Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук
Основная образовательная программа
Прикладная математика и информатика

КУРСОВАЯ РАБОТА

Исследовательский проект на тему

«Разработка и исследование метода прогнозирования популярности видео в сетях доставки контента»

Выполнил студент группы 183, 3 курса,
Рамусь Владислав Геннадьевич

Руководитель КР:
научный сотрудник Сухорослов Олег Викторович

Консультант:
научный сотрудник Пашков Василий Николаевич

Москва 2021

Аннотация

Методы машинного обучения могут использоваться для повышения качества обслуживания сетей доставки контента (CDN). Чтобы минимизировать задержку доставки, в CDN наиболее популярный видеоконтент кэшируется рядом с конечными пользователями. В данной курсовой работе производится формализация задачи прогнозирования популярности видео в CDN, осуществляется обзор существующих подходов и приводятся результаты экспериментальных исследований на основе применения линейной регрессии и градиентного бустинга.

Ключевые слова — кэширование в сетях доставки контента, прогнозирование популярности видео методами машинного обучения.

Содержание

Введение

В современном мире очень распространено использование инфокоммуникационных сетей для передачи информации во всех сферах жизни. Количество устройств, которые могут работать через сеть, очень велико, и объём передаваемых данных постоянно увеличивается. Именно поэтому возникают вопросы о том, как наиболее эффективно использовать сетевые ресурсы и технологии передачи данных, чтобы доставка контента конечному пользователю была быстрой и надёжной. Для решения данной задачи используют сети доставки и распространения содержимого (англ. Content Delivery Network или Content Distribution Network, CDN).

Сети доставки контента – географически распределённая сетевая инфраструктура для быстрой и надёжной доставки контента пользователям. Она состоит из множества узлов – кэш-серверов в разных частях страны или мира, расположенных в обслуживающих их центрах обработки данных. Когда пользователь хочет получить доступ к содержимому некоторого ресурса, подключённого к сети доставки контента, он попадает на ближайший её узел и получает данные оттуда. Такой подход помогает оптимизировать предоставление контента пользователям: они получают его с меньшей задержкой, чем если бы каждый запрос отправлялся сразу на исходный сервер.

Основная задача, которую решает CDN, – уменьшение времени загрузки страниц ресурса за счёт быстрого отклика содержимого в кэше. Таким образом клиент быстрее получает ответ на свой запрос, вследствие чего уменьшается отток клиентов из-за долгой загрузки ресурса, а значит становятся ниже финансовые потери бизнеса. Вместе с тем веб-ресурс, к которому посетители быстро получают доступ, выше располагается в поисковом рейтинге, что немаловажно для роста пользовательской базы. Помимо этого, снижается риск того, что доступ к содержимому утратится из-за сбоев исходного сервера: контент будет хранится на распределённых узлах до восстановления работоспособности системы.

CDN устроены таким образом, что система кэширует нужный контент на промежуточных узлах, находящихся в разных географических зонах. Когда пользователь отправляет запрос, он направляется на ближайший узел, содержащий требуемые данные. Вследствие этого ответ проходит меньшее расстояние, чем если бы запрос отправлялся на главный сервер ресурса.

После подключения к сети доставки контента сайт (или любой другой веб-ресурс) будет работать на том же оригинальном сервере, а на промежуточные узлы будут загружены его кэшированные части. Для конечного пользователя при этом ничего не поменяется, так как в результате его запроса к ресурсу сформируется тот же контент, с той лишь разницей, что некоторая его доля – самая востребованная или самая “тяжелая” – будет взята из промежуточного узла.

Кэширование содержимого ресурса чаще всего происходит при первичном обращении пользователя. Это значит, что максимальное количество времени, потраченное на загрузку контента будет у первого пользователя, обратившегося к основному серверу. Далее все пользователи из этого региона получат быстрый доступ к содержимому ресурса благодаря его кэшированию на ближайшем промежуточном CDN-сервере. Каждый пользовательский запрос подразумевает обращение к ближайшему промежуточному узлу, содержащему требуемый контент. А если содержимое отсутствует в данном узле, то запрос передаётся соседям и соседям соседей далее по цепочке до тех пор пока, искомый контент не будет получен.

Следует учитывать, что серверы CDN не являются хранилищами файлов. Их используют только для кэширования содержимого веб-ресурсов. Существует возможность выбрать, какие части ресурса добавить в CDN. Для веб-ресурса и мобильных приложений на промежуточных узлах можно кэшировать все страницы ресурса, только некоторые типы файлов или некоторые поддомены. Несмотря на то что в промежуточные узлы сетей доставки контента можно помещать ресурс полностью, целесообразно кэшировать элементы, занимающие много пространства, или данные, которые практически

не подвергаются изменениям.

В последние годы возросла популярность видеоконтента. Люди стали чаще обращаться как к коротким роликам, так и к длинным видеозаписям. Это большие объёмы данных, и для доставки такого рода контента может потребоваться много сетевых ресурсов, в числе которых хранение видео и технологии передачи информации. Чтобы снизить издержки, службы потокового вещания обращаются к CDN.

Пользуясь всеми преимуществами сетей доставки контента, можно обеспечить эффективное взаимодействие пользователя с системой распространения контента. Однако у сетей доставки контента существуют и недостатки, среди которых, например, задержка кэширования: на центральном сервере или в близлежащих к нему узлах контент обновился, а на переферии может храниться старая копия файла.

Также острой проблемой является то, что хранилище каждого промежуточного кэш сервера ограничено физическими характеристиками сервера (размером памяти). Это означает, что невозможно постоянно хранить весь контент, прибывший в данный узел CDN сети. Таким образом, необходимо выбирать видео объекты для каждого региона, которые следует оставлять в кэш серверах CDN сети, и выбирать объекты, которые следует удалить из CDN, то есть необходима стратегия управления и обновления контента CDN сети.

Эта стратегия может быть основана на показателях популярности и востребованности видео объекта пользователями данного региона. Поэтому актуальной является задача прогнозирования популярности видеообъекта, рассматриваемая в данной курсовой работе, так как на основе построенного прогноза можно заранее размещать в CDN потенциально наиболее востребованные пользователями видеообъекты, повысить эффективность использования CDN и сократить время доступа пользователей к наиболее популярным видеообъектам.

1 Цели и задачи работы

Целью курсовой работы является разработка и исследование метода машинного обучения, позволяющего прогнозировать популярность видео в конкретном географическом регионе для повышения эффективности размещения видео в сетях доставки контента. Для достижения поставленной цели необходимо решить следующие задачи:

- Провести формальную постановку задачи прогнозирования популярности видео.
- Провести обзор и сравнительный анализ существующих методов для прогнозирования популярности видео.
- Разработать метод построения прогноза популярности видео.
- Собрать и подготовить исходные данные для проведения экспериментов.
- Реализовать разработанный метод в виде программного средства.
- Разработать методику экспериментального исследования, провести эксперименты и проанализировать полученные результаты.

2 Постановка задачи

Введём дискретное время в модели сети CDN. Пусть t_0 – время начала работы сети доставки контента, шаг дискретизации равен одним суткам. Пусть сеть CDN характеризуется следующими параметрами:

- S_{max} – максимальный доступный объём памяти кэш-сервера, одинаковый у всех серверов сети;
- N – количество видео, которые потенциально могут размещаться в кэш-сервере;
- $V = \{v_j\}_{j=1}^{N}$ – множество всех видеообъектов.

Пусть каждый видеообъект характеризуется следующими параметрами:

- id_j — уникальный id;
- $attr_j$ — список статических и динамических признаков видео;
- $y_j(t)$ — количество запросов пользователей к объекту i_j за последние сутки $((t-1, t], t > t_0)$;
- $Y_T^j = \{y_j(t), t \in [t_0, T]\}$ — история запросов пользователей к объекту i_j с начала работы системы CDN и до текущего момента;

Функция, **определяющая объём занятой памяти** в кэш-сервере в момент времени t:

$$S(t) = \sum_{j=1}^{N} s_j$$

Свободная память вычисляется как разность между максимальным объёмом и занятой памятью:

$$S_{free}(t) = S_{max} - S(t)$$

Функцией прогнозирования назовём $F(i_j, t)$, возвращающую прогнозируемое количество запросов к информационному объекту i_j в момент времени $t + 1$.

Ряд прогнозируемых значений количества обращений пользователей к видео i_j будет основываться на показателях за предыдущие дни:

$$\hat{y}_j(t+1) = F(i_j, t)$$

Пусть M — количество объектов, которые будут добавлены в кэш-сервер CDN в момент времени $t+1$.

Введём функцию MSE для оценки точности прогноза:

$$\text{MSE}(y_j(t), \hat{y}_j(t)) = \frac{1}{N}\sum_{j=1}^{N}(y_j(t) - \hat{y}_j(t))^2$$

А также корень из MSE:

$$\text{RMSE}(y_j(t), \hat{y}_j(t)) = \sqrt{\text{MSE}}$$

Таким образом, для оценки точности прогноза мы сравниваем реальное количество обращений к объекту $y_j(t)$ с $F(i_j, t-1)$. Чем меньше значение MSE, тем лучше построенный прогноз.

Эти функции оптимизируют алгоритм обучения, однако для оценки качества построенной модели необходимо использовать *коэффициент детерминации.*

$$\text{R}^2(y_j(t), \hat{y}_j(t)) = 1 - \frac{\sum_{j=1}^{N}(y_j(t) - \hat{y}_j(t))^2}{\sum_{j=1}^{N}(y_j(t) - \overline{y})^2},$$

где $\overline{y}$ – среднее значение целевой переменной.

Коэффициент детерминации измеряет долю дисперсии, объяснённую модель, в общей дисперсии целевой переменной. Данная мера качества – это нормированная среднеквадратичная ошибка. [1]

В данной курсовой работе рассматривается одноуровневый кэш. Это значит, что необходимо определить, помещается ли видео в конкретный кэш-сервер или нет. Для построения прогноза популярности используются алгоритмы машинного обучения. Таким образом, можно декомпозировать задачу

на две подзадачи.

Первая – прогнозирование количества запросов пользователей к конкретному видео внутри CDN в момент времени $t + 1$.

Дано:

- множество I из N видео;
- история запросов к каждому видео Y_T^j.

Найти:

- Функцию прогнозирования такую, что $\text{MSE}(y_j(t), \hat{y}_j(t)) \to min$.

Вторая задача – размещение видео в кэше в момент времени $t + 1$ согласно прогнозам их популярности.

Дано:

- множество V из N видео;
- прогноз популярности $\hat{y}_j(t + 1)$ для каждого объекта из V.

Найти:

- Подмножество видео I_M^k, которые будут добавлены в CDN в момент времени $t + 1$.

Ограничения:

- $S(t + 1) \leqslant S_{max}$ — размер занятой памяти в указанный момент не превышает максимально допустимый объём хранилища сервера;

3 Обзор методов

Целью обзора является рассмотрение уже существующих практик прогнозирования популярности контента. Это позволит учесть особенности решения поставленной задачи.

Чтобы определить, насколько полезным будет применение техник, рассмотренных в публикациях, сформулируем критерии обзора. Статьи будут сравниваться исходя из того:

1. какая характеристика прогнозируется (таргет, целевая переменная);
2. какое количество моделей рассматривается;
3. откуда берутся входные данные; [2]
4. на какие признаки опирается прогноз;
5. какие стратегии размещения контента в кэше используются.

3.1 Онлайновое машинное обучение: прогнозирование на основе экспертов в сетях доставки контента

Идея, развиваемая в этой статье [3], заключается в использовании методов прогнозирования оценки будущей популярности видеоконтента, чтобы решить, что следует кэшировать. Авторы рассматривают различные методы прогнозирования, называемые «экспертами». Чтобы оценить точность прогнозов популярности, сделанных экспертами, авторы оценивают экспертов по трём критериям: совокупная потеря, максимальная мгновенная потеря и лучший рейтинг.

Вместе с тем показывается важность механизма, принимающего решения: так называемый «прогнозист». Данный программный компонент предсказывает популярность на основе прогнозов нескольких экспертов. Прогнозист, основанный на K лучших экспертах, превосходит по совокупным потерям прогнозы отдельных экспертов и прогнозы, основанные только на одном эксперте, даже если этот эксперт меняется со временем.

В ходе статьи определяются все используемые эксперты и объясняется, как они вычисляют свои предсказания. Эти предсказания используются прогнозистом для построения собственного предсказания. Авторы выделяют два типа прогнозистов: Лучший эксперт (BE, Best Expert) и K лучших экспертов (KBE). В рамках исследования рассматривается различный видеоконтент и оценивается точность предсказаний экспертов и прогнозистов.

Авторы оценивают популярность видеоконтента по количеству его запросов за определённый период времени. Предсказание популярности происходит посредством аппроксимации функции, представляющей постепенное изменение (эволюцию) запросов во времени. Для этой цели авторы оценивают различные методы предсказания, применяя их к реальным данным из YouTube CDN.

Постановка задачи. Сначала на платформе YouTube извлекаются реальные данные о случайно выбранных видео, описывающие ежедневную эволюцию запросов со временем случайно выбранного видеоконтента. Далее предполагается, что количество запросов определяет популярность видеоконтента. В итоге строится предсказание популярности видео в следующие сутки на основе данных о запросах в предыдущие дни.

Результат. В статье была проведена оценка производительности различных методов прогнозирования, среди которых: экспоненциальное сглаживание, полиномиальная регрессия, фильтрация Савицкого-Голея. Результаты моделирования показывают, что прогнозист KBE превосходит прогнозиста BE.

В дальнейшей работе авторы будут использовать прогнозы популярности контента, чтобы решить, какой контент будет кэшироваться в узлах рядом с конечным пользователем. Определенный процент размера кэша будет зарезервирован для наиболее популярного содержимого. Оставшаяся емкость хранилища будет разделена между запрошенным содержимым в зависимости от их прогнозируемой популярности. Такой метод кэширования будет сравниваться с классическими методами по коэффициенту попадания.

3.2 Применение машинного обучения в мобильных социальных сетях

Приложения для мобильных социальных сетей (MSN) стали самыми крупными платформами для пользователей, позволяющими потреблять контент и требующие большого количества данных. MSN занимает первое место среди различных популярных мобильных приложений с точки зрения количества активных пользователей в день и ежедневного времени, проводимого пользователями.

Однако провести интеллектуальную доставку контента, то есть доставить нужный контент соответствующей группе пользователей MSN, нетривиально из-за следующих проблем.

В этой статье [12] представлена основанная на машинном обучении структура для прогнозирования распределения потребностей пользователей MSN в контенте.

Фреймворк машинного обучения для CDN. Известно, что методы машинного обучения помогают прогнозировать популярность, так что поставщик может заранее кэшировать популярный контент на пограничных серверах или устройствах для быстрой доставки контента. Эта статья отвечает на следующий вопрос: какая модель машинного обучения подходит для прогнозирования популярности контента?

Структура фреймворка включает в себя уровень потребителя контента, уровень сотовой сети и пограничный уровень CDN. В данной архитектуре регион — это административное деление страны.

Предлагаемая фреймворк на основе машинного обучения выполняет две функции прогнозирования.

- **Функция 1** для прогнозирования количества запросов контента в данном регионе и за определенный период времени;
- **Функция 2** для прогнозирования количества запросов к каждому типу содержимого (то есть предпочтений пользователей по каждому типу

содержимого).

Задержка уменьшится, если алгоритм машинного обучения сможет предсказать распределение пользовательских запросов к определенному контенту в будущем, и этот контент можно будет заранее разместить на пограничном сервере. В результате контент может быть извлечен с пограничного сервера на пользовательские устройства, а не из центра обработки данных к пользователю, что экономит время передачи, то есть снижает задержку.

Вышеупомянутые две функции — это две задачи регрессии, в которых исторические данные используются для оценки или прогнозирования популярности контента в будущем. Здесь популярность контента может быть измерена количеством запросов к этому контенту, а предпочтения пользователя в отношении типа содержимого можно измерить по количеству запросов пользователя к этому типу содержимого.

Существует множество моделей машинного обучения для решения задачи регрессии. В этом исследовании сравниваются четыре модели, а именно линейную регрессию (LR), метод опорных векторов (SVR), случайный лес (RF), нейронную сеть (NN).

Модели LR, SVR, RF и NN имеют *одинаковые* входные и выходные данные для двух функций.

- В **функции 1** входные данные представляют собой набор признаков: дату, день недели / выходной день, время и количество просмотров страниц за последние U часов для каждого региона; вывод – это количество просмотров страницы за $(U+1)$-й час в заданном регионе.
- В **функции 2** входными данными является количество просмотров страниц содержимого в определенной категории (то есть определенного типа содержимого) за последние D дней для каждого региона; и вывод – сколько их будет в $(D+1)$-й день в каждом регионе.

Результат. Набор данных в этом эксперименте разделен на две части, так что первая часть, содержащая данные за первые 20 дней, используется для обучения, а другая часть, включая остальную часть набора данных, ис-

пользуется для оценки. Авторы статьи выбрали лучшие гиперпараметры с 10-кратной перекрестной проверкой при обучении модели. Экспериментальные результаты показывают, что модель нейронной сети превосходит другие подходы (LR, SVR, RF) в улучшении коэффициента совпадений на 30% и экономии стоимости сети на 15%.

3.3 Методы кэширования видео на основе их популярности

В этой статье основное внимание уделялось кэшированию на основе методов прогнозирования популярности. Популярность видео можно измерить с помощью различных показателей, среди которых чаще всего используется количество просмотров видео. Другие показатели включают время просмотра видео и рейтинг. В последнее время кэшированию уделяется больше внимания с точки зрения сети. Было опубликовано несколько обзоров сетей с поддержкой кэширования, в которых основное внимание уделяется методам размещения реплик, оптимизации кэширования, управлению ресурсами в сети и энергоэффективному кэшированию. Здесь авторы сконцентрировались только на ожидании видеоконтента. В этой статье обобщаются работы, проделанные в области кэширования видео на основе популярности, и рассматриваются открытые исследовательские задачи в этой области.

Было предложено несколько алгоритмов, которые выбирают содержимое для кэширования или удаления, и их можно классифицировать как традиционные методы кэширования или методы кэширования на основе популярности.

Традиционные схемы кэширования предполагают, что текущий популярный контент останется популярным.

В сетях кэширования до сих пор используются традиционные методы из-за их простоты и легкости реализации. Однако они игнорируют поведение пользователя и шаблоны запросов. Учитывая ограничения в хранилище и

полосе пропускания, можно утверждать, что существует потребность в улучшенных схемах кэширования, которые предсказывают наиболее запрашиваемый контент и делают его доступным в единицах кэширования до часов пик.

Основанное на популярности видео кэширование. Желаемый алгоритм должен быть быстрым, способным справиться с большой рабочей нагрузкой и обеспечивать точные прогнозы. Авторы статьи рассматривают [13] три проблемы алгоритмов прогнозирования популярности: точные и надежные прогнозы, быстрые прогнозы и масштабируемость. Затем исследуется общая структура схем кэширования на основе популярности.

Точные и надёжные прогнозы. Прогнозирование поведения пользователей – нетривиальная задача, поскольку будущая популярность контента недоступна в качестве априорной информации, но ее можно измерить с помощью таких параметров, как качество контента, исторические данные и социальные взаимодействия. Даже в этом случае фактическая популярность может сильно отличаться от предполагаемой из-за внешних факторов, неожиданных ситуаций, упущения важных признаков в прогнозе или устаревших параметров модели прогнозирования. Неточные методы прогнозирования ухудшают производительность сети и могут быть даже менее надежными, чем традиционные стратегии кэширования. Кроме того, модели распределения популярности различаются из-за социальных, пространственных и временных вариаций.

Быстрые прогнозы. Популярность некоторых видео внезапно резко возрастает вскоре после публикации, поэтому результаты прогнозов необходимо подготовить сразу после загрузки, чтобы удовлетворить большинство пользователей. Однако результаты могут быть неточными из-за компромисса между качеством прогнозирования и эффективностью разрешения. С одной стороны, быстрые прогнозы необходимы для максимального использования ресурсов. С другой стороны, более длительное ожидание дает более ценные данные, позволяя делать более точные прогнозы. Алгоритм прогнозирования

должен преодолеть эту проблему и обеспечить быстрый прогноз с разумной точностью, чтобы не пропустить огромное количество запросов на самые популярные видео только потому, что они не были кэшированы вовремя.

Основанный на популярности фреймворк. Доступная социальная и контекстная информация, такие как социальные отношения, интересы пользователей, спрос на контент, географическое положение пользователей и серверов, являются входными данными для системы.

Производительность модели машинного обучения может быть улучшена за счет использования социальных, временных и пространственных вариаций, а также может быть спроектирована так, чтобы предсказывать, где видео будет уделяться больше внимания при различных настройках детализации (например, локальном и глобальном).

Признаки для прогнозирования популярности видео. Различные характеристики (или *признаки*) помогают определить причины распространения видео, и в целом их можно разделить на четыре основные группы: статические, временные, междоменные и социальные.

Статические признаки относятся к признакам, которые обычно фиксированные и подготавливаются перед публикацией видео в Интернете. Они делятся на характеристики видео, визуальные характеристики и текстовые признаки.

К характеристикам видео можно отнести, например, продолжительность, качество, жанр и дату публикации. Визуальными характеристиками могут быть: доминирующий цвет в каждом кадре, количество лиц в кадре, количество текстовых фреймов в кадре, миниатюра видео. Текстовые признаки, как правило, выполняют описательную функцию: это могут быть заголовки, ключевые слова, возрастная категория видео.

Динамические признаки – это, как правило, численные показатели, отражающие вовлечённость пользователей. По этим характеристикам можно собрать статистику в течение произвольного промежутка времени. Среди них могут быть количество просмотров, лайков, дизлайков и комментариев.

Междоменные признаки — это переменные, полученные из внешних источников, которые можно использовать для прогнозирования будущего спроса на контент. Они являются одной из основных причин внезапного всплеска популярности.

Важно отметить, что видео могут проходить различные фазы, включая распространение, внезапный всплеск популярности и, наконец, потерю внимания пользователей. Первоначальное распространение в основном вызвано вирусным поведением из-за того, что подписчики просматривают контент. За этим следует устойчивый рост из-за эффекта миграции, когда другие зрители находят видео, выполняя поиск по рекламе или помещая его в рекомендуемый список. Позже внешние факторы могут обновить популярность в любое время, даже после исчезновения видео, к таким факторам относятся реальные действия и распространение контента через социальные сети. Однако получение междоменных признаков – очень ресурсоёмкая задача, требующая обращения к большому множеству сторонних ресурсов, что в рамках данной курсовой работы не представляется возможным.

Основанный на популярности алгоритм кэширования. В зависимости от используемых признаков, алгоритмы основанного на популярности кэширования деляется на два класса: однодоменные и междоменные. Под популярностью авторы статьи подразумевают количество просмотров, если не указано иное определение.

Однодоменные означает, что методы используют только признаки сайта обмена видео. Такие методы делятся на три категории:

- модели тенденций эволюции популярности;
- модели, основанные на метаданных;
- социально-динамические модели.

Высокая корреляция между прошлой и будущей популярностью видео указывает на то, что историческая информация может отражать грядущую тенденцию. Поэтому графики роста популярности, показывающие популярность видео в течение последовательных дней, можно использовать для про-

гнозирования популярности в будущем.

Историческая популярность старых видеороликов может предвещать их популярность в ближайшем будущем. Однако это не применимо к недавно загруженным видео, где может не быть записей о предыдущей популярности. Имея дело с недавними видеообъектами, следует учитывать метаданные видео и пользовательские данные, такие как время просмотры, репосты, подписчики и возраст видео.

Сравнение алгоритмов. Эффективная схема кэширования видео должна быть основана на популярности, что означает, что она должна определять будущую популярность видео и использовать ее при принятии соответствующих решений по кэшированию. В области кэширования видео принято множество популярных алгоритмов кэширования. Авторы статьи разделяют их на три подкласса: модели, основанные на 1) тенденциях развития популярности, 2) социальной динамике и 3) метаданных видео. Однако для анализа социальной динамики необходимо построение социального графа пользователей единиц контента, что, как и в случае с междоменными признаками, является достаточно ресурсоёмкой задачей.

1. **Тенденции эволюции популярности.** Тенденция эволюции популярности показывает, как видео «развивается» с течением времени, и выражается в исторических показателях популярности. В этой категории модели оценивают будущую популярность, фиксируя временные корреляции в последовательности значений популярности.
2. **Модели, основанные на метаданных.** Предыдущая категория зависит от значений популярности видео и не объясняет, почему видео достигает определенного значения популярности. Метаданные видео предоставляют много полезной информации, которая может показать, почему видео стало популярным, например: время просмотра, репосты, подписчики, количество просмотров и информация о канале. Таким образом, модели, основанные на метаданных, могут повысить точность прогнозирования, особенно для «молодых» видео, где информация из

графика эволюции популярности ограничена или даже не применима. Более того, метаданные видео хранятся поставщиками видео и могут быть легко доступны общественности. Они также требуют небольшого времени обработки, поскольку записываются и отслеживаются сайтом обмена видео. Это приводит к низкой сложности этапа предварительной обработки.

Результат. Машинное обучение и алгоритмы прогнозирования позволяют сети кэширования обнаруживать достаточно нетривиальные взаимосвязи между признаками, характеризующими видеообъекты. Однако необходимы дополнительные исследования, чтобы найти оптимальные способы кэширования с методами больших данных и сетевыми технологиями.

3.4 Сравнительный анализ упомянутых методов

Резюмируем подходы к прогнозированию популярности контента, рассмотренные в статьях. Ниже представлена сводная таблица, отражающая, насколько упомянутые публикации соответствуют установленным ранее критериям.

Таблица 3.1: Сводная таблица

	Таргет	**Моделей**	**Данные**	**Признаки**	**Стратетегия**
Онлайновое машинное обучение: прогнозирование на основе экспертов в сетях доставки контента	кол-во запросов к видео в $t+1$ день	2	YouTube CDN	кол-во запросов к видео в течение t дней	требуются дополнительные исследования
Применение машинного обучения в мобильных социальных сетях	кол-во запросов к видео	4	WeChat	дата, день недели, выходной день, время и количество просмотров страниц за последние U часов.	–
Методы кэширования видео на основе их популярности	бинарная классификация: (не)популярный контент	>20	–	социальные, междоменные, статические, динамические	традиционная схема; метод на основе популярности

Во всех статьях при обучении так или иначе рассматриваются несколько моделей. Кроме того, третья статья содержит исчерпывающее руководство о том, какие признаки видео особенно влияют на рост популярности видео. Также в первой и третьей статьях даётся описание используемых стратегий

кэширования, что позволяет понять, каким образом можно ранжировать видео по прогнозируемой популярности.

Таким образом, можно сделать вывод, на какие факторы стоит обратить внимание при построении метода прогнозирования видео в сетях доставки контента. Однако необходимо учитывать технические ограничения при реализации проекта курсовой работы, в частности объём данных и возможные стратегии их сбора, ограниченные квотой YouTube Data API [4].

4 Разработка метода прогнозирования и размещения объектов в кэше

Каждый алгоритм машинного обучения обладает своими преимуществами и недостатками, которые можно использовать в качестве руководства при решении конкретной задачи. Хотя один алгоритм не всегда лучше другого, у каждого из них есть набор некоторых свойств, которые при правильной настройке дают более предпочтительные результаты: параметры и гиперпараметры. Исходя из постановки задачи курсовой работы, предсказание популярности видео – это задача регрессии. Рассмотрим несколько подходов, известных в практике машинного обучения для регрессионного анализа, которые можно было бы применить для достижения поставленной цели.

4.1 Линейная регрессия

Это метод, используемый для моделирования взаимосвязи между отдельной входной независимой переменной (признаком) и выходной зависимой (целевой) переменной с использованием линейной модели, то есть прямой. Более общий случай – это линейная регрессия с несколькими независимыми входными переменными и выходной зависимой переменной. Модель является линейной в том смысле, что выход представляет собой линейную комбинацию входных переменных.

$$a(x) = w_0 + \sum_{j=1}^{d} w_j x_j.$$

Преимущества:

- Линейная регрессия быстро обучается и особенно полезна, когда моделируемые отношения не являются чрезвычайно сложными и если имеется мало данных.
- Линейная регрессия просто интерпретировать.

Недостатки:

- Линейные модели не так хороши, когда дело касается очень сложных данных.

Для построения метода прогнозирования будет использоваться линейной регрессия с L_2-регуляризацией, известная как Ridge-регрессия. [7]

В целом для предсказания популярности видео можно использовать два, на первый взгляд, похожих, но существенно различающихся в деталях реализации метода.

Первый из них предполагает, что для каждой из установленных целевых переменных (динамические статистики количества просмотров, лайков, дизлайков и комментариев) видеообъекта будет построена собственная модель, а показание популярности будет формироваться с помощью объединения результатов этих моделей.

Второй подход подразумевает одной «усреднённой модели», у которой есть риск пере- или недообучиться. В ходе экспериментов необходимо для каждой динамической статистике выяснить, какие конкретно признаки в большей степени взаимосвязаны с целевой переменной.

Таким образом, если при построении предсказательных алгоритмов для, например, количества количества просмотров и лайков, взять одинаковые признаки, качество одной из моделей может пострадать.

4.2 Градиентный бустинг

Градиентный бустинг – это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений.

Ансамбль – это набор предсказательных моделей, которые вместе дают ответ (например, средний по всем). Главной причиной, по которой используются ансамбли, является то, что вместе несколько алгоритмов могут дать более точный результат.

Алгоритм градиентного бустинга предполагает пошаговое применение

большого множества простых моделей с целью корректировки результата и улучшения предсказания.

В рамках данной курсовой работы будет использована библиотека CatBoost [8], созданная Яндексом. Она использует «небрежные» (oblivious) [9] деревья решений, чтобы вырастить сбалансированное дерево. По сравнению с классическими деревьями, небрежные деревья более эффективны при реализации на процессоре и просты в обучении. [10]

Рассмотрим общие параметры в CatBoost:

- `loss_function` – показатель, используемый для обучения;
- `eval_metric` – метрика, используемая для обнаружения переобучения.
- `iterations` – максимальное количество построенных деревьев.
- `random_seed` – случайное зерно, используемое для обучения;
- `l2_leaf_reg` – коэффициент при члене регуляризации L_2 функции потерь;
- `task_type` – использование CPU или GPU;
- `boosting_type` – схема бустинга;
- `depth` – глубина дерева;
- `grow_policy` – определяет, как будет применяться жадный алгоритм поиска.

Аналогичной модели линейной регрессии, градиентый бустинг будет использоваться для предсказания динамических статистик в отдельности.

4.3 Размещение контента в кэше

Сетям доставки контента необходимо сохранять в кэше наиболее актуальные (или востребованные видео). Данное понятие синонимично «популярности», которая, в свою очередь, формируется на основе показателей просмотров, лайков, дизлайков и комментариев. Предполагается, что данные статистики оказывают наибольшее влияние на то, чтобы видео появилось (или оставалось) в трендах YouTube [5]. Чтобы «усреднить» данные показатели,

преобразовав их в одну величину, возьмём взвешенное среднее гармоническое. При этом показатель, не являющийся валидным, то есть равный нулю или отсутствующий, не учитывается в общей формуле. Веса отражают важность (или приоритет) статистического показателя.

- $w_v = 4$ – для количества просмотров;
- $w_l = 3$ – для количества лайков;
- $w_d = 2$ – для количества дизлайков;
- $w_c = 1$ – для количества комментариев.

Таким образом, введём показатель популярности видео:

$$\rho(video\_id) = \frac{w_v + w_l + w_d + w_c}{\frac{w_v}{v} + \frac{w_l}{l} + \frac{w_d}{d} + \frac{w_c}{c}},$$

где v, l, d, c – количественные показатели просмотров, лайков, дизлайков и комментариев видеообъекта $video\_id$.

Кроме того, нужно учесть размер видеообъекта. При кэшировании можно слегка пренебречь показателем популярности в пользу объёма файла. Это следует из того, что более легкие и популярные легче загрузить заново, нежели проделывать эту процедуру с крупными видеофайлами. Таким образом снизится нагрузка на кэш-сервер.

Предположим, что размер видеофайла прямопропорционален длине видео, тогда величину объёма можно отождествить с длительностью видеообъекта. Разобъём все имеющиеся в кэш-сервере видео на 4 категории по длительности и присвоим каждой из них вес $u_i \in \{1, 2, 3, 4\}$: наибольший будет у наиболее длительных видео, наименьший – у категории коротких видео.

$$\delta(video\_id) = u_i \cdot n,$$

где n – длительность в секундах.

Сформулируем величину, по которой будут ранжироваться видео в кэш-сервере. Она будет представлять взвешенную сумму длительности и популяр-

ности.

$$\mu(video\_id) = 0.6 \cdot \delta(video\_id) + 0.4 \cdot \rho(video\_id).$$

Результат переводится в шкалу от 0 до 1, и полученный показатель используется для сортировки видеообъектов по убыванию. Первые k видео в отсортированной таблице остаются в кэше, а остальные – выгружаются.

5 Разработка программного средства прогнозирования популярности видео

Разработанное в рамках курсовой работы программное средство состоит из двух основных модулей: модуль сбора данных и видеообъектах (Scrapper) и модуль прогнозирования популярности видео (Predictor).

5.1 Разработка модуля сбора и обработки данных

Scrapper – это приложение, реализованное на языке программирования Python и предназначенное для сбора и обработки информации о видеообъектах, размещённых в трендах одного из самых популярных сервисов видеохостинга YouTube. В текущей версии приложение поддерживает 107 регионов.

В процессе разработки было рассмотрено несколько возможных стратегий сбора и обработки данных. Основным критерием для каждой из них было извлечение информации о как можно большем числе видеообъектов. Однако существует ограничение на количество запросов к YouTube API, равное 10000 обращений в сутки, поэтому при создании программного средства приходилось учитывать этот фактор.

Для доступа к API и последующего извлечения данных с помощью языка программирования требуется получить API_KEY для сервиса YouTube API Data v3 в Google Cloud Platfrom. Доступ по одному ключу и связан с суточной квотой, которую для корректной работы Scrapper нельзя превышать.

Первая из стратегий сбора и обработки данных заключалась в том, чтобы в каждый из n дней собрать как можно больше новых данных. Для этого посылались запросы к странице с трендами всех доступных в YouTube API Data v3 107 стран. Из них извлекались все доступные наиболее популярные видеообъекты, среди которых оставлялись только уникальные по всем регионам. Такой подход практически сразу стал считаться неприемлемым, потому что требует слишком много запросов. Однако были предприняты попытки со-

кратить количество регионов, чтобы влезть в данную видеохостингом квоту, но и они оказались безуспешными.

Вторая стратегия предполагала достаточно нетривиальный подход. Сначала были выбраны наиболее крупные – с географической и культурной точек зрения – страны. Количество постепенно уменьшалось и последовательно достигало 50, 30, 25, 14, 10 стран. В каждом регионе брались топ-50 видео каждый день, а затем выбрасывались повторы по всем регионам среди них. Допустим, если в парсинге участвовали 14 стран, то на выходе получалось не более $14 \cdot 50$ (исключая повторы) видеозаписей. И так день за днём, пока по каждому видеобъекту не будет собрана статистика по нужному количеству дней. Главной идеей данной стратегии было постоянное добавление новых видеобъектов до определённого момента. Скажем, если требовалось собрать показатели за n дней, то работа алгоритма занимала $(2n - 1)$ дней: первые n дней пополнялось множество исследуемых видео и статистикам по ним, а в оставшиеся $(n - 1)$ дней лишь дополнялась статистика по тем видео, которые не имеют n соответствующих показателей. В итоге, даже при самом скромной числе стран (а именно 10), не хватало суточной квоты по запросам, поэтому данная стратегия оказалась невостребованной.

Третья – финальная – стратегия учитывает недостатки предшествующих и предполагает следующий алгоритм. При первом запуске формируется множество видеообъектов, по которым в ближайшие n дней будет собрана статистика: для этого парсер отбирает видео из трендов 107 доступных регионов, и в итоге получается около 8200 уникальных видеообъектов. Затем день за днём по каждому отправляется 3 запроса к API:

- для извлечения характеристик видео;
- для извлечения характеристик канала данного видео;
- для извлечения характеристик наиболее популярного комментария данного видео.

Таким образом, требуется более 24000 запросов, что значительно превышает суточную квоту одного сервиса YouTube API Data v3. Чтобы решить

данную проблему, множество видеообъектов разбивается на несколько частей: в текущей реализации их 5. Сбор и обработка данных для каждой из частей множество происходит отдельными сервисами, следовательно общая нагрузка распределяется, что позволяет отдельно взятому сервису не превышать квоту.

Признаковое описание видеообъекта. Каждый видеообъект описывается статическими и динамическими признаками. Статические признаки – это различного вида описания и метаданные, такие как:

- идентификатор видео;
- название видео;
- качество (HD или SD);
- продолжительность (в секундах);
- дата публикации видео;
- метка: является контент лицензионным;
- есть ли субтитры;
- категория видео: примеры – образовательная, развлекательная, влоги;
- контент для детей или нет;
- статус приватности: публичное или скрытое;
- дата создания канала.

Эти данные собираются один раз и не обновляются в дальнейшем. Динамические признаки представлены следующими показателями:

- число лайков под видео;
- число дизлайков под видео;
- число комментариев под видео;
- число подписчиков канала;
- число видео на канале;
- общее число просмтров на канале;
- число лайков под самым популярным комментарием;
- число ответов под самым популярным комментарием.

Данные характеристики меняются в течение времени, и Scrapper позволяет собрать статистику по ним в течение нескольких дней. На основе некоторых подмножеств их множества перечисленных показателей будет происходить ранжирование для видео.

Полученный список «сырых» данных фильтруется – среди всех видеобъектов остаются только уникальные. Это необходимая мера, так как существуют непустые пересечения между множествами трендов в разных странах, доступных в YouTube.

Структура Scrapper. Модуль сбора и обработки данных состоит из 3-х подмодулей:

- **Parser** получает данные с сервера благодаря API.
- **Reader** обеспечивает чтение из файлов, содержащих записанные данные предыдущих дней.
- **Handler** обрабатывает новые данные, сохраняя статистику с учётом записей по предыдущим дням.

Задачей модуля Parser является разделение множества видео на несколько составляющих (в частности, на 5), а также дальнейшая отправка соответствующих запросов к серверу. Его входными параметрами являются набор идентификаторов множества видеообъектов и количество число блоков разбиения. Полученные данные буду использоваться в качестве входных параметров в Handler.

Reader считывает информацию из файлов, поддерживающих данные уже обработанных объектов: это множество видео, список стран и собранная по дням статистикам по динамическим показателям. Среди них:

- country-codes.txt
- video-set.txt
- description.tsv
- views.tsv
- likes.tsv
- dislikes.tsv

- comments.tsv
- channel-subscribers.tsv
- channel-video.tsv
- channel-views.tsv
- tc-likes.tsv
- tc-replies.tsv
- tc-published.tsv

Основная работа приложения связана с модулем Handler. В качестве входных данных он получает «сырые» данные от Parser и записи предыдущих дней от Reader. Сопоставляя полученную информацию, Handler обновляет статистику по каждому динамическому признаку, добавляя столбец с показателями очередного дня.

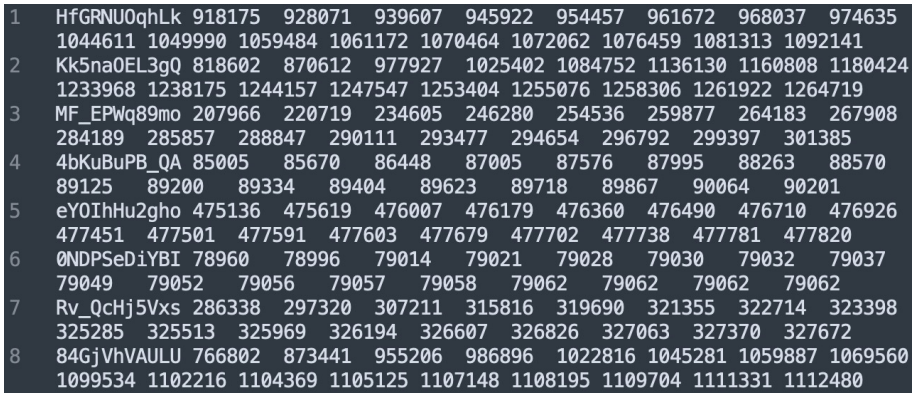

Рис. 5.1: views.tsv

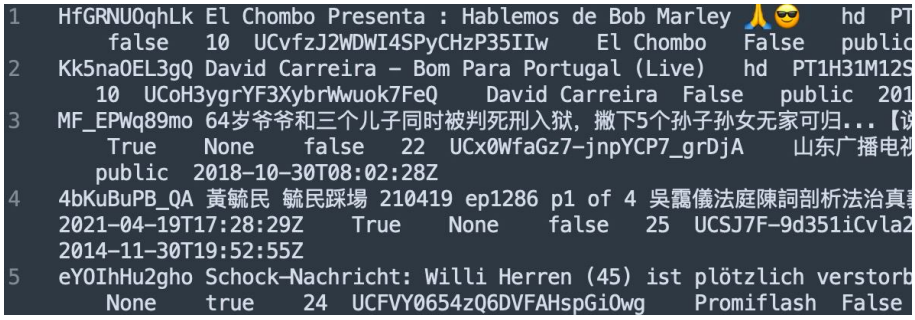

Рис. 5.2: descriptions.tsv

```
1   HfGRNUOqhLk
2   Kk5naOEL3gQ
3   MF_EPWq89mo
4   4bKuBuPB_QA
5   eYOIhHu2gho
6   0NDPSeDiYBI
7   Rv_QcHj5Vxs
8   84GjVhVAULU
9   YxvjzpH9vPs
10  4m8xhQ75gvM
11  y6Pxg27GX9Y
12  X84S0j4GPyU
13  VXYMOITklKQ
14  WRyCD2AyqP4
15  CzwpjioJIUo
16  y2_2IxRCxvM
17  koOuhXmdBdI
18  U487ngbPKKk
19  -h0hGTDzdKs
20  zST4EESh2aE
```

Рис. 5.3: video-set.txt

Рис. 5.4: Взаимодействие подмодулей

5.2 Подготовка данных для обучения

Прежде чем приступать к обучению моделей машинного и глубинного обучения, необходимо преодолеть два промежуточных этапа: объединение и предобработка данных.

Статистики по динамическим признакам на этапе сбора данных сохранялись в отдельные tsv-таблицы в виде: идентификатор - количественный показатель. Данный подход обоснован тем, что распределённое хранение позволяет работать с конкретным признаком независимо от остальных, и это облегчает процесс отладки: некорректное построение таблицы по одному признаку не касается всех остальных. Однако для обучения требуется общая таблица с данными по всем видеобъектам, поэтому прежде всего требуется «слить» все таблицы в единый датасет средствами библиотеки Pandas. Получившаяся таблица подвергается предобработке.

Предобработка – это процесс подготовки данных для обучения. Во время предобработки данных из общей таблицы удаляются объекты с пропусками в признаках, применяется one-hot кодирование для логических переменных, текстовые признаки преобразуются в количественные показатели, а также удаляются столбцы, которые не коррелируют с целевыми переменными.

Рассмотрим более подробно, каким модификациям подвергаются собранные парсером данные. Сначала необходимо избавиться от строк, содержащих неполные статистики или ячейки NaN, которые никак не заполнить. Затем проверить подозрительные признаки: допустим, есть такие, у которых одно значение встречается слишком часто – вряд ли этот признак сильно коррелирует с целевой переменной. Далее перейти к кодированию потенциально полезных признаков.

- **Заголовок:** 3 признака → количество слов, капс или нет, есть ли уточнения в названии (в скобках), наличие индикатора привлечения внимание или даже количество («премьера», год, official).

- **Качество:** hd или sd.
- **Длительность:** в секундах.
- **Теги:** количество.
- **Дата публикации:** какой день недели → one-hot.
- **Субтитры:** есть или нет.
- **Категория видео:** например, Спорт, Комедия или Образование (всего 27) → one-hot.
- **Название канала:** количество слов, капс или нет.
- **Контент для детей:** да или нет.
- **Статус приватности:** один из 3 статусов (public, private, unlisted) → one-hot.
- **Дата создания канала:** сколько дней существует.
- **Дата публикации топ-комментария:** количество дней с даты публикации.
- **Предсказание:** последнего дня по предыдущим.
- **Таргет:** просмотры, лайки, дизлайки, комментарии.

Полученный в результате датасет считается готовым и используется для обучения моделей машинного и глубинного обучения.

5.3 Разработка модуля прогнозирования

Predictor состоит из набора блокнотов формата Jupyter Notebook, подготовленных данных и скриптов на языке Python.

Блокноты содержат построенные модели прогнозирования, а также экспериментальные исследования, в ходе которых были подобраны оптимальные параметры предсказательных моделей.

Подготовленные данные представляют из себя набор таблиц в формате `.tsv`, содержащих признаковое описание и статистические показатели за 7 и 22 днях для всех видеообъектов.

Кроме того, к данному модулю относится скрипт, объединяющий табли-

цы, сфомированные модулем Handler, а также скрипт для (де)сериализации обученной модели и ранжирования видео согласно разработанной стратегии кэширования.

- `growth_df_btw_ids21.tsv` – датасет со статистикой за 22 дня;
- `growth_df_btw_ids6.tsv` – датасет со статистикой за 7 дней;
- `linreg-clean.ipynb` – построение модели линейной регрессии.
- `catboost-clean.ipynb` – построение модели градиентного бустинга;
- `merger.py` – скрипт, объединяющий статистические таблицы;
- `estimator.py` – скрипт, определяющий, какие видео останутся в кэше;
- `archive/` – архив с сопутствующими разработке файлами.

В ходе разработки и исследования были реализованы три модели градиентного бустинга. Модели различаются между собой установленными гиперпараметрами, при этом каждая из них обучена в отдельности предсказывать число просмотров, лайков, дизлайков и комментариев.

Линейная регрессия, рассмотренная в качестве предсказательной модели в первую очередь, предусматривает 2 стратегии предсказания: просмотры, лайки, дизлайки и комментарии как в отдельности, так и вместе. Во втором случае используется MultiOutputRegressor – класс из библиотеки Scikit-learn [11], позволяющий предсказывать сразу несколько целевых переменных.

6 Методика экспериментального исследования

Целью экспериментального исследования является выявление оптимальных параметров для моделей машинного обучения и выделение характеристик видеообъектов, наиболее значимых для целевой переменной. Под «оптимальностью» подразумеваются такие гиперпараметры, которые позволяют построить наиболее эффективную с точки зрения метрики качества модель. Однако такой подход допускает существование моделей с более приемлемыми результатами.

Пусть порог $\varepsilon = 0.9$, тогда требуется построить модель, для которой на тестовой выборке коэффициент детерминации R^2 как минимум равен 0.9.

В соответствие с информационно-аналитический ресурсом [6], посвященному машинному обучению, распознаванию образов и интеллектуальному анализу данных, «Для приемлемых моделей предполагается, что коэффициент детерминации должен быть хотя бы не меньше 50% (в этом случае коэффициент множественной корреляции превышает по модулю 70%). Модели с коэффициентом детерминации выше 80% можно признать достаточно хорошими (коэффициент корреляции превышает 90%). Равенство коэффициента детерминации единице означает, что объясняемая переменная в точности описывается рассматриваемой моделью».

Если разработанный алгоритм машинного обучения не достигает установленного порога ε, требуется сделать заключение о построенной модели и обосновать полученный результат.

Для достижения поставленной задачи должен быть проведён ряд экспериментов, варьирующих значения гиперпараметров моделей, а также обнаруживающих признаки, наиболее коррелирующих с целевой переменной. Кроме того, необходимо провести сравнительный анализ между обучением на статистических показателях в разные промежутки времени: это позволит выяснить, как меняется результат предсказания в зависимости от рассматриваемого периода.

7 Экспериментальное исследование метода

7.1 Корреляция признаков

Чтобы улучшить качество обучаемых моделей, в первую очередь были выявлены признаки, наиболее коррелированные с прогнозируемой величиной. В ходе экспериментального исследования эмпирическим путём были получены пороги для коэффициента корреляции Пирсона, по которым можно определить подмножество наиболее важных признаков.

Таблица 7.1: Прогнозирование показателей в 7-ой день

Статистика	**Порог**
Просмотры	0.5
Лайки	0.4
Дизлайки	0.4
Комментарии	0.25
Мульти-таргет	0.2

Таблица 7.2: Прогнозирование показателей в 22-ой день

Статистика	**Порог**
Просмотры	0.58
Лайки	0.25
Дизлайки	0.15
Комментарии	0.1
Мульти-таргет	0.2

```
array(['views_1', 'views_13', 'views_14', 'views_12', 'views_7',
       'views_9', 'views_11', 'views_8', 'views_17', 'views_10',
       'views_15', 'views_18', 'views_6', 'likes_1', 'views_20',
       'views_19', 'likes_9', 'views_16', 'likes_10', 'likes_18',
       'likes_11', 'views_5', 'likes_7', 'likes_12', 'likes_13',
       'likes_8', 'likes_17', 'views_21', 'likes_19', 'likes_6',
       'likes_20', 'dislikes_1', 'likes_21', 'likes_14', 'dislikes_12',
       'dislikes_6', 'dislikes_7', 'dislikes_10', 'likes_16',
       'dislikes_11', 'dislikes_13', 'dislikes_14', 'dislikes_8',
       'dislikes_5', 'likes_15', 'dislikes_9', 'views_4', 'dislikes_21',
       'likes_5', 'dislikes_20', 'dislikes_4', 'views_3', 'likes_4',
       'dislikes_15', 'views_2', 'dislikes_18', 'dislikes_19',
       'dislikes_3', 'dislikes_17'], dtype=object)
```

Рис. 7.1: Признаки для прогнозирования просмотров – 22

```
array(['likes_1', 'views_1', 'likes_7', 'likes_8', 'likes_9', 'likes_21',
       'likes_20', 'likes_12', 'likes_18', 'likes_6', 'likes_13',
       'dislikes_1', 'likes_10', 'likes_11', 'likes_19', 'likes_17',
       'likes_14', 'likes_16', 'likes_15', 'likes_5', 'views_12',
       'views_14', 'views_13', 'views_7', 'views_15', 'dislikes_12',
       'views_9', 'views_11', 'views_8', 'dislikes_14', 'dislikes_13',
       'dislikes_6', 'dislikes_7', 'views_17', 'dislikes_10', 'views_10',
       'dislikes_11', 'likes_4', 'views_6', 'views_18', 'dislikes_8',
       'views_19', 'views_5', 'dislikes_5', 'views_20', 'comments_11',
       'views_16', 'dislikes_9', 'dislikes_21', 'comments_1',
       'dislikes_20', 'views_21', 'top_comment_likes_1', 'dislikes_4',
       'dislikes_15', 'comments_13', 'views_4', 'likes_3', 'comments_10',
       'dislikes_3', 'comments_12', 'dislikes_19', 'dislikes_18',
       'dislikes_17', 'comments_9', 'views_3', 'views_2', 'comments_17',
       'likes_2', 'comments_16', 'dislikes_16', 'dislikes_2',
       'comments_8', 'comments_19', 'comments_20', 'comments_3',
       'comments_18', 'comments_5', 'comments_6', 'comments_15',
       'channel_views_21', 'comments_2', 'comments_21',
       'channel_subscribers_18', 'channel_views_11', 'channel_views_2',
       'channel_subscribers_7', 'channel_subscribers_16',
       'channel_subscribers_20', 'top_comment_replies_1',
       'channel_views_17', 'channel_subscribers_3', 'comments_7',
       'channel_views_3', 'channel_subscribers_11', 'comments_4',
       'channel_subscribers_21', 'channel_views_5', 'channel_views_13',
       'channel_views_6', 'channel_subscribers_13',
       'channel_subscribers_1', 'channel_views_12', 'channel_views_7',
       'channel_views_16', 'channel_subscribers_5', 'channel_views_4',
       'channel_views_8', 'channel_views_14'], dtype=object)
```

Рис. 7.2: Признаки для прогнозирования лайков – 22

```
array(['dislikes_1', 'dislikes_8', 'dislikes_7', 'views_1', 'dislikes_12',
       'dislikes_10', 'dislikes_6', 'dislikes_21', 'dislikes_14',
       'dislikes_9', 'dislikes_11', 'dislikes_20', 'dislikes_13',
       'dislikes_19', 'dislikes_17', 'dislikes_5', 'dislikes_18',
       'dislikes_15', 'dislikes_4', 'likes_11', 'likes_12', 'likes_1',
       'dislikes_16', 'likes_10', 'likes_13', 'likes_9', 'dislikes_3',
       'likes_17', 'likes_14', 'likes_18', 'likes_7', 'likes_8',
       'likes_19', 'likes_16', 'likes_6', 'views_14', 'likes_15',
       'views_12', 'views_11', 'likes_20', 'views_9', 'views_13',
       'likes_21', 'views_10', 'views_17', 'views_15', 'dislikes_2',
       'views_7', 'views_8', 'likes_5', 'views_19', 'views_18', 'views_6',
       'views_16', 'views_5', 'likes_4', 'views_20', 'views_21',
       'views_4', 'comments_13', 'comments_11', 'top_comment_likes_1',
       'comments_12', 'likes_3', 'comments_10', 'views_3', 'views_2',
       'comments_16', 'comments_9', 'comments_17', 'comments_18',
       'comments_15', 'likes_2', 'comments_8', 'comments_20',
       'comments_1', 'comments_14', 'channel_views_21', 'comments_6',
       'channel_views_11', 'comments_19', 'channel_views_3',
       'channel_views_2', 'channel_views_17', 'channel_views_13',
       'comments_5', 'comments_21', 'channel_views_6', 'channel_views_12',
       'channel_views_5', 'channel_views_14', 'channel_views_16',
       'channel_views_7', 'comments_3', 'channel_subscribers_3',
       'channel_views_8', 'channel_subscribers_18', 'channel_views_4',
       'channel_subscribers_16', 'channel_subscribers_20',
       'channel_subscribers_13', 'comments_4', 'top_comment_replies_1',
       'channel_views_18', 'channel_subscribers_7', 'comments_2',
       'comments_7', 'channel_subscribers_19', 'channel_subscribers_21'],
      dtype=object)
```

Рис. 7.3: Признаки для прогнозирования дизлайков – 22

```
array(['comments_1', 'likes_1', 'comments_11', 'comments_3', 'comments_2',
       'top_comment_likes_1', 'comments_17', 'comments_9', 'comments_13',
       'comments_12', 'comments_10', 'comments_5', 'comments_19',
       'comments_16', 'views_1', 'comments_20', 'comments_18',
       'dislikes_1', 'comments_4', 'likes_21', 'comments_8',
       'channel_subscribers_21', 'likes_20', 'comments_6', 'comments_7',
       'top_comment_likes_13', 'comments_15', 'top_comment_likes_16',
       'likes_19', 'comments_21', 'channel_subscribers_7', 'views_21',
       'views_20', 'top_comment_replies_1', 'views_19', 'likes_17',
       'views_7', 'views_8', 'views_16', 'views_17', 'likes_8',
       'likes_16', 'channel_subscribers_14', 'likes_18', 'views_6',
       'likes_7', 'channel_subscribers_11', 'channel_subscribers_1',
       'channel_subscribers_18', 'views_15', 'views_18', 'likes_6',
       'channel_subscribers_20', 'views_5', 'likes_9',
       'top_comment_likes_7', 'views_9', 'views_13',
       'channel_subscribers_4', 'views_4', 'likes_5', 'dislikes_21',
       'likes_15', 'likes_4', 'views_12', 'views_10', 'views_2',
       'views_3', 'dislikes_19', 'likes_13', 'views_14', 'likes_10',
       'likes_12', 'dislikes_20', 'likes_14', 'views_11', 'dislikes_8',
       'likes_3', 'dislikes_17', 'likes_11', 'dislikes_7', 'likes_2',
       'top_comment_likes_11', 'dislikes_6', 'channel_views_21',
       'dislikes_3', 'dislikes_14', 'dislikes_15', 'top_comment_likes_3',
       'dislikes_5', 'dislikes_18', 'dislikes_16', 'dislikes_4',
       'dislikes_9', 'channel_views_6', 'top_comment_likes_14',
       'dislikes_12', 'dislikes_10', 'dislikes_13',
       'channel_subscribers_16', 'channel_views_2', 'dislikes_2',
       'top_comment_likes_17', 'channel_subscribers_3', 'dislikes_11',
       'channel_views_5', 'top_comment_likes_6', 'channel_views_3',
       'channel_views_11', 'channel_views_1', 'channel_views_13',
       'channel_views_10', 'channel_subscribers_13'], dtype=object)
```

Рис. 7.4: Признаки для прогнозирования комментариев – 22

7.2 Оценка точности метода на основе линейной модели

При обучении моделей с использованием признаков, соответствующих указанным порогам, получились следующие показатели метрики качества (коэффициента детерминации):

Таблица 7.3: Коэффициент детерминации линейных моделей – 7

Статистика	R^2
Просмотры	0.9999622316368325
Лайки	0.9999878084487095
Дизлайки	0.9999362069677782
Комментарии	0.9999376079642144
Мульти-таргет	0.9999698817394376

Таблица 7.4: Коэффициент детерминации линейных моделей – 22

Статистика	R^2
Просмотры	0.9999971967290997
Лайки	0.999996770210786
Дизлайки	0.9999977519245685
Комментарии	0.9999978730198016
Мульти-таргет	0.9999976318741663

Исходя из полученных данных, можно сделать вывод: при прогнозировании показателей в 22-ой день получается более приемлемый результат, так как коэффициент детерминации ближе к 1. Приведём в качестве примера (вместе с параметрами) модели, которые справились с задачей лучше. Это 4 модели, которые прогнозируют количество просмотров, лайков, дизлайков и комментариев в 22-ой день на основе показателей соответствующих величин в предыдущие дни.

```
# views_21
Ridge(alpha=0.37926901907322497)
weights: [ 7.13712156e-01,  2.82810786e-02,  9.24822896e-03,
           1.68325779e-02,  1.68258215e-02,  1.86206565e-02,
           2.34337887e-02,  1.70979635e-02,  9.85501479e-03,
           1.92577880e-02,  1.58538512e-02,  2.21913036e-02,
           2.05285554e-02,  1.83982229e-04,  1.84586299e-02,
           7.11482647e-03,  1.23207755e-03,  2.36422284e-02,
          -3.75664208e-04, -1.07081206e-03, -9.50561187e-04,
```

2.06116307e−02, −1.10035210e−04, 2.55873027e−03,
−1.56309791e−03, −1.30861489e−03, −7.36605968e−04,
1.86052539e−02, 1.74279370e−03, 1.07698236e−03,
−2.54588885e−03, 3.83303974e−04, 2.90286674e−03,
−6.89295986e−04, −8.35057809e−04, −4.37555052e−04,
8.48874766e−04, 1.35255735e−03, −6.56400666e−05,
−2.36023684e−04, 1.38947526e−03, −4.97383482e−05,
1.13414993e−03, −1.41468796e−03, 5.48980113e−05,
−1.93029449e−03, 2.16876969e−02, −1.73806467e−03,
−4.17339672e−04, 8.62522975e−04, 5.48623955e−04,
3.77157908e−02, −1.15047502e−04, 1.13349305e−04,
3.24000631e−02, 1.41569627e−03, −4.89385022e−05,
3.55653342e−04, −1.72922738e−03]

likes_21

Ridge(alpha=0.37926901907322497)

weights: [8.48949478e−01, 6.60623974e−04, 1.17136272e−02,
9.84631086e−03, 9.71187575e−03, 1.29442778e−02,
6.16075073e−03, 1.37614714e−02, 9.80530205e−03,
1.34280942e−02, 1.38461731e−02, 6.27553319e−05,
1.15500130e−02, 1.09577370e−02, 3.94670138e−03,
4.12403679e−03, 5.94997733e−03, 1.31962351e−02,
1.03792427e−02, 1.30562686e−02, −1.03075933e−05,
−1.06943134e−03, 7.06610640e−04, −5.71337295e−04,
1.78893223e−03, −1.87218894e−03, −3.60701100e−04,
1.18277050e−04, 4.80565554e−04, −2.09937646e−04,
1.93344085e−03, 8.57728153e−04, −2.63915929e−04,
1.31118673e−03, −2.24905158e−04, −2.68433983e−05,
1.22515743e−03, 1.40334842e−02, −3.88450905e−04,
1.38441734e−03, −4.20513268e−04, −6.03099225e−04,

6.45132061e−04, −5.23265085e−04, 1.38791660e−04,
−1.33658956e−04, −2.26650824e−03, −1.67418449e−04,
−1.19846474e−03, 1.41311842e−04, 3.65826576e−04,
−1.25546021e−03, 1.72199416e−04, −4.36374287e−04,
6.58392404e−04, −1.49921222e−04, 1.07984875e−04,
2.59847213e−02, 2.14771601e−05, 2.99769958e−04,
5.60563272e−05, 1.06595021e−04, 5.81855705e−04,
−9.50775201e−04, 1.57701432e−05, 4.21706865e−05,
−3.55791932e−04, −1.11961485e−04, 2.97960198e−02,
−2.95487216e−04, 2.42411391e−04, 1.44222470e−04,
4.60409417e−04, 3.07076773e−04, 7.08474618e−05,
−3.05282568e−04, 5.87898072e−05, 3.61297979e−04,
−1.53598267e−04, −7.77751199e−05, 3.75530079e−05,
−4.51682733e−05, 2.54680239e−05, −4.03537016e−05,
6.44374923e−05, 1.38114943e−05, 4.83535821e−05,
4.66986120e−05, 1.93040331e−05, −1.53501062e−06,
−7.97310427e−05, 1.66938958e−05, −2.70740894e−04,
−2.56366577e−04, 4.62065104e−06, 6.40404751e−05,
−9.53522803e−06, 4.22105259e−05, 4.04896342e−05,
−8.03424662e−05, −1.44828702e−06, 1.54135385e−04,
9.71139382e−05, −3.39999733e−05, −2.32870662e−06,
2.73883872e−05, −5.26540910e−05, −2.20152292e−06,
−8.72397602e−06]

# dislikes_21
Ridge(alpha=0.01)
weights: [7.52977329e−01, 1.55786110e−02, 1.56174886e−02,
−1.22685461e−04, 1.26193959e−02, 1.84016922e−02,
1.74800978e−02, 1.35975300e−02, 6.56107755e−03,
1.35675136e−02, 1.96081230e−02, 1.44260065e−02,

2.97016441e−02, 8.29682999e−03, 6.70563071e−03,
1.69000532e−02, 2.58462733e−02, 1.64817908e−02,
1.71474681e−02, −4.68006981e−04, 3.01188816e−03,
3.42344156e−04, 2.40393527e−02, 8.11855088e−04,
−3.32835883e−03, 3.27960556e−04, 2.60803019e−02,
−9.11857451e−04, −4.21783529e−04, −2.63760952e−03,
4.76587402e−04, −1.48713701e−03, 1.56721424e−03,
1.46786007e−03, 1.21304400e−03, −1.53653176e−03,
9.58823181e−04, −4.23127715e−04, −8.41639988e−06,
−1.51487557e−03, 6.22224536e−04, 1.73000433e−03,
1.74755225e−03, −4.88921562e−04, 2.57342368e−03,
1.46963162e−03, 3.26417535e−02, −1.47635041e−03,
9.22290204e−04, −1.09247369e−03, −1.20313726e−03,
−1.58596346e−04, −2.09181434e−04, −3.54725012e−04,
−1.92233768e−04, −5.92937319e−04, −3.62232118e−04,
−1.22429396e−03, 5.65903793e−04, −1.86629707e−04,
−4.18906972e−05, −1.23085133e−04, 6.08397919e−05,
7.35615756e−04, −5.32786063e−05, 2.38407112e−04,
−3.90207746e−04, −1.01379319e−04, 6.14660864e−05,
−1.97035707e−04, −8.07225365e−06, −1.55582195e−04,
2.54663081e−05, 3.81749420e−04, 3.36553617e−04,
−3.89470224e−05, −6.22271981e−05, −7.08888742e−06,
−1.51373662e−04, −8.15616840e−06, 1.52666691e−05,
−2.90801817e−04, 3.56532885e−05, −2.21848673e−04,
2.15157519e−04, 2.96025791e−04, 4.49131999e−05,
−7.40045163e−05, 1.79209779e−05, 2.32848662e−05,
−1.83961071e−05, 1.99781587e−04, −8.94833947e−06,
−3.04754761e−04, 8.71573859e−05, 5.10910372e−05,
−1.58175244e−05, −1.23472464e−04, −3.20725227e−05,
−1.90918435e−05, 1.04777290e−05, 1.10207876e−04,

```
        -9.43032871e-06, -6.74709734e-05,  2.36534250e-05,
         5.31915494e-05, -2.37899839e-04, -5.34814715e-05,
        -2.25151345e-05]

# comments_21
Ridge(alpha=0.01)
weights: [ 9.65681497e-01,  5.42487378e-05,  2.04959206e-03,
         9.57428544e-03,  1.39517218e-02, -6.41610052e-05,
         1.38173552e-03,  2.97060241e-03,  3.13329683e-03,
         2.37199530e-03,  2.63475921e-03,  5.77453727e-03,
         1.49138941e-03,  3.35271834e-03,  1.91810103e-04,
         3.95825533e-03,  5.27002060e-03, -3.37043808e-04,
         7.26049008e-03,  7.29599506e-04,  3.12479332e-03,
         3.08630992e-05, -4.69083347e-04,  3.88100344e-03,
         4.27823672e-03,  1.55761428e-05,  4.04982258e-03,
        -8.12533923e-06, -3.13424685e-04,  2.92517655e-03,
        -1.91002409e-05,  1.05928162e-04,  1.27750614e-03,
         2.09565204e-05]
```

7.3 Оценка точности метода на основе градиентного бустинга

Также были реализованы 3 модели градиентного бустинга.

```
# CatBoost-1
CatBoostRegressor(
    iterations=8000,
    learning_rate=0.02,
    depth=10,
    random_seed=42,
    task_type='GPU',
```

```
    bagging_temperature=0.2,
    od_type='Iter',
    metric_period=400,
    od_wait=100,
)

# CatBoost-2
CatBoostRegressor(
    task_type='GPU',
    random_seed=42,
    verbose=200)

# CatBoost-3
    cb_learn_rate = 0.001
    n_iterations = 80000
    early_stop_rounds = 2000

    opt_catboost_params = {
        'iterations' : n_iterations,
        'learning_rate' : cb_learn_rate,
        'depth': 8,
        'bootstrap_type' : 'Bernoulli',
        'random_strength': 1,
        'min_data_in_leaf': 10,
        'l2_leaf_reg': 3,
        'loss_function' : 'RMSE',
        'eval_metric' : 'RMSE',
        'grow_policy' : 'Depthwise',
        'max_bin' : 1024,
        'model_size_reg' : 0,
        'task_type' : 'GPU',
```

```
    'od_type' : 'IncToDec',
    'od_wait' : 100,
    'metric_period' : 500,
    'verbose' : 500,
    'subsample' : 0.8,
    'od_pval' : 1e-10,
    'max_ctr_complexity' : 8,
    'has_time': False,
    'simple_ctr' : 'FeatureFreq',
    'combinations_ctr': 'FeatureFreq',
    'random_seed' : 13}
CatBoostRegressor(**opt_catboost_params)
```

Таблица 7.5: Коэффициент детерминации CatBoost-1

Статистика	R^2
Просмотры	0.81325878
Лайки	0.77273126
Дизлайки	0.81136465
Комментарии	0.70385731

Таблица 7.6: Коэффициент детерминации CatBoost-2

Статистика	R^2
Просмотры	0.88377347
Лайки	0.79123153
Дизлайки	0.73913935
Комментарии	0.73462535

Таблица 7.7: Коэффициент детерминации CatBoost-3

Статистика	R^2
Просмотры	0.96081155
Лайки	0.95395155
Дизлайки	0.81882261
Комментарии	0.90889016

7.4 Выводы

Исходя из полученных результатов, можно сделать вывод: линейная модель справилась лучше как при прогнозировании на 7-ой день, так и на 22-ой день. Об этом свидетельствуют значения коэффициента детерминации для каждой из применённых моделей: чем ближе показатель к 1, тем качественнее обученный алгоритм. Из этого следует, что между признаковым описанием видеобъектов и целевыми переменными присутствует линейная зависимость.

Стоит обратить внимание на то, что при прогнозировании показателей в 22-ой день были изменены пороги коэффициента корреляции Пирсона. Вместе с большим количеством ежедневных численных показателей это дало прирост показателя качества модели на порядок.

Для оценки качества разработанного метода размещения видео в кэше требуются дополнительные исследования.

Заключение

Работа над данным проектом подразумевала исследовательскую деятельность в сфере прогнозирования популярности видеоконтента, а также реализацию методов, способных предсказывать популярность отдельно взятых видеообъектов. В результате было разработано программное средство прогнозирования популярности видео, состоящее из двух модулей: Scrapper и Predictor.

Scrapper – это приложение, реализованное на языке программирования Python и предназначенное для сбора и обработки информации о видеобъектах, размещённых в трендах одного из самых популярных сервисов видеохостинга YouTube. Этот модуль позволяет получать описательные характеристики, а также статистические показатели видео за определённый промежуток времени: чтобы получить данные за n дней, необходимо единожды запускать приложение в каждый из подряд идущих n дней. Среди численных метрик присутствуют: просмотры, лайки, дизлайки, комментарии, количество подписчиков канала, количество видео на канале, число лайков и ответов под самым популярным комментарием.

Predictor – набор блокнотов формата Jupyter Notebook, в каждом из которых построены модели прогнозирования и проведены экспериментальные исследования. В контексте поставленной задачи под популярностью подразумевалось количество просмотров видеообъекта, поэтому в терминах машинного обучения решалась задачи регрессии. Были рассмотрены несколько алгоритмов машинного обучения, лучшим среди которых с точки зрения установленной метрики качества оказался метод линейной Ridge-регрессии.

Полученные результаты говорят о том, что необходимы дополнительные исследования в области прогнозирования популярности видеоконтента. В дальнейшем эта задача не потеряет свою актуальность, в силу того что рост объёма хранения необходимого контента превышает технические возможности сетей доставки контента и в любом случае необходимо постоянно

выбирать, какой контент следует сохранять на отдельно взятом сервере.

Во время реализации текущей версии программного средства были сформулированы некоторые из возможных в дальнейшем улучшений, которые потенциально существенно увеличат качество предсказания популярности видеообъектов.

В первую очередь необходимо уделить большее внимание таким описательным характеристикам, как заголовок видео, название канала, теги и описание видео. Есть основания думать, что применение методов NLP позволит улучшит качество модели, ведь смысловая составляющая публикаций в интернете напрямую влияет на распространение контента. Помимо этого имеет смысл учесть корреляцию описательных характеристик отдельно взятого видео с частыми поисковыми запросами внутри социальной сети YouTube.

Не менее важным является пересмотр множества полезных для предсказаний признаков видеообъектов. Среди характеристик видео, предоставляемых YouTube API, необходимо получить как можно большее количество и с помощью более тщательного анализа выделить те, от которых значительно зависит актуальность и востребованность предлагаемого пользователям контента.

Кроме того, есть предположение, которые необходимо проверить: обучаемый датасет должен учитывать региональные особенности потребителей контента. Это можно сделать, например, обучив модели для видеообъектов из каких-то отдельных регионов или групп регионов. В этой связи необходимо либо сгруппировать имеющиеся, либо использовать наиболее важные с культурной и графической точек зрения регионы.

Актуальным с точки зрения машинного обучения является вопрос о размере данных, на которых обучается модель. В текущей работе был использован датасет, изначально содержащий ≈ 8200 объектов. После предобработки данных и удаления невалидных объектов осталось ≈ 7600 объектов. Вероятно, качество моделей упиралось в существенный недостаток объектов для обучения. В дальнейшем возможно увеличить как количество видео, так

и объём признакового описания. Если задачу извлечения как можно большего числа характеристик решить несложно – YouTube API практически не ограничивает возможности внутри запроса к видео, – то стратегия рассмотрения в обучении большего числа объектов требует тщательного переосмысления. Это во многом связано с квотой на количество запросов к API в течение суток. Однако с помощью различных эвристик можно разработать более продвинутый алгоритм сбора и обработки данных, способный выявлять намного большее количество «полезных» для обучения видеообъектов.

В рамках данной курсовой работы популярность характеризовалась числом просмотров и/или лайков, дизлайков и комментариев. В дальнейших исследованиях нелишним будет пересмотр показателя востребованности видеообъекта.

Список литературы

[1] Е. А. Соколов, «Машинное обучение и приложения. Лекции», 2020, https://github.com/esokolov/ml-course-hse.

[2] YouTube CDN, https://cloud.google.com/cdn/docs/locations.

[3] Nesrine Ben Hassine, Dana Marinca, Pascale Minet, Dominique Barth, Expert-based On-line Learning and Prediction in Content Delivery Networks, IEEE, 2016, https://ieeexplore.ieee.org/document/7577054.

[4] YouTube Data API, https://developers.google.com/youtube/v3.

[5] YouTube Trends, https://www.youtube.com/feed/trending.

[6] MachineLearning.ru, http://www.machinelearning.ru.

[7] Ridge Regression, https://scikit-learn.org/stable/modules/linear_model.html.

[8] CatBoost, https://catboost.ai.

[9] Daniel Chepenko, Introduction to Gradient Boosting on Decision Trees with Catboost, Towards Data Science, 2019, https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14.

[10] Derrick Mwiti, Fast Gradient Boosting with CatBoost, KDnuggets, 2020: https://www.kdnuggets.com/2020/10/fast-gradient-boosting-catboost.html.

[11] Multi Target Regression, https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html.

[12] Yu Tao, Kaigui Bian, Chengliang Gao, Yuanxing Zhang, Lingyang Song, Shaolin Dong, Machine Learning Assisted Content Delivery at Edge of Mobile Social Networks, IEEE, 2019, https://ieeexplore.ieee.org/document/8923784.

[13] Huda S. Goian, Omar Al-Jarrah, Sami Muhaidat, Yousof Al-Hammadi, Paul Yoo, Mehrdad Dianati, Popularity-Based Video Caching Techniques for Cache-Enabled Networks: A Survey, IEEE, 2019, https://ieeexplore.ieee.org/document/8658196.