МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт№8 «Информационные технологии и прикладная математика» **Кафедра** 810Б

Направление подготовки 02.04.02 ФИИТ **Группа** М8О-203М-18

Квалификация (степень) магистр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

На тему: Разработка модели прогнозов загрузки узлов логистической сети

Автор диссертации Чебенева Екатерина Алексеевна _____
(Фамилия, имя, отчество) подпись

Научный руководитель Ревизников Дмитрий Леонидович _____
(Фамилия, имя, отчество) подпись

Рецензент Хатунцева Ольга Николаевна _____
(Фамилия, имя, отчество) подпись

К з а щ и т е д о п у с т и т ь

Зав. кафедрой Абгарян Каринэ Карленовна _____
(Фамилия, инициалы) подпись

« 24 » мая 2020 г.

Москва 2020 г.

РЕФЕРАТ

Магистерская диссертация содержит 59 страниц, 20 рисунков, 4 таблицы, 19 использованных источников.

МОДЕЛИ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ВРЕМЕННЫХ РЯДОВ, ЛОГИСТИЧЕСКАЯ СЕТЬ, ПРЕДСКАЗАНИЯ ВРЕМЕННЫХ РЯДОВ, ЯЗЫК SCALA, ЯЗЫК PYTHON, APACHE SPARK.

В данной магистерской диссертации проведен анализ, реализована и протестирована модель для прогнозирования количества посылок в узле для последующего принятия решений по регулированию нагрузки на логистический узел.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5

ОСНОВНАЯ ЧАСТЬ 7

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 8

1.1 Термины 8

1.2 Описание целевой переменной 8

1.3 Структура данных 10

1.4 Стек используемых технологий 14

1.4.1 Python 14

1.4.2 Apache Spark 15

1.5 Временные ряды 16

1.6 Описание моделей прогнозирования. Проверка остатков 19

1.6.1 ARIMA 21

1.6.2 SARIMA 24

1.6.3 Градиентный бустинг 25

1.6.4 XGBoost 28

1.6.5 LightGBM 28

1.6.6 Prophet 29

1.6.7 LSTM 30

1.6.8 Проверка остатков моделей 31

1.7 Выбор метрик 32

1.7.1 Средняя абсолютная ошибка 32

1.7.2 Среднеквадратичная ошибка 32

1.7.3 Информационный критерий Акайке 33

2 ПРАКТИЧЕСКАЯ ЧАСТЬ 35
2.1 Архитектура решения 35
2.2 Генерирование признаков 36
2.3 Результаты 44
ЗАКЛЮЧЕНИЕ 57
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 58

ВВЕДЕНИЕ

На данный момент правила обработки посылок, участвующих в логистической сети, построены таким образом, что в аэропортах может скапливаться большое количество отправлений, которые становится сложно обработать в короткие сроки. Данная работа может помочь регулировать количество посылок в аэропорту, для избегания залежей большого количество посылок, например, перед праздничными днями. Была поставлена задача разработать модель, позволяющую предсказывать будущее значение количества посылок в аэропорту относительно следующего объекта назначения. Например, задав аэропорт и следующий объект, куда направляется посылка, вывести временные промежутки на ближайшие три дня и количество посылок, которые будут находиться в аэропорту относительно следующего места назначения.

Данная задача была решена путем использования фреймворка Apache Spark, который позволяет быстро обрабатывать большие потоки данных и содержит библиотеки для машинного обучения, предоставляющие различные алгоритмы. В работе было опробовано несколько подходов к прогнозированию временных рядов и был выбран градиентный бустинг в качестве итоговой модели, который в последствии был реализован на Spark.

Цель работы: реализовать прогнозирование количества посылок в узле для последующего принятия решений по регулированию нагрузки на логистический узел.

Для достижения данной цели, необходимо решить ряд задач:

- провести анализ существующих подходов к прогнозированию временных рядов
- выполнить очистку и предобработку данных
- реализовать и протестировать модели
- выбрать лучшую модель на основе отобранных метрик

- интегрировать выбранную модель в производственный процесс

ОСНОВНАЯ ЧАСТЬ

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Термины

Логистическая сеть (рисунок 1.1) – это большое количество звеньев логистической системы, находящихся во взаимосвязи между собой по материальным или сопутствующим им информационным и денежным потокам в границах логистической системы [1].

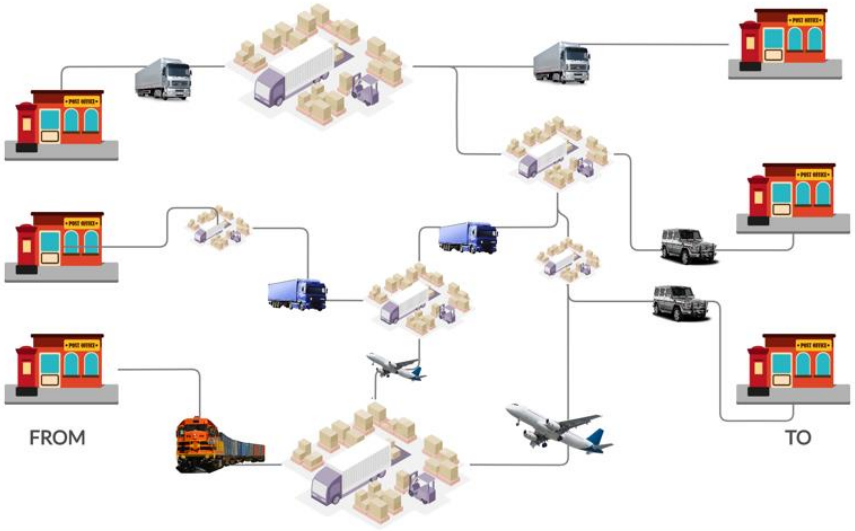

Рисунок 1.1 – Пример логистической сети

РПО – регистрируемое почтовое отправление. Это любая посылка в почтовой сети, имеющая идентификатор.

АОПП – авиационное отделение перевозки почты.

ОПС – объект почтовой связи, на рисунке 1 представляет собой узел графа, например, аэропорт (АОПП), сортировочные центы.

Плечо – это ребро между двумя ОПС.

1.2 Описание целевой переменной

В качестве целевой переменной выступают остатки РПО в узлах логистической сети. Узлами логистической сети в данной работе являются московские АОПП (Домодедово АОПП, Внуково АОПП, Шереметьево АОПП).

В АОПП есть приход (accepted) и уход (left), показаны на рисунке 1.2. Для расчета целевой переменной сделаем предположение, что хотя бы раз за период 6 месяцев происходит обнуление остатков, то есть все РПО, которые необходимо отправить на плечо – отправлены.

Рисунок 1.2 – Пример прихода и ухода на плечо

Формула расчета целевой переменной:

$$residuals = \max(left - accepted) + accepted - left$$

Относительно каждого АОПП отбирается следующее место назначения, которое не должно находится в одном макрорегионе с АОПП, важно, чтобы РПО было отправлено именно воздушным способом, также в качестве ограничения служит количество РПО на плече за 6 месяцев, отобраны те плечи, у которых за период было более 100000 РПО.

Остатки считаются по временной сетке 8 часов. Задача стоит в предсказании количества остатков на ближайшие три дня.

Пример целевой переменной представлен на рисунке 1.3.

```
times_dt
2019-11-13 16:00:00    19560
2019-11-14 00:00:00    18325
2019-11-14 08:00:00    17990
2019-11-14 16:00:00    18424
2019-11-15 00:00:00    18928
2019-11-15 08:00:00    18252
2019-11-15 16:00:00    18741
2019-11-16 00:00:00    18867
2019-11-16 08:00:00    10485
2019-11-16 16:00:00     8617
2019-11-17 00:00:00    19825
Name: residuals_new, dtype: int64
```

Рисунок 1.3 – Пример целевой функции

1.3 Структура данных

В работе используется исходная ненормализованная таблица matreshka_grouped, содержащая 50 миллионов записей РПО, далее таблица разворачивается и отбираются необходимые столбцы, таким образом получается таблица raw_data, которая не содержит типы struct и array.

Далее, необходимо было получить информацию о том, куда направляется РПО после АОПП. Для этого были получены записи по каждому РПО в виде:

<РПО 1> <Объект1> <Время начала выполнения операций в Объекте1>…

<РПО 1> <Объект1> <Время окончания выполнения операций в Объекте1>…

<РПО 1> <Объект2> <Время начала выполнения операций в Объекте2>…

<РПО 1> <Объект2> <Время окончания выполнения операций в Объекте2>…

В последствии, данные были свернуты в вид:

<РПО 1><Объект1><Время начала выполнения операций в Объекте1><Время окончания выполнения операций в Объекте1> <Предыдущий Объект (в данном примере null)>> <Следующий объект почтовой связи (в данном примере Объект2)>…

<РПО 1><Объект2><Время начала выполнения операций в Объекте2><Время окончания выполнения операций в Объекте2> <Предыдущий Объект (в данном примере Объект1)>> <Следующий объект почтовой связи (в данном примере null)>…

В таблице matreshka_grouped содержаться исходные данные, основные столбцы в данной таблице:

1. bar_code – штриховой почтовый идентификатор (ШПИ) регистрируемого почтового отправления (РПО), 16-ти значный идентификатор документа (уникальное значение)
2. opers (тип array):
    - oper_date_time – время выполнения операции над ШПИ

- index_oper – индекс объекта, в котором выпоняется операция
- index_to – индекс назначения (индекс объекта в который следует документ)
- document_form – тип документа
- oper_type – тип операции (примеры: 1001 - Вскрытие, 1 - Прием, 1018 - Отправка)
- oper_attr – атрибут операции (примеры: 20 - Покинуло место возврата/досылки, 2 - Невостребовано)
- type_of_operand1 – тип объекта, над которым совершается операция (1 - РПО, 2 - емкость, 3 - документ)
- id_of_operand1 – id объекта, над которым совершается операция
- country_oper – страна операции

3. rpo_info (тип struct)
   - index_from_rf – почтовый индекс ОПС приема/импорта в РФ
   - country_from – код страны подачи международного отправления, в соответствии со справочником кодов стран пересылки почтовых отправлений
   - country_to – код страны, в направлении которого подано международное отправление, в соответствии со справочником кодов стран пересылки почтовых отправлений
   - index_to – почтовый индекс ОПС адресата
   - inter_type – код типа международного отправления
   - insr_rate – сумма платы за объявленную ценность в копейках
   - air_rate – сумма платы за пересылку воздушным транспортом
   - trans_type – способ пересылки
   - mail_type – код вида отправления
   - mail_ctg – код категории почтового отправления
   - mail_rank – код разряда почтового отправления

- send_ctg – категория отправителя
- post_mark – отметки РПО
- mass – вес отправления в граммах
- mass_rate – общая сумма платы за пересылку наземным и воздушным транспортом в копейках
- ad_val_tax – сумма налога на добавленную стоимость в копейках
- rate – дополнительный тарифный сбор в копейках
- custom_duty – сумма таможенной пошлины в копейках
- payment – сумма наложенного платежа в копейках
- value – сумма объявленной ценности в копейках
- pay_type – код способа и форм оплаты
- direct_ctg – классификации отправления
- volume_weight – объемный вес отправления в граммах
- send_address.type – тип адреса отправителя
- recv_address.type – тип адреса получателя

Ниже приведены расшифровки операций.

Способ пересылки (trans_type):

- Наземный 1
- Авиа 2
- Комбинированный 3
- Системой ускоренной почты 4
- Электронный 5
- Стандарт 6

Категория отправителя (send_ctg):

- Категория отправителя
- Население 1
- Бюджетная организация 2
- Хозрасчетная организация 3
- Международный оператор 4

- Корпоративный клиент 5
- Почтовый оператор 6

Столбцы таблицы dry_data: bar_code, index_from_rf, index_to_rpo, inter_type, insr_rate, air_rate, trans_type, mail_type, mail_ctg, mail_rank, send_ctg, post_mark, mass, mass_rate, ad_val_tax, rate, payment, value, pay_type, index_oper_new, document_form, oper_type, oper_attr, type_of_operand1, id_of_operand1, start_dt, end_dt, prev, nex, first_index, first_dt, air_form, statuses_sum, processed_time.

Описание столбцов:

- start_dt – время начала выполнения операции в ОПС
- end_dt – время окончания выполнения операции в ОПС
- prev – предыдущий объект
- nex – следующий объект
- first_index – первый объект в трассировке
- first_dt – время окончания операции в первом объекте трассировке
- air_form – количество раз, когда в трассировке РПО встречаются значения document_form связанные с отправкой на рейс
- statuses_sum – количество раз, когда в трассировке РПО встречаются операции, связанные с возвратом
- processed_time – время выполнении операций в объекте

В таблице dry_data в результате собраны все комбинации (РПО – объекты, в которых находилось РПО).

Пример данных в таблице matreshka_grouped:

ШПИ	array<struct<ДатаВремя, Объект, Тип операции...>>	struct<Почтовый индекс приема, Способ пересылки, Сумма платы за пересылку воздушным транспортом,...>
1400159391235660	<<Дата1 10:20, Люберцы 140015, Прием...>, <Дата1 10:45, Люберцы 140015, Отправка..>, <Дата1 18:20, Домодедово АОПП, Прием...> <Дата1 19:20, Домодедово АОПП, Сортировка...> <Дата1 22:20, Домодедово АОПП, Отправка...> <Дата2 9:05, Иркутск АОПП, Прием...>...>	<Люберцы 140015, Авиа, 120р.>
3131293500043133	<<Дата1 10:21, Сургут 628402, Прием...>, <Дата1 12:45, Сургут 628402, Отправка..>, <Дата1 20:20, Сортировочный центр, Прием...>...>	<Сургут 628402, Наземный, 0>

Далее таблица разворачивается, избавляемся от типов struct и array, находим следующее место назначения, прошлый объект, время начала и окончания операций в текущем объекте. Исходные данные, приведенные выше, преобразуются в следующий вид:

ШПИ	Объект	Время начала выполнения операций	Время окончания операций	Следующий объект	Предыдущий объект	Почтовый индекс приема	Способ пересылки	Сумма платы за пересылку воздушным транспортом
1400159391235660	Люберцы 140015	Дата1 10:20	Дата1 10:45	Домодедово АОПП	null	Люберцы 140015	Авиа	120р.
1400159391235660	Домодедово АОПП	Дата1 18:20	Дата1 22:20	Иркутск АОПП	Люберцы 140015	Люберцы 140015	Авиа	120р.
1400159391235660	Иркутск АОПП	Дата2 9:05	...	null	Домодедово АОПП	Люберцы 140015	Авиа	120р.
3131293500043133	Сургут 628402	Дата1 10:21	Дата1 12:45	Сортировочный центр	null	Сургут 628402	Наземный	0р.
3131293500043133	Сортировочный центр	Дата1 20:20	...	null	Сургут 628402	Сургут 628402	Наземный	0р.

При условии рассмотрении плеча Домодедово АОПП – Иркутск АОПП получаем только одну интересующую запись в таблице:

ШПИ	Объект	Время начала выполнения операций	Время окончания операций	Следующий объект	Предыдущий объект	Почтовый индекс приема	Способ пересылки	Сумма платы за пересылку воздушным транспортом
1400159391235660	Домодедово АОПП	Дата1 18:20	Дата1 22:20	Иркутск АОПП	Люберцы 140015	Люберцы 140015	Авиа	120р.

В результате целевая переменная выглядит следующим образом:

Дата время	Кол-во РПО ДомодедовоАОПП-ИркутскАОПП
Дата1 0:00	...
Дата1 8:00	...
Дата1 16:00	1
Дата2 0:00	...

1.4 Стек используемых технологий

1.4.1 Python

Python – это интерпретируемый объектно-ориентированный язык программирования высокого уровня с динамической семантикой [2]. Python поддерживает модули и пакеты, что способствует модульности программы и повторному использованию кода. Интерпретатор Python и обширная стандартная библиотека доступны в исходном или двоичном виде бесплатно для всех основных платформ и могут свободно распространяться.

Причиной растущего успеха Python является наличие постоянно обновляющихся библиотек для обработки и анализа данных. Многие библиотеки доступны для анализа данных только на Python:

- NumPy важен для выполнения научных вычислений с Python. Он включает в себя ассортимент математических функций высокого уровня для работы с многомерными массивами и матрицами [3].
- Scikit-learn предоставляет широкий выбор алгоритмов обучения с учителем и без учителя [4].
- Pandas – библиотека, разработанная на основе NumPy для обработки и анализа данных, предоставляет структуры данных и операции для изменения числовых таблиц и временных рядов [5].
- Matplotlib – библиотека для визуализации двумерных и трехмерных данных [6].

1.4.2 Apache Spark

Spark является движком для обработки больших объемов неструктурированных и слабоструктурированных данных [7].

Spark имеет множество преимуществ:

- Обеспечивает высоконадежный быстрый расчет.
- Возможности отказоустойчивости из-за неизменной первичной абстракции RDD.
- Встроенные библиотеки машинного обучения.
- Spark совместим с несколькими языками программирования, такими как R, Java, Python, Scala и т. д. Scala более чем в 10 раз быстрее, чем Python. Scala использует виртуальную машину Java (JVM) во время выполнения, что в большинстве случаев дает более высокую скорость по сравнению с Python.

Если необходимо повысить скорость обработки данных для принятия более быстрых решений, Spark определенно может предложить передовое преимущество. Данные обрабатываются в Spark циклически, и механизм выполнения разделяет данные в памяти. Поддержка механизма Directed

Acyclic Graph (DAG) позволяет механизму Spark обрабатывать одновременные задания с одинаковыми наборами данных. Данные обрабатываются механизмом Spark в 100 раз быстрее по сравнению с Hadoop MapReduce.

Spark и Scala позволяют овладеть мощью различных структур данных, поскольку Spark способен обращаться к Tachyon, Hive, HBase, Hadoop, Cassandra и другим. Spark может быть развернут через YARN или другую распределенную среду, а также на отдельном сервере.

1.5 Временные ряды

Временной ряд — это последовательность, в которой метрика записывается через регулярные промежутки времени [8]. Прогнозирование временных рядов (например, спроса и продаж) часто имеет огромную коммерческую ценность, определяет основные направления бизнес-планирования, закупок и производственной деятельности. Любые ошибки в прогнозах будут отражаться на всей цепочке поставок. Поэтому очень важно получить точные прогнозы, чтобы сэкономить на затратах.

Для применения эконометрических моделей для прогнозирования временных рядов первоначально необходимо привести временной ряд к стационарности. Стационарный временной ряд — это когда среднее значение, дисперсия и автокорреляционная структура, то есть характеристики временного ряда постоянны во времени [9].

Известно, что линейная регрессия работает лучше всего, если предикторы (переменные X) не коррелируют друг с другом. Таким образом, стационаризация ряда решает эту проблему, поскольку она устраняет любую устойчивую автокорреляцию, тем самым делая предикторы (лаги ряда) в моделях прогнозирования почти независимыми.

Стационарность временного ряда можно установить, посмотрев на график временного ряда. Другой метод заключается в разделении ряда на 2

или более смежных частей и вычислении суммарной статистики, такой как среднее значение, дисперсия и автокорреляция. Если статистика сильно отличается, то временной ряд не является стационарным [9].

Также существуют методы, чтобы количественно определить, является ли данный ряд стационарным или нет. Это можно сделать с помощью статистических тестов, называемых "тестами единичного корня". Существует несколько вариантов этого метода, когда тесты проверяют, является ли временной ряд нестационарным и имеет ли он единичный корень. Наиболее часто используется тест Дикки-Фуллера (ADH Test) [10].

Нулевая гипотеза теста Дикки-Фуллера состоит в том, что временной ряд обладает единичным корнем и является нестационарным. Таким образом, если p-значение теста меньше уровня значимости (0,05), то вы отвергаете нулевую гипотезу и делаете вывод, что временной ряд действительно стационарен.

Необходимо различать белый шум и стационарный ряд. Как и стационарный ряд, белый шум также не является функцией времени, то есть его среднее значение и дисперсия не меняются с течением времени. Но разница в том, что белый шум является полностью случайным со средним значением 0. В белом шуме вообще нет никакой закономерности. Математическая последовательность совершенно случайных чисел со средним нулем — это пример белого шума.

На рисунке 4 представлен пример стационарного временного ряда по критерию Дики-Фуллера, ACF и PACF функций.

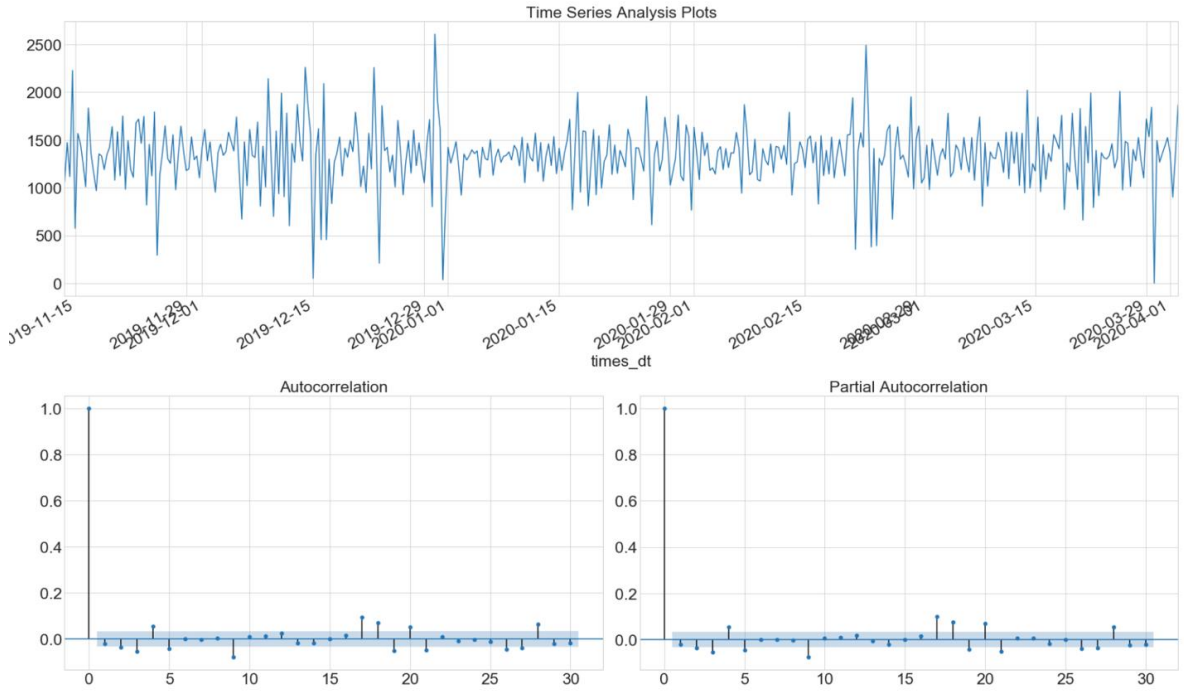

Рисунок 1.4 – Пример стационарного временного ряда, ACF и PACF функций

Автокорреляция относится к тому, насколько коррелирован временной ряд с его прошлыми значениями, тогда как ACF — это график, используемый для просмотра корреляции между точками, вплоть до единицы запаздывания. В ACF коэффициент корреляции находится на оси X, тогда как количество лагов показано на оси Y. График функции автокорреляции позволит вам узнать, как данный временной ряд коррелирует с самим собой [9].

Частичная автокорреляция (PACF) – это изложение связи между наблюдением во временном ряду и наблюдениями на предыдущих временных этапах с удалением связей промежуточных наблюдений [11].

В общем случае частичная корреляция — это условная корреляция. Это корреляция между двумя переменными в предположении, что мы знаем и учитываем значения некоторого другого набора переменных. Например, рассмотрим регрессионный контекст, в котором y-переменная ответа, а x1, x2 и x3-переменные предиктора. Частичная корреляция между y и x3-это корреляция между переменными, определяемая с учетом того, как y и x3 связаны с x1 и x2.

В работе применяются способы перехода к стационарному временному ряду. Самый распространенный подход заключается в том, чтобы вычесть предыдущее значение из текущего значения. Иногда, в зависимости от сложности ряда, может потребоваться более одного дифференцирования.

Правильный порядок дифференцирования — это минимальное дифференцирование, необходимое для получения почти стационарного ряда, который перемещается вокруг определенного среднего, и график ACF довольно быстро достигает нуля [9].

Если автокорреляции положительны для большого числа лагов (10 и более), то ряд нуждается в дальнейшем дифференцировании. Если возникает проблема выбора между двумя порядками дифференцирования, то следует выбрать порядок, который дает наименьшее стандартное отклонение в дифференцированном ряду.

1.6 Описание моделей прогнозирования. Проверка остатков

В логистике существует несколько подходов к прогнозированию. Классификация методов прогнозирования по различным признакам приведена ниже (таблица 1.1) [12].

Таблица 1.1 – Методы прогнозирования по различным признакам

Признак классификации	**Вид метода**	**Описание**
По характеру исходных данных	Фактографический	Основан на использовании источников фактической информации
	Статистический	Основан на анализе динамических рядов параметров ОП
	Патентный	Основан на оценке изобретений и исследований динамики их патентования
	Экспертный	Основан на использовании экспертной информации

Продолжение таблицы

По используемому подходу к прогнозированию	Экспертных оценок	Основан на субъективной оценке экспертов текущего момента и перспектив развития, учитывает знания, опыт, интуицию экспертов
	Анализ и прогнозирование рядов данных	Связан с исследованием рядов значений показателей, выявлением зависимости показателей, тенденций и использованием их для прогноза (если независимый показатель – время, то ряд называется временным)
	Причинно-следственные	Основаны на поиске факторов, определяющие поведение ОП, построения и использования для прогнозов соответствующей модели его поведения
По способу обработки и анализа исходных данных и формированию прогноза	Сглаживание	Преобразование исходных динамических рядов данных в ряды со сглаженными (уменьшенными) отклонениями от предполагаемого тренда
	Экстраполяция	Определение будущих значений величин на основе имеющихся данных о тенденциях их изменений в прошлые периоды
	Интерполяция	Определение промежуточного значения параметра Y на основе данных о его зависимости от X, полученных на некотором интервале значений параметра X
	Аналогия	Основан на установлении и использовании для прогнозирования аналогии ОП с другими объектами по некоторым общим чертам

Продолжение таблицы

	Моделирование	На основе математических и им моделей прогнозируются возмо ОП при различных значениях ис
	Прогнозный сценарий	Основан на установлении логической последовательности состояния ОП во времени при различных условиях для определения целей развития этой объекта
	Морфологический анализ	Строится матрица параметров ОП и их возможных значений с последующим перебором и оценкой вариантов сочетаний этих значений

В работе используются эконометрический подход к прогнозированию остатков в АОПП и метод градиентного бустинга.

1.6.1 ARIMA

Популярным и широко используемым статистическим методом для прогнозирования временных рядов является модель ARIMA. ARIMA - алгоритм прогнозирования, основанный на идее, что информация в прошлых значениях временного ряда может быть использована для прогнозирования будущих значений.

ARIMA – это сокращение, обозначающее Autoregressive integrated moving average. Это класс моделей, который фиксирует набор различных стандартных временных структур во временных рядах [10].

Аббревиатура модели носит описательный характер, охватывая ключевые аспекты самой модели. Вкратце, это:

- AR: Авторегрессия (Autoregression). Термин "Авторегрессия" в ARIMA означает, что это линейная регрессионная модель, которая использует свои собственные лаги в качестве предикторов. Линейные регрессионные модели, лучше всего работают, когда предикторы не коррелируют и не зависят друг от друга [9].

- I: Интегрированный (Integrated). Использование разности необработанных наблюдений (например, вычитание наблюдения из наблюдения на предыдущем временном шаге), чтобы сделать временной ряд стационарным [9].
- MA: Скользящее среднее (Moving Average). Модель, которая использует зависимость между наблюдением и остаточной ошибкой от модели авторегрессии, примененной к лаговым наблюдениям [9].

Каждый из этих компонентов явно указан в модели в качестве параметра. Используется стандартная нотация ARIMA(p, d, q), где параметры заменяются целыми значениями, чтобы быстро указать конкретную используемую модель ARIMA.

Параметры модели ARIMA определяются следующим образом:

- p: число лаговых наблюдений, включенных в модель, также называется порядком отставания (порядок AR) [9]
- d: количество раз, когда необработанные наблюдения отличаются друг от друга, также называется степенью различия (число разностей, необходимых для того, чтобы сделать временной ряд стационарным, а если временной ряд уже стационарен, то d = 0) [9]
- q: размер окна скользящей средней, также называемый порядком скользящей средней (порядок MA) [9]

Перед применением модели АРИМА выполняется подготовка временного ряда, приведение к стационарному виду, то есть устраняются трендовые и сезонные структуры, негативно влияющие на регрессионную модель.

Значение 0 может быть использовано для параметра, который указывает на то, что этот элемент модели не используется. Таким образом, модель ARIMA может быть сконфигурирована для выполнения функций модели ARMA и даже простой модели AR, I или MA.

Чистая авторегрессионная модель (только AR) — это модель, в которой Y_t зависит только от собственных лагов. То есть Y_t — это функция "лагов Y_t" [12]:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \varepsilon_t$$

Y_{t-1} – первый лаг временного ряда

Y_t – стационарный ряд со средним μ

$\beta_1, \dots, \beta_p$ – константы

β_1 – коэффициент первого лага, который оценивает модель

α – коэффициент перехвата, также оцениваемый моделью

$$\alpha = \mu(1 - \beta_1 - \cdots - \beta_p)$$

ε_t – гауссов белый шум с нулевым средним и постоянной дисперсией σ_ϵ^2

Чистая модель скользящей средней (только MA) — это модель, в которой Y_t зависит только от лаговых ошибок прогноза [13]:

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

Y_t – стационарный ряд

$\phi_1, \dots, \phi_q$ – константы

$\alpha = \mu$, если Y_t имеет среднее значение μ

ϵ_t – это ошибки авторегрессионных моделей соответствующих лагов. Ошибки ϵ_t и ϵ_{t-1} являются ошибками из следующих уравнений [13]:

$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_0 Y_0 + \epsilon_t$$

$$Y_{t-1} = \beta_2 Y_{t-2} + \beta_3 Y_{t-3} + .. + \beta_0 Y_0 + \epsilon_{t-1}$$

Модель ARIMA — это модель, в которой временной ряд был дифференцирован по крайней мере один раз, чтобы сделать его стационарным, объединяем термины AR и MA. Таким образом, уравнение становится [13]:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

Y_t – стационарный ряд со средним μ

ARIMA – это метод прогнозирования для одномерных данных временных рядов. Как следует из его названия, он поддерживает как авторегрессионные, так и скользящие средние элементы. Интегральный элемент относится к

дифференцированию, позволяющему методу поддерживать данные временных рядов с трендом.

Проблема с ARIMA заключается в том, что она не поддерживает сезонные данные. Это временной ряд с повторяющимся циклом. ARIMA ожидает данные, которые либо не являются сезонными, либо имеют скорректированную сезонную составляющую, например, скорректированную с помощью таких методов, как сезонная дифференциация.

1.6.2 SARIMA

ARIMA, является одним из наиболее широко используемых методов для одномерного прогнозирования данных временных рядов. Метод может обрабатывать данные с трендом, но он не поддерживает временные ряды с сезонной составляющей. Расширение к ARIMA, которое поддерживает прямое моделирование сезонной составляющей ряда, называется SARIMA$(p,d,q)(P,D,Q)_m$. Сезонная модель ARIMA формируется путем включения дополнительных сезонных терминов в модели ARIMA [9].

(p,d,q) - несезонная часть

$(P,D,Q)_m$ - сезонная часть

Есть три элемента тренда, которые требуют настройки. Они такие же, как в модели ARIMA [9]:

p – порядок авторегрессии тренда

d – порядок дифференцирования

q – порядок скользящей средней

Дополнительно в модели имеется четыре сезонных элемента, которые не являются частью ARIMA, которые должны быть настроены [9]:

P – сезонный авторегрессионный порядок

D – порядок сезонного дифференцирвания

Q – сезонный порядок скользящих средних

m – количество временных шагов за один сезонный период

Модель имеет преимущества в виде поддержки сезонной компоненты и четкого математико-статистическое обоснования. Недостаток модели SARIMA в том, что она не оптимальный способ по времени. Подготовка данных, подбор оптимальных значений, обучение, находить каждый раз оптимальный подход к ряду – это занимает много времени, алгоритм имеет плохую обобщающую способность.

1.6.3 Градиентный бустинг

Часть моделей могут не подойти для соотношения быстро-качественно для выведения в производственный процесс, так как либо требуют слишком больших затрат по подготовке данных, либо сложно настраиваются, либо требуют частого переобучения на новых данных. Примером моделей, которые имеют все вышеперечисленные проблемы, являются ARIMA, SARIMA. Зачастую эффективнее бывает выделить признаки из временного ряда и построить по ним линейную регрессию, решающий лес, градиентный бустинг и т.д.

Бустинг строит алгоритмы последовательно и каждый следующий алгоритм приближает вектор антиградиента на обучающей выборке. Бустинг относится к классу ансамблевых алгоритмов машинного обучения, которые могут быть использованы для задач классификации или регрессионного прогностического моделирования [14]. Общая формула для бустинга:

$$a_N(x) = \sum_{n=1}^{N} b_n,$$

где b_n – базовый алгоритм

Самый частый подход в бустинге, когда базовый алгоритм представляет собой дерево решений. Деревья добавляются по одному в ансамбль и подходят для исправления ошибок предсказания, сделанных предыдущими моделями.

Решающее дерево разбивает пространство на области непересекающихся объектов выборки $R_1, \dots, R_J$. В каждой области получаем константное предсказание $b_1, \dots, b_J$ ($R_1 \to b_1, R_2 \to b_2$):

$$b(x) = \sum_{j=1}^{J} [x \in R_j] b_j$$

Формула градиентного бустинга для случая, когда базовый алгоритм представляет собой решающие деревья:

$$a_N(x) = a_{N-1}(x) + b_N(x),$$

$$b_N(x) = \sum_{j=1}^{J} [x \in R_{N_j}] b_{N_j}$$

В результате получаем:

$$a_N(x) = a_{N-1}(x) + \sum_{j=1}^{J} [x \in R_{N_j}] b_{N_j}$$

Перенастроим b_{N_j} так, чтобы прогноз был оптимален для исходной функции потерь L. Получаем задачу в следующем виде:

$$\sum_{i=1}^{l} L(y_i, a_{N-1}(x) + \sum_{j=1}^{J} [x \in R_{N_j}] b_{N_j}) \to min_{b_1,\dots,b_J}$$

Получаем сумму ошибок уже построенной композиции $a_{N-1}(x)$ и ответа нового решающего дерева по всем объектам. Необходимо найти такие прогнозы в листьях b_{N_j}, чтобы как можно значительнее уменьшить функцию потерь L.

Задача распадается на J подзадач, сколько листьев в дереве, столько и подзадач получаем. Поиск оптимального прогноза в J-ом листе, будет выглядеть следующим образом:

$$b_{N_j} = \underset{\gamma \in R}{\operatorname{argmin}} \sum_{x_i \in R_j} L(y_i, a_{N-1}(x) + \gamma)$$

Минимизируем сумму ошибок на объектах, которые попали в данный лист, то есть в область R_j. Прогноз получается из построенной композиции и

прогноза в листе γ. Гамма (γ) находится различными подходами, например, аналитически, по сетке возможных значений или градиентным методом.

Ответы в листьях можно перенастроить так, чтобы они были оптимальными не относительно исходной функции потерь L. Таким образом, сходимость градиентного бустинга очень сильно ускорится. Таким образом, перенастраивается в алгоритме градиентного бустинга над решающими деревьями прогнозы в листьях, что повышает значительно сходимость градиентного бустинга.

Xgboost и lightGBM – оптимизированные реализации градиентного бустинга (Gradient Boosted Trees). Разница между xgboost и lightGBM заключается в специфике оптимизаций [15].

При обучении каждого отдельного дерева решений и разбиении данных можно использовать две стратегии: уровневую и листовую (пример показан на рисунке 1.5). Стратегия с учетом уровня поддерживает сбалансированное дерево, тогда как стратегия с учетом листьев расщепляет лист, что в наибольшей степени уменьшает потери.

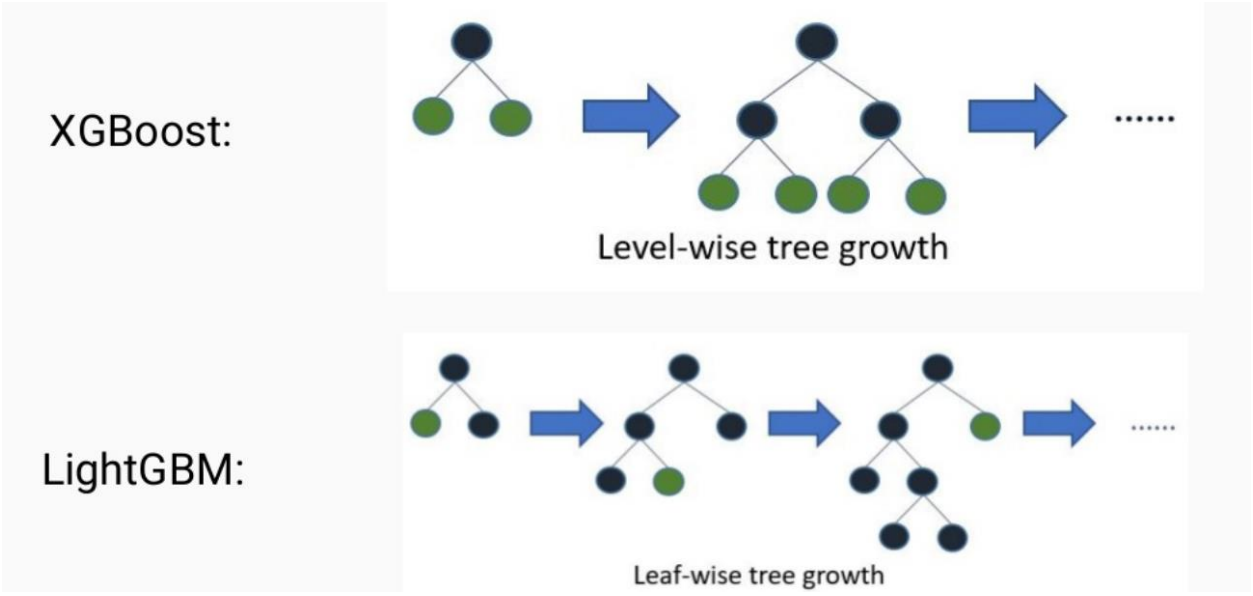

Рисунок 1.5 – Примеры стратегий построения деревьев

Обучение по уровням можно рассматривать как форму регуляризованного обучения, поскольку обучение по листьям может построить любое дерево, которое можно построить по уровням, в то время как

противоположное не имеет места. Таким образом, обучение по листьям более склонно к переобучению, но более гибко.

Количество времени, которое требуется для построения дерева, пропорционально количеству разбиений, которые должны быть оценены. Методы, основанные на гистограммах, используют этот факт, группируя объекты в набор ячеек и выполняя разбиение на ячейки вместо объектов. Это эквивалентно подсчету количества разбиений, которое вычисляет модель. Поскольку объекты могут быть объединены перед построением каждого дерева, этот метод может значительно ускорить обучение.

1.6.4 XGBoost

XGBoost является методом основанном на гистограммах и аппроксимации функций путем оптимизации конкретных функций потерь, а также применения нескольких методов регуляризации [16].

Целевая функция (функция потерь и регуляризация), которую нам нужно минимизировать, состоит из двух слагаемых: функции потерь и регуляризатора $\Omega(f_k)$ для каждого из K деревьев, где $f_k -$ прогноз k-ого дерева, выглядит следующим образом [15]:

$$loss = \sum_{i}^{l} l(y_i - \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Первое слагаемое γT штрафует модель за большое число листьев T, а второе $\frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$ контролирует сумму весов модели в листьях.

1.6.5 LightGBM

Light GBM имеет префикс «Light» из-за его высокоскоростной работы. Легкий GBM может обрабатывать большие объемы данных и занимает

меньше памяти. Поскольку он основан на алгоритмах дерева решений, он разделяет лист дерева с наилучшим соответствием, тогда как другие алгоритмы повышения делят дерево по глубине или уровню, а не по листу. Таким образом, при выращивании на одном листе в Light GBM, листовой алгоритм может уменьшить больше потерь, чем поуровневый алгоритм, и, следовательно, дает гораздо лучшую точность [17].

LightGBM использует новую технику односторонней выборки на основе градиента (GOSS) для фильтрации экземпляров данных для поиска значения разделения, в то время как XGBoost использует предварительно отсортированный алгоритм и алгоритм на основе гистограммы для вычисления наилучшего разделения.

Проще говоря, алгоритм на основе гистограммы разделяет все точки данных для объекта на дискретные элементы и использует эти элементы для нахождения значения разделения гистограммы. Несмотря на то, что он эффективнее, чем предварительно отсортированный алгоритм в скорости обучения, который перечисляет все возможные точки разделения на предварительно отсортированные значения признаков, он все еще отстает от GOSS с точки зрения скорости [17].

1.6.6 Prophet

По своей сути модель Prophet представляет собой аддитивную регрессионную модель с четырьмя основными компонентами [18]:

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t)$$

- $g(t)$ – трендовая компонента
- $s(t)$ – сезонные компоненты, отвечают за моделирование периодических изменений
- $h(t)$ отвечает за заданные пользователем аномальные дни
- $\epsilon(t)$ охватывает специфические изменения, не учитываемые моделью

Сезонные компоненты s(t) отвечают за моделирование периодических изменений, связанных с недельной и годовой сезонностью. Еженедельная сезонность моделируется с использованием бинарных переменных. Генерируются 6 дополнительных бинарных признаков, например, понедельник, вторник, среда, четверг, пятница, суббота, которые принимают значения 0 и 1. Атрибут, соответствующий воскресному дню недели, добавляться не будет, поскольку он будет зависеть от других дней недели, и это повлияет на модель. Годовая сезонность моделируется рядом Фурье [18].

Тренд g(t) — это кусочно-линейная или логистическая функция [18]. Логистическая функция $g(t) = \frac{C}{1+\exp(-k(t-b))}$ позволяет имитировать рост с насыщением, когда при увеличении индикатора скорость его роста уменьшается. Например, рост аудитории приложения или сайта.

1.6.7 LSTM

Сети долговременной кратковременной памяти, обычно называемые просто «LSTM», представляют собой особый вид рекуррентных нейронных сетей (RNN), способный к обучению долговременным зависимостям. Последовательность хранится в виде матрицы, где каждая строка представляет собой вектор признаков, описывающий ее. RNNs содержат циклы. Каждая единица имеет состояние и получает два входных сигнала-состояния от предыдущего слоя и статистику от этого слоя с предыдущего временного шага [19].

LSTMs предназначены для того, чтобы избежать долгосрочной проблемы зависимости. Запоминание информации в течение длительного периода времени основная задача LSTM сетей [19].

В работе использовалась нейронная рекуррентная сеть, состоящая из 32 LSTM ячеек, с помощью которой составлялся прогноз. Обучение длилось 1000 эпох.

1.6.8 Проверка остатков моделей

Остатки – разность между фактом и прогнозом:

$$\hat{\epsilon} = y_t - \hat{y}_t$$

Необходимо выполнять проверку, обладают ли остатки определенными свойствами [9]:

- несмещенность – равенство среднего значения нулю (можно проверить с помощью критерия Стьюдента гипотезу $H_0: \epsilon = 0$)
- стационарность – отсутствие зависимости от времени (можно проверить с помощью критерия Дики-Фуллера)
- неавтокоррелированность – отсутствие зависимости от предыдущих наблюдений (можно проверить по коррелограмме и с помощью Q-критерия Льюнга-Бокса)

Первое, что мы должны проверить, это то, являются ли остатки смещенными или нет. Известно, что среднее значение остатков равно нулю, поэтому мы можем начать проверять с помощью t-критерия Стьюдента, верно ли это или нет для нашей выборки.

t-критерия Стьюдента [9]:

H0: математическое ожидание равно нулю

Ha: математическое ожидание не равно нулю

Если p-значение больше 5%, мы не можем отвергнуть нулевую гипотезу и можем сказать, что среднее значение остатков несогласных статистически похоже на 0.

В модели, которая нашла зависимости в данных, остатки независимы друг от друга, что означает, что в остатках нет никакого паттерна автокорреляции. Q-тест Льюнга-Бокса проверяет наличие автокорреляционного паттерна в остатках.

Q-тест Льюнга-Бокса [9]:

H0: данные являются случайными (т.е. представляют собой белый шум).

Ha: данные не являются случайными.

1.7 Выбор метрик

1.7.1 Средняя абсолютная ошибка

Средняя абсолютная ошибка – это среднее значение абсолютных значений отклонения [9]. Этот тип измерения погрешности полезен при измерении ошибок предсказания в той же единице измерения, что и исходная серия. MAE – это линейная оценка, которая означает, что все индивидуальные различия взвешиваются одинаково в среднем. Ниже приведена формула [9].

$$MAE = \frac{\sum_{i=0}^{n} |y_i - \hat{y}_i|}{n}$$

1.7.2 Среднеквадратичная ошибка

Среднеквадратичная ошибка (Mean Square Error) — это среднее значение квадрата ошибки прогноза [9]. Поскольку берется квадрат ошибок, то получается, что более крупные ошибки имеют больший вес.

$$MSE = \frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{n}$$

Это одна из наиболее эффективных мер для оценки и поиска моделей, но как только модель найдена, обычно используются другие меры ошибки, такие как MAE. Например, обнаружение того, что средний прогноз может быть отклонен на ±5%, само по себе является полезным результатом, но MSE 65.34 труднее понять. Один из способов решения этой проблемы заключается в том, чтобы взять корень из суммы ошибок, прежде чем разделить его с размером выборки. Это называется Root Mean Square Error и имеет то преимущество, что находится в той же единице, что и переменная прогноза.

Кроме того, поскольку и MSE, и RMSE принимают квадрат ошибок, выбросы будут иметь огромное влияние на результирующую ошибку. Формула для Root Mean Square Error приведена ниже [9].

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{n}}$$

Сравнение MAE и RMSE:

Сходство: и MAE, и RMSE выражают среднюю ошибку прогнозирования модели в единицах интересующей переменной. Обе метрики могут варьироваться от 0 до ∞ и безразличны к направлению ошибок.

Различия: взятие квадратного корня из средних квадратов ошибок имеет некоторые интересные последствия для RMSE. Поскольку ошибки возводятся в квадрат до их усреднения, RMSE придает относительно высокий вес большим ошибкам. Это означает, что RMSE должен быть более полезен, когда большие ошибки особенно нежелательны.

MAE и RMSE могут использоваться вместе для диагностики вариации ошибок в наборе прогнозов. RMSE всегда будет больше или равен MAE, чем больше разница между ними, тем больше дисперсия индивидуальных ошибок в выборке. Если RMSE=MAE, то все ошибки имеют одинаковую величину.

В работе используются MAE и RMSE вместе.

1.7.3 Информационный критерий Акайке

Для сравнения параметров моделей АРИМА и САРИМА используется информационный критерий Акайке. Информационный критерий Акайке (AIC) является оценкой относительного качества статистических моделей для данного набора данных [9]. AIC по существу является оценочным показателем качества каждой из имеющихся эконометрических моделей, поскольку они соотносятся друг с другом для определенного набора данных, что делает его идеальным методом для выбора модели.

Критерий является не статистическим, а информационным, поскольку основан на оценке потери информации при уменьшении числа параметров модели. В общем случае AIC вычисляется по формуле [9]:

$$AIC = 2 \cdot k - 2 \cdot \ln(L),$$

где k — число параметров модели, L — максимизированное значение функции правдоподобия модели. Лучшей признается та модель, для которой значение AIC минимально.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Архитектура решения

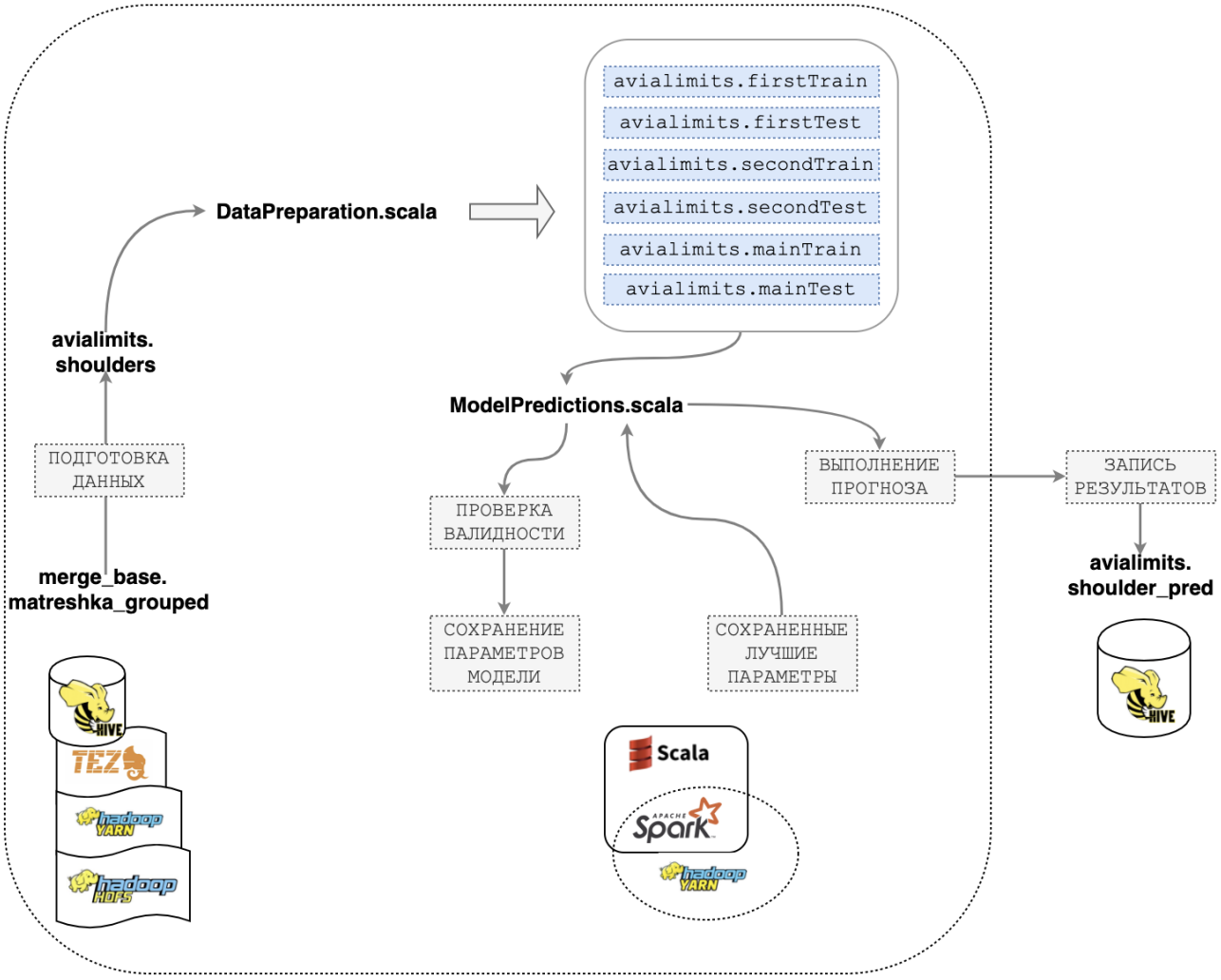

Рисунок 2.1 – Архитектура программного решения

На рисунке 2.1 изображена архитектура программного решения. На первом этапе выполняется подготовка данных, хранящихся в HDFS. HDFS — распределенная файловая система, предназначенная для хранения файлов больших размеров, поблочно распределённых между узлами вычислительного кластера. Для распределения ресурсов используется Hadoop YARN. Для обработки данных предназначен Tez. Apache Tez — это свободный фреймворк, работающий поверх Apache Hadoop YARN и предоставляющий возможность использовать сложный направленный ациклический граф (DAG) задач для обработки данных. Запрос для подготовки итоговой таблицы

реализуется в SQL интерфейсе к данным в Hadoop – Apache Hive. Позволяет выполнять запросы, агрегировать и анализировать данные, хранящиеся в Apache Hadoop.

На втором этапе выгружаются данные из HDFS, выполняется генерация признаков, настройка модели и получение предсказаний модели. Результаты предсказаний записываются в HDFS.

Для реализации модели было написано два модуля на Spark Scala. Первый класс (DataPreparation.scala) выполняет подготовку данных в виде трех комбинаций тренировочных и тестовых выборок. Первые две выборки используется для кросс-валидации по временному ряду. Второй модуль (ModelPredictions.scala) выполняет проверку валидности модели, подбирает параметры и выполняет предсказания.

Кросс-валидационные выборки используются для проверки валидности модели, тестируется качество на данных выборках и сравнивается с предсказанием среднего, если качество на кросс-валидационных выборках лучше, то выполняется подбор параметров для главной тренировочной выборки, выполняются предсказания и записываются в HDFS.

2.2 Генерирование признаков

Стандартными признаками для временных рядов являются лаги целевой переменной, генерируется 30 лагов. Много информации содержат дата и время, которые позволяют добавить такие признаки, как: час, день недели, бинарный признак выходной день, месяц, день месяца, бинарный признак утро, бинарный признак ночь, количество дней в месяце.

Время является примером циклических данных: минуты, часы, секунды, день недели, неделя месяца, месяц, сезон и так далее следуют циклам. Необходимо донести цикличность до модели. Создадим две новые функции, считающие синусное преобразование и косинусное преобразование функции:

$$val\_\sin = \sin\left(\frac{val \cdot 2 \cdot \pi}{period}\right)$$

$$val\_\cos = \cos\left(\frac{val \cdot 2 \cdot \pi}{period}\right)$$

Для дня недели период равен 7, для часов период равен 24, для дня месяца – количеству дней в месяце и так далее.

Построим преобразование двух объектов в виде 24-часовых часов (рисунок 2.2). В результате, расстояние между двумя точками соответствует разнице во времени, как мы ожидаем от 24-часового цикла.

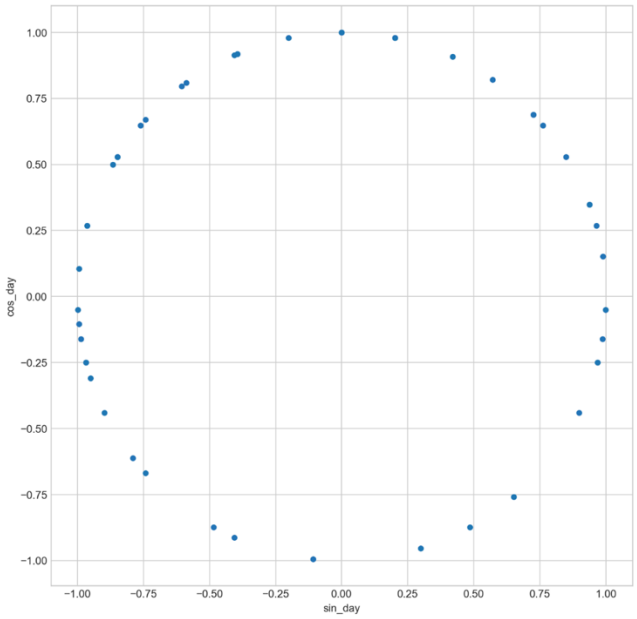

Рисунок 2.2 – Проецирование признака дня месяца

Переход к синусу и косинусу был выполнен для признаков: день недели, день месяца, месяц, часы.

Вещественные признаки, такие как "insr_rate", "mass_rate", "air_rate", "mass", были преобразованы в группы по квантилям и прологарифмированы.

```
df[colName+'_groups'] = df[colName].apply(lambda x: 5 if np.isnan(x) else
 (0 if (x>= d['min'] and x<d['25%']) else
(1 if (x>=d['25%'] and x<d['50%']) else
    (2 if (x>=d['50%'] and x<d['75%']) else
        (3 if (x>=d['75%'] and x <d['max']) else 4)))))
```

Когда распределение непрерывных данных ненормально, применяются преобразования данных, чтобы сделать данные как можно более "нормальными". Логарифмическое преобразование, возможно, является наиболее популярным среди различных типов преобразований, используемых для преобразования искаженных данных, чтобы они приблизительно соответствовали нормальности. Логарифмирование выполняется, когда переменные охватывают несколько порядков.

Пример распределения признака air_rate до логарифмирования представлен на рисунке 2.3 слева (как видно на рисунке большой правый «хвост») и после логарифмирования на рисунке справа.

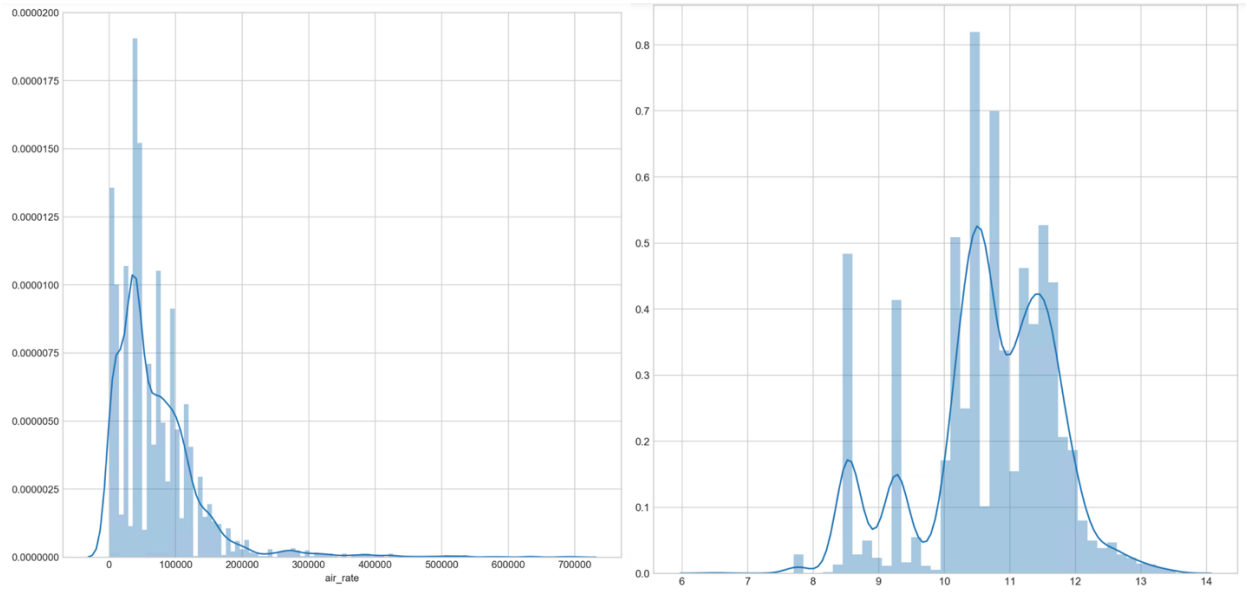

Рисунок 2.3 – Признак до логарифмирования (слева) и признак после логарифмирования (справа)

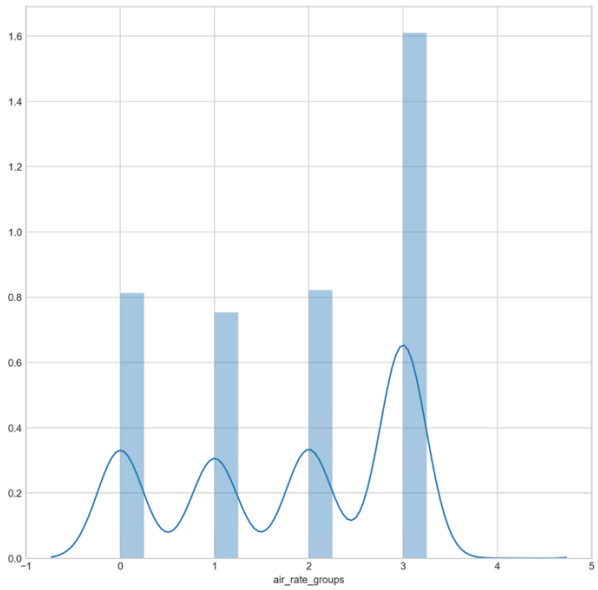

Рисунок 2.4 – Деление на группы признака air_rate

К признакам "rate", "payment", "value", "ad_val_tax" был применен только логарифм, без дробления на группы. На рисунке 2.5 приведен пример логарифмирования признака payment. На рисунке слева виден большой «хвост», от которого избавляемся при помощи применения логарифмической функции.

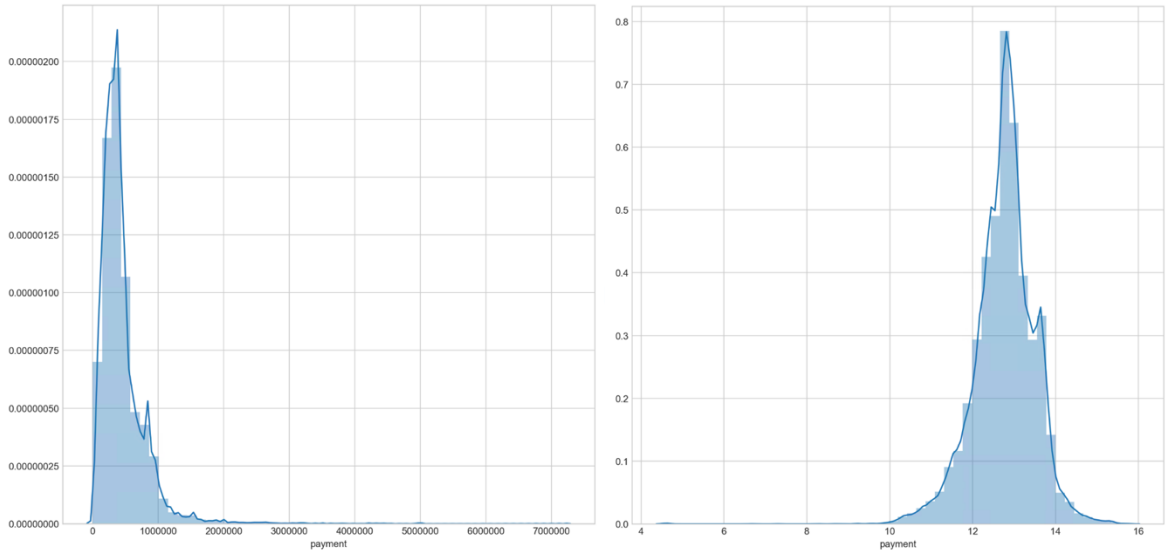

Рисунок 2.5 – Признак payment до логарифмирования (слева) и после логарифмирования (справа)

Для того, чтобы учесть особенности передвижений РПО, были созданы следующие признаки:

- index_from_to_mr = index_to_rpo_macroregion + index_from_rf_macroregion
- index_from_to_obj = preparation_index_from_rf + preparation_index_to_rpo
- index_from_to_obj_wd = preparation_index_from_rf + preparation_index_to_rpo + weekday

index_from_to_mr позволяет учесть макрорегионы из которого едет РПО и в который. index_from_to_obj позволяет учесть объекты из которого едет РПО и в который.

index_from_to_obj_wd позволяет учесть объекты из которого едет РПО и в который, учитывая день недели.

В дальнейшем данные признаки используется для подсчета количества РПО, которые имеют данное значение признака в промежутке 8 часов.

Например:

ШПИ	Макрорегион, из которого отправлено РПО	Макрорегион, в который направлено РПО	index_from_to_mr	Время выполнения операции в АОПП
1292948284920349	11	2	112	День 1 12:01
4294923423949239	5	3	53	День 1 16:30
1231634583483033	11	2	112	День 1 7:05

Из примера выше, получаются тренировочные данные в виде:

Время	Количество РПО с index_from_to_mr = 112	Количество РПО с index_from_to_mr = 53
День 1 0:00	1	0
День 1 8:00	1	0

День 1 16:00	0	1
День 2 0:00	0	0

Для агрегирования данных были созданы такие признаки как:

- hours_weekday = hours + weekday
- hours_day = hours + day
- morning_weekday = morning + weekday
- night_weekday = night + weekday
- morning_weekend = morning + is_weekend

Были также добавлены признаки am_air_form, am_statuses, которые позволяют посчитать количество РПО, которые пришли на плечо в промежуток времени с особенностями в виде: имеют форму документа, связанную с отправкой на рейс и подвергались ли они возврату соответственно.

Для признаков "trans_type", "mail_ctg", "mail_rank", "send_ctg", "post_mark", "first_index", "index_to_rpo", "prev", "index_from_to_mr", "index_from_to_obj", "index_from_to_obj_wd", "preparation_index_from_rf", "preparation_index_to_rpo", "insr_rate_groups", "mass_groups", "mass_rate_groups", "air_rate_groups", "has_ad_val_tax", ”oper_type","pay_type" были найдены популярные уникальные значения и посчитано количество данных уникальных значений в промежутки времени.

Например:

ШПИ	trans_type	mass_groups	has_ad_val_tax	Время выполнения операции в АОПП
1292948284920349	1	1	1	День 1 12:01
4294923423949239	2	2	1	День 1 16:30
1231634583483033	1	2	0	День 1 7:05
3429423949204240	1	1	1	День 1 10:20

Из примера выше, получается тренировочные данные в виде:

Время	Количество РПО с trans_type = 1	Количество РПО с trans_type = 2	Количество РПО с mass_groups = 1	Количество РПО с mass_groups = 2	Количество РПО с has_ad_val_= 1	Количество РПО с has_ad_val_= 0
День 1 0:00	1	0	0	1	0	1
День 1 8:00	2	0	2	0	2	0
День 1 16:00	0	1	0	1	1	0
День 2 0:00	0	0	0	0	0	0

Для того, чтобы отловить момент отгрузки РПО на самолет был взят признак id_of_operand1 (id объекта, над которым совершается операция) и по нему выполнена группировка и агрегирование в виде максимального значения времени для каждой группы. Далее, группировка по полученному времени и, таким образом, подсчет количества вылетов. В итоговые тестовый и тренировочные датасеты количество вылетов агрегируется по часам и

считается относительная частота вылета в данный момент времени. Пример приведен ниже.

Время	Количество вылетов	Среднее количество вылетов mean_am_flights в определенный час
День 1 0:00	7	13/3 = 4.(3)
День 1 8:00	20	45/2 = 5.625
День 1 16:00	2	4/2 = 2.0
День 2 0:00	1	4.(3)
День 2 8:00	25	5.625
День 2 16:00	2	2.0
День 3 0:00	5	4.(3)

Из таблицы выше получаем:

Час	Количество вылетов	Суммарное количество вылетов	Относительная частота
0	13	62	21%
8	45	62	73%
16	4	62	6%

Результирующая таблица соединяется с тренировочными и тестовыми данными по полю час. Как итог:

Время	Час	Относительная частота вылета
День 1 0:00	0	21%
День 1 8:00	8	73%
День 1 16:00	16	6%
День 2 0:00	0	21%
День 2 8:00	8	73%
День 2 16:00	16	6%
День 3 0:00	0	21%

Также количество вылетов агрегировано усеченным средним по дню недели. Усеченное среднее – это метод усреднения, который удаляет отведенный процент наибольший и наименьших значений перед вычислением среднего значения. После удаления указанных данных усеченное среднее вычисляется с использованием стандартной арифметической формулы

расчета среднего значения. Использование усеченного среднего помогает устранить влияние «хвостов» на данные, которые могут несправедливо повлиять на традиционное среднее.

Целевая переменная была агрегирована через "weekday" и "hours_weekday". По каждой группе были получены максимальное, минимальное значения, медиана, среднее, усеченное среднее, стандартное отклонение, усеченное стандартное отклонение целевой переменной. Также добавлены признаки усеченное среднее +- усеченное стандартное отклонение, разница между максимальным и минимальным значениями.

Для уникальных значений признаков "trans_type", "mail_ctg", "mail_rank", "send_ctg", "post_mark", "first_index", "index_to_rpo", "prev", "index_from_to_mr", "index_from_to_obj", "index_from_to_obj_wd", "preparation_index_from_rf", "preparation_index_to_rpo", "insr_rate_groups", "mass_groups", "mass_rate_groups", "air_rate_groups", "has_ad_val_tax", ”oper_type","pay_type" выполнена агрегация по "hours_weekday". Для нахождения агрегирующей функции для данных признаков была обучена модель LightGBM с агрегацией упомянутых выше признаков по weekday, hours_weekday, morning_weekday, night_weekday, morning_weekend и на основе значимости признаков была отобрана агрегация по "hours_weekday".

Вещественные признаки "mass", "insr_rate", "air_rate", "mass_rate", "rate", "payment", "value", "log_ad_val_tax", "log_air_rate", "log_value", "log_mass", "log_rate", "log_payment", "log_insr_rate", "log_mass_rate" агрегированы по weekday и получены среднее, минимальное, максимальное значения и стандартное отклонение.

2.3 Результаты

Рассмотрение плеча Шереметьево АОПП – Иркутск АОПП.

Описание целевой переменной:

- mean 4704.041162 – среднее значение

- std 1114.129429 – стандартное отклонение
- min 0.000000 – минимальное значение
- 25% 4128.000000 – 25-ый квантиль
- 50% 4834.000000 – 50-ый квантиль
- 75% 5355.000000 – 75-ый квантиль
- max 8610.000000 – максимальное значение

Для построения прогноза моделью АРИМА выполняется перебор параметров (p,d,q). На основе информационного критерия Акайке выбирается наилучший набор параметров. После отбора параметров прогнозируются остатки в АОПП Шереметьево на три дня вперёд и выполняется проверка остатков модели АРИМА.

Остатки несмещены (подтверждается критерием Стьюдента), стационарны (подтверждается критерием Дики-Фуллера и визуально), неавтокоррелированы (подтверждается критерием Льюнга-Бокса и коррелограммой). Таким образом, остатки проходят проверку, из чего следует, что в остатках нет полезной информации и модель нашла зависимость в данных.

Вывод модели АРИМА для Шереметьево АОПП – Иркутск АОПП:

```
ARIMA(1, 0, 2) - AIC:6970.102416091977
ARIMA(2, 0, 1) - AIC:6967.6381256759505
ARIMA(2, 0, 2) - AIC:6967.229435092737
ARIMA(3, 0, 1) - AIC:6967.473382467178
ARIMA(3, 0, 2) - AIC:6968.00757237602
ARIMA(4, 0, 1) - AIC:6971.60275830623
ARIMA(4, 0, 2) - AIC:6973.5630949508795
ARIMA(4, 0, 3) - AIC:6974.8902082823115
ARIMA(5, 0, 1) - AIC:6973.590349596436
ARIMA(5, 0, 2) - AIC:6969.51232804561
ARIMA(5, 0, 3) - AIC:6969.713331969455
ARIMA(5, 0, 4) - AIC:6973.276090503472
ARIMA(5, 0, 5) - AIC:6963.596704243062
ARIMA(6, 0, 1) - AIC:6971.119834478031
ARIMA(6, 0, 2) - AIC:6972.316869126344
ARIMA(6, 0, 5) - AIC:6966.421173717044
```

```
_____________________________________________________________
Best params:(5, 0, 5) - min AIC:6963.596704243062
_____________________________________________________________
```

Проверка остатков:

Нулевая гипотеза критерия Стьюдента не отвергается, среднее значение остатков статистически похоже на 0

Критерий Стьюдента: p=0.992934

Остатки стационарны по критерию Дики-Фуллера

Критерий Дики-Фуллера: p=0.0

Гипотеза о случайности остатков не отвергается

Остатки прошли проверку

Best ARIMA model parameters: (5, 0, 5)

Рисунок 2.6 – Результаты предсказания для плеча Шереметьево АОПП – Иркутск АОПП

На рисунке 2.6 на первом графике представлены предсказанные значения целевой переменной и действительные. На втором графике представлены остатки модели, визуально остатки похожи на белый шум, стационарны, имеют примерно нулевое среднее, что было подтверждено критериями. На

третьем графике представлена автокорреляция остатков, как видно, значимых лагов нет, из чего следует, что остатки не автокоррелированы.

Модель SARIMA:

Модель имеет 6 параметров (p,d,q)x(P,D,Q). (p,d,q) параметры являются лучшими параметрами для АРИМА модели, параметры сезонной составляющей подбираются перебором. На рисунке 12 показаны результаты. Пример вывода модели SARIMA:

```
SARIMA(5, 0, 5)x(2, 0, 1, 21) - AIC:6196.137834669251
SARIMA(5, 0, 5)x(2, 0, 2, 21) - AIC:6179.0748044597185
SARIMA(5, 0, 5)x(2, 0, 3, 21) - AIC:5830.806319368153
SARIMA(5, 0, 5)x(2, 0, 4, 21) - AIC:5484.600407927389
SARIMA(5, 0, 5)x(2, 0, 5, 21) - AIC:9005.428529187855
SARIMA(5, 0, 5)x(2, 0, 6, 21) - AIC:19858.31728064217
SARIMA(5, 0, 5)x(3, 0, 1, 21) - AIC:5850.810286616545
SARIMA(5, 0, 5)x(3, 0, 2, 21) - AIC:5844.394734886764
SARIMA(5, 0, 5)x(3, 0, 3, 21) - AIC:5831.602650109433
SARIMA(5, 0, 5)x(3, 0, 4, 21) - AIC:5489.837496215511
SARIMA(5, 0, 5)x(3, 0, 5, 21) - AIC:11339.203445251027
SARIMA(5, 0, 5)x(3, 0, 6, 21) - AIC:4768.87753394756
SARIMA(5, 0, 5)x(4, 0, 1, 21) - AIC:5502.568611437853
SARIMA(5, 0, 5)x(4, 0, 2, 21) - AIC:5504.156987635577
SARIMA(5, 0, 5)x(4, 0, 3, 21) - AIC:5506.311562041115
SARIMA(5, 0, 5)x(4, 0, 4, 21) - AIC:5492.116754587787
SARIMA(5, 0, 5)x(4, 0, 5, 21) - AIC:11082.174322805226
SARIMA(5, 0, 5)x(4, 0, 6, 21) - AIC:4768.209406359081
SARIMA(5, 0, 5)x(5, 0, 1, 21) - AIC:5137.2425634850515
SARIMA(5, 0, 5)x(5, 0, 2, 21) - AIC:5136.253331041022
SARIMA(5, 0, 5)x(5, 0, 3, 21) - AIC:5138.853219838769
SARIMA(5, 0, 5)x(5, 0, 4, 21) - AIC:5140.941614172652
SARIMA(5, 0, 5)x(5, 0, 5, 21) - AIC:11986.285480191502
SARIMA(5, 0, 5)x(5, 0, 6, 21) - AIC:4769.382395170242
SARIMA(5, 0, 5)x(6, 0, 1, 21) - AIC:4782.291807601179
SARIMA(5, 0, 5)x(6, 0, 2, 21) - AIC:4783.780334352878
SARIMA(5, 0, 5)x(6, 0, 3, 21) - AIC:4781.900494521925
SARIMA(5, 0, 5)x(6, 0, 4, 21) - AIC:4785.292902396915
SARIMA(5, 0, 5)x(6, 0, 5, 21) - AIC:10689.505679716927
SARIMA(5, 0, 5)x(6, 0, 6, 21) - AIC:4771.055020660906
SARIMA(5, 0, 5)x(7, 0, 1, 21) - AIC:4436.179161551505
SARIMA(5, 0, 5)x(7, 0, 2, 21) - AIC:4438.3910071162
SARIMA(5, 0, 5)x(7, 0, 3, 21) - AIC:4441.013303996504
SARIMA(5, 0, 5)x(7, 0, 4, 21) - AIC:4440.969063835709
SARIMA(5, 0, 5)x(7, 0, 5, 21) - AIC:19184.38516668608
```

```
SARIMA(5, 0, 5)x(7, 0, 6, 21) - AIC:4444.244369919669
SARIMA(5, 0, 5)x(8, 0, 1, 21) - AIC:4085.564174326081
SARIMA(5, 0, 5)x(8, 0, 2, 21) - AIC:4087.4880379348087
SARIMA(5, 0, 5)x(8, 0, 3, 21) - AIC:4089.422619740897
SARIMA(5, 0, 5)x(8, 0, 4, 21) - AIC:4091.0241856587245
SARIMA(5, 0, 5)x(8, 0, 5, 21) - AIC:17656.030024031847
SARIMA(5, 0, 5)x(8, 0, 6, 21) - AIC:4095.0434950084136
SARIMA(5, 0, 5)x(9, 0, 1, 21) - AIC:3736.5371359591213
SARIMA(5, 0, 5)x(9, 0, 2, 21) - AIC:3740.1701074454204
SARIMA(5, 0, 5)x(9, 0, 3, 21) - AIC:3740.2388644916437
SARIMA(5, 0, 5)x(9, 0, 4, 21) - AIC:3741.180511041003
SARIMA(5, 0, 5)x(9, 0, 5, 21) - AIC:16149.124848664098
SARIMA(5, 0, 5)x(9, 0, 6, 21) - AIC:3748.662068512559


_____________________________________________________________

Best params:(9, 0, 1, 21) - min AIC:3736.5371359591213
```

_____

Проверка остатков:

Нулевая гипотеза критерия Стьюдента не отвергается, среднее значение остатков статистически похоже на 0

Критерий Стьюдента: p=0.604307

Остатки стационарны по критерию Дики-Фуллера

Критерий Дики-Фуллера: p=2.256173750392114e-22

Гипотеза о случайности остатков не отвергается

Остатки прошли проверку

Рисунок 2.7 – Результаты предсказания модели SARIMA для плеча Шереметьево АОПП – Иркутск АОПП

Ниже приведены общие графики всех рассматриваемых моделей и плечей.

Рисунок 2.8 – Результаты для плеча Шереметьево АОПП – Иркутск АОПП

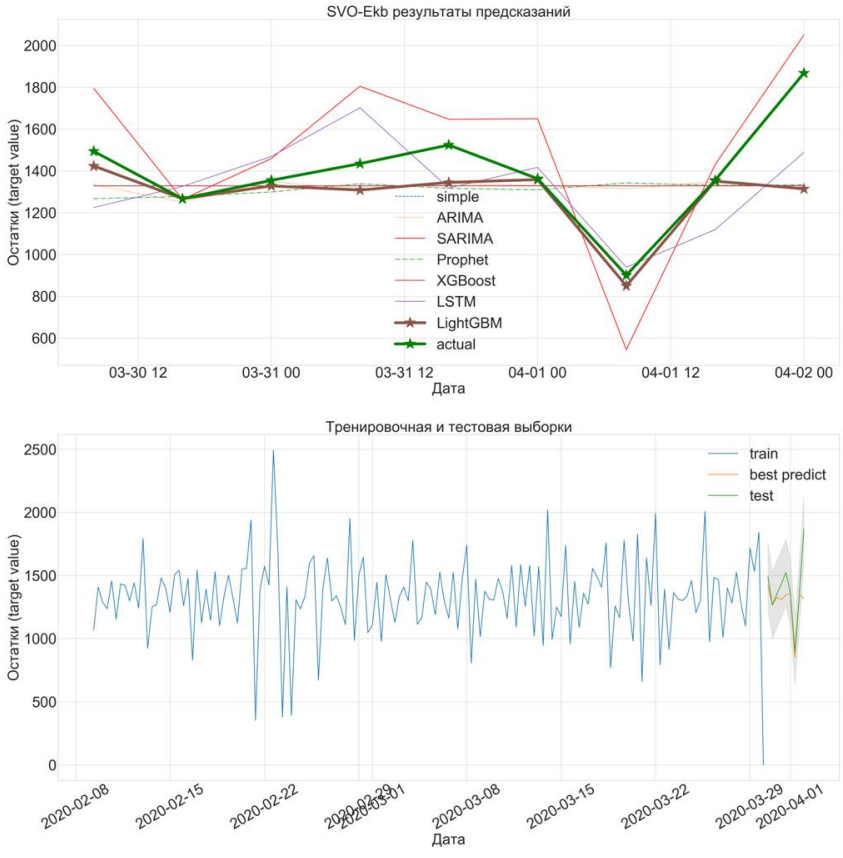

Рисунок 2.9 – Результаты для плеча Шереметьево АОПП – Екатеринбург АОПП

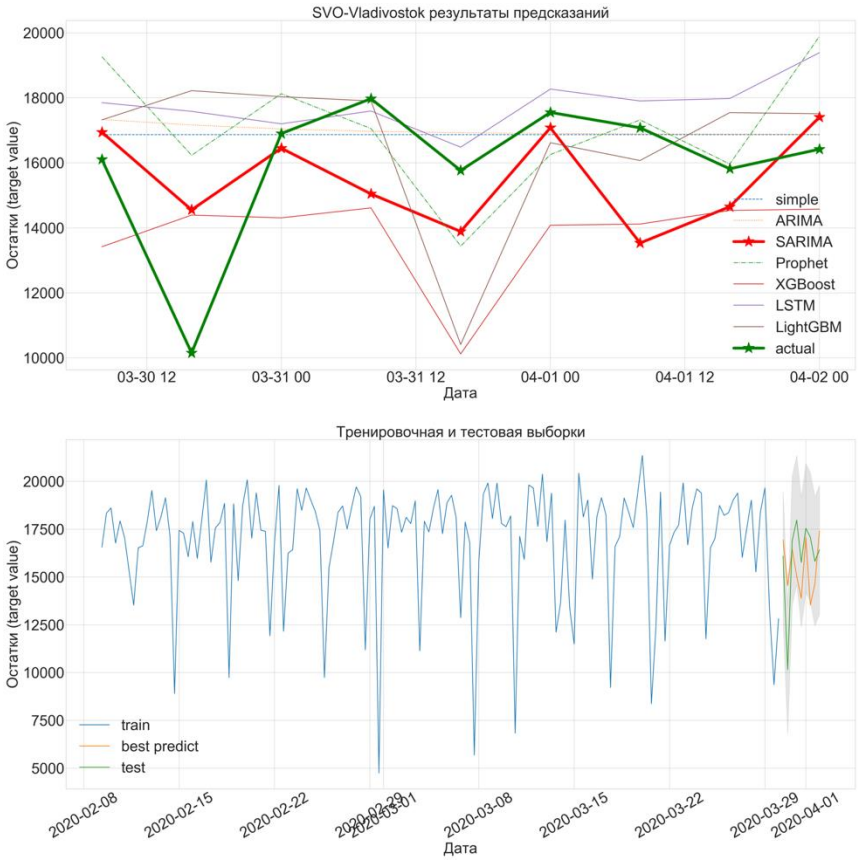

Рисунок 2.10 – Результаты для плеча Шереметьево АОПП – Владивосток АОПП

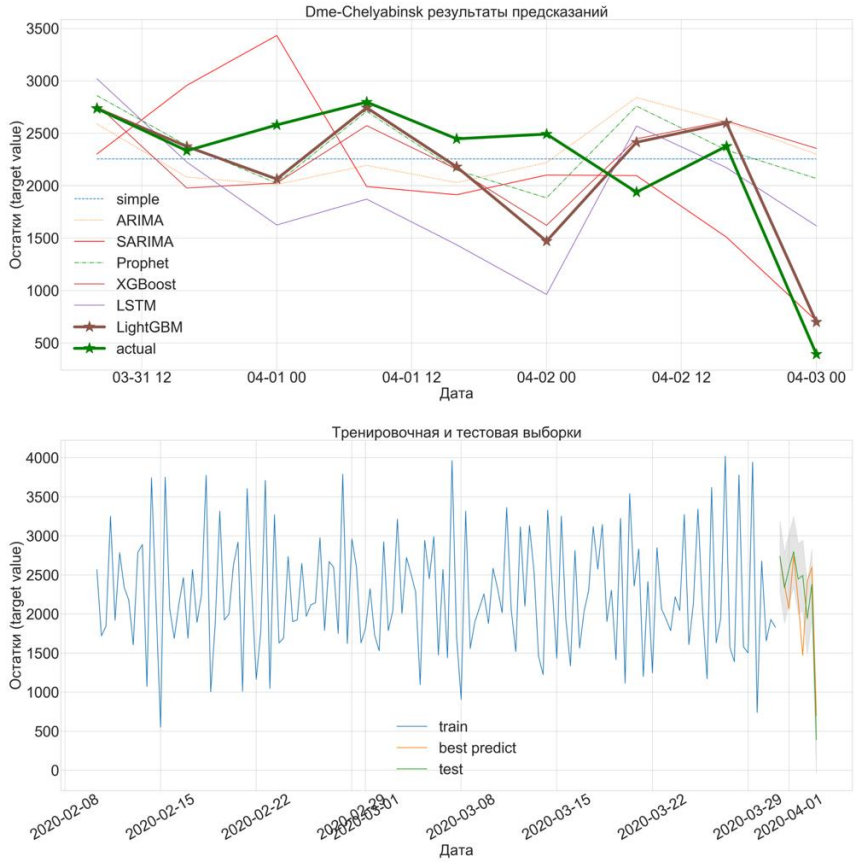

Рисунок 2.11 – Результаты для плеча Домодедово АОПП – Челябинск АОПП

Рисунок 2.12 – Результаты для плеча Домодедово АОПП – Красноярск АОПП

Рисунок 2.13 – Результаты для плеча Домодедово АОПП – Новосибирск АОПП

Рисунок 2.14 – Результаты для плеча Внуково АОПП – Казань АОПП

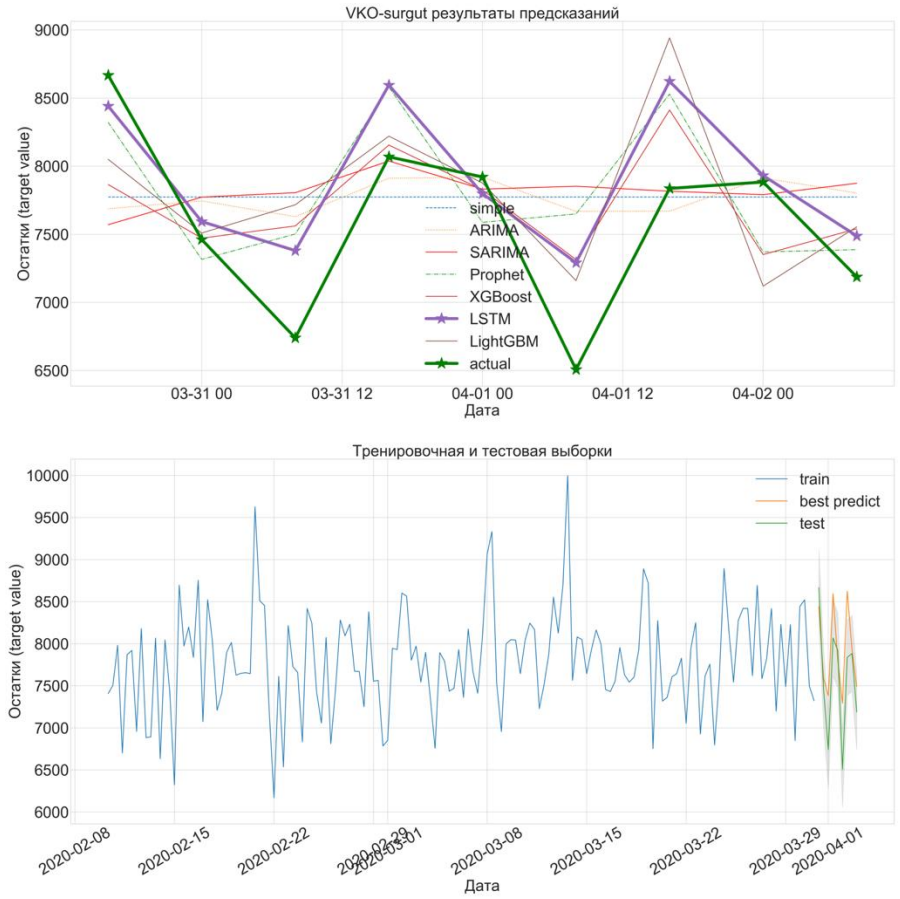

Рисунок 2.15 – Результаты для плеча Внуково АОПП – Сургут АОПП

Таблица 2.1 – Результаты для метрики Mean Absolute Error (MAE)

Модель / Плечо	ARIMA	SARIMA	XGBoost	LightGBM	Prophet	LSTM	Simple	Mean
SVO – Владивосток	1435	1852	3120	2289	2096	1916	1346	16871
SVO –Иркутск	560	728	601	558	577	466	560	4704
SVO – Екб	166	165	176	131	183	160	175	1329
DME – Новосибирск	1932	2259	2559	2246	2321	2541	1969	38616
DME – Челябинск	589	501	561	344	473	763	461	2255
DME – Красноярск	791	596	747	859	1178	890	792	15367
VKO – Сургут	476	526	454	525	516	396	522	7774
VKO – Казань	235	260	477	396	532	353	245	3095

Таблица 2.2 – Результаты для метрики Root Mean Square Error (RMSE)

Модель / Плечо	ARIMA	SARIMA	XGBoost	LightGBM	Prophet	LSTM	Simple model
SVO – Владивосток	2469	2297	3348	3376	2758	2862	2356
SVO – Иркутск	764	745	844	822	822	543	734
SVO – Екб	248	236	248	200	256	212	248
DME – Новосибирск	2615	3014	3219	2765	3004	2733	2753
DME – Челябинск	783	602	781	441	689	892	692
DME – Красноярск	1177	1215	1100	1225	1482	1341	1177
VKO – Сургут	633	725	550	643	594	482	669
VKO – Казань	427	510	590	513	631	428	501

Таблица 2.3 – Результаты при подсчете количества раз, сколько предсказание отклонилось от истинного значения более чем на 20% от среднего значения таргета (из 9 отсчетов)

Модель / Плечо	ARIMA	SARIMA	XGBoost	LightGBM	Prophet	LSTM
SVO – Владивосток	5	6	8	7	6	4
SVO – Иркутск	1	2	1	2	2	2
SVO – Екб	2	2	2	1	2	2
DME-Новосибирск	0	0	0	0	0	0
DME – Челябинск	4	4	4	3	4	6
DME – Красноярск	0	0	0	0	0	0
VKO – Сургут	4	4	5	5	5	4
VKO – Казань	2	2	4	2	5	2

В таблицах 2.1-2.3 зеленым цветом выделены лучшие модели, красным – худшие.

В результате проделанной работы по соотношению скорости выполнения и показанных результатов по значениям метрики, была отобрана модель градиентного бустинга LightGBM для выведения в производственный процесс.

ЗАКЛЮЧЕНИЕ

В магистерской диссертации был проведен анализ подходов к прогнозированию временных рядов, изученные подходы были протестированы и отобран лучший в виде модели градиентного бустинга, которая была интегрирована в производственный процесс.

В ходе выполнения возникали трудности с улучшением качества прогнозов, которые были решены путем добавления признаков, которые позволили лучше описать целевую функцию. Также в работе возникали трудности с выгрузкой большого объема данных, применение более оптимизированных SQL-запросов ускорило время выполнения обработки данных.

В результате проделанной работы, полученная информация о количестве остатков в аэропорту используется для снижения нагрузки на авиасообщения и приведет к более быстрому решению доставки посылок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мишина Л.А. Конспект лекций по логистике. Эксмо, 2008. – 160 с.
2. Документация Python [Электронный ресурс]. URL: https://docs.python.org/3/.
3. Travis E. Oliphant A guide to NumPy, USA: Trelgol Publishing, 2006.
4. Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Edouard Duchesnay, Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 2011.
5. Wes McKinney, Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 2010.
6. John D. Hunter, Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 2007
7. Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills, Advanced Analytics with Spark: Patterns for Learning from Data at Scale . – 2nd Edition. O'Reilly Media, 2017. – 280 с.
8. Douglas C. Montgomery, Cheryl L. Jennings, Murat Kulahci Introduction to Time Series Analysis and Forecasting. – 1st Edition.
9. Hyndman, R.J., & Athanasopoulos, G, Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2, 2018.
10. Statistical forecasting: notes on regression and time series analysis [Электронный ресурс]. URL: https://people.duke.edu/~rnau/411home.htm
11. Course of statistical methods [Электронный ресурс]. URL: https://online.stat.psu.edu/stat510/
12. Алесинская Т.В. Основы логистики. Общие вопросы логистического управления. - Таганрог: ТРТУ, 2005.

13. ARIMA Model – Complete Guide to Time Series Forecasting in Python [Электронный ресурс]. URL: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
14. Gradient Boosting with Scikit-Learn, XGBoost, LightGBM, and CatBoost [Электронный ресурс]. URL: https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/
15. Butch Quinto Next-Generation Machine Learning with Spark: Covers XGBoost, LightGBM, Spark NLP, Distributed Deep Learning with Keras, and More. Apress, 2020.
16. Tianqi Chen, Carlos Guestrin XGBoost: A Scalable Tree Boosting System, 2016.
17. Guolin Ke, Qi Meng , Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu LightGBM: A Highly Efficient Gradient Boosting Decision Tree // Microsoft Research . Advances in Neural Information Processing Systems 30 (NIP 2017), 2017.
18. Sean J. Taylor, Ben Letham Prophet: forecasting at scale, 2017
19. Steven Elsworth and Stefan Guttel, Time Series Forecasting Using LSTM Networks: A Symbolic Approach, 2020