УДК 004.852

ИСПОЛЬЗОВАНИЕ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРОГНОЗИРОВАНИЯ И ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ БАЗ ДАННЫХ

Исомадинов А.Х., студент гр. ИТб-202, III курс
Научный руководитель: Ванеев О.Н, к.т.н., доцент
Кузбасский государственный технический университет
имени Т. Ф. Горбачева
г. Кемерово

Современные приложения и системы обрабатывают огромные объемы данных и для их хранения используют базы данных. Однако, с увеличением объемов данных и количества пользователей производительность баз данных может снижаться, что приводит к задержкам и сбоям системы. В данной статье мы рассмотрим использование машинного обучения для прогнозирования и оптимизации производительности баз данных.

В данной статье мы предлагаем использовать машинное обучение для создания моделей, которые могут прогнозировать, какие запросы будут выполняться наиболее часто и какие данные будут наиболее активно использоваться в базах данных. Затем мы можем использовать эти модели для оптимизации производительности баз данных, например, путем предварительной загрузки данных или настройки индексов на основе прогнозов модели.

Шаг 1: Сбор данных

Первый шаг в этом процессе - это сбор данных из базы данных. Мы можем использовать данные о выполненных запросах, типах запросов и использованных данных, чтобы создать обучающий набор данных для нашей модели машинного обучения.

Шаг 2: Подготовка данных

После сбора данных мы должны подготовить их для использования в модели машинного обучения. Это может включать в себя предобработку данных, такую как очистку, нормализацию и фильтрацию.

Шаг 3: Создание модели машинного обучения

Следующим шагом будет создание модели машинного обучения на основе нашего обучающего набора данных. Мы можем использовать различные алгоритмы машинного обучения, такие как решающие деревья, случайный лес или глубокие нейронные сети, чтобы создать модель, которая может прогнозировать, какие запросы будут выполняться наиболее часто и какие данные будут наиболее активно использоваться в базах данных.

Шаг 4: Оптимизация производительности баз данных

Последний шаг заключается в использовании прогнозов модели для оптимизации производительности баз данных. Мы можем использовать эти прогнозы, чтобы предварительно загружать данные или настраивать индексы, что

позволит ускорить выполнение запросов и уменьшить нагрузку на базу данных.

Конкретный пример использования существующих моделей для прогнозирования и оптимизации производительности баз данных - это использование модели прогнозирования нагрузки для оптимизации работы базы данных.

Одной из существующих моделей для прогнозирования нагрузки на базу данных является модель ARIMA (Autoregressive Integrated Moving Average), которая используется для прогнозирования временных рядов. Эта модель может быть применена для прогнозирования количества запросов к базе данных в будущем на основе исторических данных.

Другой пример - использование алгоритма кластеризации, такого как K-means, для оптимизации расположения данных в базе данных. Алгоритм K-means может использоваться для группирования данных, которые часто запрашиваются вместе, и размещения их на одном физическом диске, что может существенно снизить время выполнения запросов к базе данных.

Также существуют различные средства управления базами данных, которые используют машинное обучение для оптимизации производительности, такие как IBM Db2 AI, Oracle Autonomous Database и Microsoft Azure SQL Database. Они используют алгоритмы машинного обучения для автоматической настройки баз данных, в том числе настройки индексов, фрагментации данных и сжатия, для оптимизации производительности и снижения нагрузки на базу данных.

Для использования модели прогнозирования нагрузки на базу данных с помощью SQL, необходимо выполнить следующие шаги:

1.Извлечь данные о нагрузке на базу данных из SQL базы данных. Для этого можно использовать SQL-запросы. Если вы хотите получить данные о выполненных запросах, вы можете использовать DMV sys.dm_exec _query_stats, который содержит статистику выполнения запросов.

Пример запроса, который возвращает дату и количество запросов для каждой даты из DMV представлен на рисунке 1.

```
SELECT CAST(DATEADD(DAY, DATEDIFF(DAY, 0, creation_time), 0) AS DATE) AS
query_date, COUNT(*) AS query_count
FROM sys.dm_exec_query_stats
GROUP BY CAST(DATEADD(DAY, DATEDIFF(DAY, 0, creation_time), 0) AS DATE)
ORDER BY query_date ASC
```

Рисунок 1

Этот запрос использует функции DATEDIFF и DATEADD для извлечения даты из столбца «creation_time» DMV sys.dm_exec_query_stats. Затем он считает количество запросов в каждую дату и сортирует результаты по дате.

2.Сохранить данные в формате CSV. Формат CSV удобен для работы с табличными данными, так как он позволяет легко читать и записывать данные, а также быстро обрабатывать их в различных языках программирования.

```
-- Создаем временную таблицу для хранения результатов запроса
CREATE TABLE temp_results (
    date DATE,
    query_count INT
)

-- Выполняем запрос и сохраняем результаты во временную таблицу
INSERT INTO temp_results
SELECT CAST(DATEADD(DAY, DATEDIFF(DAY, 0, creation_time), 0) AS DATE) AS
query_date, COUNT(*) AS query_count
FROM sys.dm_exec_query_stats
GROUP BY CAST(DATEADD(DAY, DATEDIFF(DAY, 0, creation_time), 0) AS DATE)
ORDER BY query_date ASC
      — - Выводим результат. —-
select * from temp_results
 - - удаляем временную таблицу - -
drop table temp_results
```

Рисунок 2

На рисунке 2 представлен SQL-скрипт, которой создает временную таблицу temp_results, выполняет запрос на извлечение данных из таблицы sys.dm_exec_query_stats и сохраняет результаты в созданную таблицу temp_results. Далее, производится выборка всех данных из этой таблицы с помощью команды ‘select * from temp_results’. И наконец, временная таблица удаляется с помощью команды ‘drop table temp_results’. В результате выполнения этого скрипта мы получим все данные из таблицы ‘sys.dm_exec_query_stats’ в виде таблицы, которую можно просмотреть и экспортировать для дальнейшего анализа.

Для сохранения результатов запроса в формате CSV можно выполнить следующие действия:

1.Выполнить запрос и получить результаты в окне «Результаты».

2.Правой кнопкой мыши щелкнуть по результатам запроса и выбрать «Сохранить как...» (рисунок 3).

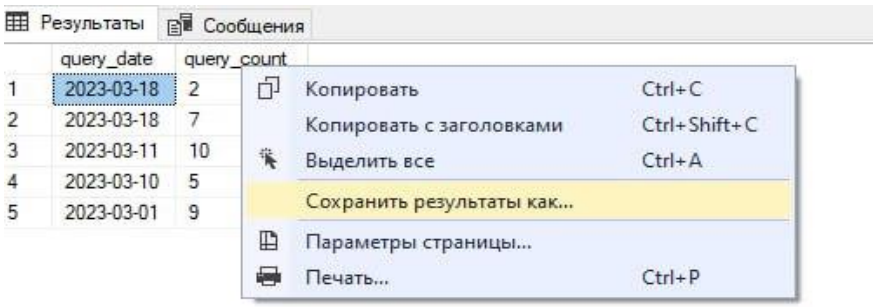

Рисунок 3

3.В диалоговом окне «Сохранить как» выбрать формат «CSV (разделители-запятые) (*.csv)» (рисунок 4).

4.Выбрать место для сохранения файла и нажать «Сохранить».

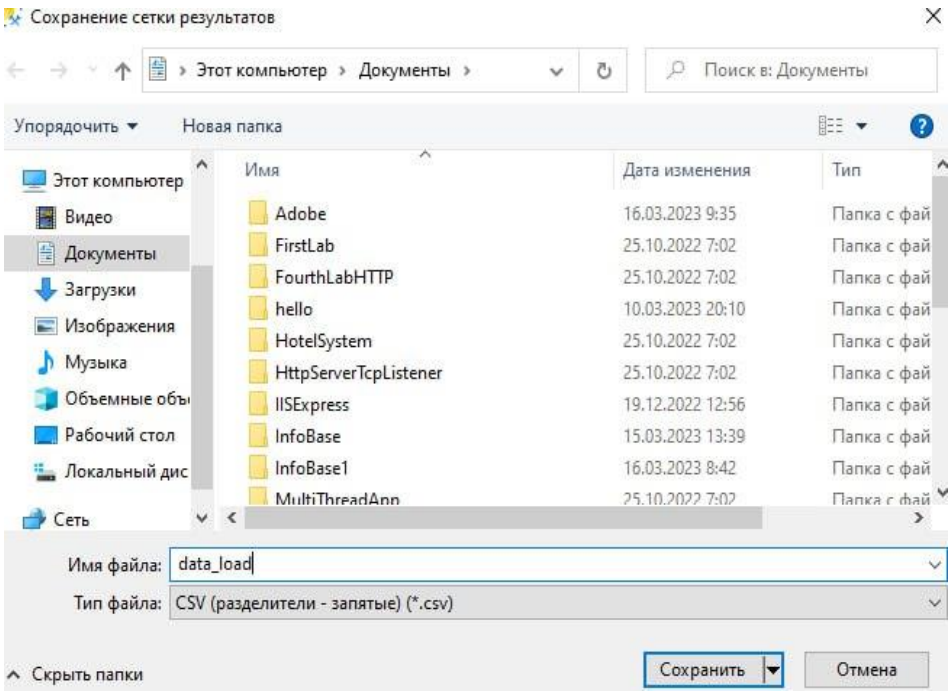

Рисунок 4

Таким образом, данные будут сохранены в формате CSV (рисунок 5).

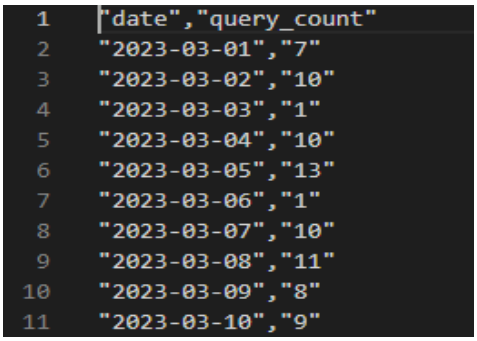

Рисунок 5

Если у вас не создались столбцы «date», «query_count», то можно прописать в файле самому, как показано в примере.

В данном примере было принято решение увеличить объем данных для сохранения в формате CSV с целью повышения точности модели.

3.Использовать код модели для прогнозирования нагрузки на базу данных, используя извлеченные данные в формате CSV.

Код (рисунок 6) реализует метод ARIMA (Autoregressive Integrated Moving Average) для прогнозирования временных рядов, используя библиотеки pandas, numpy и statsmodels.tsa на языке Python.

```
import pandas as pd
import numpy as np

#это модуль библиотеки statsmodels, который предоставляет классы и функции для
оценки моделей авторегрессии.
from statsmodels.tsa.arima.model import ARIMA
# загрузка данных о количестве запросов к базе данных из CSV файла
data = pd.read_csv('data_load.csv', index_col='date', parse_dates=['date'])

# построение модели ARIMA для прогнозирования количества запросов на следующие
10 дней
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit()

# создание прогнозов на следующие 10 дней
next_month = pd.date_range(start=data.index[-1], periods=10, freq='D')
forecast = model_fit.forecast(steps=10)

# вывод результатов
print('Прогноз нагрузки на следующие 10 дней:')
print(forecast)
```

Рисунок 6

Библиотека pandas используется для загрузки и обработки временных рядов. Она предоставляет удобный интерфейс для работы с данными, включая функции для загрузки и хранения временных рядов, агрегирования, фильтрации и сортировки данных.

Библиотека numpy используется для работы с массивами и матрицами чисел. Она предоставляет эффективные алгоритмы для работы с данными и вычислений на многомерных массивах, что делает ее идеальной для научных вычислений, таких как обработка данных временных рядов.

Библиотека statsmodels.tsa предоставляет класс ARIMA для моделирования и прогнозирования временных рядов. ARIMA является стандартным методом прогнозирования временных рядов, который учитывает автокорреляцию, тренд и сезонность.

Выбор прогноза на 10 дней обусловлен ограниченным объемом данных в базе данных запросов. Для получения точных прогнозов необходимо иметь достаточно большой объем данных. В данном случае, прогноз на 10 дней выбран как наиболее оптимальный для имеющегося объема данных.

Таким образом, использование библиотек pandas, numpy и statsmodels.tsa позволяет эффективно обрабатывать временные ряды, моделировать и прогнозировать их значения.

Структура программных компонентов реализующих пример обработки с использованием модели Arima представлена на рисунке 7.

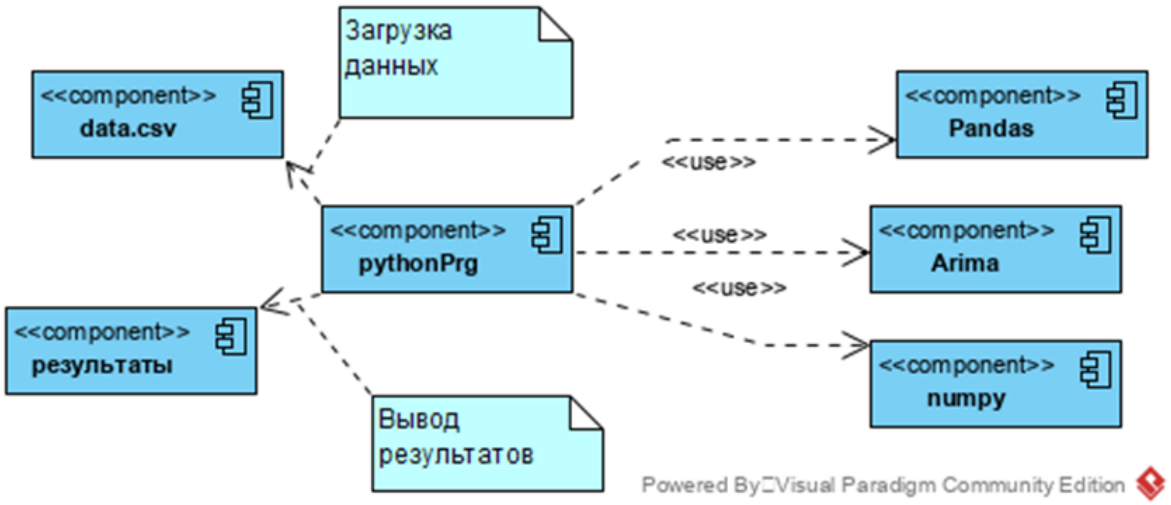

Рисунок 7

Результат работы программы показан на рисунке 8.

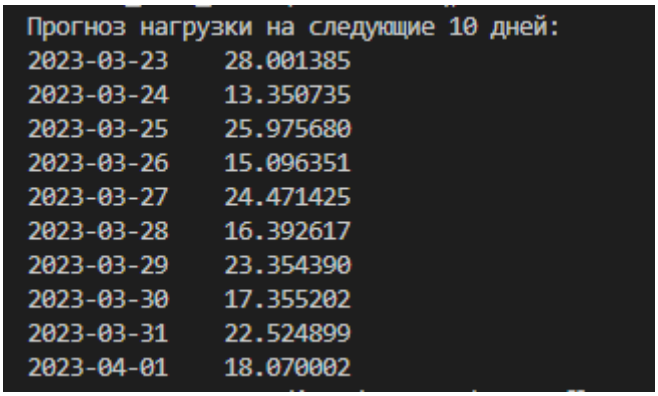

Рисунок 8

4.Использовать полученные прогнозы для оптимизации работы базы данных, например, для настройки параметров базы данных, таких как размер буферного кэша или количество соединений, чтобы улучшить производительность в соответствии с прогнозируемой нагрузкой. Обратите внимание, что при использовании модели для прогнозирования нагрузки на базу данных с помощью SQL, необходимо также учитывать возможные изменения в поведении пользователей и другие факторы, которые могут повлиять на количество запросов к базе данных.

В данной статье рассмотрено использование машинного обучения для прогнозирования и оптимизации производительности баз данных. Описано методология, включающая сбор и подготовку данных, создание модели машинного обучения и использование прогнозов модели для оптимизации производительности баз данных.

Применение машинного обучения для оптимизации производительности баз данных может привести к значительному улучшению производительности и уменьшению нагрузки на базу данных. Кроме того, эта методология может

быть использована для автоматизации процесса оптимизации производительности баз данных, что позволит сэкономить время и усилия администраторов баз данных.

В будущем исследования в этой области могут включать более точные методы прогнозирования и оптимизации производительности баз данных с помощью машинного обучения, а также применение этой методологии для различных типов баз данных и сред разработки.

Список литературы:

1. Машинное обучение в среде SQL Server [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/520112/, свободный. (Дата обращения 11.03.2023)

2. Лучшие базы данных, поддерживающие машинное обучение в базе данных [Электронный ресурс]. – Режим доступа: https://www.shunlongwei.com/ru/top-databases-supporting-in-database-machine-learning/, свободный. (Дата обращения 11.03.2023)

3. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.