Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Зав. кафедрой ИТиСУ
_____ Е.В. Кислицын
(подпись) (Ф.И.О.)
«29» 05 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ИССЛЕДОВАНИЕ МЕТОДОВ АВТОМАТИЧЕСКОГО МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧЕ ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ

Научный руководитель: Ронкин Михаил Владимирович
к.т.н., доцент
_____
подпись

Нормоконтролер: Бредихина Наталья Сергеевна
_____
подпись

Студент группы: РИМ-220906 Зенков Мирослав Андреевич
Зенков
подпись

Екатеринбург
2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления
Направление подготовки 09.04.01 Информатика и вычислительная техника
Образовательная программа Инженерия искусственного интеллекта

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Зенкова Мирослава Андреевича группы РИМ-220906
(фамилия, имя, отчество)

1. Тема выпускной квалификационной работы Исследование методов автоматического машинного обучения в задаче прогнозирования временных рядов

Утверждена распоряжением по институту от «4» декабря 2023 г. № 33.02-05/298

2. Научный руководитель Ронкин Михаил Владимирович, доцент, кандидат технических наук
(Ф.И.О., должность, ученая степень, ученое звание)

3. Исходные данные к работе _____

4. Перечень демонстрационных материалов *презентация в MS PowerPoint*

5. Календарный план

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы	Отметка о выполнении
1.	Глава 1. Анализ предметной области инструментов автоматического машинного обучения	до 23.03.2024 г.	✓
2.	Глава 2. Проведение экспериментов с пакетами автоматического машинного обучения	до 29.04.2024 г.	✓
3.	Глава 3. Оценка и интерпретация результатов экспериментов	до 19.05.2024 г.	✓
4.	ВКР в целом	до 20.05.2024 г.	✓

Научный руководитель Ронкин М.В. _____
Ф.И.О. (подпись)

Студент задание принял к исполнению 18.09-2024 Зенков
дата (подпись)

6. Допустить Зенкова Мирослава Андреевича к защите выпускной квалификационной работы в экзаменационной комиссии

Зав. кафедрой ИТиСУ _____ Е.В. Кислицын
(подпись) Ф.И.О.

РЕФЕРАТ

Выпускная квалификационная работа 57 с., 26 рис., 5 табл., 47 источн.

ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ, АВТОМАТИЧЕСКОЕ МАШИННОЕ ОБУЧЕНИЕ, СРАВНЕНИЕ, ОПТИМИЗАЦИЯ ГИПЕРПАРАМЕТРОВ

Объект исследования — пакеты автоматизированного машинного обучения для прогнозирования временных рядов.

Предмет исследования — алгоритмы оптимизации гиперпараметров применяемые в ряде выбранных пакетов.

Цель работы — проведение сравнения пакетов автоматизированного машинного обучения в контексте задачи прогнозирования временных рядов и выявление особенностей подходов к оптимизации гиперпараметров используемых в каждом пакете.

Методы исследования: проведение теоретического анализа доступной литературы по теме исследования, изучение документации к задействованным в работе пакетам автоматического машинного обучения, проведение экспериментов, сравнение и оценка результатов прогнозирования с помощью построенных конвейеров, обобщение и интерпретация полученных результатов.

Результаты работы: выделены особенности в реализации алгоритмов оптимизации гиперпараметров для рассматриваемых библиотек.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей работе приведены следующие термины с соответствующими определениями:

Автоматическое машинное обучение (*англ*. AutoML) — это область машинного обучения, которая стремится автоматизировать процесс разработки моделей машинного обучения.

Гиперпараметры (*англ*. Hyperparameters) — это параметры, которые контролируют процесс обучения модели машинного обучения. В отличие от параметров модели, которые выводятся из данных во время обучения, гиперпараметры устанавливаются до начала обучения и остаются неизменными в течение всего процесса.

Фреймворк (*англ*. Framework) — это программная платформа, которая предоставляет набор инструментов, библиотек и API для упрощения разработки, обучения и развертывания моделей машинного обучения.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе приведены следующие обозначения и сокращения:

ARIMA (*англ*. Autoregressive Integrated Moving Average) — интегрированная модель авторегрессии - скользящего среднего. Это класс моделей, который отражает набор различных стандартных временных структур в данных временных рядов.

AutoML (*англ*. Automated Machine Learning) — это область машинного обучения, которая стремится автоматизировать процесс применения машинного обучения к задачам реального мира.

DAG (*англ*. Directed Acyclic Graph) — структура, в которой узлы представляют переменные, а направленные ребра показывают зависимости между этими переменными, используется в структурном обучении для представления моделей причинно-следственных связей.

ETS (*англ*. Error-Trend-Seasonality) — класс моделей для анализа и прогнозирования временных рядов, учитывающих ошибку, тренд и сезонность.

GBM (*англ*. Gradient Boosting Machine) — алгоритм машинного обучения, который использует градиентный бустинг для построения ансамбля деревьев решений.

GLM (*англ*. Generalized Linear Model) — статистическая модель, которая обобщает линейную регрессию, позволяя использовать различные типы зависимых переменных. GLM предполагает, что зависимая переменная связана с линейной комбинацией независимых переменных через функцию связи.

HPO (*англ*. Hyper Parameter Optimization) — это процесс поиска наилучших значений гиперпараметров для модели машинного обучения, чтобы добиться максимальной производительности.

MAE (*англ*. Mean Absolute Error) — средняя абсолютная ошибка, метрика для оценки качества моделей, измеряющая среднюю величину абсолютного отклонения предсказанных значений от фактических.

MSE (*англ*. Mean Squared Error) — среднеквадратичная ошибка, является средним значением квадратов отклонений прогнозов модели от фактических значений.

NAS (*англ*. Neural Architecture Search) — процесс автоматического поиска оптимальной архитектуры нейронной сети для конкретной задачи.

NBEATS (*англ*. Neural Basis Expansion Analysis and Transformation) — архитектура нейронной сети для временных рядов, которая использует блоки сезонности для прогнозирования.

RMSE (*англ*. Root Mean Squared Error) — корень из среднеквадратичной ошибки.

RNN (*англ*. Recurrent Neural Network) — Тип нейронной сети, специализирующийся на обработке последовательных данных (например, текст, временные ряды). RNN обладает циклическими связями, позволяющими сети "запоминать" предыдущие входные данные и использовать их для предсказания следующего элемента последовательности.

ROC AUC (*англ*. Receiver Operating Characteristic Area Under the Curve) — это метрика, используемая для оценки качества бинарной классификации модели. Она измеряет площадь под кривой ROC (Receiver Operating Characteristic), которая отражает отношение между чувствительностью (вероятность правильного определения положительного класса) и специфичностью (вероятность правильного определения отрицательного класса) модели при изменении порога классификации.

SMAC (*англ*. Sequential Model-based Algorithm Configuration) — метод оптимизации параметров алгоритмов машинного обучения, основанный на последовательной моделировании и адаптации.

SMAPE (*англ*. Symmetric Mean Absolute Percentage Error) — симметричная средняя абсолютная процентная ошибка, используется для измерения точности прогнозов модели в процентах.

SMBO (*англ*. Sequential Model-Based Optimization) — техника оптимизации, которая использует модель для поиска оптимальных параметров функции. SMBO последовательно строит и улучшает модель, используя информацию из предыдущих оценок функции.

SOTA (*англ*. State-of-the-Art) — понятие, обозначающее лучшие известные на определенный момент методы и модели в области машинного обучения.

SSM (*англ*. State Space Model) — математическая модель, используемая для описания динамической системы. SSM представляет систему как набор скрытых состояний, которые развиваются во времени. Модель позволяет прогнозировать будущие значения системы, используя текущее состояние и входные данные.

SVM (*англ*. Support Vector Machine) — метод обучения с учителем, используемый для задач классификации и регрессии, который строит разделяющую гиперплоскость в пространстве признаков.

ВВЕДЕНИЕ

Прогнозирование временных рядов имеет фундаментальное значение для различных областей жизни. Оно применяется при прогнозировании нагрузки на сеть, при прогнозировании потока клиентов у терминала, при прогнозировании котировок акций, объемов продаж товаров, численности заболевших при развитии эпидемий и т.д. От качества прогноза напрямую зависит качество принимаемых решений. В условиях, когда требуются обработка огромных объемов данных и привлечение различных математических инструментов, не обойтись без использования соответствующих моделей машинного обучения.

Подготовка решения для конкретного случая требует кропотливого и долгого процесса проектирования. Предварительная обработка данных, инжиниринг признаков, обучение моделей, оптимизация гиперпараметров, оценка результатов – каждый этап нетривиален и требует вмешательства человека.

Если пространство для поиска велико, человеку не хватит времени, чтобы перебрать все комбинации вручную, однако хорошее решение должно быть найдено, комбинации необходимо перебрать и более того, необходимо сделать это эффективно за разумное время.

Кроме того, обучающая выборка в полном объеме также доступна не всегда. Существуют системы, данные в которые поступают динамически раз в определенный промежуток времени. В таком случае возникает потребность периодически переобучать модель с учетом новых, только что поступивших данных, однако расходовать время специалиста на повторяемую из раза в раз рутинную работу это не самый оптимальный подход.

AutoML, автоматизированное машинное обучение, призвано решить подобные проблемы. Цель AutoML состоит в том, чтобы автоматизировать процесс применения машинного обучения к задачам реального мира минимизируя необходимость человеческого вмешательства. Регулярная

монотонная работа перекладывается на плечи автоматической системы, которая сама способна обработать данные, протестировать различные комбинации моделей и гиперпараметров, оценить их и вернуть лучшее найденное решение.

Вместо того, чтобы осуществлять исчерпывающий перебор всех возможных вариантов конфигураций модели, методов предобработки, инжиниринга признаков AutoML фреймворки полагаются на специальные методы и стратегии, позволяющие сократить перебор.

Для решения задачи оптимизации гиперпараметров могут быть применены следующие методы:

– Поиск по сетке;
– Случайный поиск;
– Вероятностные методы;
– Метод градиентного спуска;
– Эволюционные алгоритмы;
– Нейросетевой подход.

Реализация подходов к поиску гиперпараметров зачастую скрыта от пользователя и фреймворк используется просто как инструмент.

Цель данной работы состоит в том, чтобы сравнить несколько популярных AutoML-фреймворков для решения задачи прогнозирования временных рядов, учитывая не только полученные метрики и качество прогнозов, но также и тонкости их получения, особенности работы фреймворков по оптимизации гиперпараметров.

Для достижения поставленной цели необходимо решить следующие задачи:

– изучение материалов, посвященных AutoML-инструментам;
– изучение современных библиотек автоматизированного машинного обучения, применяемых для прогнозирования временных рядов;

– реализация экспериментов, позволяющих оценить качество прогнозирования, время обучения, и расходы вычислительных ресурсов для каждого из решений;

– обобщение и интерпретация результатов с точки зрения особенностей рассматриваемых библиотек.

Объектом исследования являются пакеты автоматизированного машинного обучения для прогнозирования временных рядов.

Предметом исследования вступают алгоритмы оптимизации гиперпараметров, применяемые в ряде пакетов.

Методы исследования включают в себя:

– проведение теоретического анализа, связанного с темой исследования источников;

– изучение документации к рассматриваемым в работе решениям в области автоматизированного машинного обучения;

– проведение экспериментов, сравнение качества прогнозов, получаемых с помощью конвейеров, подобранных в процессе работы библиотек;

– сравнение потребления вычислительных ресурсов, требуемых каждым решением;

– сравнение подходов к оптимизации, применяемых в выбранных библиотеках.

Результаты работы будут содержать перечень особенностей в реализации алгоритмов оптимизации гиперпараметров, выделенных для каждого рассматриваемого в ходе экспериментов фреймворка.

1. Обзор материалов посвящённых применению инструментов автоматического машинного обучения

В существующих научных трудах сравнение результатов работы AutoML-фреймворков и исследование используемых ими подходов к оптимизации зачастую разделены. Как правило, авторы ограничиваются простым упоминанием или перечислением фактов.

Найденные материалы можно разделить на несколько групп:

– Сравнение обобщающей способности конвейеров, получаемых при помощи библиотек автоматического машинного обучения друг с другом и с другими методами прогнозирования;

– Общее сравнение функционала пакетов AutoML безотносительно конкретной задачи;

– Создание нового решения в области AutoML и последующее сравнение с существующими решениями.

1.1 Сравнение обобщающей способности конвейеров

Первой группой являются статьи посвященные непосредственно сравнению качества предсказаний AutoML пакетов для одной или нескольких задач.

Так в работе «Does AutoML Outperform Naive Forecasting?» было приведено сравнение четырёх решений: AutoGluon, H2O, TPOT, Auto-Sklearn с традиционными стратегиями для задачи прогнозирования временных рядов [30]. Под традиционными стратегиями авторами подразумеваются метод Хольта-Уинтера, экспоненциальное сглаживание и наивные предсказания [43]. Наборами данных выступали ElectricityLoadDiagrams20112014 с почасовыми показаниями о потреблении электроэнергии и EXUSEU, коллекция ежедневных обменных курсов. Лимиты времени для AutoML решений составляли 60, 120 и 300 секунд. Авторами было обнаружено, что

качество прогнозов библиотек автоматического машинного обучения сильно зависит от выделяемого на обучение времени, и в условиях недостатка времени фреймворки зачастую работают хуже традиционных методов. При выделении же слишком большого количества времени, часть из рассматриваемых пакетов начинает склоняться к запоминанию обучающей выборки, к переобучению. Традиционные методы в среднем оказались более эффективны, чем AutoML, однако при сравнении результатов особенности фреймворков не учитывались, в статье просто перечислен набор фактов о каждом пакете, список моделей и возможностей предобработки данных. Полученные авторами результаты представлены ниже на Рисунке 1.

Model	Wins	Losses	Model	Wins	Losses
Holt-Winters (add_d-add)	146	46	h2o_v3_300s	66	126
Holt-Winters (add-add)	146	46	autogluon_v3_300s	65	127
SimpleExpSmoothing	139	53	tpot_v3_60s	65	127
Holt-Winters (add_d-mul)	138	54	h2o_v5_60s	65	127
Holt-Winters (add-mul)	138	54	h2o_v3_60s	63	129
h2o_v1_120s	80	112	h2o_v8_60s	62	130
h2o_v1_300s	79	113	tpot_v5_60s	62	130
autosklearn_v1_300s	75	117	autosklearn_v1_60s	61	131
h2o_v1_60s	75	117	tpot_v8_60s	60	132
h2o_v8_120s	74	118	autogluon_v5_300s	60	132
autogluon_v1_300s	74	118	tpot_v8_120s	60	132
tpot_v1_120s	74	118	autogluon_v8_300s	59	133
tpot_v8_300s	74	118	autogluon_v8_120s	59	133
autogluon_v1_60s	73	119	autogluon_v5_120s	59	133
tpot_v1_60s	72	120	autogluon_v8_60s	57	135
tpot_v1_300s	71	121	autogluon_v5_60s	53	139
h2o_v5_300s	69	123	autosklearn_v3_300s	51	141
autogluon_v1_120s	69	123	h2o_v5_120s	50	142
autogluon_v3_120s	69	123	autosklearn_v5_300s	36	156
h2o_v8_300s	68	124	autosklearn_v3_120s	18	174
tpot_v5_120s	67	125	autosklearn_v5_120s	17	175
autogluon_v3_60s	67	125	autosklearn_v5_60s	14	178
autosklearn_v1_120s	67	125	autosklearn_v8_300s	14	178
tpot_v5_300s	66	126	autosklearn_v8_60s	9	183
tpot_v3_300s	66	126	autosklearn_v8_120s	5	187
tpot_v3_120s	66	126	autosklearn_v3_60s	2	190
h2o_v3_120s	66	126	Naive	Base	Base

Рисунок 1 – Сравнение производительности для AutoML фреймворков и наивных предсказаний [30]

Помимо сравнения библиотек автоматического машинного обучения друг с другом, с классическим ML и со статистическими методами были также встречены сравнения с решениями, написанными человеком.

В одной из найденных работ было представлено сравнение качества прогнозов четырех фреймворков TPOT, H2O, Auto-Sklearn, AutoGluon с решениями ML сообщества [25]. Проверки проводились на 12 наборах данных из OpenML (6 для задачи регрессии, 6 для классификации) и 1 реальном наборе данных. Метриками выступали ROC AUC и Accuracy для классификации и RMSE, MAE для регрессии. В 7 из 12 случаев автоматически сформированные решения показали сравнимые с человеком или лучшие значения метрик. В эти 7 случаев входят задачи регрессии и «простые» задачи классификации, которые по заверениями авторов хорошо решались и людьми, и пакетами AutoML. Задача контролируемой регрессии в OpenML была недостаточно проработана. Авторами отмечено, что для большей части рассмотренных наборов характерно либо полное превосходство автоматических решений, либо их заметное отставание. Особенности фреймворков, которые могли бы объяснить полученные результаты, приведены не были. Таблица полученных исследователями результатов представлена на Рисунке 2.

Number	OpenML task name	OpenML task id	Type	Best human score			Best score by AutoML (auc resp. rmse)	AutoML better or equal in at least one metric?	Best AutoML framework outperforms humans by ... %	Number of AutoML frameworks better or equal
1	supervised classification credit-g	31	classification	metric	acc	auc	0.799	NO	-0.97	0
					0.786	0.807				
2	supervised classification blood-transfusion	10101	classification	metric	acc	auc	0.754982	NO	-0.16	0
					0.8021	0.756				
3	supervised classification wilt	9914	classification	metric	acc	auc	0.995346	NO	-0.08	0
					0.9897	0.996				
4	Supervised classification tic-tac-toe	145804	classification	metric	acc	auc	1	YES	0	4
					1	1				
5	Supervised Classification on monks-problems-2	146065	classification	metric	acc	auc	1	YES	0	3
					1	1				
6	Supervised Classification on monks-problems-1	146064	classification	metric	acc	auc	1	YES	0	4
					1	1				
7	Supervised Regression on cholesterol	2295	regression	metric	mae	rmse	49.72166	YES	2.04	4
					38.6187	50.76				
8	Supervised Regression on liver-disorders	52948	regression	metric	mae	rmse	2.995219	NO	-1.83	4
					2.3088	2.941				
9	Supervised Regression on analcatdata_negotiation	4823	regression	metric	mae	rmse	0.79956	NO	-1.33	0
					0.5766	0.789				
10	Supervised Regression on cleveland	2285	regression	metric	mae	rmse	0.856123	YES	1.93	1
					0.6407	0.873				
11	Supervised Regression on fri_c3_100_25	4958	regression	metric	mae	rmse	0.255819	YES	72.73	4
					0.7575	0.938				
12	Supervised Regression on kin8nm	2280	regression	metric	mae	rmse	0.005561	YES	97.26	4
					0.1634	0.203				

Рисунок 2 – Сравнение AutoML и решений сообщества [25]

Использование материалов из репозитория OpenML также упоминается в работе, посвященной сравнению производительности и функционала конвейеров AutoML для задач бинарной классификации, многоклассовой классификации и регрессии [38]. Авторами утверждалось, что этап предобработки во всех рассмотренных фреймворках нестабилен и требует вмешательства человека. Для части рассмотренных решений дано краткое описание принципов работы. По результатам тестов однозначного лидера обнаружить не удалось. Также авторами отмечено, что хотя с увеличением лимита времени все результаты улучшились, скорость сходимости в каждом из случаев разная. С какими особенностями рассмотренных фреймворков связаны полученные результаты, не уточняется. Сравнение функционала представлено на Рисунке 3.

Tool	Platform	Input data sources		Data pre-processing	Data types detected					Feature engineering				ML Tasks		Model selection and Hyperparameter optimization				
		Spreadsheet datasets	Image, text		Numerical	Categorical	Datetime	Time-series	Other (Hierarchical types) (7*)	Datetime, categorical processing	Imbalance, missing values	Feature selection, reduction	Advanced feature extraction (8*)	Supervised learning (9*)	Unsupervised learning (10*)	Ensemble	Genetic algorithm	Random search	Bayesian search	Neural architecture search
TransmogrifAI	Apache Spark	Y	N	Y(*)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Y	Y	N
H2O-AutoML	AWS, GCP, Azure	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	N	Y	N	Y	N	Y	N	N
Darwin (+)	GCP	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y
DataRobot (+)	AWS, GCP, Azure	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
Google AutoML (+)	Google Cloud	N	Y	Y						N	Y	Y	Y	Y	Y		Y	Y	Y	Y
Auto-sklearn		Y	N	N	N	N	N	N	N	Y(2*)	Y	Y	Y	Y	N	Y	N	Y	Y	N
MLjar (+)	MLJAR Cloud	Y(3*)	N	Y	Y	Y	N	N	N	Y	Y(4*)	N	N	Y(5*)	N	Y	N	Y	N	N
Auto_ml		Y	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	N	Y	N	Y	Y	N
TPOT		Y	N	N	N	N	N	N	N	N	Y	N	Y	Y	N	Y	Y	N	N	N
Auto-keras		Y	Y	N	N	N	N	N	N	N	Y	Y	N	Y	N	N	N	Y	Y	Y
Ludwig		Y	Y	Y(*)	Y	Y	N	Y	Y	N	Y	Y	Y	Y	N	Y	N	Y	Y	Y
Auto-Weka		Y	N	N	Y	Y	N	N	N	N	Y	Y	N	Y	N	Y	N	Y	Y	N
Azure ML (+)	Azure	Y	Y	Y(6*)	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	N	Y	Y	N
H2O-Driverless AI (+)	AWS, GCP, Azure	Y(3*)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N

Рисунок 3 – Сравнение AutoML и решений сообщества [38]

Авторами «Open Source AutoML Benchmark» был создан универсальный бенчмарк для унификации сравнения AutoML конвейеров [24]. Составлена таблица для упрощенного сравнения механизмов выбора моделей в пакетах Auto-Weka, Auto-Sklearn, TPOT и H2O, кратко описаны используемые в каждом пакете подходы. Было отобрано 39 наборов данных из OpenML для задачи классификации. Для бинарной классификации использовалась метрика ROC AUC, для многоклассовой – log loss. При проведении тестов исследователи старались сохранять значения гиперпараметров «по умолчанию», за исключением тех, которые отвечают за выделяемые вычислительные ресурсы и время обучения. Было проведено два теста с лимитом в 1 час и 4 часа. Базовой модели для сравнения выступал случайный лес, во многих случаях AutoML пакеты показали лучшие результаты, но однозначного победителя выявить не удалось. Авторами было замечено, что

на некоторых наборах данных часть пакетов работала значительно хуже других, однако однозначных выводов о причинах подобной разницы в производительности предоставлено не было. Сравнительная таблица с результатами исследования представлена на Рисунке 4.

Framework: Binary tasks:	auto-sklearn	Auto-WEKA	H2O AutoML	RandomForest	TPOT
adult	1.045	1.000	1.049	1.000	1.048
airlines	1.403	1.016	1.435	0.997	1.343
albert	1.009		1.115	1.001	0.981
amazon_employee...	0.972*	0.886	1.048	1.003	1.012
apsfailure	1.000	0.985	1.001	1.000	1.001
australian	1.010	1.015	0.909	1.010	1.011
bank-marketing	1.012	0.950	1.015	1.000	1.008
blood-transfusion	1.495	1.379	1.532	0.985	1.149
christine	1.072	0.998	1.048	0.988	1.029
credit-g	0.970*	0.829	0.991	1.004	0.924
guiellermo	1.004	0.934	1.024	0.999	0.878
higgs	1.018*	0.845	1.041	0.999	1.005
jasmine	0.987	0.939	1.001	0.998	1.004
kc1	0.999*	0.934	0.992	0.987	1.013
kddcup09_appetency	1.181*	1.043	1.176	1.016	1.134
kr-vs-kp	1.000*	0.959	1.000	0.999	0.999
miniboone	1.008	0.957	1.010	0.999	1.001
nomao	1.002	0.973	1.002	1.000	1.001
numerai28.6	1.679	1.544	1.730	1.042	1.428
phoneme	0.993*	0.998	1.005	1.000	1.015
riccardo	1.000	0.996	1.000	0.999	0.992
sylvine	1.013	0.985	1.011	0.999	1.023
Multi-class tasks:					
car	1.030	0.906	1.060	0.878	1.060
cnae-9	1.069	0.541	1.076	0.999	1.057
connect-4	1.184	-1.565	1.409	0.954	1.276
covertype	0.976	-0.361	0.856	0.944	0.933
dilbert	1.182	0.459	1.205	0.979	1.111
dionis	0.580	0.590		1.002	
fabert	1.026	-5.235	1.049	1.004	1.005
fashion-mnist	0.995	0.717	1.052	0.993	0.841
helena	0.660	-18.420	1.905	0.970	1.676
jannis	1.083	-1.989	1.065	0.973	0.987
jungle_chess...	1.299	-3.309	1.235	0.933	1.459
mfeat-factors	1.059*	0.789	1.053	0.992	1.018
robert	-0.001		1.545	1.000	0.640
segment	1.004	0.808	1.012	0.992	1.008
shuttle	1.000	0.979	1.000	1.000	1.000
vehicle	1.102	-4.630	1.166	0.986	1.099
volkert	1.002	-5.585	1.111	0.954	0.945

Рисунок 4 – Сравнение производительности AutoML пакетов и базовой модели [24]

В рамках WSDM Cup 2020 было проведено сравнение решений задачи автоматизированной регрессии временных рядов пакетом AutoGluon и участниками соревнования AutoSeries [40]. Было предложено 10 наборов

данных, для ранжирования решений участников использовалась метрики RMSE и SMAPE после проведения испытаний. Фреймворку удалось обойти предоставленный авторами baseline, однако достичь метрик близких к решению победителя соревнования удалось не во всех случаях. Положение AutoGluon улучшилось только после добавления ручного инжиниринга признаков. Отмечено, что в занявших лучшие места работах реализованы модули для инженерии признаков и тонкой настройки моделей, однако как реализован подобный подход в AutoGluon никак не упоминается. Сравнение результатов по RMSE и SMAPE представлено на Рисунке 5.

Dataset	Phase	Baseline		1st DV		AutoGluon		FE + AutoGluon	
		RMSE	SMAPE	RMSE	SMAPE	RMSE	SMAPE	RMSE	SMAPE
fph1	Feedback	99.04	142.59	40.69	102.19	90.19	**26.45**	**40.57**	105.31
fph2	Feedback	17563	142.64	**236.6**	26.63	14978	59.94	263.74	**25.51**
fph3	Feedback	3337	38.49	**623.32**	**4.99**	6365	116.14	3159	31.08
fph4	Feedback	6.91	**187.58**	**3.66**	190.94	NA	NA	NA	NA
fph5	Feedback	8.63	174.45	**5.76**	**173.54**	NA	NA	NA	NA
pph1	Private	422.37	12.65	218.83	6.11	2770.70	9.46	**212.68**	**5.85**
pph2	Private	16851	139.31	**242.41**	23.46	15028	57.04	269.85	**22.98**
pph3	Private	8.78	178.45	**6.21**	**177.08**	NA	NA	NA	NA
pph4	Private	11.54	174.94	**3.74**	**168.4**	NA	NA	NA	NA
pph5	Private	309.33	39.2	**50.37**	**5.91**	949.4	20.52	65.22	6.65

Рисунок 5 – Сравнение метрик решений победителей соревнования, базового решения и AutoGluon [40]

В одной из работ ориентированных на исследование практического использования автоматизированного машинного обучения рассматривалось применение AutoML в области производственного инжиниринга [19]. Использовалось 6 наборов данных. Сравнение качества прогнозов фреймворков с решением сделанным вручную показало, что пакеты автоматического машинного обучения способны обеспечить неплохой базовый уровень, однако автоматизированный feature-engineering заметно уступает ручной генерации признаков. Исследователями отмечено, что обучение на предварительно обработанном наборе данных, позволило

AutoML библиотекам превзойти решения человека. Связано ли это с используемыми в пакетах подходами к оптимизации гиперпараметров, авторы не уточнили.

Вопрос влияния инжиниринга признаков на производительность AutoML решений также был поднят в [37]. В качестве фреймворка кандидата выступал Auto-Sklearn, за feature-engineering отвечала библиотека TSFresh [18]. Было проведено три типа экспериментов: отдельно с автоматическим выбором размера окна (W), отдельно с добавлением признаков из TSFresh (T), и с тем, и с другим одновременно (WT). Применялись две стратегии прогнозирования: многовыходная и рекурсивная. Тестирование проводилось на наборах данных из CompEngine, 20 различных категорий, среди которых присутствовали как синтетические, так и настоящие данные [6]. Для оценки использовалась RMSE. Сравнение с базовыми решениями GBM, SVM, NBEATS, Auto-Keras и Vanilla Auto-Sklearn показало эффективность многошаговой стратегии для всех кандидатов кроме NBEATS (ввиду отсутствия поддержки рекурсивных прогнозов). Vanilla Auto-Sklearn вышел лучшим в 8 из 20 экспериментов, при этом, Auto-Sklearn после применения модификаций (W, T, WT) продемонстрировал преимущество в 14 из 20 случаев. Был кратко упомянут используемый подход, однако его влияние на полученные результаты озвучено не было. Созданную авторами схему рабочего процесса для пакета Auto-Sklearn можно наблюдать на Рисунке 6.

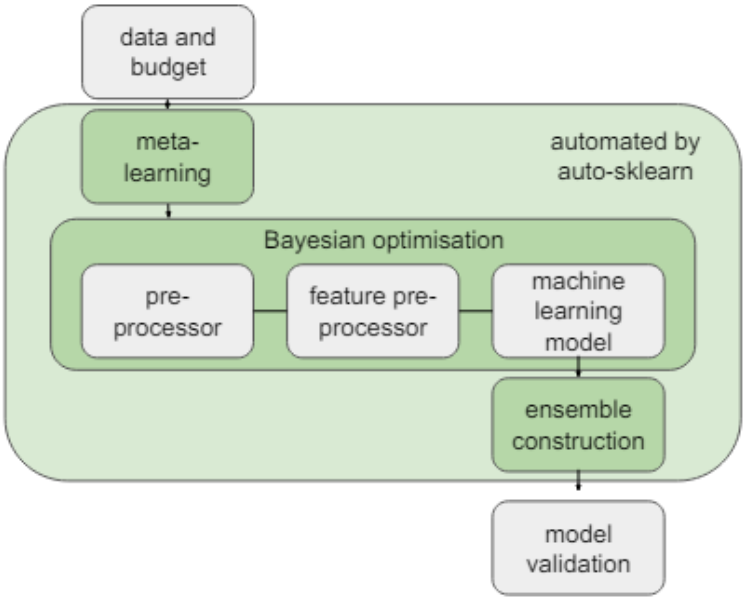

Рисунок 6 – Схема рабочего процесса фреймворка Auto-Sklearn [37]

В работе «Benchmarking AutoML for regression tasks on small tabular data in materials design» было проведено сравнение качества предсказаний для задачи регрессии на небольших наборах данных [1]. При подборе фреймворков кандидатов, авторы исключили проприетарные решения, пакеты поддержка которых прекращена, библиотеки, ориентированные на другие задачи (требуемая задача – регрессия на табличных данных), решения, полагающиеся на нейронные сети (из-за малого объема данных). Рассмотрены пакеты H2O, TPOT, Auto-Sklearn и MLjar. Тестирование проводилось на двенадцати наборах данных, с использованием метрик RMSE, MAE, MAPE в зависимости от набора. В среднем, решения AutoML превзошли решения из литературы, исключениями выступили только самые малые из представленных наборов данных. Для каждого фреймворка было приведено краткое описание реализуемого подхода, однако связи между методом и полученными результатами не прослеживалось. Для сопоставления функционала пакетов авторами была построена сравнительная таблица (Рисунок 7).

Framework	Version	Preparation			Models					HyperOpt					XAI			
		Cat. Encoding	Preprocessing	Feature Engineering	Regression	Naive Bayes	Support Vector Machine	Tree-based	Neural Network	Grid Search	Random Search	Bayesian Optimization	Genetic Programming	Neural Architecture Search	Feature Importance	Advanced	Ensembles	GPU support
Auto-sklearn	v0.14.2	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗
H2O	v3.34.0.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓
MLjar	v0.10.6	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗
TPOT	v0.11.7	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✓	✓

Рисунок 7 – Функционал пакетов Auto-Sklearn, H2O, MLjar, TPOT [1]

1.2 Обзорные статьи по пакетам автоматического машинного обучения

Ко второй группе были отнесены обзорные статьи, которые оказались не богаты на детальные сравнения.

В первой обнаруженной статье была представлена общая схема типичного конвейера AutoML [39]. Перечислены различные методы оптимизации гиперпараметров, приведены примеры фреймворков AutoML и описаны используемые в каждом из них подходы. Сравнение функционала пакетов было оформлено в виде таблицы (Рисунок 8), а сравнения качества прогнозов не проводилось.

Method	Optimization Algorithm	Data Pre-Processing	Feature Engineering	Model Selection	HPO
Auto-WEKA [67,70]	Bayesian Optimization (SMAC)	✓		✓	✓
Auto-Sklearn [71,77]	Joint Bayesian Optimization and Bandit Search (BOHB)	✓		✓	✓
TPOT [78]	Evolutionary Algorithm	✓	✓	✓	✓
TuPAQ [81]	Bandit Search			✓	✓
ATM [83]	Joint Bayesian Optimization and Bandit Search		✓		✓
Automatic Frankensteining [84]	Bayesian Optimization			✓	✓
ML-Plan [86]	Hierarchical Task Networks (HTN)	✓		✓	✓
Autostacker [90]	Evolutionary Algorithm			✓	✓
AlphaD3M [92]	Reinforcement Learning/Monte Carlo Tree Search	✓		✓	✓
Collaborative Filtering [94]	Probabilistic Matrix Factorization	✓		✓	✓

Рисунок 8 – Сравнение функционала AutoML решений [39]

Основные подходы к оптимизации гиперпараметров были перечислены в [31]. Были описаны принципы работы Байесовской оптимизации, нелинейного программирования, эволюционных алгоритмов, эвристических и мета-эвристических подходов. Отмечены случаи, когда и в каких работах использовались данные методы поиска. Исследователями выделены три подхода к выбору метода прогнозирования: эвристический, эмпирический и на основе модели принятия решений. Для различных моделей и соответствующих им методов выбора представлена сравнительная таблица. Также была создана таблица для отбора мета-признаков и используемого метода агрегации для моделей принятия решений (Рисунок 9). Приведена таблица для упомянутых в научных трудах этапов разработки конвейера предсказаний, однако сравнений методов предсказаний на типовых задачах не проводилось.

Optimized Forecasting Method(s)	**Optimized Hyper-parameters**	**Performance Metric**	**Validation Sample**		**Optimization Method**
			in-sample	out-of-sample	
sARIMA	p, q		-	-	ACF & PACF
ARIMA, SETARMA	p, q		-	-	testing, ACF & PACF, checking
AR, VAR	p, s	BIC	X	-	grid search, checking
MA	q	user-defined	X	-	grid search
ARIMA	p, d, q	user-defined	X	-	testing, ACF & PACF, grid-search
ARIMA	p, d, q	GOF	X	-	grid search, checking
sARIMA	p, d, q, P, D, Q, s	AIC	X	-	testing, two-stage grid search
sARIMA	p, d, q, P, D, Q	user-defined	X	X	testing, grid search, checking
sARIMA	p, d, q, P, D, Q, s	AIC	X	-	var. minimization, two-stage grid search
sARIMA	p, d, q, P, D, Q, s	user-defined	X	X	two-stage MIQP & grid search
sARIMA	p, d, q, P, D, Q	MAE	-	X	BO GP
ETS	E, T, S	AIC	X	-	grid search
TES	$\alpha, \beta, \gamma, \varphi, p$	RMSE, MAD MAPE	X	-	multiobjective NLP
Theta	φ	MAE	X	-	Brent
	T, S				testing, grid search
UC	T, S, I	BIC	X	-	grid search
kNN	k		-	-	heuristic
LASSO	λ	MAPE	-	X	grid search

Рисунок 9 – Таблицы моделей и методов их оптимизации [31]

В работе «Инструменты построения моделей машинного обучения» представлена общая архитектура AutoML системы на примере Auto-Sklearn

[13]. Были кратко описаны принципы работы и перечислены отличительные черты фреймворков Auto-Weka, Auto-Sklearn, TPOT, Google Vizer, однако сравнения качества получаемых прогнозов не проводилось.

В работе «Review of ML and AutoML Solutions to Forecast Time-Series Data» представлен обзор как ML, так и AutoML решений для прогнозирования временных рядов [16]. Перечислены наиболее популярные методы оптимизации гиперпараметров. Кратко описаны общие принципы работы для 10 пакетов. Авторами было приведено сравнение функционала фреймворков и списка поддерживаемых ими моделей, однако качество получаемых прогнозов не сравнивалось.

В работе исследователей из Санкт-Петербургского государственного электротехнического университета были перечислены наиболее популярные методы HPO [13]. Приведены примеры AutoML фреймворков и их короткое описание. Тестов или сравнений не проводилось.

В работе помимо перечисления самых распространенных подходов к оптимизации гиперапарметров были также выделены ключевые преимущества и недостатки каждого подхода [12].

1.3 Создание новых решений в области автоматического машинного обучения

Наконец, третья группа - труды, посвященные созданию новых инструментов в области AutoML. В отличие от статей, ориентированных на обзор или сравнение библиотек, они предполагают более тесную связь подробного описания и сравнения. Поскольку фреймворк представляется впервые, подобные статьи обязаны иметь детальное объяснение используемого подхода. Сравнение же с существующими методами должно присутствовать как доказательство эффективности нового предлагаемого решения.

Так в работе посвященной реализации нового подхода к отбору моделей авторами был представлен фреймворк AutoXPCR [22]. При создании пакета авторы уделили внимание не только качеству предсказаний (P - predictive error), но и таким немаловажным характеристикам, как объяснимость (C - complexity) и экономичность (R - resource). При отборе моделей использовалось мета-обучение [4]. Выбор модели был сформулирован как задача многоцелевой оптимизации, характеристики временного ряда рассматривались системой в качестве признаков, каждая модель оценивалась с точки зрения PCR, а не только прогностической производительности. Предложенное решение сравнивалось с методом грубого перебора, с AutoForecast, и AutoKeras [27]. В ходе сравнения было обнаружено, что предлагаемый метод слегка уступает конкурирующим решениям в точности, но значительно превосходит их в плане потребления ресурсов (Рисунок 10). При сохранении до 90% от наилучшего возможного качества, предложенный метод требует всего 20% вычислительных затрат.

Data set	AutoXPCR			AutoForecast			Exhaustive			AutoKeras	
	F(x)	MASE	kWh	F(x)	MASE	kWh	F(x)	MASE	kWh	MASE	kWh
Aust..and	**0.70**	**1.24**	**0.98**	0.21	1.69	29.7	**0.70**	**1.24**	914	13.0	69.9
Car Parts	0.66	0.50	**0.70**	0.38	**0.47**	1.55	**0.67**	0.75	56.3	0.97	8.47
CIF 2016	**0.78**	1.08	**0.07**	0.27	**1.03**	0.33	**0.78**	1.08	9.86	16.2	2.46
Dominick	**0.74**	**1.55**	**13.3**	0.49	1.77	28.9	**0.74**	**1.55**	1000	2.53	245
Elec..kly	**0.76**	**1.65**	**0.42**	0.39	2.79	0.97	**0.76**	**1.65**	52.5	19.8	12.2
FRED-MD	**0.73**	**0.71**	**0.15**	0.26	1.92	0.57	**0.73**	**0.71**	12.3	991	6.32
Hospital	**0.72**	0.84	**0.96**	0.45	**0.76**	3.92	**0.72**	0.84	52.2	75.5	20.0
M1 M..hly	0.62	**1.40**	**0.76**	0.39	1.50	4.09	**0.70**	1.67	55.3	1172	21.6
M1 Q..rly	0.60	3.15	**0.30**	0.37	**1.78**	0.85	**0.68**	1.99	16.5	335	2.23
M3 M..hly	**0.61**	**1.12**	**2.32**	0.38	1.16	11.1	**0.61**	**1.12**	273	2.57	80.6
M3 Q..rly	**0.58**	**1.41**	**1.90**	0.35	2.08	3.20	**0.58**	**1.41**	86.2	2.67	7.81
M4 Hourly	**0.66**	5.09	**0.48**	0.22	**3.14**	7.16	**0.66**	5.09	414	7.7e4	213
M4 Weekly	0.69	**2.79**	**1.31**	0.39	4.13	4.52	**0.70**	3.10	71.0	25.7	24.6
NN5 Daily	**0.66**	**0.65**	**0.23**	0.42	0.68	0.80	**0.66**	**0.65**	42.7	1.21	29.7
NN5 ..kly	**0.74**	0.95	**0.19**	0.38	**0.88**	1.56	**0.74**	0.95	38.9	1.58	2.16
Sola..kly	**0.70**	1.47	**0.10**	0.44	**0.83**	0.48	**0.70**	1.47	23.9	1.00	1.24
Tour..hly	0.59	1.70	**0.48**	0.34	**1.44**	3.30	**0.64**	2.79	58.5	6.94	41.3
Tour..rly	**0.77**	1.96	**0.27**	0.48	**1.67**	0.66	**0.77**	1.96	24.2	12.9	11.9
Traf..kly	**0.77**	1.69	**0.66**	0.34	**1.53**	5.72	**0.77**	1.69	107	5.06	9.74

Рисунок 10 – Сравнение результатов AutoXPCR, AutoForecast, AutoKeras и полного перебора [22]

Повторное использование заранее вычисленных результатов применяется не только в XPCR. Пакет auto-sktime и ранее упомянутый AutoForecast также полагаются на мета-обучение [8].

В работе связанной с фреймворком AutoForecast был предложен новый метод выбора модели прогнозирования для неизвестного ранее набора данных без необходимости получения исчерпывающих оценок всех имеющихся моделей [14]. Подход базируется на мета-обучателях для предсказания производительности модели на основе сходства с известным набором данных и для прогнозирования производительности модели на различных временных окнах в зависимости от предыдущих вычисленных значений. Приведены детальное описание принципов работы и общая схема фреймворка (Рисунок 11). По качеству получаемых прогнозов предлагаемое решение превзошло другие SOTA-методы на основе мета-обучения.

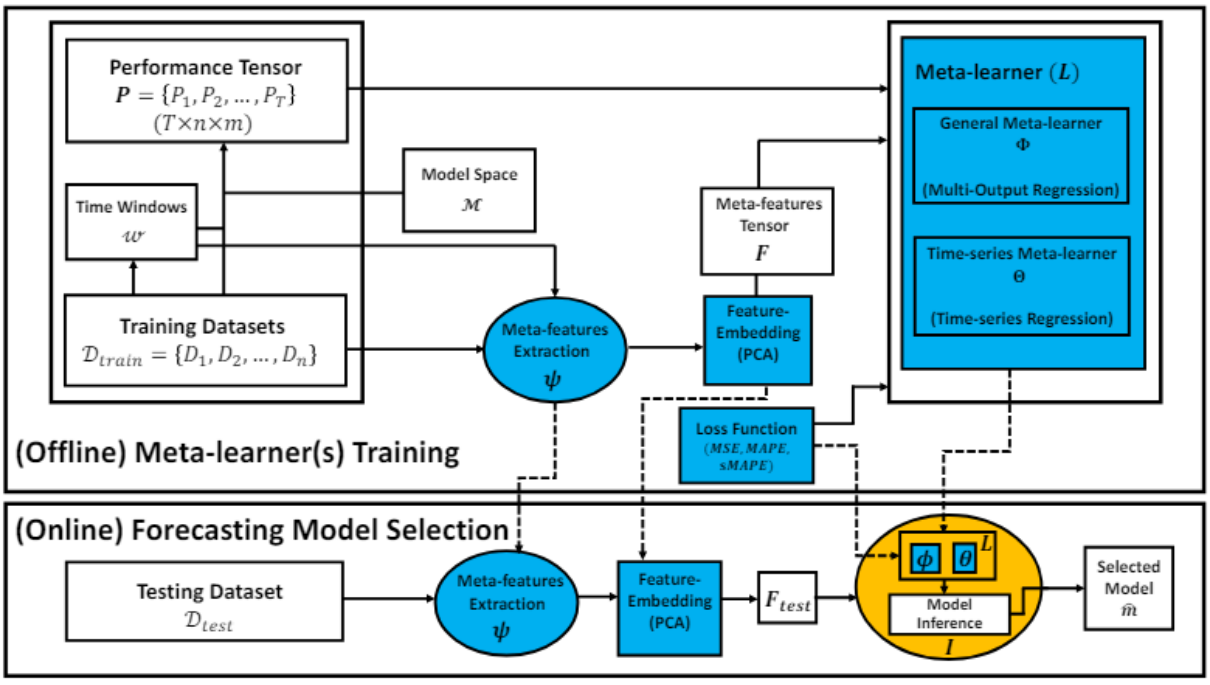

Рисунок 11 – Устройство AutoForecast [14]

В материалах по пакету auto-sktime исследователями был предложен комбинированный подход совмещающий в себе поиск гиперпараметров с помощью Байесовской оптимизации и идею мета-обучения [42]. Было создано

три типа шаблонов: для статистических моделей, моделей машинного обучения и нейронных сетей. Первоначально по шаблонам создавалось множество моделей кандидатов. Далее с помощью мета-обучения отбирались кандидаты, наиболее подходящие под рассматриваемый набор данных. Наконец, когда лучшие модели отобраны, запускалась процедура Байесовской оптимизации. Детально описан сам метод, приведено сравнение с AutoGluon, AutoTS, DeepAR, ETS, PMDARIMA. Метод оказался жизнеспособен и сумел превзойти лучшие существующие решения SOTA (Рисунок 12).

Framework	MASE	Ranking	Time
APT-TS	3.12 ± 5.41	4.51 ± 2.49	242.2 ± 134.9
AUTO-SKTIME	$\mathbf{1.68 \pm 1.89}$	$\mathbf{3.19 \pm 1.88}$	326.1 ± 43.4
AUTOGLUON	6.00 ± 18.17	6.76 ± 2.35	144.9 ± 160.1
AUTOTS	4.26 ± 10.94	5.38 ± 2.35	321.1 ± 52.7
DEEPAR	11.53 ± 51.51	7.23 ± 1.99	61.3 ± 106.0
ETS	2.57 ± 2.78	5.83 ± 2.38	$\mathbf{2.0 \pm 7.6}$
HYPERTS	2.88 ± 5.03	4.75 ± 2.60	300.0 ± 89.4
PMDARIMA	2.62 ± 3.21	5.45 ± 2.65	47.1 ± 98.4
PYAF	3.22 ± 5.33	5.39 ± 2.56	22.6 ± 56.5
TFT	9.79 ± 50.62	6.70 ± 1.92	76.9 ± 118.3

Рисунок 12 – Сравнение auto-sktime с state-of-the-art решениями [42]

В работе посвященной созданию пакета Auto-PyTorch-TS авторами было предложено решение для поиска архитектуры нейронной сети (NAS) [20]. Решение базировалось на Байесовской оптимизации с многофакторной оптимизацией. Гиперпараметры подбирались с помощью SMAC, для моделирования распределения потерь в пространстве конфигураций применялся случайный лес [29]. Для эксперименты были выбраны наборы данных из Monash Time Series Forecasting Repository. На 18 из 24 наборов Auto-Pytorch-TS оказался лучшим (Рисунок 13).

	Auto-PyTorch-TS						
	# Epochs	Resolution	# Series	# SMPs per Ser.	Vanilla BO	Best dataset-specific baseline	Overall single best baseline
M3 Yearly	2.73(0.10)	2.66(0.05)	2.76(0.09)	**2.64**(0.09)	2.68(0.08)	2.77(0.00)	3.13(0.00)
M3 Quarterly	**1.08**(0.01)	1.10(0.01)	1.10(0.01)	1.09(0.02)	1.12(0.03)	1.12(0.00)	1.26(0.00)
M3 Monthly	**0.85**(0.01)	0.89(0.02)	0.86(0.01)	0.87(0.04)	0.86(0.02)	0.86(0.00)	0.86(0.00)
M3 Other	1.90(0.07)	1.82(0.03)	1.98(0.13)	1.92(0.05)	1.95(0.15)	**1.81**(0.00)	1.85(0.00)
M4 Quarterly	1.15(0.01)	1.13(0.01)	**1.13**(0.01)	1.15(0.01)	1.15(0.02)	1.16(0.00)	1.19(0.00)
M4 Monthly	0.93(0.02)	0.93(0.02)	0.93(0.02)	**0.93**(0.02)	0.96(0.02)	0.95(0.00)	1.05(0.00)
M4 Weekly	0.44(0.01)	0.45(0.02)	**0.43**(0.02)	0.44(0.02)	0.45(0.01)	0.48(0.00)	0.50(0.00)
M4 Daily	1.14(0.01)	1.18(0.07)	1.16(0.06)	1.14(0.04)	1.38(0.41)	**1.13**(0.02)	1.16(0.00)
M4 Hourly	0.86(0.12)	0.95(0.11)	**0.78**(0.07)	0.85(0.07)	0.85(0.06)	1.66(0.00)	2.66(0.00)
M4 Yearly	**3.05**(0.03)	3.08(0.04)	3.05(0.01)	3.09(0.04)	3.10(0.02)	3.38(0.00)	3.44(0.00)
Tourism Quarterly	1.61(0.03)	1.57(0.05)	1.59(0.05)	1.59(0.02)	1.55(0.03)	**1.50**(0.01)	1.83(0.00)
Tourism Monthly	**1.42**(0.03)	1.44(0.03)	1.45(0.04)	1.47(0.02)	1.42(0.02)	1.44(0.02)	1.75(0.00)
Dominick	0.51(0.04)	0.49(0.00)	**0.49**(0.01)	0.49(0.01)	0.49(0.01)	0.51(0.00)	0.72(0.00)
Kdd Cup	1.20(0.02)	1.18(0.02)	1.18(0.03)	1.18(0.03)	1.20(0.03)	**1.17**(0.01)	1.39(0.00)
Weather	0.63(0.08)	0.58(0.04)	0.59(0.02)	0.59(0.06)	**0.57**(0.00)	0.64(0.01)	0.69(0.00)
NN5 Daily	0.79(0.01)	0.80(0.01)	0.81(0.04)	**0.78**(0.01)	0.79(0.01)	0.86(0.00)	0.86(0.00)
NN5 Weekly	0.76(0.01)	0.76(0.03)	0.76(0.01)	0.77(0.01)	**0.76**(0.01)	0.77(0.01)	0.87(0.00)
Hospital	0.76(0.01)	0.76(0.00)	0.76(0.00)	**0.75**(0.01)	0.75(0.01)	0.76(0.00)	0.77(0.00)
Traffic Weekly	1.04(0.07)	1.10(0.03)	1.04(0.05)	1.08(0.09)	1.03(0.07)	**0.99**(0.03)	1.15(0.00)
Electricity Weekly	0.78(0.04)	1.06(0.13)	0.80(0.04)	**0.74**(0.07)	0.85(0.11)	0.76(0.01)	0.79(0.00)
Electricity Hourly	1.52(0.05)	1.54(0.00)	1.58(0.08)	1.54(0.06)	**1.51**(0.05)	1.60(0.02)	3.69(0.00)
Kaggle Web Traffic Weekly	0.56(0.01)	0.56(0.01)	**0.55**(0.00)	0.57(0.01)	0.59(0.01)	0.61(0.02)	0.62(0.00)
Covid Deaths	5.11(1.60)	4.54(0.05)	**4.43**(0.13)	4.58(0.30)	4.53(0.24)	5.16(0.04)	5.72(0.00)
Temperature Rain	0.76(0.05)	0.75(0.01)	0.73(0.02)	0.73(0.03)	0.71(0.04)	**0.71**(0.03)	1.23(0.00)
∅ Rel. Impr Best	0.82	0.83	**0.81**	0.81	0.82	0.86	1.0
∅ Rel. Impr Oracle	0.96	0.97	**0.95**	0.95	0.96	1.0	1.17

Рисунок 13 – Результаты, достигнутые Auto-Pytorch-TS [20]

Помимо мета-обучения и байесовской оптимизации существуют и другие подходы к поиску моделей. Авторами пакета AutoGluon утверждалось, что в большей части существующих пакетов основное внимание уделяют оптимизацию гиперпараметров одиночных моделей, в то время как предлагаемое ими решение на создание многослойных ансамблей [21]. После подгонки одиночных моделей на основе их предсказаний и исходных данных обучается отдельная stacker модель. При разработке исследователи вдохновлялись глубокими нейронными сетями. Имеется собственный аналог skip connection, получаемый ансамбль содержит несколько слоев, stacker-ы более высокого уровня будут использовать предсказание нижних в качестве входных данных, в конце производится агрегация прогнозов (Рисунок 14). Представлен алгоритм, описывающий стратегию обучения. Предложенное решение сравнивалось с Auto-Weka, Auto-Sklearn, TPOT и H2O. Тесты

проводились на открытых наборах данных с Kaggle, OpenML и в большинстве случаев подход AutoGluon-Tabular оказалось лучшим.

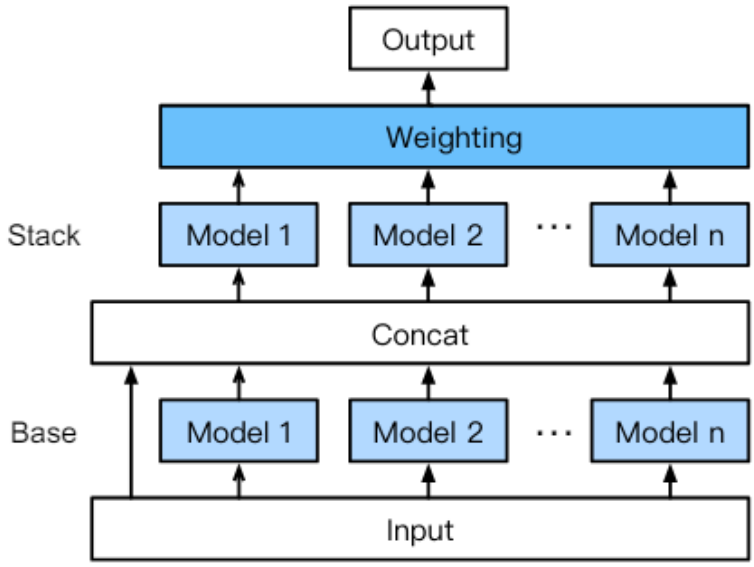

Рисунок 14 – Применяемый в AutoGluon многослойный стекинг [21]

В работе «A revolutionary framework for landslide hazard analysis» отмечено, что для получения итоговой модели AutoGluon полагается не на глубокую оптимизацию гиперпараметров отдельных моделей, а на формирование стекового ансамбля из лучших моделей, однако деталей реализации предоставлено не было [34]. По ROC-AUC метрике полученный ансамбль превзошел KNN, Extra Trees, Random Forest и Catboost.

После создания предиктора для табличных данных функционал фреймворка AutoGluon многократно дорабатывался. В рамках статьи представлен модуль AutoGluon Time Series, ориентированный на работу с временными рядами [36]. Новый модуль использует существующие в рамках проекта AutoGluon алгоритмы построения ансамблей и прямого отбора. При сравнении с существующими решениями на базе статистических методов, машинного обучения и глубоких нейронных сетей AutoGluon–TimeSeries продемонстрировал более долгое время обучения, однако обеспечил более

высокое качество прогнозов и вышел победителем для 19 из 29 наборов данных (Рисунок 15).

Framework	Wins	Losses	Ties	Failures	Champion	Average rank	Average rescaled error	Win rate vs. baseline
AutoGluon (MASE)	-	-	-	**0**	**19**	**2.08**	**0.073**	**100.0%**
StatEnsemble	6	20	0	3	3	3.12	0.238	82.8 %
AutoPyTorch (MASE)	4	25	0	**0**	2	4.12	0.257	93.1%
AutoETS	4	25	0	**0**	1	4.64	0.374	75.9 %
AutoTheta	4	23	0	2	0	4.92	0.427	72.4 %
DeepAR	4	24	0	1	2	5.08	0.434	93.1 %
AutoARIMA	4	22	0	3	1	5.92	0.612	79.3 %
TFT	2	27	0	**0**	1	6.12	0.635	75.9 %

Рисунок 15 – Сравнение AutoGluon–TimeSeries с другими фреймвоками [36]

В процессе создания фреймворка AutoAI-TS для оценки конвейеров авторами был предложен механизм T-Daub [35]. В начале производилась проверка временного ряда и поиск доступных преобразований. Следующим шагом обучалась нулевая константная модель и определялась длина look-back окна. После этого система начинала создавать конвейеры на основе заранее заготовленных шаблонов. Конвейеры ранжировались на основе прогнозируемой кривой обучения. Первоначально конвейеры использовали небольшую часть обучающей выборки, кандидаты, обладающие хорошими метриками, получали в распоряжение больше данных для обучения. В конце самый лучший найденный конвейер использовал всю обучающую выборку. Предложенное решение было протестировано как на синтетическом датасете, так и на реальных наборах данных (62 одномерных и 9 многомерных рядов). Конкурентами были решения из SOTA: DeepAR, Prophet, pmdarima, PyAF, NBEATS. По метрике SMAPE AutoAI-TS в 17 случаях занял первое место, в 11 второе и по 9 – третье и четвертое места. Архитектура созданной системы представлена на Рисунке 16.

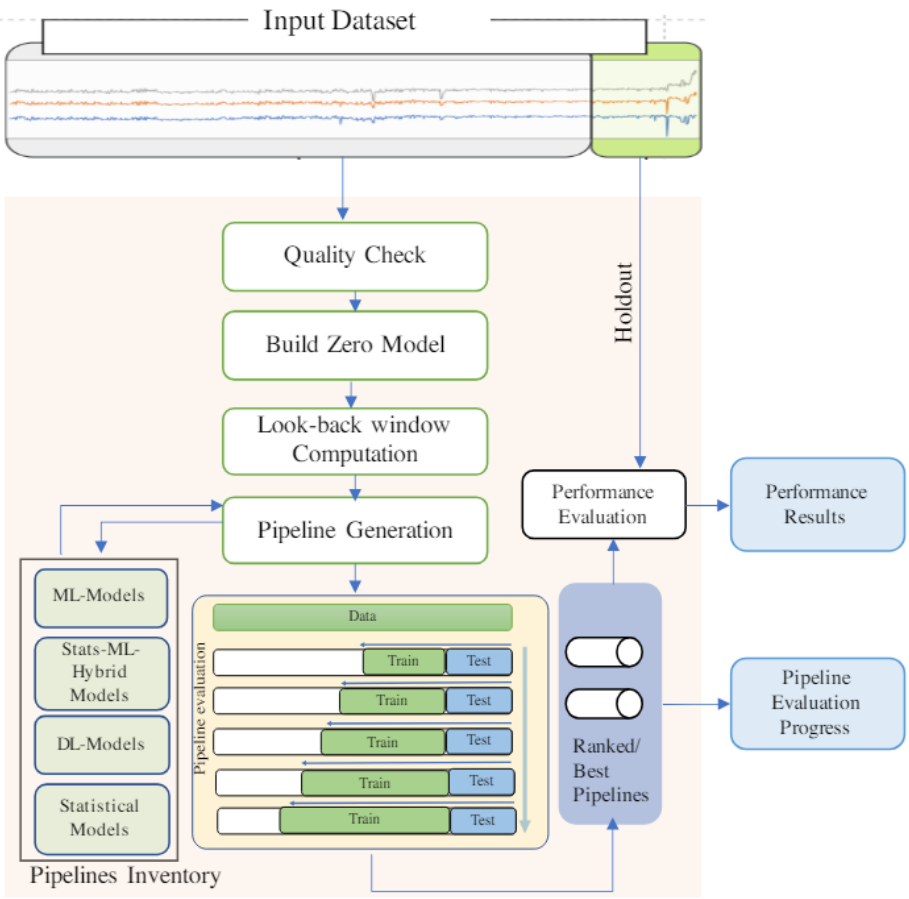

Рисунок 16 – Архитектура системы AutoAI-TS [35]

1.4 Постановка задачи исследования

Итак, при сравнении качества прогнозов, получаемых при помощи инструментов автоматического машинного обучения, можно сделать следующий общий вывод: внутреннему устройству пакетов и используемым ими подходам к поиску моделей и оптимизации гиперпараметров уделяется недостаточно много внимания, что может сказаться как на качестве самого прогноза, так и на удобстве использования соответствующего сервиса и прозрачности и понятности реализуемых им алгоритмов.

В зависимости от статьи, применяемый метод либо упоминается вскользь, перечисляется как факт безотносительно результата, либо вовсе опускается.

Если же статья предполагает детальное описание метода, при сравнении явного указания на особенности метода, которые привели к полученным результатам, не делается.

Целью данной работы является сравнение AutoML-фреймворков для задачи прогнозирования временных рядов с повышенным вниманием не только к качеству получаемых прогнозов, но также тому, как именно прогнозы были получены и каковы особенности работы фреймворков по оптимизации гиперпараметров.

Список задач, способствующих достижению поставленной цели, включает в себя:

– изучение пакетов AutoML, поддерживающих прогнозирование временных рядов;

– реализацию экспериментов, позволяющих оценить качество прогнозирования, время обучения, и расходы вычислительных ресурсов для каждого из решений;

– обобщение и интерпретацию результатов с точки зрения особенностей каждого из рассматриваемых пакетов.

2 Подготовка и проведение экспериментов

2.1 Набор данных для экспериментов

Набором данных для экспериментов был выбран ETTh1. ETTh1 — это набор данных, предназначенный для обучения и тестирования моделей прогнозирования временных рядов. Он является частью более крупной коллекции ETT (Electricity Transformer Temperature), которая содержит данные о нагрузке трансформаторов электроэнергии [26].

В наборе содержится 17420 точек измерений с 1 июля 2016 года по 26 июня 2018 года. Частота дискретизации составляет 1 час, каждая точка данных состоит метки времени, целевого значения «oil temperature» (OT) и шести характеристик силовой нагрузки (HUFL, HULL, MUFL, MULL, LUFL, LULL). Пропусков набор данных не содержит. Изменения значений целевой переменной показаны на Рисунке 1.

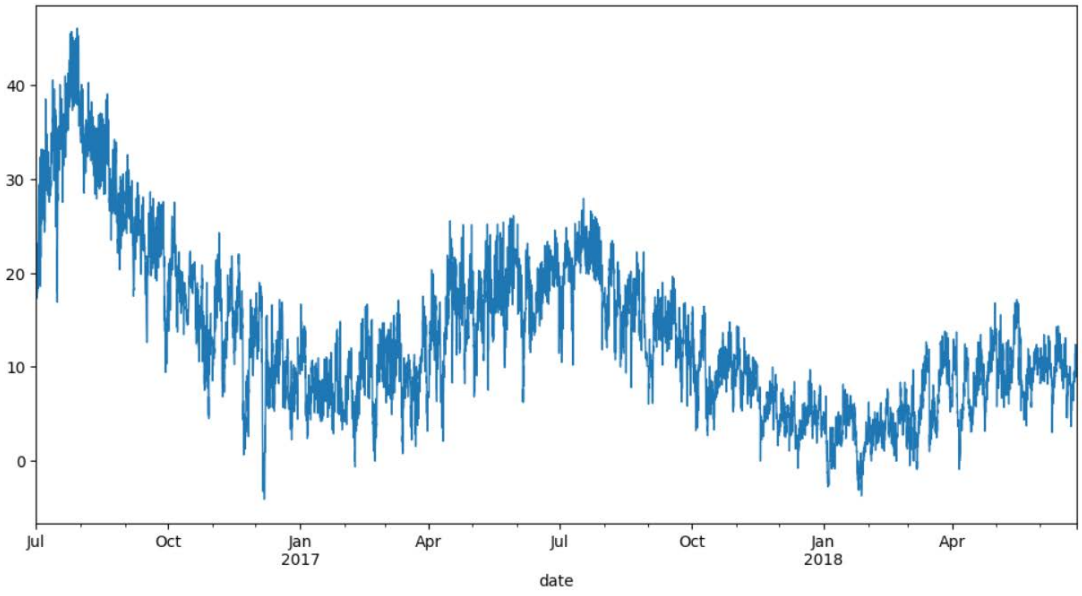

Рисунок 1 – Значения целевой переменной в ETTh1

Гистограмма распределения целевой переменной представлена на Рисунке 2.

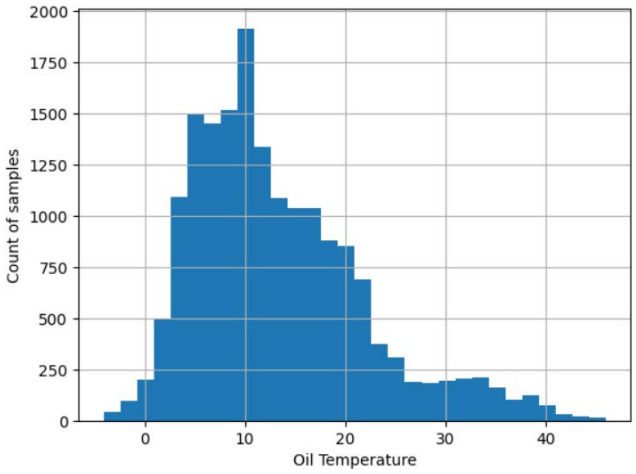

Рисунок 2 – Распределение значений целевой переменной в ETTh1

Коллекция ETT была собрана в рамках исследования посвященного созданию Informer, новой архитектуры нейронной сети для прогнозирования временных рядов [41]. Данные из ETT использовались для оценки производительности полученной модели.

2.2 Поиск библиотек и подбор кандидатов для экспериментов

Существует много AutoML-инструментов, направленных на решение разнообразного перечня задач. Необходимо сформулировать критерии, по которым будут подбираться пакеты.

На первом этапе были выдвинуты два требования и пакеты подбирались по следующим критериям:

– Во фреймворке имеется поддержка прогнозирования временных рядов.

– Решение является открытым, исходный код находится в публичном репозитории. Исследование особенностей пакета предполагает наличие доступа не только к официальной документации, но и к исходному коду. Данное требование призвано исключить из выборки проприетарные инструменты от крупных компаний.

Помимо ссылок на репозитории проектов и статистических данных с Github был проведен поиск статей, в которых упоминается каждый рассматриваемый фреймворк. Результаты представлены в Таблице 1.

Таблица 1 – AutoML решения с поддержкой временных рядов

Название	Ссылка	Первая версия	Последняя версия	Статьи
AutoKeras	https://github.com/keras-team/autokeras	17.01.2020	21.03.2024	+
Auto_TS	https://github.com/AutoViML/Auto_TS	14.06.2020	25.07.2020	+
AutoTS	https://github.com/winedarksea/AutoTS	31.05.2020	08.04.2024	-
ETNA	https://github.com/tinkoff-ai/etna	17.09.2021	08.08.2023	-
Auto-Sklearn	https://github.com/automl/auto-sklearn	17.10.2016	13.02.2023	+
AutoGluon	https://github.com/autogluon/autogluon	25.03.2020	17.04.2024	+
Merlion	https://github.com/salesforce/Merlion	24.09.2021	15.02.2023	-
TPOT	https://github.com/epistasislab/tpot	06.11.2019	24.02.2024	+
Auto-Sktime	https://github.com/Ennosigaeon/auto-sktime	29.03.2023	29.03.2023	+
Auto-PyTorch	https://github.com/automl/Auto-PyTorch	09.10.2019	23.08.2022	-
HyperTS	https://github.com/DataCanvasIO/HyperTS	08.08.2022	22.12.2023	-
AutoWeka	https://github.com/automl/autoweka	07.06.2021	07.03.2022	+
H2O	https://github.com/h2oai/h2o-3	06.02.2015	13.03.2024	+
FEDOT	https://github.com/aimclub/FEDOT	21.01.2021	15.03.2024	+

При поиске дат было обнаружено, что не все группы разработчиков явно помечают версии пакетов как Release. В случае отсутствия отметок, датой первой (или последней) версии считается дата первой (или последней) версии из вкладки Releases в репозитории проекта. Также стоит отметить, что статистика по количеству пользователей или «звезд» у репозитория не может объективным показателем. Это связано с тем, что среди найденных фреймворков присутствуют как решения, изначально ориентированные на временные ряды, так и инструменты AutoML общего назначения, где есть функционал для прогнозирования временных рядов [3].

На втором этапе были выдвинуты следующие требования:

– Последняя версия не старше года, что свидетельствует об активной поддержке проекта;

– Пакет относительно новый, первая версия не старше 5 лет (данное требование исключает «старые» проекты, которые уже были широко исследованы множеством других авторов);

– Выпущено более 10 Release версий пакета.

Результаты фильтрации по перечисленным требованиям представлены в Таблице 2.

Таблица 2 – AutoML решения с поддержкой временных рядов после применения фильтров

Название	Github	Первая версия	Последняя версия	Статьи
AutoKeras	https://github.com/keras-team/autokeras	17.01.2020	21.03.2024	+
AutoTS	https://github.com/winedarksea/AutoTS	31.05.2020	08.04.2024	-
ETNA	https://github.com/tinkoff-ai/etna	17.09.2021	11.04.2024	-
AutoGluon	https://github.com/autogluon/autogluon	25.03.2020	17.04.2024	+
TPOT	https://github.com/epistasislab/tpot	06.11.2019	24.02.2024	+
HyperTS	https://github.com/DataCanvasIO/HyperTS	08.08.2022	22.12.2023	-
FEDOT	https://github.com/aimclub/FEDOT	21.01.2021	15.03.2024	+

Из отфильтрованного списка пакетов было выбрано 4 фреймворка: AutoTS, AutoGluon, ETNA, FEDOT.

Научных статей, посвященных созданию пакетов AutoTS и ETNA, найти не удалось. Для AutoGluon были найдены статьи по созданию предиктора для табличных данных и по добавлению в пакет функционала для прогнозирования временных рядов.

Из выбранных пакетов AutoTS имеет самый широкий список поддерживаемых моделей [5]. AutoTS может использовать модели из пакетов:

– statsmodels,
– gluonts,
– prophet,
– scikit-learn,
– pytorch-forecasting,
– scipy,
– tensorflow,
– neuralforecast.

Список доступных моделей для AutoGluon включает в себя:

– Статистические модели ETS, ARIMA и их автоматически настраиваемые варианты;
– Модели для прогнозирования спроса CrostonClassic, CrostonSBA, CrostonOptimezed;
– Модели нейронных сетей и трансформеров DeepAR, DLinerar, PatchTST, WaveNet, TFT [28];
– Предварительно обученные модели Chronos [17];
– Собственный табличный предиктор, адаптированный для работы с временными рядами.

Доступные в ETNA модели это:

– Статистические модели ETS, ARIMA и их автоматически настраиваемые варианты;
– Модели из пакетов StatsForecast, CatBoost, Scikit-learn;

– Модели нейронных сетей и трансформеров RNN, MLP, NBeats, SSM, DeepAR, TFT, PatchTST [32];

Самый малым арсеналом моделей для предсказания временных рядов обладает Fedot. Fedot поддерживает:

– ARIMA (+ вариант с декомпозицией ряда через STL),
– CGRU,
– ETS,
– GLM,
– Наивные прогнозы,
– Полиномиальную аппроксимацию,
– Регрессоры из Scikit-learn [2].

2.3 Условия проведения экспериментов

Эксперименты проводились локально, машиной для проведения выступал компьютер под управление ОС Windows 11 Pro 23H2 с процессором Intel Core i5 12500H, объемом оперативной памяти в 32 гигабайта и видеоадаптером Nvidia GeForce RTX 3050Ti laptop с 4 гигабайтами видеопамяти.

Ввиду отсутствия поддержки новых версий Python в части рассматриваемых пакетов, использовался Python версии 3.10.11.

В целях снижения вероятности возникновения конфликтов зависимостей, под каждый фреймворк было создано отдельное виртуальное окружение [11].

Для AutoTS была выбрана версия 0.6.11, для AutoGluon 1.1.0, для ETNA 2.6.0, для Fedot версия 0.7.3.1.

Каждое из рассматриваемых решений имеет интеграции со сторонними библиотеками и может использовать предоставляемые ими модели. Для реализации возможности использовать все поддерживаемые модели были

установлены «расширенные» версии каждого из AutoML фреймворков, которые предполагают установку всех опциональных зависимостей. Зависимости для каждого пакета представлены в Таблице 3.

Таблица 3 – Пакеты AutoML и их опциональные зависимости

Пакет	Опциональные зависимости
AutoTS	prophet gluonts mxnet tensorflow lightgbm xgboost tensorflow-probability greykite pytorch-forecasting neuralprophet scipy arch
AutoGluon	torch, lightning, pytorch_lightning, transformers[sentencepiece], accelerate, gluonts statsforecast, mlforecast, utilsforecast,
ETNA	prophet, torch, pytorch-forecasting, pytorch-lightning, wandb, optuna, statsforecast
Fedot	tensorflow, torch

Для каждого фреймворка был выставлен следующий набор ограничений:

– Лимит времени обучения 12 часов (720 минут или 43200 секунд в зависимости от используемых в пакете единиц измерения). Для AutoTS ввиду отсутствия общего лимита определены количество поколений генетического алгоритма равное 4 и время на обработку одного поколения равное 3 часам [44];

– Используемая метрика - MSE;

– Если фреймворк поддерживает предварительные настройки, выбиралась настройка, предназначенная для прогнозирования временных рядов, в случае отсутствия таковой, выбиралась настройка, обеспечивающая наилучший уровнем качества (best, best_quality и т.д.);

– Ввиду малого объема видеопамяти из списков моделей были исключены предварительно обученные модели загружаемые с HuggingFace (модели семейства Chronos в AutoGluon [7]);

– Поскольку набор данных небольшой, количество разбиений на кросс-валидации было ограничено двумя.

Гиперпараметры не связанные с описанными ограничениями были оставлены в значении «по умолчанию».

Стандартное разбиение, использовавшееся в оригинальной работе [41] для набора ETTh1 предполагало выделение 1 года под обучающую выборку, 4-х месяцев под набор для валидации и 4-х месяцев под тестовый, однако в данной работе было решено объединить обучающую и валидационную выборки. Сделано это по нескольким причинам. Валидационный набор предназначен для тонкой настройки гиперпарметров модели с проверкой на не известных ранее данных, однако, в контексте AutoML, настройка гиперпараметров отдельных моделей автоматически выполняется самим фреймворком [47]. Кроме того, из рассматриваемых библиотек только AutoGluon поддерживает передачу отдельного набора для валидации, при этом кросс-валидация реализована каждом из пакетов [10].

Одной из особенностей набора данных является отсутствие пропусков, поэтому дополнительной обработки пропусков произведено не было. Набор данных был предварительно стандартизован при помощи StandardScaler из scikit-learn [9]. Разделение набора и стандартизация были проведены заранее, при запуске только осуществлялось чтение уже подготовленных данных.

Каждый из экспериментов был оформлен в виде python-скрипта. Для получения информации о последовательности вызовов и времени выполнения применялся профилировщик profilehooks [23]. За снятие данных об объеме используемой оперативной памяти отвечал memory-profiler [46].

3. Результаты работы

3.1 Полученная статистика

Для каждого фреймворка было проведено 20 запусков. Результаты по времени обучения, зафиксированному для каждого из запусков представлены в Таблице 4.

Таблица 4 – Время обучения для каждого запуска, время приведено в секундах

Номер эксперимента	AutoTS	AutoGluon	ETNA	Fedot
1	15732.0	662.6	261.0	-
2	15601.9	652.4	275.4	-
3	17247.5	641.3	265.0	-
4	16585.4	715.0	271.8	-
5	14030.0	690.0	261.0	-
6	15483.3	667.5	249.6	-
7	16067.1	683.5	279.5	-
8	16297.2	706.3	273.6	-
9	17314.0	639.9	249.8	-
10	15678.4	639.6	280.9	-
11	17519.1	673.7	287.6	-
12	16535.2	658.5	273.3	-
13	16470.9	682.7	258.7	-
14	15726.3	633.9	287.3	-
15	15201.5	625.0	263.6	-
16	13494.2	611.4	293.8	-
17	15536.0	655.3	300.8	-
18	14825.4	664.0	276.0	-
19	17043.8	698.8	239.9	-
20	16155.4	614.2	266.9	-

По собранным данным была посчитана описательная статистика. Статистика по времени обучения приведена в Таблице 5.

Таблица 5 – Статистика по времени обучения, время приведено в секундах

Пакет	Среднее, с	Отклонение, с	Мин, с	Медиана, с	Макс, с
AutoTS	15927.2	1041.5	13494.2	15899.5	17519.1
AutoGluon	660.8	29.4	611.4	660.6	715.1
ETNA	270.8	15.4	239.9	272.6	300.8
FEDOT	-	-	-	-	-

При установленном лимите в 12 часов 3 из 4 фреймворков уложились в назначенное время, для одного из пакетов не получилось добиться стабильного воспроизведения. Зафиксированные значения времени для каждого из запусков представлены на Рисунке 1.

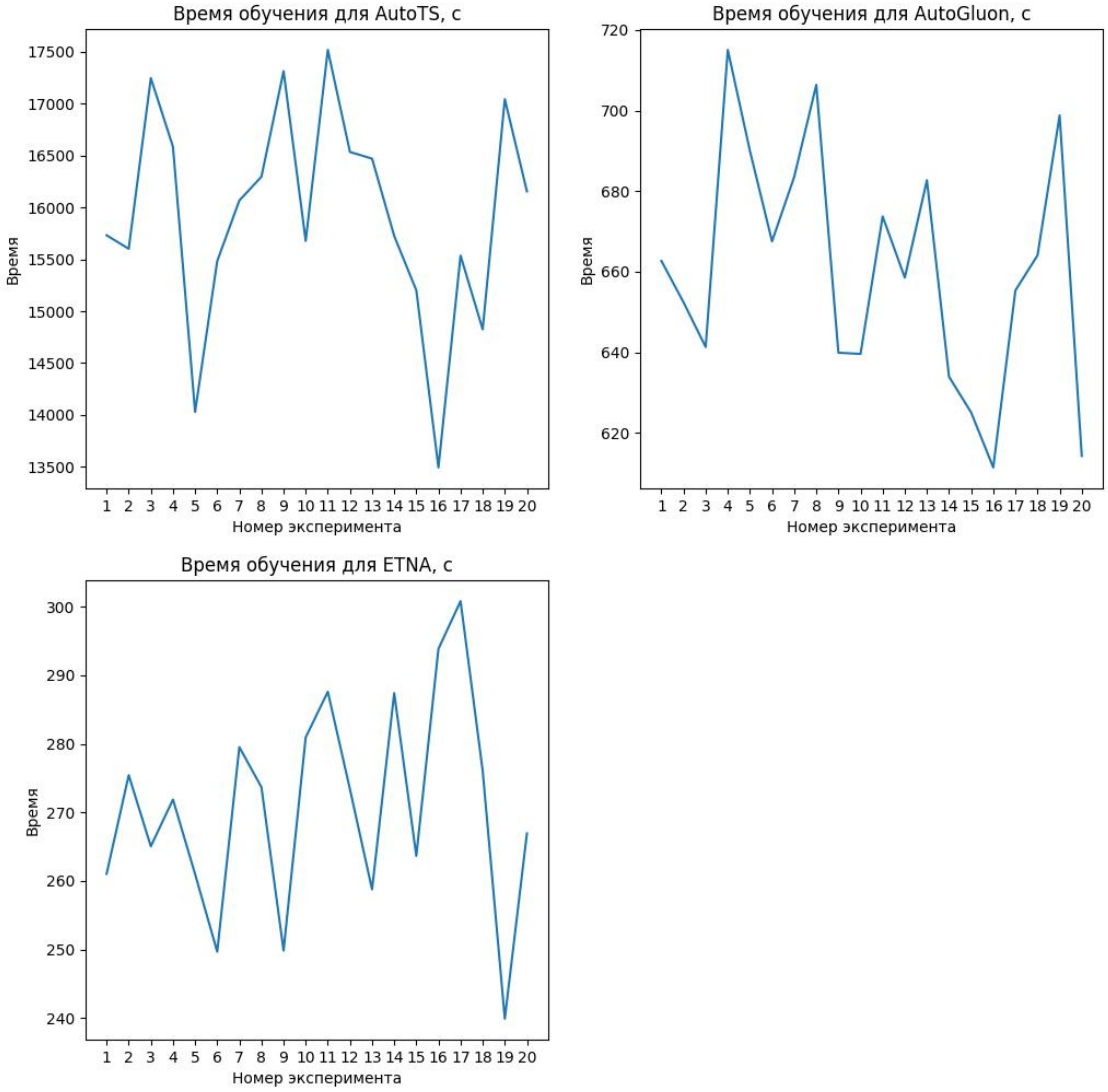

Рисунок 1 – Значения времени для каждого эксперимента

Для каждого запуска были вычислены значения метрики MSE на тестовой выборке. Значения метрики также были усреднены. Усредненные значения метрики MSE представлены на Рисунке 2.

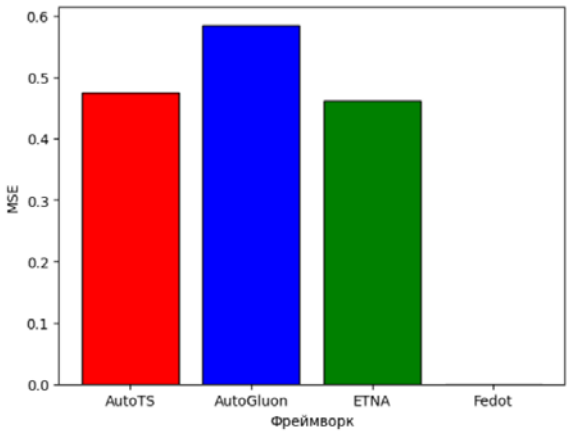

Рисунок 2 – Значения MSE для рассмотренных пакетов

Визуализация предсказаний конвейеров, подобранных каждым из пакетов, и сравнение предсказаний с реальными данными, представлены на Рисунке 3.

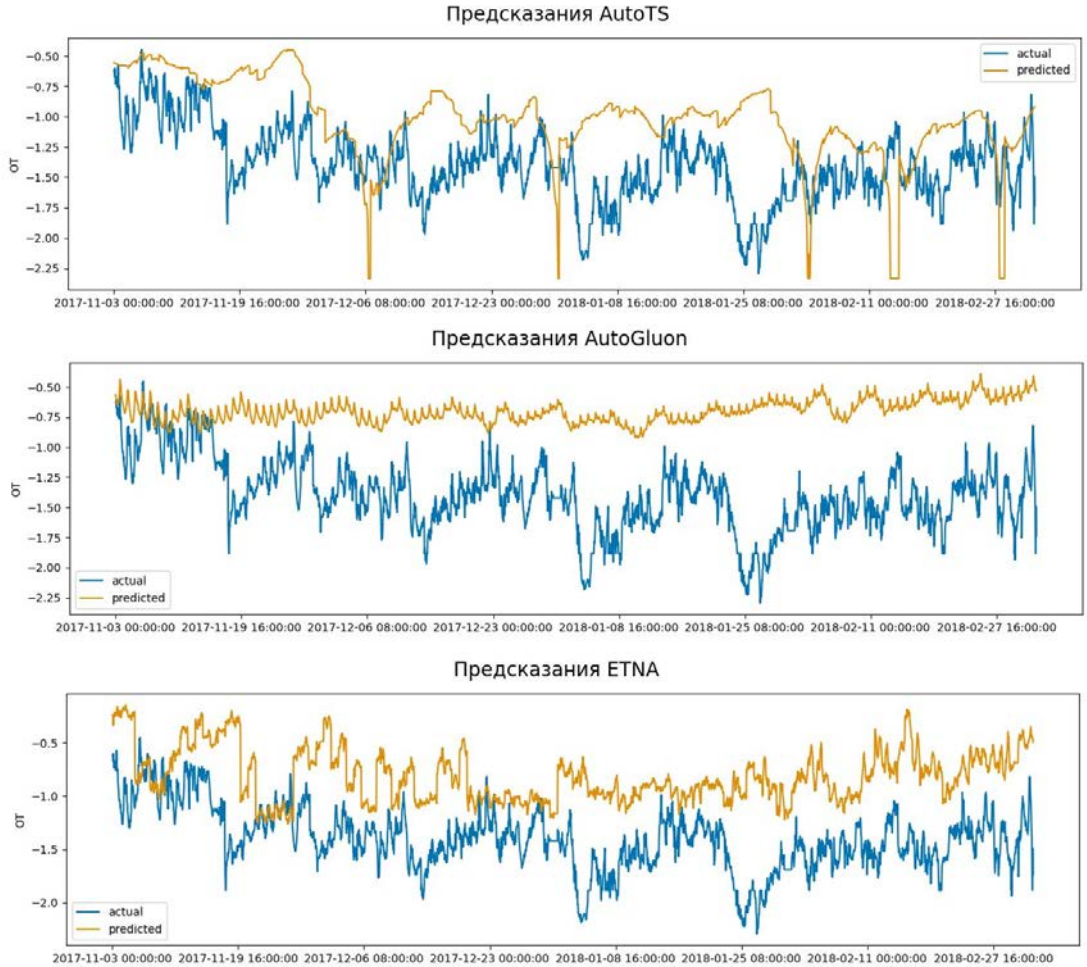

Рисунок 3 – Сравнение предсказаний с реальными данными

3.2 Интерпретация результатов

AutoTS продемонстрировал самое долгое время обучения и самое большое количество потребляемой памяти. Поиск оптимальной модели занимал от 3,7 до 4,9 часов. Значение метрики MSE для полученного с помощью AutoTS конвейера составило 0,475. Потребление памяти в пике составило 28 635 Мб. Изменения в объеме требуемой оперативной памяти представлены на Рисунке 4.

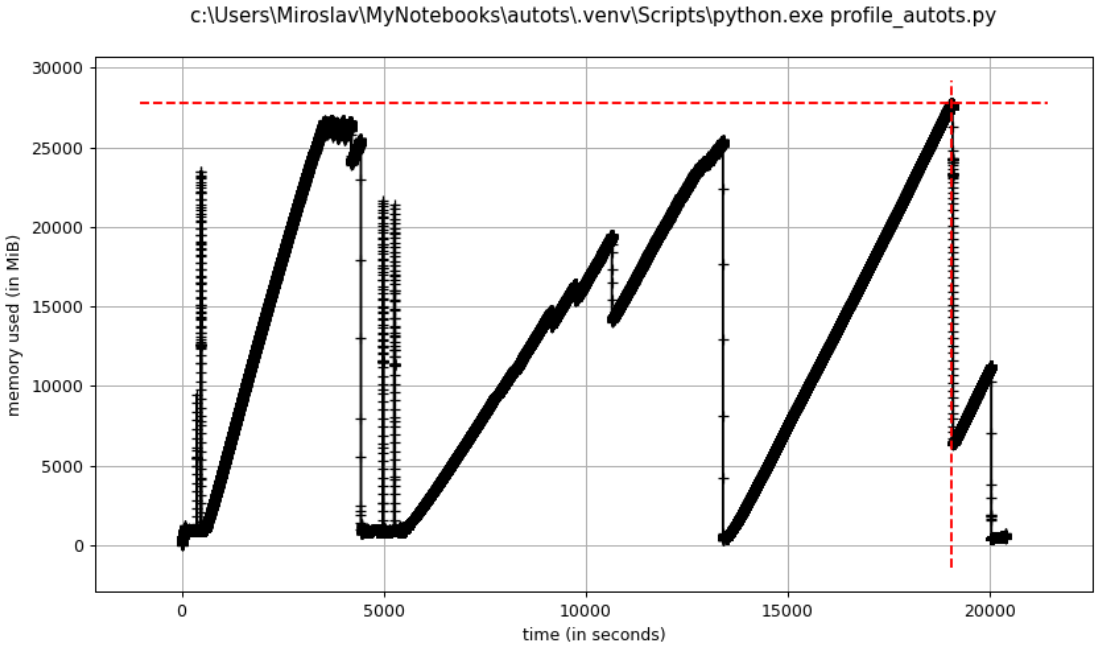

Рисунок 4 – Характер изменения потребляемой памяти в AutoTS

Пилообразный характер нагрузки обусловлен используемым алгоритмом. AutoTS полагается на генетическое программирование. Постепенный рост нагрузки связан с генерацией большого количества моделей в новой популяции на текущей эпохе (150 моделей на одно поколение), резкие падения потребления связаны окончанием определенной эпохи генетического алгоритма.

Было замечено, что, при лимите в 12 часов, время самого долгого эксперимента не превысило 5 часов и ни одна из эпох эволюционного алгоритма не израсходовала положенных 3-х часов времени. Результаты AutoTS можно улучшить путем увеличения количества поколений генетического алгоритма и уменьшения времени на одну эпоху, однако это негативно скажется на объёме потребляемых ресурсов. При использовании 6 эпох по 2 часа значение MSE составило 0,32, однако была израсходована вся оперативная память и время обучения превысило допустимый лимит в 12 часов.

Можно сделать вывод, что использование AutoTS оправдано при наличии большого количества времени и вычислительных ресурсов. Эффективность применяемого алгоритма растет с увеличением количества эпох, однако фреймворк предъявляет высокие требования к аппаратному обеспечению и в краткосрочной перспективе получаемое решение может уступать решениями из других пакетов.

Потребление памяти в эксперименте с AutoGluon было ниже, но оставались достаточно высоким (12 582 Мб). Обучение завершалось в течении 10-12 минут, но результаты подобранной модели оказались хуже, чем у конвейера из предыдущего эксперимента. Изменения в объеме требуемой оперативной памяти представлены на Рисунке 5. Большую часть времени потребление не превышало 4000 Мб.

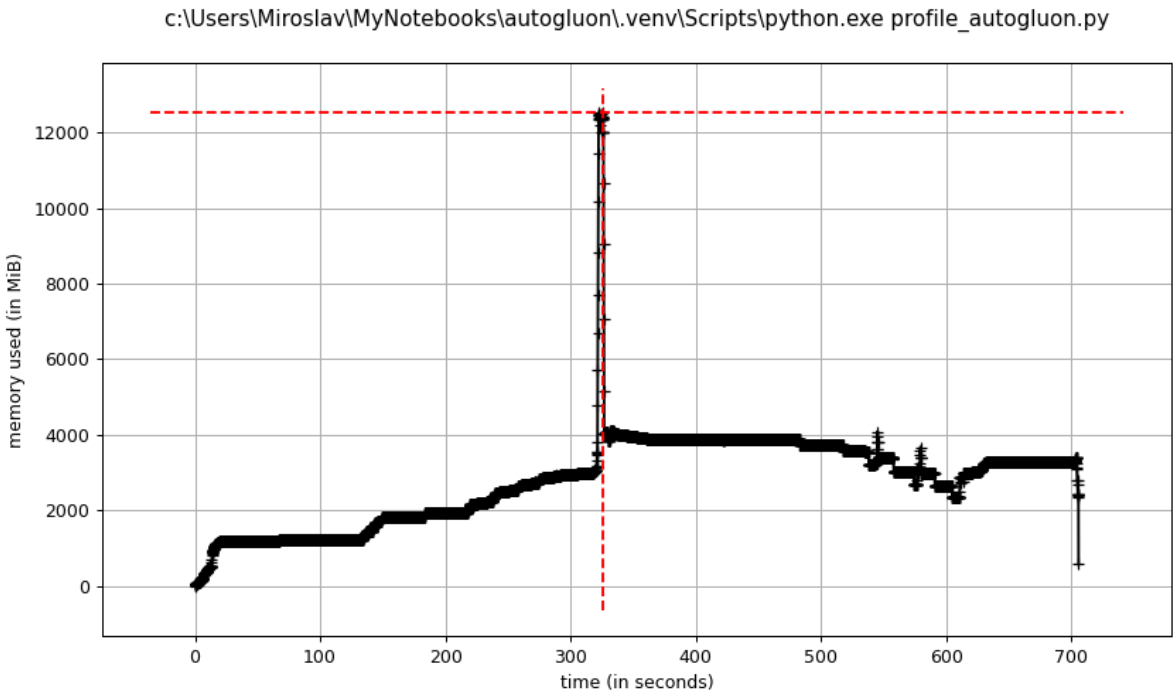

Рисунок 5 – Характер изменения потребляемой памяти в AutoGluon

Выбранный авторами AutoGluon алгоритм оказался более экономичным. Вместо создания множества разных моделей и глубокой подгонки каждой из них, метод полагается на построение ансамбля из базовых моделей со стандартными наборами гиперпарметров. После подбора моделей нижнего уровня начинается обучение агрегирующих моделей (стекеров), которые представляют собой такие же базовые модели. Особенность стекеров состоит в том, что они имеют доступ не только к предсказаниям моделей уровнем ниже, но и к изначальным входным данным (skip-connection). По строению получаемый на выходе многослойный ансамбль во многом схож с нейронной сетью. Отдельными нейронами выступают либо не подвергавшиеся глубокой оптимизации базовые модели, либо модели стекеры. За счет меньшего количества вариантов для перебора метод оказался значительно быстрее, однако из-за недостаточно тонкой настройки каждой базовой модели значение метрики MSE вышло хуже, чем у AutoTS. Пример структуры получаемой модели WeightedEnsemble представлен на Рисунке 6.

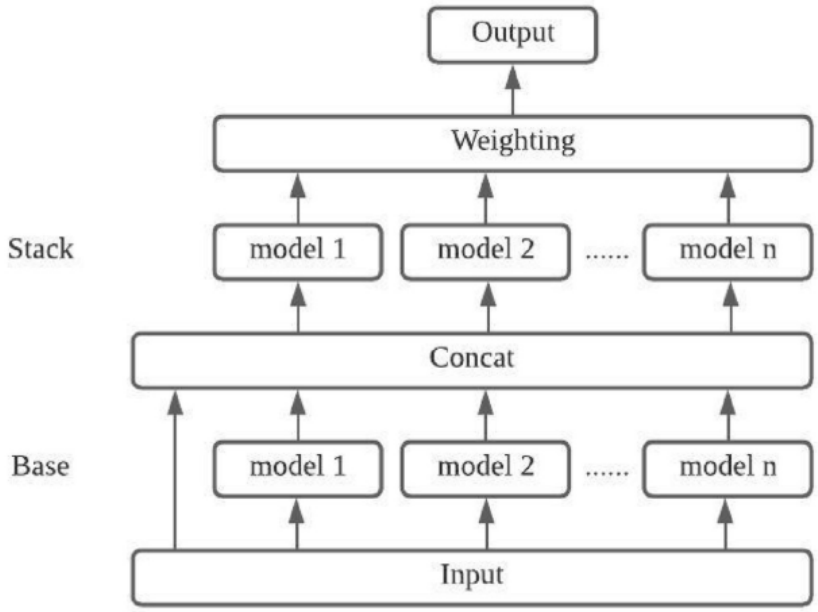

Рисунок 6 – Структура WeightedEnsemble

Самым быстрым оказалось решение пакета ETNA. За 4-5 минут была подобрана модель по метрике MSE не уступавшая решению из AutoTS. Пиковое потребление памяти составило 1285 Мб, меньше, чем среднее в двух предыдущих экспериментах. Изменения в объеме требуемой оперативной памяти представлены на Рисунке 7. Следует заметить, что в отличие от предыдущих решений, конвейер, получаемый через ETNA, не является уже обученным и требует обучения перед непосредственным использованием. Также необходимо учесть, что в отличие от AutoTS и AutoGluon, ETNA не полагается на собственное решение для оптимизации.

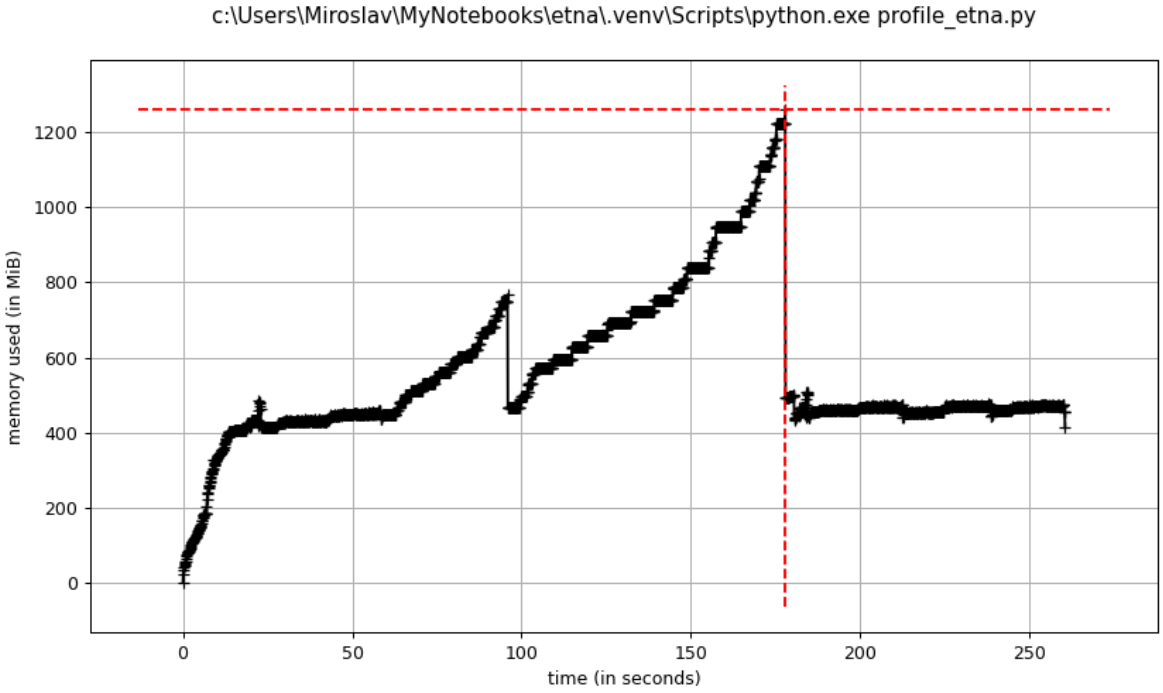

Рисунок 7 – Характер изменения потребляемой памяти в ETNA

В пакете ETNA поиск оптимальной конфигурации гиперпараметров делегирован фреймворку Optuna [15]. Используемый метод Sequential Model-Based Optimization (SMBO) основан на байесовской оптимизации. При выборе лучшей области пространства гиперпараметров учитываются исторические данные, полученные значения целевой функции для проверенных точек, где уже были обучены модели.

Основными компонентами являются суррогатная модель и функция сбора.

Строится предположение о распределении значений гиперпараметров. После обучения модели с определенной конфигурацией гиперпараметров и оценки качества происходит обновление суррогатной модели.

Для выбора следующей конфигурации применяется функция сбора, которая учитывает оценки предыдущих построенных моделей и выбирает набор гиперпараметров, который с наибольшей вероятностью будет оптимальным.

Для сокращения пространства поиска применяется двухэтапный алгоритм усечения, который удаляет области, в которых качество модели оказалось ниже медианы из уже рассмотренных. На первом этапе происходит отслеживание промежуточных значений цели, на втором этапе проверка обозначенных условий и прекращение признанных «бесперспективными» испытаний. Применяемый алгоритм Asynchronous Successive Halving может эффективно распараллеливаться.

В целом, использование AutoGluon и ETNA может быть оправдано, если решение требуется подготовить за сравнительно небольшое время. Конвейеры, полученные с помощью AutoGluon и ETNA могут уступать моделям из AutoTS по качеству прогнозов, однако время поиска и объем необходимых ресурсов оказываются зачастую ниже.

Поскольку в эксперименте с Fedot не удалось добиться стабильного воспроизведения результата, данные для Fedot в таблицах отсутствуют. Нестабильность была связана с используемым в библиотеке алгоритмом поиска конвейера.

Осуществлялся поиск композитной модели. Структура итоговой композитной модели представлялась в виде направленного ациклического графа (DAG) узлами которого выступают модели машинного обучения или процедуры обработки данных. В процессе обучения подбиралась оптимальная структура и параметры для каждого узла. С точки зрения эволюционной оптимизации генотипом выступала структура графа, а фенотипом - интегральные характеристики конкретной реализации. В начале создавалась популяция моделей, структура которых случайна. При помощи метода предпочтительного отбора определялись лучшие из кандидатов, и на их основе путем применения кроссинговера или мутаций строились новые модели. После обучения и проверки на тестовой выборке новые модели формировали новую популяцию и процесс повторялся. Для повышения степени автоматизации используется беспарамтерическая многоцелевая генетическая схема GPComp@Free позволяющая регулировать размер

популяции, скорость кроссинговера и мутации в процессе эволюции [33]. Схема предложенного алгоритма представлена на Рисунке 8.

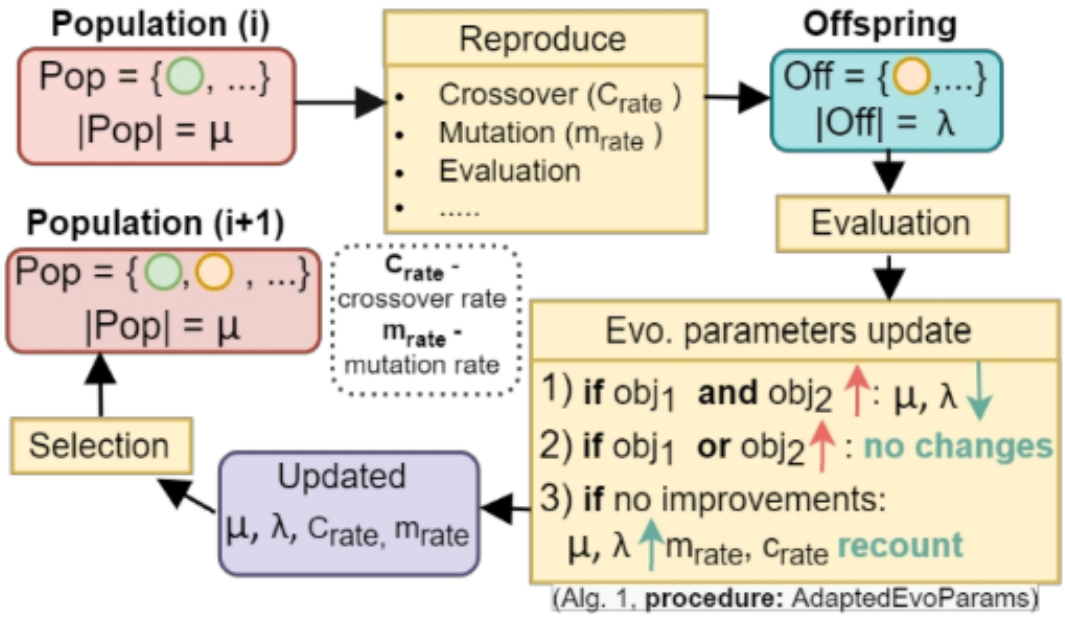

Рисунок 8 – Одна итерация эволюционного алгоритма [33]

Эволюционный алгоритм имеет стохастическую природу и в процессе своей работы опирается на значение гиперпараметра seed, который либо вручную определяется пользователем до начала обучения, либо генерируется случайным образом [45]. От значения seed зависят структура сгенерированных в первом поколении конвейеров и результат случайных мутаций. При случайной генерации итоговый используемый seed не выводится в логи в явном виде, что усложняет дальнейшие повторение эксперимента с таким же значением seed.

Ручной поиск значения seed не увенчался успехом, подобрать значение, при котором обучение успешно завершалось в обозначенные 12 часов не удалось. Seed может быть подобран экспериментально, т.е. путем многократного перезапуска алгоритма с заранее сгенерированным значением, однако данный метод обладает случайным временем сходимости и

неэффективен в плане затрат (фиксация неудачи по истечении лимита времени в 12 часов).

Кроме того, подобный подход не согласуется с самой идеологией AutoML. Чтобы автоматическое машинное обучение работало необходимо осуществить аналогичный RandomSearch перебор значений определенного гиперпараметра, в то время как цель AutoML – избавление от подобного ручного перебора.

В качестве частичного решения данной проблемы при использовании Fedot допустимо явно генерировать случайный seed и фиксировать его згначение с самого начала экспериментов.

С точки зрения доработки функционала пакета, можно добавить вывод значения seed в логи в начале обучения и после окончания обучения.

Также допустима запись использованного значения в файл во время сохранения шаблона полученного конвейера.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены современные подходы к задаче оптимизации гиперпараметров в AutoML фреймворках для прогнозирования временных рядов. В частности, представлен обзор методов, основанных на уже устоявшихся традиционных подходах, и передовых решений в области оптимизации гиперпараметров.

В существующих научных работах сравнение качества получаемых решений зачастую не учитывает алгоритма оптимизации, используемого в каждом конкретном случае.

В ходе работы было проведено несколько экспериментов для сравнения качества предсказаний конвейеров получаемых с помощью пакетов автоматизированного машинного обучения. При анализе результатов повышенное внимание было уделено особенностям работы фреймворков области оптимизации гиперпараметров.

По итогу проводимых в данной работе исследований, был сделан вывод о том, что пакеты автоматического машинного обучения вполне успешно могут применяться для прогнозирования временных рядов. Тем не менее, при выборе фреймворка для конкретной задачи необходимо учитывать индивидуальные особенности, принципы по которым функционирует пакет.

При наличии большого количества времени и памяти можно воспользоваться фреймворком AutoTS. В пакете имеется поддержка множества современных моделей прогнозирования, для поиска оптимальной модели применяется генетический алгоритм, который тяжел в вычислительном плане. В краткосрочной перспективе находимые конвейеры либо сравнимы, либо уступают решениям, найденным другими фреймворками, однако с увеличением числа поколений алгоритмом будут подбираются конвейеры, способные обеспечить наилучшее качество прогнозов.

Если же необходимо быстро найти сравнительно неплохое решение, а объем вычислительных ресурсов и времени сильно ограничены, можно воспользоваться функционалом AutoGluon и ETNA.

AutoGluon полагается на построение ансамблей из базовых моделей без их глубокой настройки. Применение данного подхода положительно сказывается на расходе ресурсов и времени, однако из-за недостаточно гибкой настройки, получаемый конвейер может уступать решениям из других пакетов.

В процессе работы ETNA полагается на реализованную в Optuna Байесовскую оптимизацию. Пространство гиперпараметров сокращается за счет применения двухэтапного алгоритма усечения, который удаляет области пространства, где качество модели оказалось ниже медианного показателя уже проведенных испытаний. В экспериментах с набором данных ETTh1 найденные ETNA конвейеры продемонстрировали наилучшее качество прогнозов при наименьшем времени обучения.

Наконец, для поиска конвейера, можно воспользоваться пакетом Fedot. В пакете применяется генетический алгоритм, эффективность работы которого во многом зависит от конфигураций конвейеров, сгенерированных случайным образом в начале работы и от результатов применения случайных мутаций. Для воспроизводимости результатов потребуется подобрать экспериментальным путем начальное значение для алгоритма генерации случайных чисел и зафиксировать значение при запуске обучения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Conrad, F. Benchmarking AutoML for regression tasks on small tabular data in materials design // Scientific Reports. 2022. Vol. 12. № 1. – P. 19350.
2. 1.1. Linear Models : сайт. – URL: https://scikit-learn/stable/modules/linear_model.html (дата обращения: 27.05.2024) – Текст: электронный.
3. All-in-One & AutoML : сайт – Time Series All-in-One Python Packages. – URL: https://opentimeseries.com/python_packages/all_in_one/opentimeseries.com/python_packages/all_in_one/ (дата обращения: 27.05.2024) – Текст: электронный.
4. AutoML | Meta-Learning : сайт. – URL: https://www.automl.org/meta-learning/ (дата обращения: 27.05.2024) – Текст: электронный.
5. AutoTS/extended_tutorial.md at master · winedarksea/AutoTS : сайт. – URL: https://github.com/winedarksea/AutoTS/blob/master/extended_tutorial.md (дата обращения: 27.05.2024) – Текст: электронный.
6. CompEngine : сайт – A self-organizing database of time-series data. – URL: https://www.comp-engine.org/ (дата обращения: 27.05.2024) – Текст: электронный.
7. Forecasting Time Series - Model Zoo : сайт. – URL: https://auto.gluon.ai/tutorials/timeseries/forecasting-model-zoo.html (дата обращения: 24.05.2024) – Текст: электронный.
8. Meta Learning (AutoML) | Machine & Deep Learning Compendium : сайт. – URL: https://oricohen.gitbook.io/machine-and-deep-learning-compendium/machine-learning/meta-learning (дата обращения: 27.05.2024) – Текст: электронный.
9. StandardScaler : сайт. – URL: https://scikit-learn/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (дата обращения: 24.05.2024) – Текст: электронный.

10. TimeSeriesSplit : сайт. – URL: https://scikit-learn/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html (дата обращения: 27.05.2024) – Текст: электронный.
11. venv — Creation of virtual environments : сайт. – URL: https://docs.python.org/3/library/venv.html (дата обращения: 27.05.2024) – Текст: электронный.
12. Анафиев, А. С. Обзор подходов к решению задачи оптимизации гиперпараметров для алгоритмов машинного обучения / А.С. Анафиев, А.С. Карюк – Текст : непосредственный. // Таврический вестник информатики и математики. 2022. № 2 (55). – С. 30-37.
13. Матвеев, А. Н. Инструменты построения моделей машинного обучения / А.Н. Матвеев – Текст : непосредственный. // E-Scio. 2023. № 6 (81). – С. 71-77.
14. Abdallah, M. AutoForecast: Automatic Time-Series Forecasting Model Selection // Proceedings of the 31st ACM International Conference on Information & Knowledge Management : CIKM ’22. – New York, NY, USA: Association for Computing Machinery, 2022. AutoForecast. – С. 5-14.
15. Akiba, T. Optuna: A Next-generation Hyperparameter Optimization Framework. 2019. Optuna.
16. (PDF) Review of ML and AutoML Solutions to Forecast Time-Series Data A. Alsharef, K. Aggarwal, Sonia [и др.]. – URL: https://www.researchgate.net/publication/361026590_Review_of_ML_and_AutoML_Solutions_to_Forecast_Time-Series_Data (дата обращения: 21.05.2024) – Текст: электронный.
17. Ansari, A. F. Chronos: Learning the Language of Time Series. 2024. Chronos.
18. Christ, M. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package) // Neurocomputing. 2018. Т. 307. – С. 72-77.
19. Conrad, F. AutoML Applied to Time Series Analysis Tasks in Production Engineering : 5th International Conference on Industry 4.0 and Smart

Manufacturing (ISM 2023) // Procedia Computer Science. 2024. Т. 232. – С. 849-860.
20. Deng, D. Efficient Automated Deep Learning for Time Series Forecasting. 2022.
21. Erickson, N. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. 2020. AutoGluon-Tabular.
22. Fischer, R. AutoXPCR: Automated Multi-Objective Model Selection for Time Series Forecasting / R. Fischer, A. Saadallah – Текст : непосредственный. 2023. AutoXPCR.
23. Gedminas, M. mgedmin/profilehooks : сайт / M. Gedminas. – 2024.
24. Gijsbers, P. An Open Source AutoML Benchmark. 2019.
25. Hanussek, M. Can AutoML outperform humans? An evaluation on popular OpenML datasets using AutoML Benchmark / M. Hanussek, M. Blohm, M. Kintz – Текст : непосредственный. 2020. Can AutoML outperform humans?
26. Haoyi. zhouhaoyi/ETDataset / Haoyi. – 2024.
27. Jin, H. Auto-Keras: An Efficient Neural Architecture Search System / H. Jin, Q. Song, X. Hu – Текст : непосредственный. 2019. Auto-Keras.
28. Lim, B. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. 2020.
29. Lindauer, M. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization.Т. 23.SMAC3 / M. Lindauer, K. Eggensperger, M. Feurer [и др.]. – 2022.
30. Engineering Proceedings | Free Full-Text | Does AutoML Outperform Naive Forecasting? G. Marco Paldino, J. De Stefani, F. De Caro, G. Bontempi. – URL: https://www.mdpi.com/2673-4591/5/1/36 (дата обращения: 21.05.2024) – Текст: электронный.
31. Meisenbacher, S. Review of automated time series forecasting pipelines // WIREs Data Mining and Knowledge Discovery. 2022. Т. 12. № 6. – С. e1475.
32. Nie, Y. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. 2023. A Time Series is Worth 64 Words.

33. Polonskaia, I. S. Multi-Objective Evolutionary Design of Composite Data-Driven Models. 2021.
34. Qi, W. AutoGluon: A revolutionary framework for landslide hazard analysis / W. Qi, C. Xu, X. Xu – Текст : непосредственный. // Natural Hazards Research. 2021. Т. 1. AutoGluon. № 3. – С. 103-108.
35. Shah, S. Y. AutoAI-TS: AutoAI for Time Series Forecasting. 2021. AutoAI-TS.
36. Shchur, O. AutoGluon-TimeSeries: AutoML for Probabilistic Time Series Forecasting. 2023. AutoGluon-TimeSeries.
37. Engineering Proceedings | Free Full-Text | Towards Time-Series Feature Engineering in Automated Machine Learning for Multi-Step-Ahead Forecasting T. Thomas Bäck, H. Hoos, M. Olhofer [и др.]. – URL: https://www.mdpi.com/2673-4591/18/1/17 (дата обращения: 21.05.2024) – Текст: электронный.
38. Truong, A. Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools // 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI) / arXiv:1908.05557 [cs, stat]. 2019. Towards Automated Machine Learning. – С. 1471-1479.
39. Waring, J. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare / J. Waring, C. Lindvall, R. Umeton – Текст : непосредственный. // Artificial Intelligence in Medicine. 2020. Т. 104. Automated machine learning. – С. 101822.
40. Xu, Z. AutoML Meets Time Series Regression Design and Analysis of the AutoSeries Challenge / Z. Xu, W.-W. Tu, I. Guyon – Текст : непосредственный. 2021.
41. Zhou, H. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. 2021. Informer.
42. Zöller, M.-A. auto-sktime: Automated Time Series Forecasting / M.-A. Zöller, M. Lindauer, M.F. Huber – Текст : непосредственный. 2024. auto-sktime.

43. 11 Classical Time Series Forecasting Methods in Python (Cheat Sheet) : сайт. – URL: https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/ (дата обращения: 28.05.2024) – Текст: электронный.
44. autots package — AutoTS 0.6.13 documentation : сайт. – URL: https://winedarksea.github.io/AutoTS/build/html/source/autots.html#autots.AutoTS (дата обращения: 24.05.2024) – Текст: электронный.
45. FEDOT API — FEDOT 0.7.3.2 documentation : сайт. – URL: https://fedot.readthedocs.io/en/latest/api/api.html#fedot.api.main.Fedot (дата обращения: 24.05.2024) – Текст: электронный.
46. pythonprofilers/memory_profiler : сайт. – pythonprofilers, 2024.
47. What is the Difference Between Test and Validation Datasets? : сайт. – URL: https://machinelearningmastery.com/difference-test-validation-datasets/ (дата обращения: 27.05.2024) – Текст: электронный.