОБРАБОТКИ, АВТОНОМНОГО ОЦЕНИВАНИЯ И ПРОГНОЗИРОВАНИЯ СОСТОЯНИЯ СЛОЖНЫХ ОБЪЕКТОВ И ПРОЦЕССОВ НА ОСНОВЕ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ СОБЫТИЙ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ И НЕДОСТОВЕРНОСТИ ДАННЫХ

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.00853-01 13 12

ЛИСТОВ 40

2022

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

АННОТАЦИЯ

Документ содержит описание программы «Компонент библиотеки алгоритмов сильного ИИ в части алгоритмов предобработки, автономного оценивания и прогнозирования состояния сложных объектов и процессов на основе интеллектуальной обработки событий в условиях неопределенности и недостоверности данных» RU.СНАБ.00853-01 13 12. Данное программное обеспечение предназначено для предобработки, автономного оценивания и прогнозирования состояния сложных объектов и процессов и входит в состав инструментального ПО, разрабатываемого в рамках плана Исследовательского центра в сфере искусственного интеллекта «Сильный ИИ в промышленности» (ИЦ ИИ) в соответствии с соглашением 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. ОБЩИЕ СВЕДЕНИЯ

— Программа «Компонент предобработки, автономного оценивания и прогнозирования состояния сложных объектов и процессов на основе интеллектуальной обработки событий в условиях неопределенности и недостоверности данных» (сокр. ПАОПС) библиотеки алгоритмов сильного ИИ RU.СНАБ.00853-01 13 12 разработан в соответствии с мероприятием М1.3.3 Разработка и испытание экспериментального образца библиотеки алгоритмов сильного ИИ в части алгоритмов автономного оценивания и прогнозирования состояния сложных объектов и процессов на основе интеллектуальной обработки событий в условиях неопределенности и недостоверности данных (п.1.3.3 Плана деятельности СИИП).

— Компонент предназначен для определения значений характеристик сложных технических объектов и процессов (сокр. СлОП) в текущий, прошедшие и будущие моменты времени с требуемыми точностью и достоверностью.

— Компонент разработан на языке Python (версия не ниже 3.7) с использованием следующих библиотек: data_classes, gzip, keraskeras, matplotlib, numpy, os, pandas, pickle, sklearn, tensorflow.

— Компонент размещен в репозитории по адресу https://github.com/labcomsec/aopssop_lib.

— Для использования необходим интерпретатор Python (версия не ниже 3.7).

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1 Назначение программного компонента

Назначением программного компонента является предоставление базового функционала для создания и программной реализации перспективных методов ИИ, направленных на решение задач, связанных с оцениванием и прогнозированием состояния СлОП. При этом в части сильного ИИ, назначением программного комплекса является следующее (реализованное в рамках отдельных алгоритмов):

— улучшение качества предоставляемых данных, в том числе сырых и нечетких данных[1], что может быть задействовано на подготовительной стадии оценивания и прогнозирования состояния СлОП (алгоритм ПОСНД);

— построение модели СлОП, основанной на знаниях, что обеспечивается путем извлечения множества знаний, описывающих связи между понятиями (коэффициентами дискретного вейвлет-преобразования) и действиями в предметной области оценивания и

[1] Дискретная оптимизация и моделирование в условиях неопределенности данных. Перепелица В. А., Тебуева Ф. Б. Издательство: Академия Естествознания. 2007. ISBN: 978-5-91327-013-9.

прогнозирования состояния СлОП, а также извлечением множества действий (выводов), вытекающих из этих знаний (алгоритм ИЗСНД);

— автономное[2] оценивание состояний СлОП, что обеспечивается путем автоматического извлечения высокоуровневых представлений исходных данных и взаимосвязей между ними; при этом, 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 (алгоритм АОТССОП);

— автономное прогнозирование состояний СлОП; что обеспечивается наличием большого числа гиперпараметров и настраиваемых весов, свойственных для глубоких НС, что позволяет с достаточно высоким качеством выявлять закономерности между признаками состояния системы и прогнозировать последующие состояния за заданный отрезок времени (алгоритм АПССОП).

Таким образом, разработанные алгоритмы, а также их совместное применение, реализуют Национальную стратегию развития искусственного интеллекта в части разработки перспективных методов искусственного интеллекта, а именно методов, направленных на автономное решение задач оценки и прогнозирования состояний сложных объектов и процессов.

2.2 Классы решаемых задач

Компонент позволяет решить следующие классы задач:

1) Предобработка сырых и нечетких данных о СлОП (алгоритмы ПОСНД (коррекция типов, устранение неполноты и мультиколлинеарности входных данных) и ИЗСНД (извлечения фрагментов знаний, имеющихся в данных, в виде ассоциативных правил вида «ЕСЛИ <посылка>, ТО <следствие>»)).

2) Автономное оценивание текущего состояния СлОП (алгоритм АОТССОП (определение наличия или отсутствия в текущий момент времени определенного вида функциональных неисправностей, дефектов и атакующих воздействий, свойственных целевой системе)).

3) Автономное прогнозирование состояний СлОП (алгоритм АПССОП (обучение модели прогнозирования состояний СлОП на основе исторических данных в виде временного ряда; прогнозирование состояний, описываемых вектором характеристик, за заданный отрезок времени)).

Отметим, что перечень конкретных решаемых задач показывает, что каждый алгоритм в отдельности уже может быть полезен разработчикам, однако для решения классов технических задач выгодно совместное их использование.

---

[2]Термин автономный используется для обозначения характеристики процесса, выполняемого с максимальным отсутствием вмешательства человека.

2.3 Функциональные ограничения на применение

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

1) для алгоритма ПОСНД устанавливаются:

— порог уникальности значений признаков, устанавливаемый пользователем (значение по умолчанию: 0.70, пользователю рекомендуется изменять данное значение в соответствии с решаемой задачей и используемым набором данных, рекомендуемый диапазон [0.65; 0.95]);

— порог количества уникальных значений признака для отнесения к категориальному типу данных (значение по умолчанию: 10, пользователю рекомендуется изменять данное значение в соответствии с решаемой задачей и используемым набором данных, в том числе в процентном соотношении к количеству строк в наборе данных);

— максимальное количество кластеров, на которые предполагается разбивать данные (значение по умолчанию: 10, пользователю рекомендуется изменять данное значение в соответствии с решаемой задачей и используемым набором данных);

— максимальное количество итераций процесса кластеризации (значение по умолчанию: 10, пользователю рекомендуется изменять данное значение в соответствии с решаемой задачей и используемым набором данных);

— максимальное количество узлов, реализующих параллельную обработку кластеров (значение по умолчанию: 2, пользователю рекомендуется изменять данное значение в соответствии с параметрами вычислительной техники, на которой используется алгоритм);

— минимальное/максимальное значение порога информативности признаков (значение по умолчанию: 0.70, пользователю рекомендуется изменять данное значение в соответствии с решаемой задачей и используемым набором данных, рекомендуемый диапазон [0.65; 0.95]);

2) для алгоритма ИЗСНД устанавливаются:

— максимальный размер / объем обучающих данных, характеризующих СлОП, заданных множеством описаний (ограничен объемом оперативной и долговременной памяти машины, алгоритмических ограничений нет);

— максимальное количество агрегатов, которые объединяют отдельные значения каждого признака (ограничено значением M * L * 2, где M – это количество признаков в наборе данных, L – количество различных меток класса в наборе данных);

— максимальное количество ассоциативных правил (ограничено максимальным количеством агрегатов);

— минимальное/максимальное пороговое значение метрики информативности (должно быть в интервале [0; 1] для стандартизованных метрик);

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 (стандартным соотношением обучающей и тестовой выборок считаются 8:2 или 9:1, однако итоговое решение зачастую зависит от решаемой задачи и используемого набора данных, поэтому рекомендуется воспользоваться хорошо известными рекомендациями по избавлению от таких проблем формирования выборок, как отсутствие данных, недостаточное количество данных, разбалансировка, ложные зависимости, ограниченный набор источников, изменение генеральной совокупности во времени и др.);

— 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 (ограничения на длину вектора не накладываются);

— прогнозирование с использованием обученной модели осуществляется только для фиксированного вектора характеристик, заложенного в обученную модель;

— значение длины исторической последовательности для обучения модели прогнозирования не может быть больше, чем 90% от длины обучающей выборки (ограничения на длину обучающей выборки не накладываются);

— 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.

2.4 Область применения

Модули разработанного компонента могут быть применены для таких объектов, как 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.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1 Используемые методы

Для удобства описания, представление задействованных методов было условно разделено в соответствии с классами задач, на решение которых они направлены: предобработка данных о СлОП и снижение их размерности; автономное оценивание

состояния СлОП; автономное прогнозирование состояния СлОП. Рассмотрим каждый набор методов более подробно.

1) Предобработка данных о СлОП и снижение их размерности.

Используемые в данном классе задач методы решают задачи коррекции типов данных, устранения неполноты и мультиколлинеарности данных, а также извлечения фрагментов знаний в виде ассоциативных правил. При этом для работы методов используются входные данные в табличной форме следующего вида:

$$D^I = (F^M, F^T, F^S, L^R, L^T, L^S, E^C),$$

где $F^M = \{F^M_{on}\},\ o \in 1..O, n \in 1..N$ – матрица признаков, O – количество объектов, N – количество признаков; $F^T = \{F^T_n\}, n \in 1..N$ – вектор типов признаков; $F^S = \{F^S_n\}, n \in 1..N$ – вектор имен признаков; $L^R = \{L^R_{om}\},\ o \in 1..O, m \in 1..M$ – матрица лейблов, M – количество лейблов; $L^T = \{L^T_m\}, m \in 1..M$ – вектор типов лейблов; $L^S = \{L^S_m\}, m \in 1..M$ – вектор имен лейблов; $E^C = \{E^C_k\}, k \in 1..K$ – вектор координат пустых значений, K – количество пустых значений.

Рассмотрим данные методы более подробно.

Коррекция типов данных происходит на основе статистических критериев, связанных с количеством уникальных значений признака, обнаружением последовательностей значений признака, анализом текстовых названий признаков, а также анализом дробных значений для представления некорректных FLOAT (*.0(0)) в качестве INT.

Устранение неполноты данных происходит на основе статистического усреднения данных в рамках отдельных кластеров. Разбиение данных на кластеры возможно на основе таких подходов, как K-means, Mini Batch K-means, Mean-shift, OPTICS, Bisecting K-means.

Устранение мультиколлинеарности данных происходит на основе коэффициентов корреляции Пирсона и Спирмена, а также дисперсионного анализа (ANOVA).

В основе методов извлечения фрагментов знаний в виде ассоциативных правил лежат следующие математические соотношения. Обучающие данные, характеризующие СлОП, заданы множеством описаний **X** = {X_i}, i = 1, …, I, СлОП (экземпляров), каждому из которых поставлена в соответствие метка класса из множества **Y** = {Y_j}, j = 1, …, J, задающая состояние СлОП в некоторый момент времени. Метки классов $Y_j \in$ **Y** (состояния СлОП) могут быть представлены в категориальный шкале (в простейшем случае, булевыми) или быть линейно упорядоченными.

Признаки $F_k \in$ **F**, каждый из которых имеет область допустимых значений **D**$_k$ = {d^l_k}, k = 1, …, K, l =1, ..., L, характеризующие описания СлОП, $X_i \in$ **X**, также могут быть

представлены в категориальной, булевой и/или числовой шкалах, а также могут быть текстами на естественном языке. Числовые признаки должны предварительно быть квантованы и далее могут рассматриваться как дискретные линейно упорядоченные признаки. Для этого можно использовать стандартные процедуры дискретизации. Для текстов необходимо предварительно выполнить процедуру векторизации с помощью стандартной процедуры, например, используя метрику TF-IDF.

Для построения ассоциативных правил на множестве обучающих данных на первом шаге необходимо объединить отдельные значения $d_k^l \in \mathbf{D}_k$ каждого признака в агрегаты $\mathbf{A}_k^{(j)}$, обладающие общим свойством по отношению к некоторому состоянию СлОП, заданному меткой класса $Y_j \in \mathbf{Y}$. Таким образом, в ходе построения агрегатов область значений $\mathbf{D}_k$ каждого атрибута $F_k \in \mathbf{F}$ разделяется на непересекающиеся подмножества $\mathbf{A}_k^{(j)} \subseteq \mathbf{D}_k$, поставленные в соответствие некоторым классам множества $Y_j \in \mathbf{Y}$. Подмножества $\mathbf{A}_k^{(j)}$ строятся таким образом, чтобы каждое из них содержало те значения признака $F_k \in \mathbf{F}$ из множества его значений $\mathbf{D}_k$, которые более характерны для описаний СлОП, $X_i \in \mathbf{X}$, принадлежащих классу Y_j, чем для описаний, принадлежащих другим классам.

Для построений подмножеств-агрегатов могут быть использованы различные метрики оценки частотности. При этом предлагается использовать эвристический метод, основанный на наивном байесовском подходе, выбор которого обусловлен результатами, полученными в выполненных ранее экспериментах. А именно, предполагается, что некоторое значение $d_k^l \in \mathbf{D}_k$ признака $F_k \in \mathbf{F}$ принадлежит подмножеству $\mathbf{A}_k^{(j)}$, $d_k^l \in \mathbf{A}_k^{(j)}$, в том случае, когда

$$p(Y_j \mid d_k^l) > p(\bar{Y}_j \mid d_k^l) + \delta. \qquad (3.1.1)$$

где $p(Y_j \mid d_k^l)$ – условная вероятность класса Y_j при $F_k = d_k^l$, $p(\bar{Y}_j \mid d_k^l)$ – это условная вероятность любого класса, кроме класса Y_j, при $F_k = d_k^l$, а δ – некоторая положительная константа, значение которой позволяет регулировать мощность формируемых подмножеств-агрегатов.

Отметим, что с помощью теоремы Байеса выражение (3.1.1) можно привести к виду, требующему вычисления только $p(d_k^l \mid Y_j)$ *и* $p(d_k^l \mid \bar{Y}_j)$ и априорных вероятностей классов. И такие вероятности могут быть подсчитаны всего за один проход по обучающим данным, что делает описываемый подход вычислительно эффективным.

Далее предлагается построить унарные предикаты $U_k^{(j)}(d_k^l \in \mathbf{A}_k^{(j)})$, каждый из которых принимает истинное значение в том случае, когда значение d_k^l признака F_k

для некоторого описания СлОП, $X_i \in \mathbf{X}$ принадлежит подмножеству-агрегату $\mathbf{A}_k^{(j)}$, $d_k^l \in \mathbf{A}_k^{(j)}$. Таким образом, подмножество $\mathbf{A}_k^{(j)}$ является областью истинности унарного предиката $U_k^{(j)}$, а для значений $d_k^l \notin \mathbf{A}_k^{(j)}$ предикат $U_k^{(j)}$ принимает значение «ложь».

Используя построенные предикаты далее в качестве посылки, а метки класса $F_k \in \mathbf{F}$, соответствующие некоторому состоянию СлОП, в качестве следствия, могут быть сформированы ассоциативные правила класса следующего вида:

$$U_k^{(j)}\left(d_k^l \in \mathbf{A}_k^{(j)}\right) \to Y_j, j = 1, \dots, J;\ k = 1, \dots, K;\ l = 1, \dots, L. \quad (3.1.2)$$

Такие правила можно интерпретировать как знания, описывающие связи между понятиями и действиями в предметной области оценивания и прогнозирования состояния СлОП, а также множество действий (выводов), вытекающих из знаний.

С целью сокращения множества подученных правил и выявления наиболее информативных из них для каждого правила далее необходимо рассчитать значение метрики информативности $\mu(U_k^{(j)}, Y_j)$. Предполагается, что метрика информативности поступает на вход метода в качестве параметра. В простейшем случае может использоваться классическая метрика уверенности или другая, реже используемая метрика, например, мера Клозгена или коэффициент регрессии событий, которые претендуют на то, чтобы оценивать также силу причинной связи между посылкой и следствием правила. Важным условием для метрики является ее несимметричность.

После вычисления значений выбранной метрики для каждого правила, множество правил сортируется в порядке убывания модуля значения метрики. В итоговом множестве правил останутся правила вида (3.1.1), удовлетворяющие условию: $\left|\mu\left(U_k^{(j)}, Y_j\right)\right| \geq \mathrm{M}$, где M – это пороговое значение метрики, которое выбирается экспериментально, исходя из мощности полученного множества правил и среднего значения модуля метрики для правил.

В результате такой фильтрации итоговое множество правил будет содержать только наиболее «сильные» правила, отражающие наиболее «сильные» связи параметров описаний СлОП с его состоянием. Полученное множество правил можно также интерпретировать как новое булево признаковое пространство, соответственно к нему можно применять любые методы сокращения пространства признаков, позволяющие обрабатывать признаки в булевой шкале.

2) Автономное оценивание состояния СлОП

Автономное оценивание текущего состояния сложных объектов и процессов основывается на математических соотношениях из линейной алгебры

и дифференциального исчисления. При этом автономность процесса заключается в решении задачи при минимальном участии человека. В частности, метод градиентного спуска является основой метода обратного распространения ошибки, который используется для обучения нейросетевых структур данного вида. Концептуальная модель системы, на оценивание и прогнозирование состояний и процессов функционирования которой направлены разрабатываемые методы, включает элементы $\{V_t\}_{t\in T}$, где $V_t = V_t(p_1, \dots, p_n)$ – кортеж, включающий n переменных $p_i, i = 1 \dots n$, описывающих состояние системы в момент времени t. Множество переменных $P = P_d \cup P_e$, где P_d – характеристики (параметры) системы, пригодные для технического диагностирования системы [17], P_e – параметры внешнего окружения системы и внешних воздействий. Множество рассматриваемых кортежей включает следующие выборки: обучающую выборку признаков $X_{ij}, = 1 \dots N$; обучающую выборку меток классов состояний сложных объектов $Y_{ij} = 1 \dots N$; тестовую выборку признаков $Z_{ij} = 1 \dots M$.

При этом предварительно выполняется предобработка входных данных, которая осуществляется при помощи метода minmax-нормализации и описывается следующей формулой:

$$G: \mathrm{X_{ij}} \to \frac{\mathrm{X_{ij}} - \min\limits_{k=1..N} \mathrm{X_{ik}}}{\max\limits_{k=1..N} \mathrm{X_{ik}} - \min\limits_{k=1..N} \mathrm{X_{ik}}}$$

Решаемая научная задача по разработке методов оценивания состояния сложных объектов и процессов описывается функцией F, сопоставляющей следующие входные и выходные данные:

$$F: \left\{\left(p_1^{(t_i)}, \dots p_n^{(t_i)}, t_i\right)\right\}_{i\in\mathbb{N}, t\in T_0, T_0\subseteq T} \to \left\{\left(m_1^{(t_i)}, \dots m_N^{(t_i)}, pr_i\right)\right\}_{i\in\mathbb{N}, t\in T_0, T_0\subseteq T}, \quad (3.1.3)$$

где $m_j^{(t_i)}$ – значение целевого показателя системы в момент времени t_0. Величина pr_i – скалярная (либо векторная) априорно формируемая вероятностная оценка соответствия результирующего вектора $(m_1^{(t_i)}, \dots m_r^{(t_i)})$ фактическому состоянию системы в момент времени t_0. В рамках решаемой задачи набор представленных параметров $p_1^{(t_i)}, \dots p_n^{(t_i)}$ представляет собой вектор признаков, характеризующих состояние системы в момент времени t_0, а параметры $m_1^{(t_i)}, \dots m_N^{(t_i)}, pr_i$ задают вероятностную принадлежность к возможным классам состояния системы в этот момент времени. Тем самым функция F предназначена для классификации объекта или процесса в соответствии с его описательными параметрами $p_1^{(t_i)}, \dots p_n^{(t_i)}$.

Каждый из показателей $m_j^{(t_i)}$ может принимать бинарное или числовое значение. Примерами показателей со значениями $m_j^{(t_i)}$ являются:

— наличие или отсутствие функциональных неисправностей в системе в момент времени t_0;

— наличие или отсутствия в системе дефектов материалов физических конструкций системы в момент времени t_0;

— наличие или отсутствие непреднамеренных и преднамеренных воздействий на систему в момент времени t_0.

3) Автономное прогнозирование состояния СлОП

Автономное прогнозирование состояний сложных объектов и процессов основывается на следующих математических соотношениях. Обучающие данные, характеризующие СлОП, заданы множеством описаний (экземпляров), представляющих собой упорядоченную во времени последовательность векторов характеристик СлОП:

$$\mathbf{X} = \{X_t\},\ t = 1, \ldots, T, \tag{3.1.3}$$

где T – длина временного ряда (количество экземпляров).

Каждый экземпляр содержит числовой вектор характеристик состояния СлОП:

$$X_t = \{x_{mt}\},\ m = 1, \ldots, M, \tag{3.1.4}$$

где M – фиксированная длина вектора (количество характеристик).

Для обучаемой модели АПССОП задается параметр длины исторической последовательности данных для прогнозирования L – количество упорядоченных во времени экземпляров $\{X_t, \ldots, X_{t+L}\}$, необходимых для прогнозирования последующего экземпляра X_{t+L+1}.

На основе обучающих данных при помощи функции G производится формирование генератора временных рядов, представляющего собой формат данных, состоящих из пары (X^p, X^d):

$$G\colon \mathbf{X} \rightarrow (\mathbf{X}^p, \mathbf{X}^d)$$
$$\mathbf{X}^p = \left\{X_1^p, \ldots, X_{T-L-1}^p\right\},\ X_t^p = \{X_t, \ldots, X_{t+L}\},\ X_t^p \in \mathbf{X}^p \tag{3.1.5}$$
$$\mathbf{X}^d = \left\{X_1^d, \ldots, X_{T-L-1}^d\right\},\ X_t^d = X_{t+L+1},\ X_t^d \in \mathbf{X}^d,$$

где $\mathbf{X}^p$ – множество пакетов, упорядоченных во времени исторических экземпляров для прогнозирования, $\mathbf{X}^d$ – множество целевых экземпляров для прогнозирования. Целевые экземпляры представляют собой ожидаемый результат прогнозирования: любому пакету X_i^d, соответствует последующий во времени экземпляр X_i^d.

Множество рассматриваемых кортежей включает следующие выборки: обучающую выборку $\boldsymbol{X}_r = \{X_1, \dots, X_t\}$, $t < T$, и тестовую выборку $\boldsymbol{X}_e = \{X_{t+1}, \dots, X_T\}$. Выборки строятся таким образом, что обучающая выборка содержит временной ряд данных, предшествующий временному ряду в тестовой выборке. Соотношение объемов обучающей и тестовой выборок является опциональным. Классическим способом разделения исходной выборки является разделение с соотношением 8:2 (80% выборки является обучающей, 20% – тестовой) или 9:1 (90% выборки является обучающей, 10% – тестовой).

Прогноз осуществляется таким образом, что для предсказания параметров событий в следующий момент времени необходимо проанализировать предшествующие события во временном окне размерностью L. Задача прогнозирования состояний сложных объектов и процессов может быть описана при помощи функции ϕ, сопоставляющей следующие входные и выходные данные:

$$\begin{gathered} \phi\colon\ \{X_t, \dots, X_{t+L}\} \to X_{t+L+1}, \\ \text{или } \phi\colon\ X_t^p \to X_t^d, t \in T. \end{gathered} \tag{3.1.6}$$

Для прогнозирования используются блоки долгой краткосрочной памяти (перев. на англ. Long Short-Term Memory, аббр. LSTM). Блоки LSTM используются для кодирования признаков исторических экземпляров X^h, поступающих на вход нейронной сети. LSTM-блоки содержат вентили (гейты) i_t, f_t, o_t, реализованные в виде логистических функций, для контроля потоков информации на входах и на выходах данных блоков, а также вектор состояний c_t в качестве внутренней памяти, которая обновляется с использованием текущего и предыдущего состояний:

$$LSTM = \begin{cases} i_t = \sigma_g(W_{xi}X_t^h + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t = \sigma_g\left(W_{xf}X_t^h + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \\ o_t = \sigma_g\left(W_{xo}X_t^h + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o\right) \\ c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c\left(W_{xc}X_t^h + W_{hc}h_{t-1} + b_c\right) \\ h_t = o_t \circ \sigma_t(c_t) \end{cases}, \tag{3.1.7}$$

где h_t –вектор текущего скрытого состояния;

h_{t-1} – вектор предыдущего скрытого состояния,

i_t – вектор входного вентиля (вес получения новой информации);

f_t – вектор вентиля забывания (вес запоминания старой информации);

o_t – вектор выходного вентиля (кандидат на выход);

W_{jk} – матрица весов для преобразования вектора j в компоненту вектора k;

b_j – вектор смещения вектора j;

σ_g – функция активации на основе сигмоиды;

σ_c – функция активации на основе гиперболического тангенса;

$\circ$ – произведение Адамара (покомпонентное произведение двух матриц).

Обозначим нейронную сеть из N слоев как $LSTM^{[N]}$, тогда ее скрытые состояния вычисляются рекуррентно и последнее скрытое состояние является выходным вектором:

$$\phi = \begin{cases} h_t^{[1]} = LSTM^{[1]}\left(X_t^d\right) \\ h_t^{[2]} = LSTM^{[2]}\left(h_t^{[1]}\right) \\ \dots \\ X_t^d = h_t^{[N]} = LSTM^{[N]}\left(h_t^{[N-1]}\right) \end{cases}, \quad (3.1.8)$$

где $[n], n = 1, \dots, N$ – порядковый номер слоя нейронной сети.

Таким образом,

$$X_t^d = \phi\left(X_t^h\right).$$

Выходными данными являются прогнозируемые состояния СлОП за заданный отрезок времени Δ:

$$\Phi: \ \mathbf{X} \to \mathbf{X}^*, \quad \mathbf{X}^* = \{X_{T+1}^*, \dots, X_{T+\Delta}^*\}, \quad (3.1.9)$$

где **X*** – множество прогнозируемых векторов состояний СлОП.

3.2 Алгоритмы компонента

3.2.1 Алгоритм ПОСНД

Алгоритм предназначен для работы с подаваемыми на вход наборами данных и включает в себя коррекцию типов данных, а также устранение неполноты и мультиколлинеарности данных о сложных объектах или процессах.

Алгоритм является вспомогательным для других алгоритмов, реализует функции сильного ИИ в части улучшения качества предоставляемых данных и может быть задействован на подготовительной стадии оценивания и прогнозирования состояния.

Блок-схема алгоритма представлена на рис. 3.2.1.

На первом шаге алгоритма осуществляется анализ корректности типов данных для каждого признака отдельно. Корректность типа данных оценивается на основе количества уникальных значений признака (блок «Уникальные значения» на рис. 3.2.1), обнаруженных последовательностей (блок «Последовательности» на рис. 3.2.1), анализа названий признаков (блок «Названия признаков» на рис. 3.2.1), а также на основе обоснованности использования дробных значений, в случае их наличия (блок «Дробные значения» на рис. 3.2.1). Вердикт о корректности типа данных выносится алгоритмами оценки отдельно, в то время как итоговое решение принимается на основе всех четырех

оценок с учетом их весовых коэффициентов. Незаполненные поля игнорируются. Информация о корректировке типов данных сохраняется. И т.к. решение по каждому признаку принимается независимо от решений по остальным, то обработка каждого признака происходит в параллельном режиме.

Второй шаг алгоритма направлен на устранение неполноты данных на основе статистического усреднения данных в рамках отдельных кластеров. Вначале данные разбиваются на отдельные кластеры на основе таких подходов, как K-means, Mini Batch K-means, Mean-shift, OPTICS, Bisecting K-means. Метод кластеризации может быть задан вручную оператором или выбран алгоритмом автоматически в соответствии с заданным по умолчанию значением. В дальнейшем, каждый признак каждого кластера обрабатывается параллельно. Устранение неполноты данных представляет собой автоматическое заполнение пропущенных значений признаков на основе среднего, медианного или регрессионного значения по кластеру, а также на основе замены специальным значением, указанным оператором (например, EMPTY). Данные о произведенных преобразованиях сохраняются.

На третьем шаге осуществляется устранение мультиколлинеарности данных за счет удаления признаков низкой информативности. Анализ информативности признаков осуществляется параллельно для каждой пары признаков. При этом в зависимости от типов признаков (только численные, только категориальные, численные и категориальные), информативность вычисляется с помощью критерия корреляции Пирсона или Спирмена, а также методом дисперсионного анализа (Крамер). По умолчанию, логика работы алгоритма, следующая: если признаки категориальные, то используется дисперсионный анализ Крамера, численные – корреляция Пирсона, один численный, другой категориальный (или типы неизвестны) – Спирмена. Если необходимо, для пользователя предусмотрена возможность изменения задействованных критериев. Порог информативности также может быть задан пользователем, в противном случае алгоритм использует значение по умолчанию, равное 0.70. Итогом работы алгоритма являются плоские данные в табличной форме, в которых указаны корректные типы данных, заполнены пропущенные значения, а также удалены неинформативные признаки.

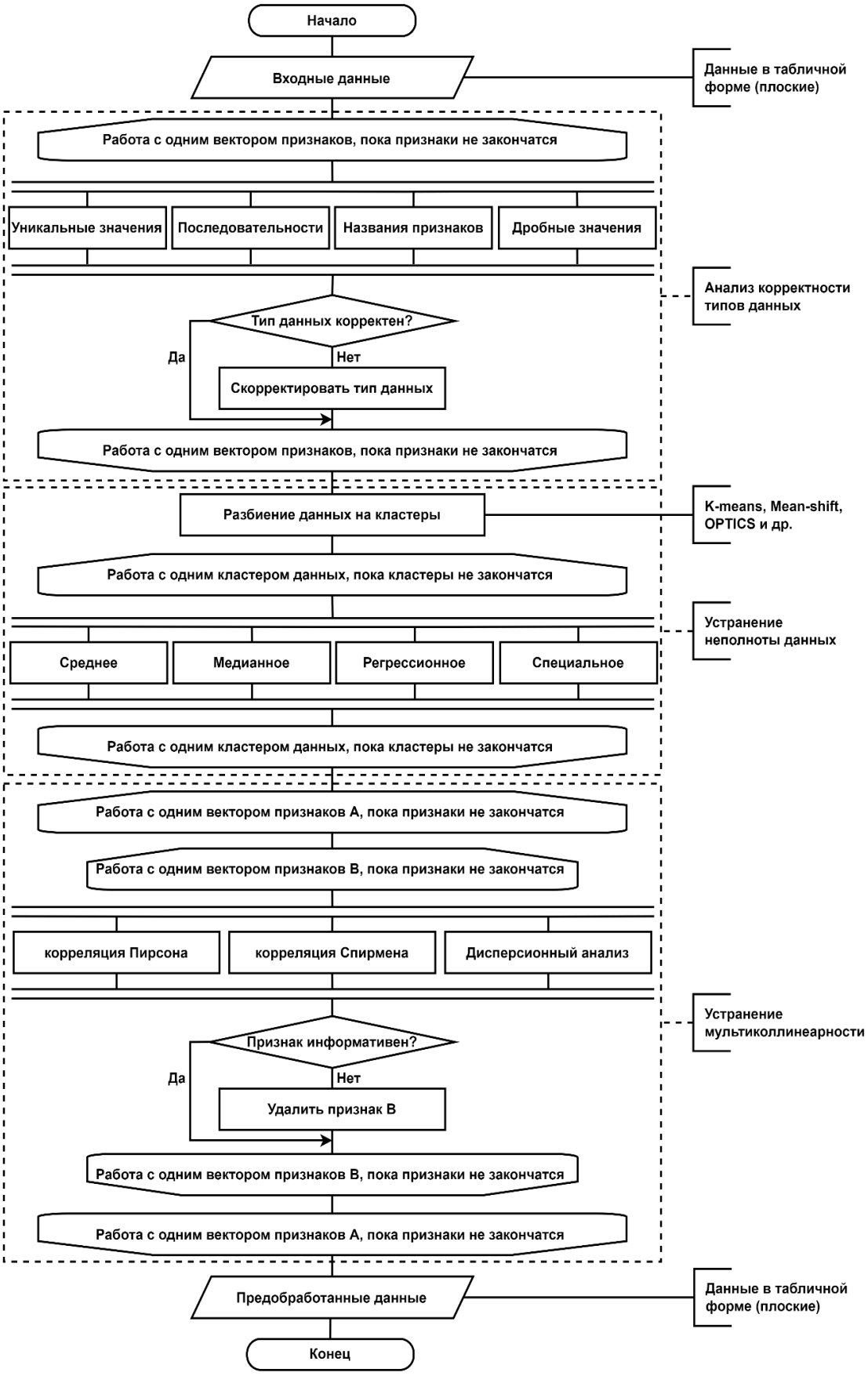

Рисунок 3.2.1 – Блок-схема алгоритма ПОСНД

3.2.2 Алгоритм ИЗСНД

Алгоритм предназначен для извлечения фрагментов знаний, имеющихся в данных о сложных объектах, в виде ассоциативных правил (вида «ЕСЛИ <посылка>, ТО <следствие>»), содержащих в правой части (следствии) метку класса (англ. class association rules). Блок-схема алгоритма ИЗСНД представлена на рис. 3.2.2.

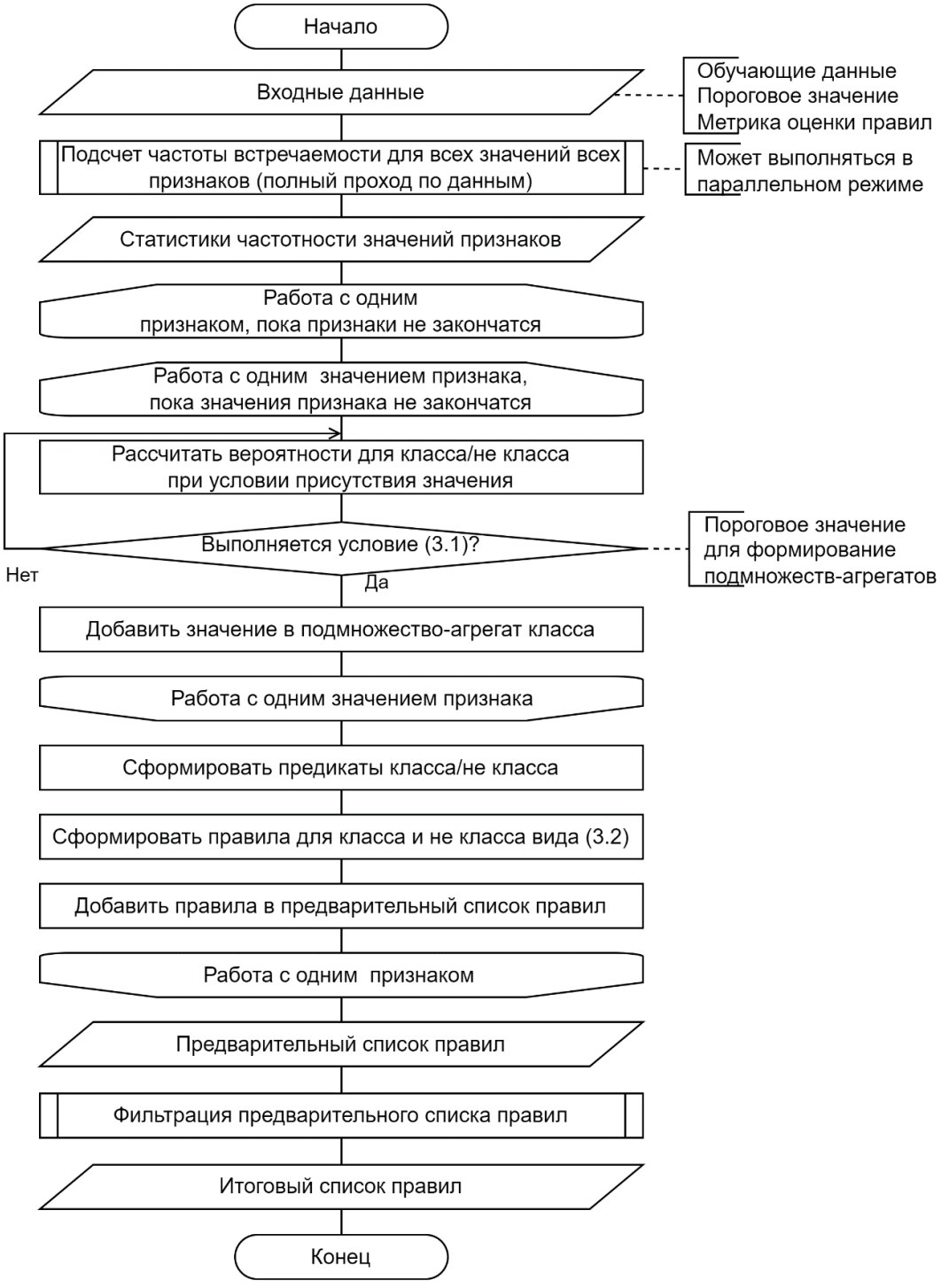

Рисунок 3.2.2 – Блок-схема алгоритма ИЗСНД

На первом шаге алгоритма вычисляется частота встречаемости всех уникальных дискретных значений всех признаков и условные вероятности для классов на их основе. Это может быть выполнено за один проход по данным. Далее в циклах выполняется проход по всем атрибутам и всем уникальным значениям этих признаков, подсчет условных вероятностей для этих значений и проверка условия (3.1.1). Эти шаги могут выполняться в параллельном режиме. Значения, удовлетворяющие условию (3.1.1), добавляются в подмножества-агрегаты, для которых в конце каждой итерации внешнего цикла формируются предикаты и правила класса вида (3.1.2) на их основе. В результате выполнения циклов формируется предварительный список правил. Далее для каждого правила из предварительного списка рассчитывается метрика оценки силы правила и выполняется их фильтрация на основании значения метрики. Для этого не требуется дополнительный проход по данным, так как можно использовать статистики, рассчитанные на первом шаге алгоритма. Результатом работы алгоритма будет являться множество ассоциативных правил класса, отражающих наиболее сильные связи параметров описаний СлОП с его состоянием.

3.2.3 Алгоритм АОТССОП

Алгоритм предназначен для автономного оценивания текущего состояния сложных объектов и процессов, включающего наличие или отсутствие в текущий момент определенного вида функциональных неисправностей, дефектов и атакующих воздействий, свойственных целевой системе. Алгоритм строится на основе глубокой нейронной сети прямого распространения сигнала, обучение которой производится на объединенном наборе данных, полученном из открытых источников и включающем данные о функционировании нескольких видов индустриальных систем.

Блок-схема алгоритма представлена на рис. 3.2.3.

Начальный этап включает задание величины среднеквадратичной ошибки (СКО, *MSE*) и числа эпох обучения НС. Далее выполняется разбиение исходного набора данных в заданном пользователем соотношении, одна из образовавшихся подвыборок используется в качестве обучающей (X_{ij} – признаки, Y_{ij} – метки классов) для настройки весовых коэффициентов НС, другая часть – для тестирования. В процессе обучения выполняется корректировка весовых коэффициентов нейронной сети до тех пор, пока не будет достигнута требуемая величина СКО или число эпох обучения не превысит заданного значения. После этого вычисляется показатель точности (*acc*) НС на элементах тестовой

выборки (Z_{ij}), на основании этого значения принимается решение о повторном обучении нейросетевой модели или ее применении для оценки состояния объекта.

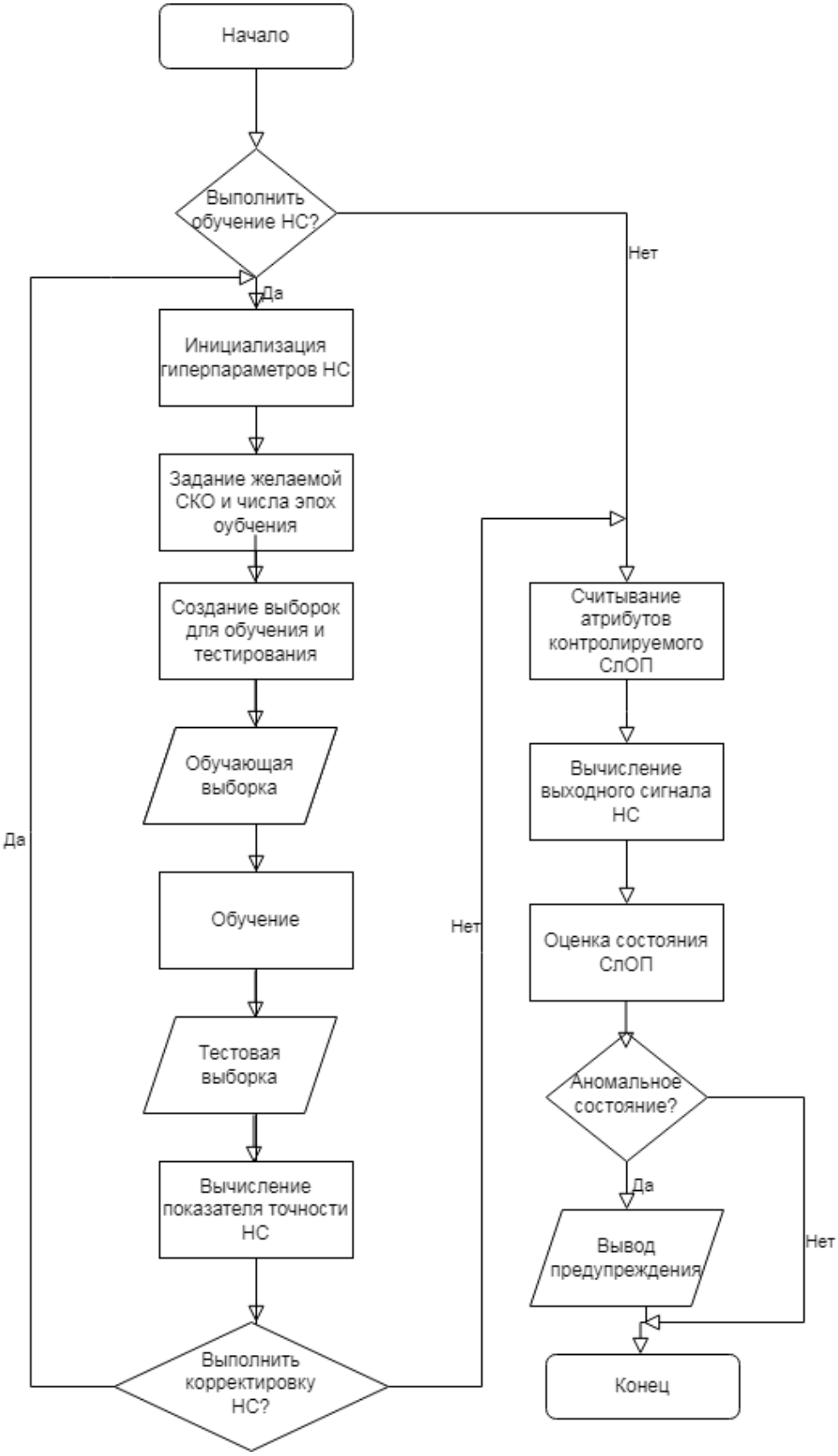

Рисунок 3.2.3 – Блок-схема алгоритма АОТССОП

После обучения НС выполняется считывание атрибутов контролируемого сложного объекта, что подразумевает мониторинг параметров его состояния в дискретные моменты времени. На основе этих параметров формируется входной сигнал для НС. Далее выполняется построение выходного сигнала НС, с этой целью последовательно применяются композиции линейных и нелинейных преобразований над входным сигналом. Выходное значение НС рассматривается как признак наличия или отсутствия аномального состояния у контролируемого объекта.

3.2.4 Алгоритм АПССОП

Алгоритм предназначен для автономного прогнозирования состояний сложных объектов и процессов, включающего наличие или отсутствие в некоторый момент времени в будущем определенного вида функциональных неисправностей, дефектов и влияния как непреднамеренных воздействий, так и преднамеренных (атакующих) воздействий, свойственных целевой системе.

Алгоритм строится на основе рекуррентной глубокой нейронной сети, обучение которой производится на семействе наборов данных, полученных из открытых источников и включающих данные о функционировании нескольких видов индустриальных систем.

Алгоритм АПССОП реализует функции сильного ИИ в части выполнения автономного прогнозирования состояний сложных объектов и процессов. Наличие большого числа гиперпараметров и настраиваемых весов, свойственных для глубоких нейронных сетей, позволяет с достаточно высоким качеством выявлять закономерности между признаками прогнозируемого состояния системы и ожидаемой меткой его класса.

Блок-схема алгоритма представлена на рис. 3.2.4.

Начало реализации алгоритма автономного прогнозирования состояний сложных объектов и процессов состоит в чтении следующих входных данных: упорядоченную во времени последовательность векторов характеристик СлОП (X) и параметра длины исторической последовательности данных для прогнозирования (L).

Далее производится формирование генератора временных рядов (3.1.6). При необходимости обучить модель прогнозирования на входных данных осуществляется построение выборок – обучающей ($\boldsymbol{X}r$) и тестовой ($\boldsymbol{X}e$) – таким образом, что обучающая выборка содержит временной ряд данных, предшествующий временному ряду в тестовой выборке. Разбиение исходного набора выполняется в соответствии с заданным пользователем соотношением.

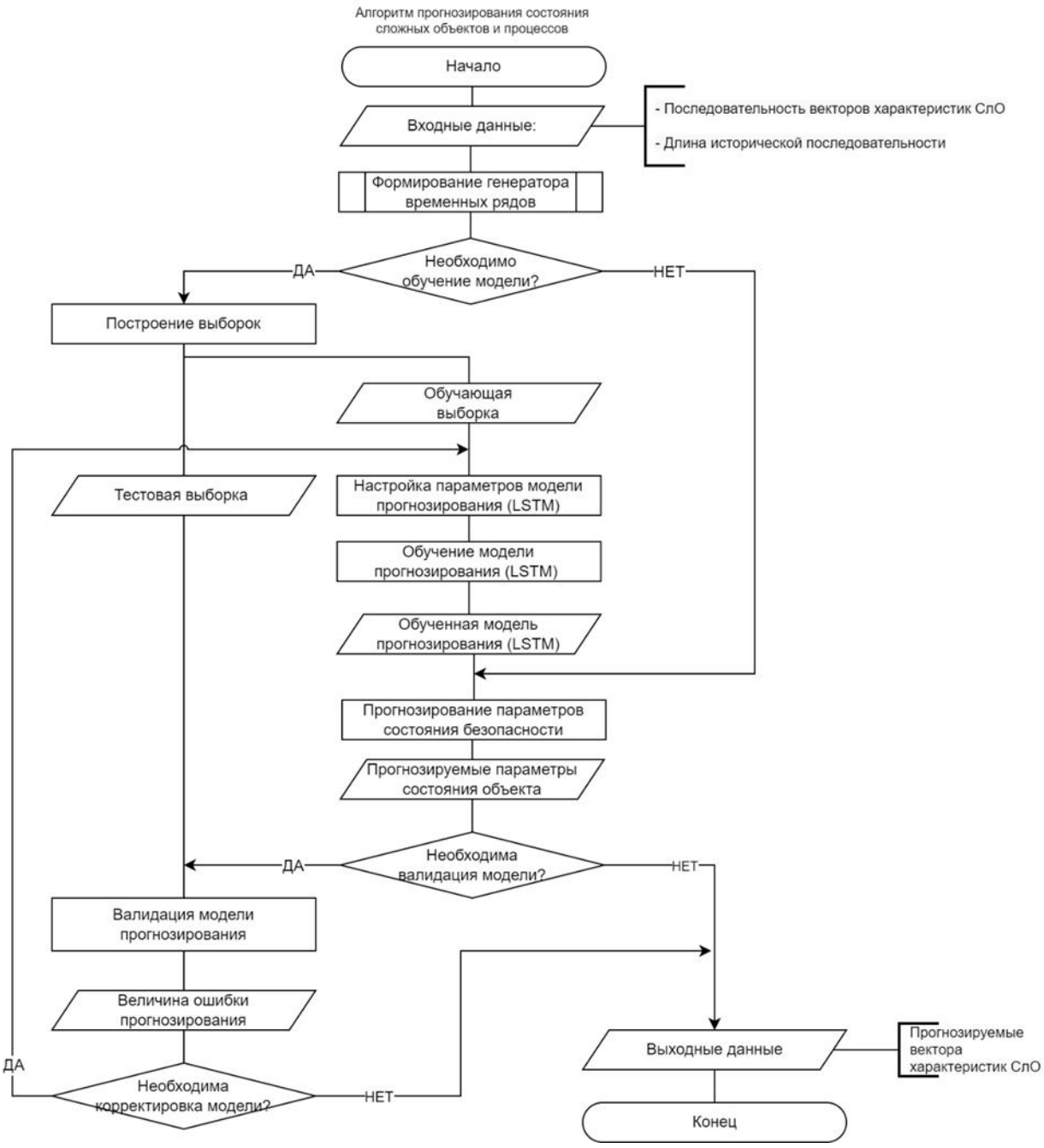

Рисунок 3.2.4 – Блок-схема алгоритма АПССОП

На основе обучающей выборки производятся последовательно настройка параметров модели прогнозирования и ее обучение. В качестве интеллектуальной модели прогнозирования выбрана рекуррентная нейронная сеть (перев. на англ. Recurrent Neural Network, аббр. RNN), представленная долгой краткосрочной памятью (перев. на англ. Long Short-Term Memory, аббр. LSTM). Сеть LSTM является разновидностью рекуррентных нейронных сетей и позволяет хорошо анализировать данные в том случае, когда важные события разделены временными лагами неопределенной продолжительности.

Модель прогнозирования обучается путем поиска закономерностей в векторах характеристик состояний СлОП для заданного временного окна и прогнозированием вектора значений в последующий момент времени (3.1.7).

Обученная модель прогнозирования LSTM применяется затем для прогнозирования параметров состояния (3.1.8). Далее, при необходимости, выполняется валидация модели прогнозирования на тестовых данных, результатом которой является величина ошибки прогнозирования, представленная кортежем <*mse, mae*>, где *mse* – среднеквадратичная ошибка, *mae* – средняя абсолютная ошибка. После этого принимается решение о необходимости корректировки модели прогнозирования, и при положительном решении заново происходит настройка параметров модели прогнозирования. Результатом выполнения данного алгоритма является набор прогнозируемых характеристик состояний объекта.

3.3 Структура компонента

Компонент состоит из модулей, каждый из которых реализует описанные ранее алгоритмы, а именно ПОСНД, ИЗСНД, АОТССОП И АПССОП (см. раздел 3.2).

1) Модуль ПОСНД состоит из следующих программных элементов (реализованных с помощью программных классов), обладающих собственным целевым предназначением (см. рис. 3.3.1):

— *CheckDataTypes* – класс для коррекции типов данных;

— *ClusterFilling* – класс устранения неполноты данных;

— *Informativity* – класс для анализа информативности признаков;

— *MultiCollinear* – класс для устранения мультиколлинеарности;

— *Data* – класс для хранения данных и результатов их обработки;

— *PrintLog* – класс для журналирования работы модуля и представления результатов пользователю.

2) Модуль ИЗСНД состоит из следующих программных элементов (реализованных с помощью программных классов), обладающих собственным целевым предназначением (см. рис. 3.3.2):

— *Predicate* – класс для представления логических предикатов;

— *IzdapAlgo* – класс, реализующий логику алгоритма ИЗДАП.

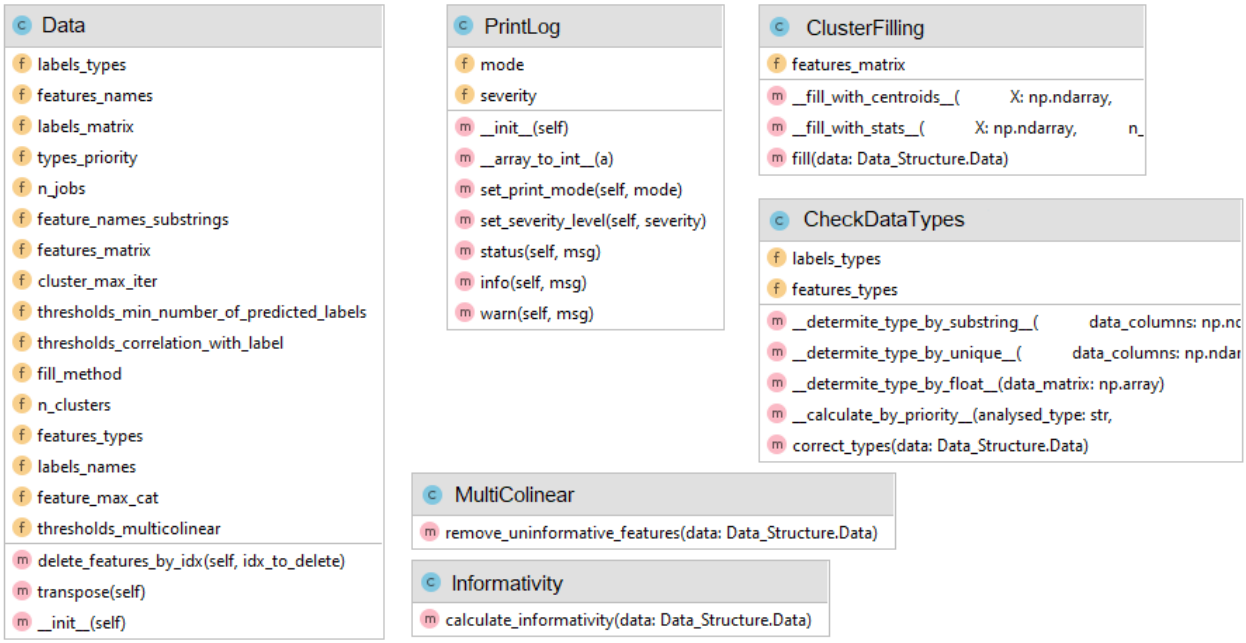

Рисунок 3.3.1 – Диаграмма классов модуля ПОСНД

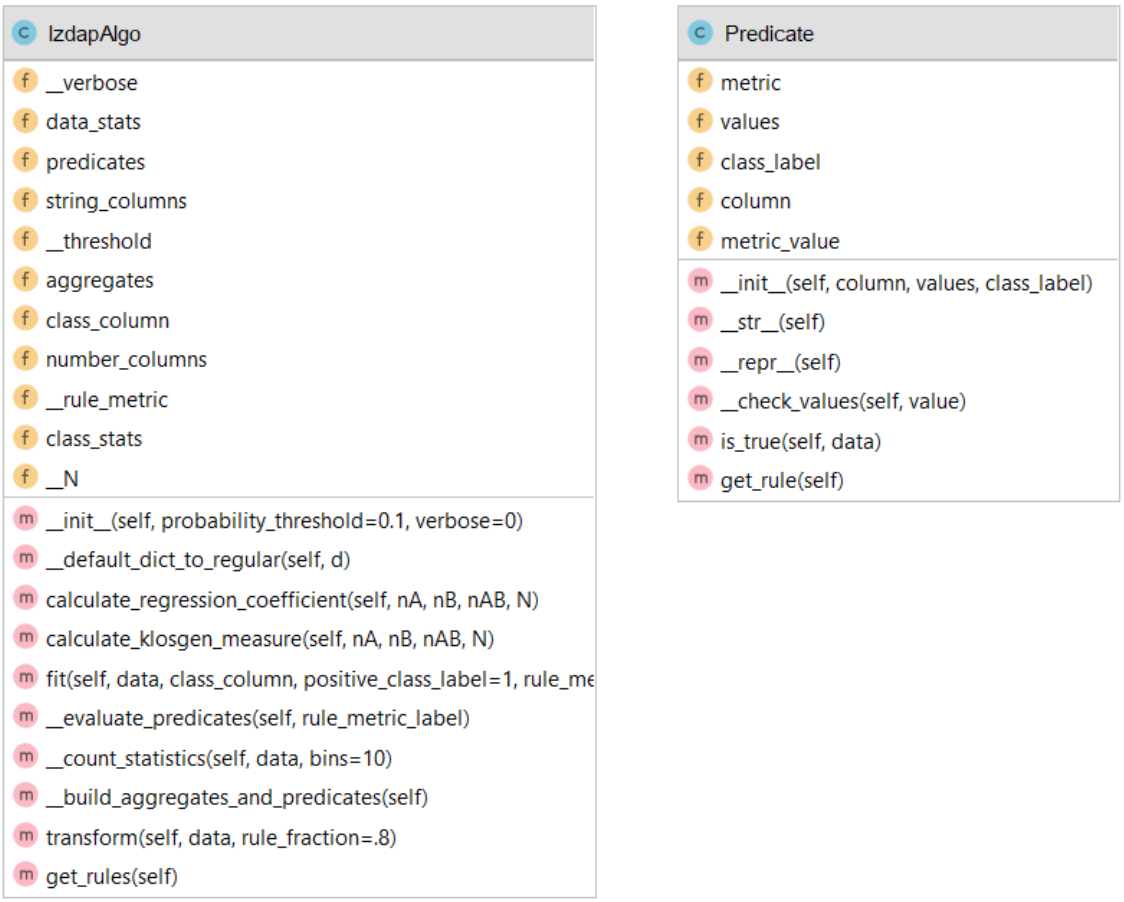

Рисунок 3.3.2 – Диаграмма классов модуля ИЗСНД

3) Модуль АОТССОП состоит из следующих программных элементов (реализованных с помощью программных классов), обладающих собственным целевым предназначением (см. рис. 3.3.3):

— *SAIClassifier* – элемент для селективной классификации, позволяющий выбрать один из следующих их типов: дерево решений, наивный Байес, логистическая регрессия и искусственная нейронная сеть;

— *FormatDetector* – элемент для определения формата входных данных;

— *DataLoader* – элемент для загрузки набора входных данных;

— *ClsEstimator* – элемент для оценивания эффективности классификаторов.

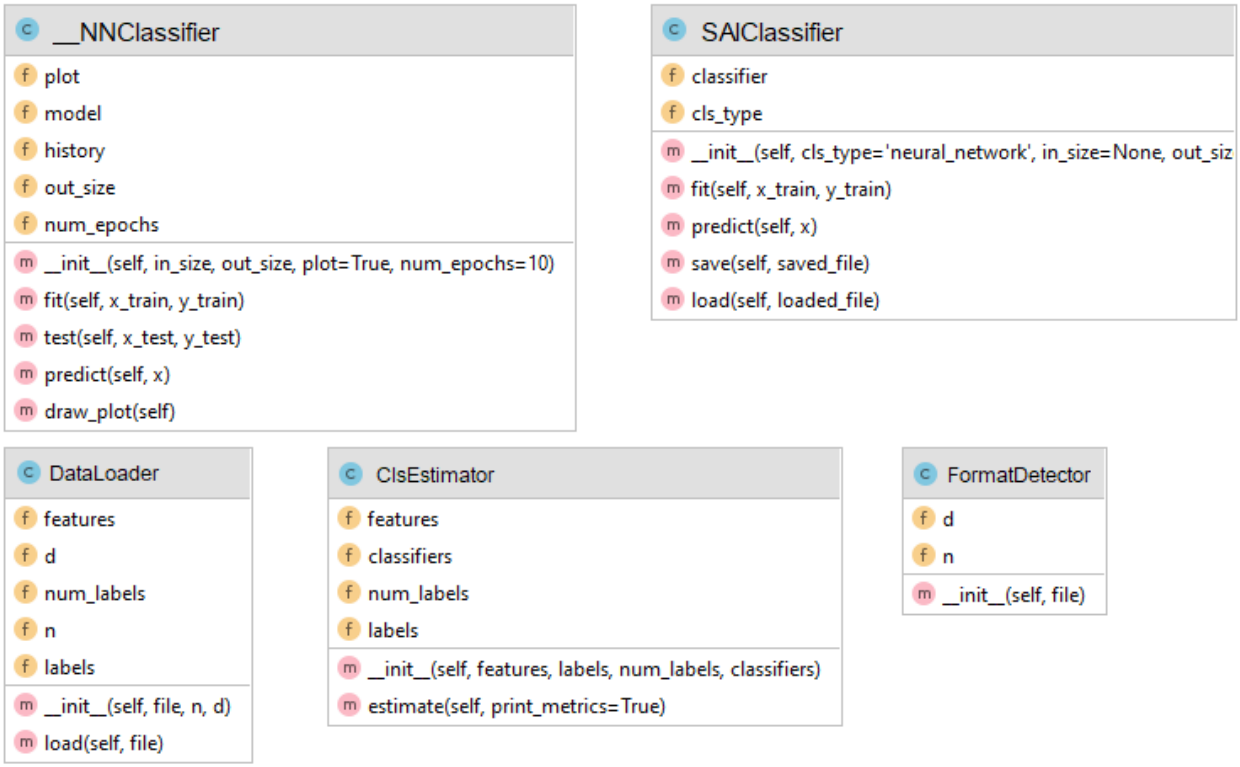

Рисунок 3.3.3 – Диаграмма классов модуля АОТССОП

4) Модуль АПССОП состоит из следующих программных элементов (реализованных с помощью программных классов), обладающих собственным целевым предназначением (см. рис. 3.3.4).

— *AIForecaster* – элемент для прогнозирования СлОП. Атрибуты класса: *model* – нейросетевая модель прогнозирования (объект keras.models.Sequential); *time_window_length* – длина исторической последовательности (целое число); *n_features* – количество признаков (целое число); *model_path* – путь для внешнего сохранения модели прогнозирования (текстовая строка); *epochs* – количество эпох для обучения модели

прогнозирования (целое число); *batch_size* – размер пакетов для обучения (целое число); *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 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).

— *DataScaler* – элемент для нормализации данных. Атрибуты класса: *scaler* – объект модели нормализации (объект MinMaxScaler); *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.

— *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).

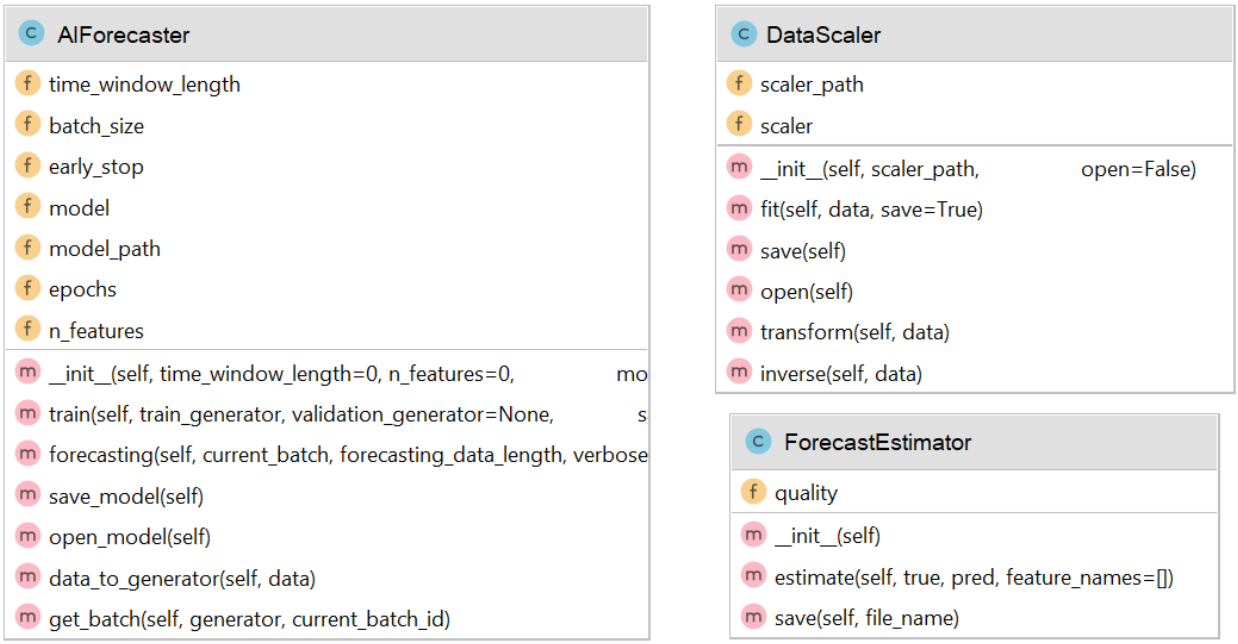

Рисунок 3.3.4 – Диаграмма классов модуля АПССОП

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Техническими средствами являются электронно-вычислительные машины и устройства, которые используются при работе программы, должны иметь минимально необходимые характеристики, представленные в табл. 4.1. В качестве возможных операционных систем подходят любые из семейства Linux и Windows, совместимые с Python версии 3.7 и выше.

Таблица 4.1 – 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, МГц**	**Оперативная память, Гб**	**Дисковая память, Гб**
Рабочая станция	1 x 8	3.8	1 x 3584	1480	32	2000

5. ВЫЗОВ И ЗАГРУЗКА

5.1 Базовые функции компонента

1) Модуль ПОСНД

Начало работы с набором данных:

```
# создание объекта данных
data = Data()
# передача пути к набору данных
titanic_path = '../datasets/titanic.csv'
# считывание данных в pd.Dataframe
titanic = pd.read_csv(titanic_path)
# сохранение наименований признаков данных
data.features_names = ["PassengerId", "Pclass", "Age", "SibSp", "Parch"]
# сохранение наименований меток данных
data.labels_names = ["Survived", "Fare"]
# сохранение матрицы признаков
data.features_matrix = np.array(titanic[data.features_names])
# сохранение матрицы меток
data.labels_matrix = np.array(titanic[data.labels_names])
# сохранение типов данных признаков
data.features_types = ["cat", "cat", "num", None, None]
# сохранение типов данных меток
data.labels_types = ["cat", None]
```

Подключение журналирования:

```
# подключение журналирования (True)
__Verbose__.PrintLog.instance().set_print_mode(True)
# задание степени подробности журналирования (status)
__Verbose__.PrintLog.instance().set_severity_level("status")
```

Запуск алгоритмов модуля:

```
# устранение некорректности типов данных
CheckDataTypes.CheckDataTypes.correct_types(data)
# устранение неполноты данных
ClusterFilling.ClusterFilling.fill(data)
# анализ информативности данных
Informativity.Informativity.calculate_informativity(data)
# устранение мультиколинеарности данных
Multicolinear.MultiCollinear.remove_uninformative_features(data)
```

2) 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)
```

3) Модуль АОТССОП

Модель инициализируется путем создания объекта SAIClassifier, которому передается тип классификатора, количество нейронов на распределительном и выходном слоях:

```
SAIClassifier('neural_network', 10, 1)
```

Объект FormatDetector создается путем вызова соответствующего конструктора с наименованием файла, в котором содержится обрабатываемый набор данных:

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

Объект DataLoader создается путем вызова соответствующего конструктора с наименованием файла, в котором содержится обрабатываемый набор данных, количество полей в записи, разделитель полей в записи:

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

Объект ClsEstimator создается путем вызова соответствующего конструктора с набором данных (векторы признаков, метки объектов) и типом классификатора:

```
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]
```

4) Модуль АПССОП

Новая модель прогнозирования инициализируется путем создания объекта AIForecaster, которому передается количество эпох обучения (n_epochs), длина исторической последовательности для прогнозирования (time_window_length), количество признаков данных (n_features) и путь к внешнему файлу модели для сохранения (model_path):

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

Существующая модель прогнозирования инициализируется путем создания объекта AIForecaster, которому передается путь к внешнему файлу модели для загрузки (model_path) и положительное булевое значения для команды загрузки модели (open):

```
forecasting_model = AIForecaster (model_path = ”…/forecasting_model, open = True)
```

Формирование генератора временных рядов осуществляется путем вызова метода data_to_generator объекта AIForecaster, которому передается многомерный массив данных (trainX):

```
generatorX = forecasting_model.data_to_generator(trainX)
```

Обучение модели объекта AIForecaster:

```
forecasting_model.train(generatorX)
```

Прогнозирование данных осуществляется путем вызова метода forecasting объекта AIForecaster, которому передаются пакет данных для прогнозирования (current_batch) и длина прогнозируемой последовательности (forecasting_data_length):

```
prediction = forecasting_model.forecasting(current_batch = batch,
forecasting_data_length = 1000)
```

Новая модель нормализации данных инициализируется путем создания объекта DataScaler, которому передается путь к внешнему файлу модели для сохранения (scaler_path):

```
scaler = DataScaler(scaler_path = ”scaler.pkl”)
```

Существующая модель нормализации данных инициализируется путем создания объекта DataScaler, которому передается путь к внешнему файлу модели для загрузки (scaler_path) и положительное булевое значения для команды загрузки модели (open):

```
scaler = DataScaler(scaler_path = “scaler.pkl”, open=True)
```

Обучение модели нормализации данных осуществляется путем вызова метода fit объекта DataScaler, которому передается многомерный массив данных (trainX):

```
scaler.fit(data = trainX)
```

Нормализация данных осуществляется путем вызова метода transform объекта DataScaler, которому передается многомерный массив данных (trainX):

```
scaled_trainX = scaler.transform(trainX)
```

Оценка эффективности прогнозирования осуществляется путем создания объекта ForecastEstimator и выхода метода estimate, которому передаются фактические значения признаков данных (true) и прогнозируемые значения признаков данных (pred):

```
estimator = ForecastEstimator()
quality = estimator.estimate(true = scaled_testX, pred = prediction)
```

6. ВХОДНЫЕ ДАННЫЕ

6.1 Состав и структура входных данных

1) Входные данные модуля ПОСНД сведены в единую табл. 6.1.1.

Таблица – 6.1.1. Входные данные модуля ПОСНД

Наименование данных	**Обозначение**	**Структура данных**	**Способ ввода данных**	**Ограничения**
Матрица признаков	F^M	Двумерный массив, состоящий из	В виде аргумента *features_matrix*	Только численные значения

		объектов и их признаков *(numpy.array)*		
Типы данных признаков	F^T	Одномерный массив, длина которого соответствует количеству признаков *(numpy.array)*	В виде аргумента *features_types*	Возможные значения элементов *["num", "cat", None]*
Наименования признаков	F^S	Одномерный массив, длина которого соответствует количеству признаков *(numpy.array)*	В виде аргумента *features_names*	Только строковые значения
Матрица меток	L^R	Двумерный массив, состоящий из объектов и их меток *(numpy.array)*	В виде аргумента *labels_matrix*	Только численные значения
Типы данных меток	L^T	Одномерный массив, длина которого соответствует количеству меток*(numpy.array)*	В виде аргумента *labels_types*	Возможные значения элементов *["num", "cat", None]*
Наименования меток	L^S	Одномерный массив, длина которого соответствует количеству меток*(numpy.array)*	В виде аргумента *labels_names*	Только строковые значения

2) Входные данные модуля ИЗСНД сведены в единую табл. 6.1.2.

Таблица 6.1.2 – Входные данные модуля ИЗСНД

Наименование данных	Обозначение	Структура данных	Способ ввода данных	Ограничения
Вероятность, задающая порог отсечения значений для построения предикатов	δ	Вещественное число	В виде аргумента *probability_threshold*	$\delta \in [0, 1]$
Метрика оценки правил (название в строковом формате)	μ	Строковое значение	В виде аргумента *rule_metric*	Строка из фиксированного списка реализованных метрик
Признаки и метки обучающей выборки	$X_i, i=1..N$, $Y_i, i=1..N$	Многомерный массив	В виде аргумента *data*	Объем массива данных ограничен объемом оперативной и долговременной памяти
Метка положительного класса	Y_j	Вещественное число или строковое значение	В виде аргумента *positive_class_label*	Метка должна присутствовать в обучающей выборке
Имя признака, содержащего метки обучающей выборки	–	Строковое значение	В виде аргумента *class_column*	Признак должен присутствовать в обучающей выборке

3) Входные данные модуля АОТССОП сведены в единую табл. 6.1.3

Таблица 6.1.3 – Входные данные модуля АОТССОП

Наименование данных	Обозначение	Структура данных	Способ ввода данных	Ограничения
Признаки обучающей выборки	–	Двумерный массив	В виде аргумента *x_train*	Нет
Метки обучающей выборки	–	Одномерный массив	В виде аргумента *y_train*	Нет

Признаки тестовой выборки	–	Двумерный массив	В виде аргумента *x_test*	Нет
Метки тестовой выборки	–	Одномерный массив	В виде аргумента *y_test*	Нет
Объект для определения класса с помощью глубокой нейронной сети	–	Двумерный массив	В виде аргумента *x*	Нет
Файл для сериализации	–	keras и pickle	В виде аргумента *saved_file* с путем к бинарному файлу	Существование пути к файлу с моделью
Файл для десериализации	–	keras и pickle	В виде аргумента *loaded_file* с путем к бинарному файлу	Существование пути к файлу с моделью; Корректность формата файла
Файл для определения его типа и загрузки данных	–	Набор строк из элементов, разделенных символами ‘,’ или ‘;’ (может быть запакован с помощью gzip).	В виде аргумента *file* с путем к файлу	Существование файла с моделью; Корректность формата файла
Признаки данных сложного объекта	–	Массив с данными	В виде аргумента *features*	Нет
Метки данных сложного объекта	–	Массив с данными	В виде аргумента *labels(num_labels)*	Нет
Выбранные ранее классификаторы	–	Программный объект	В виде аргумента *classifiers*	Нет

4) Входные данные модуля АПССОП сведены в единую табл. 6.1.4

Таблица 6.1.4 – Входные данные модуля АПССОП

Наименование данных	**Обозначение**	**Структура данных**	**Способ ввода данных**	**Ограничения**
Путь к модели прогнозирования (*model_path*)	–	Текстовая строка	В виде аргумента *model_path* с путем к бинарному файлу	Не более 255 символов
Путь к модели нормализации (*scaler_path*)	–	Текстовая строка	В виде аргумента *scaler_path* с путем к бинарному файлу	Не более 255 символов
Размер временного окна при обучении, длина исторической последовательности (*time_window_length*)	*L*	Число	В виде аргумента *time_window_length*	Не более 90% от числа экземпляров исходного набора данных
Размер временного окна для прогнозирования (*forecast_len)*	Δ	Число	В виде аргумента *forecast_len*	Нет
Матрица признаков исходного набора данных (*data*)	**X**	Многомерный массив числовых значений	В виде аргумента *data*	Нет

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 передаются в виде пути к файлу, содержащему данные, нуждающиеся в предобработке. При этом пользователю необходимо отделить признаки данных от их меток в соответствии с описанием использованного набора данных.

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

3) Модуль АОТССОП: модель сериализуется во внешний файл, путь к которому передается через аргумент – saved_file, и который имеет бинарный формат, реализованный в библиотеке pickle (файл создается автоматически и не требует формирования пользователем); модель десериализуется из внешнего файла, путь к которому передается через аргумент – loader_file, и который имеет бинарный формат, реализованный в библиотеке pickle (файл создается автоматически и не требует формирования пользователем). Входной набор данных для определения типа и загрузки загружается из файла, путь к которому передается через аргумент – file, и который 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, путь к которому передается через аргумент – model_path, и который имеет бинарный формат, реализованный в библиотеке keras.models (файл создается автоматически и не требует формирования пользователем). Модель нормализации загружается из файла, путь к которому передается через аргумент – scaler_path, и который имеет бинарный формат, реализованный в библиотеке pickle (файл создается автоматически и не требует формирования пользователем).

7. ВЫХОДНЫЕ ДАННЫЕ

7.1 Состав и структура выходных данных

1) Выходные данные модуля ПОСНД сведены в единую табл. 7.1.1.

Таблица 7.1.1 – Выходные данные модуля ПОСНД

Наименование данных	**Обозначение**	**Структура данных**	**Способ вывода данных**	**Ограничения**
Результаты анализа корректности типов данных	O^T	Строковые данные *(str)*	Вывод в консоль	Нет
Результаты устранения неполноты данных	O^E	Строковые данные *(str)*	Вывод в консоль	Нет

Результаты анализа информативности данных	O^I	Строковые данные *(str)*	Вывод в консоль	Нет
Результаты анализа информативности данных	O^M	Строковые данные *(str)*	Вывод в консоль	Нет

2) Выходные данные модуля ИЗСНД сведены в единую табл. 7.1.2.

Таблица 7.1.2 – Выходные данные модуля ИЗСНД

Наименование данных	**Обозначение**	**Структура данных**	**Способ вывода данных**	**Ограничения**
Множество построенных ассоциативных правил класса	$U_k^{(j)}\left(d_k^l \in \mathbf{A}_k^{(j)}\right) \to Y_j$	Массив объектов класса Rules	Вывод в консоль	Нет
Преобразованные данные	$\boldsymbol{X}^m = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$	Объект pandas.DataFrame	Возвращаемый атрибут функции	Нет

3) Выходные данные модуля АОТССОП сведены в единую табл. 7.1.3.

Таблица 7.1.3 – Выходные данные модуля АОТССОП

Наименование данных	**Обозначение**	**Структура данных**	**Способ вывода данных**	**Ограничения**
Оцениваемые значения	X_{ij}	Двумерный массив данных	Вывод в консоль	Нет
Показатели качества оценивания	–	Словарь	Вывод в консоль	Нет
Обученная модель по некоторому датасету	F	Внутренняя программная структура (keras, pickle)	Бинарный файл в файловой системе ОС	Нет

4) Выходные данные модуля АПССОП сведены в единую табл. 7.1.4.

Таблица 7.1.4 – Выходные данные модуля АПССОП

Наименование данных	**Обозначение**	**Структура данных**	**Способ вывода данных**	**Ограничения**
Прогнозируемые значения	$\boldsymbol{X}^*$	Многомерный массив числовых значений	Консоль, файл формата CSV	Нет
Показатели качества прогнозирования	–	Многомерный массив числовых значений	Консоль, файл формата CSV	Нет
Обученная модель прогнозирования по некоторому датасету	$LSTM^{[N]}$	Программный формат модели (библиотека keras)	Файл формата H5	Нет
Обученная модель нормализации по некоторому датасету	–	Программный формат модели (библиотека sklearn)	Файл формата pickle	Нет

7.2 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: не сохраняет выходные данные во внешние файлы.

4) Модуль АПССОП: Модель прогнозирования сериализуется во внешний файл, путь к которому передается через аргумент – model_path; и файл который имеет бинарный формат, реализованный в библиотеке keras (файл создается автоматически и не требует обработки пользователем). Модель нормализации сериализуется во внешний файл, путь к которому передается через аргумент – scaler_path; файл, и который имеет бинарный формат, реализованный в библиотеке pickle (файл создается автоматически и не требует обработки пользователем). Результаты оценки прогнозирования сохраняются во внешний файл формата CSV – file_name.

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					