

Аннотация

Целью данной работы является изучение возможности определять неработающие видео по статистике просмотров. В ходе работы был предложен ряд моделей и проведены численные эксперименты. Наилучшей точности классификации удалось достичнуть с использованием классификатора из библиотеки CatBoost [13]. Наилучший классификатор показал точность в 80.3 % и полноту в 51.4 %. Точность классификации удалось улучшить на 9 % с помощью добавления в данные событий плеера и создания отдельной модели для каждого плеера. В данном случае удалось достичнуть точности классификации в 89.3 % при той же полноте. На основании данной работы рекомендуется внедрение классификатора в сервис Yandex Video, а также дальнейшее изучение свойств модели и ее последующее улучшение.

Содержание

1. Используемые определения	4
1.1. Видеоплеер	4
1.2. Неработающие видео	4
1.3. Бинарная классификация	4
1.4. Решающие деревья	5
1.5. Bootstrap aggregation	7
1.6. Модель extremely randomized trees	7
1.7. Gradient boosting	8
1.8. Yandex Video	8
1.9. Yandex Toloka	8
2. Введение	10
3. Предыдущие работы	11
4. Подготовка данных	12
4.1. Исходные данные	12
4.2. Выбор признаков	12
5. Обучение классификатора	14
5.1. Выбор модели	14
5.2. Обучение модели	14
5.3. Анализ модели	15
5.4. Одноплеерная модель	19
6. Заключение	23
7. Список литературы	24

Используемые определения

1.1 Видеоплеер

Под видеоплеером (плеером) в данной работе подразумевается элемент web страницы, позволяющий пользователю просматривать видеозаписи (смотри рисунок 1). Как правило, один видеоплеер умеет воспроизводить ряд видеозаписей, хранящихся на сервере провайдера контента.

Помимо предоставления конечным пользователям возможности просматривать видеозаписи, видеохостинги часто предоставляют возможность другим интернет ресурсам интегрировать этот контент в свои страницы. Для этого, как правило, используется тэг iframe языка разметки HTML. Данный тэг, согласно спецификации [10], позволяет web странице открывать внутри себя вложенные страницы, например, с видеоплеером.

Кроме того ряд крупных видеохостингов предоставляют более обширное API для интеграции контента. В частности, некоторые плееры открывают доступ к так называемым событиям плеера, то есть оповещениям о некоторых событиях, которые дают возможность оценивать состояние контента. К таким событиям чаще всего относятся события старта, паузы и ошибки плеера. Примером плеера, который предоставляет подобную функциональность может служить плеер видеохостинга YouTube [17].

1.2 Неработающие видео

Под неработающим видео (рисунок 2) в данной работе поднимается любая видеозапись, которую пользователь не может просмотреть. Videозапись может прекратить работу по ряду причин: отключение сервера видеохостинга, на котором данная видеозапись хранилась или блокировка просмотра видеозаписи на основании обращения правообладателя контента.

1.3 Бинарная классификация

В данной работе рассматривается задача бинарной классификации, то есть разбиения множества объектов на два класса. Были выбраны следующие обозначения: под классом 0 (нулевым классом, отрицательным классом) подразумевается класс работающих видео, а под классом 1 (положительным классом) подразумевается класс неработающих видео, детектирование которых и является целью данной работы.

При анализе предсказания модели в задаче бинарной классификации принято говорить о четырех группах объектов: ложные отрицательные объекты (false negatives, *FN*), ложные

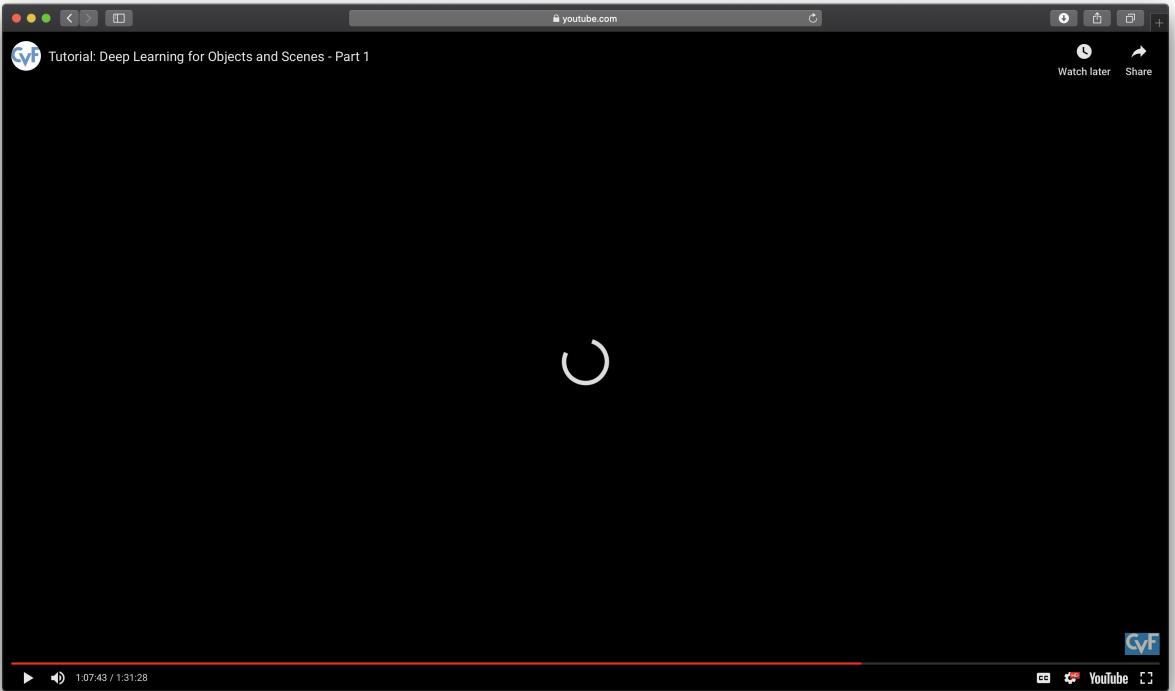


Рисунок 1 — Видеоплеер YouTube.

положительные объекты (false positives, FP), верно отрицательные объекты (true negatives, TN) и верно положительные объекты (true positives, TP).

Для оценки модели в задаче бинарной классификации принято использовать метрики точности (precision, P) и полноты (recall, R), которые вводятся следующим образом:

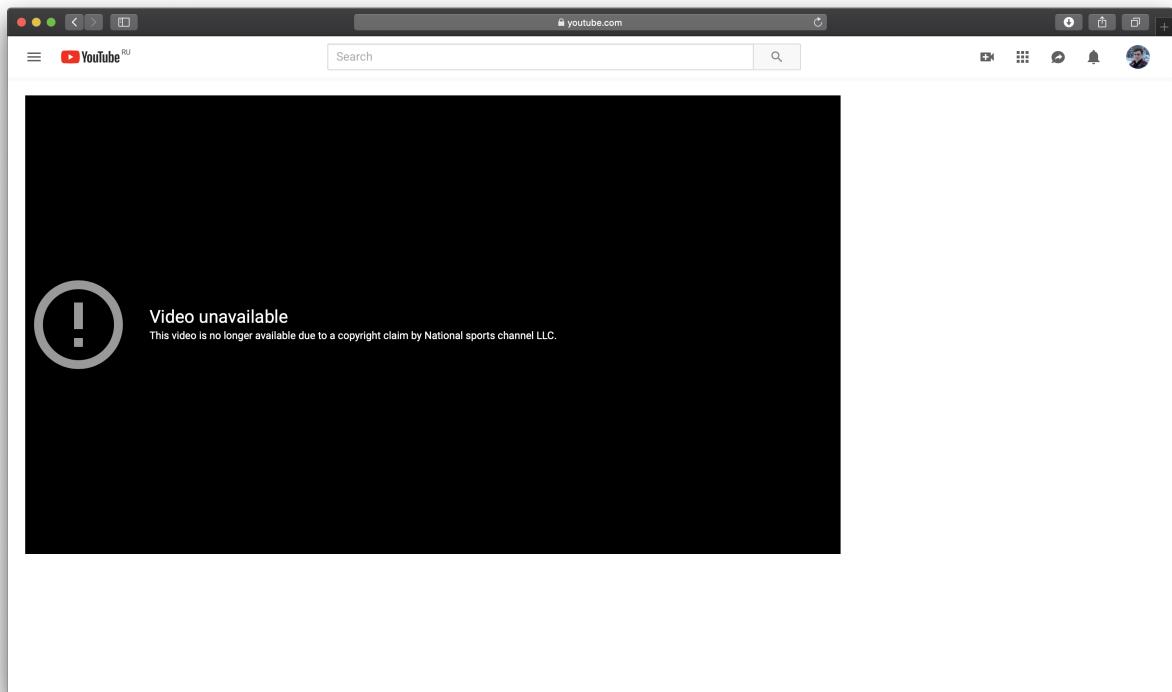
$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}.$$

Здесь и далее в данной работе под точностью и полнотой классификации подразумеваются именно введенные выше метрики.

1.4 Решающие деревья

Решающие деревья [3] — это модель машинного обучения, которая позволяет решать задачи классификации и регрессии с помощью построения двоичного дерева, подобного дереву поиска. В каждом внутреннем узле данного дерева находится условие, исходя из соответствия которому для конкретного объекта поиск спускается либо в правое, либо в левое поддерево. В листьях данного дерева находится предсказание модели, то есть метка класса в случае задачи классификации либо численное значение в случае задачи регрессии.

Решающие деревья являются одной из самых интерпретируемых моделей машинного обучения, так как финальная модель может быть очевидным способом разложена в ряд логических предикатов и соответствующих им значений предсказания. Кроме того данная модель является устойчивой к высокой размерности данных и зависимости среди признаков, что позволяет сводить предобработку данных к минимуму при работе с данной моделью. Тем



A screenshot of a Yandex search results page. The search query is 'Бразилия - боливия 3-0 обзор матча кубок америки 15.06.' The results show several video thumbnails related to the match. One thumbnail is highlighted, showing a video from YouTube. The video player interface on the right side of the result shows a black screen with a large exclamation mark and the message 'Video unavailable' and 'This video is no longer available due to a copyright claim by National sports channel LLC.' The Yandex search bar is at the top, and the page includes various navigation links and ads typical of a search engine results page.

Рисунок 2. Пример неработающего видео на сайте www.youtube.com и на сайте Yandex Video.

не менее данная модель имеет и ряд существенных минусов. Так предсказание решающего дерева очень чувствительно к гиперпараметру глубины дерева. Действительно, если глубина не ограничена сверху, то модель легко переобучается (производя деление до тех пор, пока в каждом листе не останется единственный объект обучающей выборки). Предсказания такой модели скорее всего выйдут очень шумными, а обобщающая способность модели окажется крайне низкой. В то же время слишком жесткое ограничение на глубину деления вызовет создание модели, сложности которой недостаточно для описания зависимостей в данных. В то же время, даже обладая априорным знанием оптимальной глубины дерева, нахождение оптимального дерева остается нетривиальной задачей, которой посвящен ряд статей (например [8]). Все эти проблемы вместе с развитием других моделей машинного обучения привели к тому что на данный момент решающие деревья в чистом виде практически не используются, однако они являются важным составляющим компонентом для более сложных и эффективных моделей.

1.5 Bootstrap aggregation

Bootstrap aggregation (bagging) [2] — это метод, применяемый в моделях машинного обучения для улучшения стабильности и точности предсказания. Метод заключается в следующем: пусть у нас есть выборка из n объектов, давайте сгенерируем из нее m подвыборок размера n' путем последовательного выбора произвольного объекта исходной выборки. Да-вайте теперь обучим на полученных выборках модели и усредним их предсказания. Можно доказать, что в случае независимости предсказаний полученных моделей, дисперсия результирующего предсказания падает с увеличением их числа.

Изначально метод никак не связан с решающими деревьями, однако на данный момент чаще всего bagging применяется именно для моделей решающего дерева. Это связано с тем, что для таких моделей проще обеспечить независимость предсказания отдельных моделей путем дополнительной подвыборки признаков, по которым разрешено ветвиться каждому конкретному дереву в ансамбле, а также довольно мягкому ограничению глубины деревьев (предполагается, что переобученные на разных подвыборках деревья будут давать существенно независимые предсказания). В таком случае также не сильно существенно строить оптимальное дерево каждый раз. Поэтому чаще всего выбор признака и значения для дальнейшего ветвления на каждом шагу производится жадным образом, что позволяет значительно ускорить время обучения модели. На этом основана модель random forest [7], которая использовалась в данной работе, как базовая модель. Реализация модели была взята из библиотеки Scikit-learn [12]. С ее помощью удалось достигнуть точности предсказания неработающих видео в 72.0 % и полноты предсказания в 50.8 %.

1.6 Модель extremely randomized trees

Модель extremely randomized trees [6] — это модель, во многом похожая на модель random forest. Для ее построения также производится генерация подвыборок в соответствии с методом bagging и обучение на подвыборках моделей решающего дерева. Однако в отличии

от жадного поиска оптимального признака и значения для ветвления при обучении каждого дерева в ансамбле, как это происходит при обучении модели random forest, модель extremely randomized trees произвольно генерирует несколько предполагаемых вариантов и выбирает оптимальный среди них. Это позволяет сделать деревья в ансамбле еще более независимыми, что в ряде случаев позволяет улучшить точность предсказания. Реализация данной модели также была взята из библиотеки Scikit-learn [12]. Использование позволило увеличить точность предсказания до 74.3 %, однако при этом полнота упала до 47.0 %.

1.7 Gradient boosting

Gradient boosting (boosting) [5] — это метод машинного обучения, позволяющий создавать ансамбль из простых моделей, каждая из которых обладает достаточно низкой предсказательной способностью, обладающий высокой точностью предсказания. Этого удается достичь путем последовательного обучения моделей: первая модель обучается на исходных данных, каждая последующая же модель учится предсказывать уже не искомое значение, а ошибку получившегося до нее ансамбля (в случае задачи регрессии, в случае задачи классификации она учится предсказывать класс объекта, но больший вес отдается тем объектам, на которых предыдущий ансамбль ошибается). В соответствии с точностью ее предсказания, получившейся модели присваивается вес и она пополняет ансамбль.

Несмотря на то, что данный метод также не имеет непосредственного отношения к решающим деревьям, чаще всего в качестве базовых моделей в ансамбле выбираются именно решающие деревья небольшой глубины. Реализация данной модели была взята из библиотеки CatBoost [13]. Модель показала наилучшую точность из полученных в 80.3 %. Полнота получившейся модели оказалась также выше чем у моделей random forest и extremely randomized trees и составила 51.4 %. Также для данной модели был произведен эксперимент с предсказанием неработающих видеозаписей для одного плеера. В данном эксперименте удалось достигнуть точности в 89.3 % при той же полноте.

1.8 Yandex Video

Yandex Video [16] — это сервис, позволяющий пользователям смотреть видеозаписи, собранные с множества различных видеохостингов. Из-за большого объема контента и невозможности моментально реагировать на его изменения, хранить видео на серверах компании Yandex не представляется возможным. Поэтому видеозаписи показываются пользователям в виде интегрированных с помощью тэга iframe видеоплееров.

1.9 Yandex Toloka

Yandex Toloka [15] — это сервис, позволяющий публиковать некоторые несложные задания, которые другие пользователи могут выполнять за материальное вознаграждение. Данный сервис использовался в работе для получения экспертной разметки данных: пользователям, взявшимся за выполнение задания, показывался iframe с видеозаписью и требовалось отве-

тить, проигрывается эта видеозапись или нет. Для увеличения точности разметки, одна и та же видеозапись размечалась несколько раз разными пользователями. На разметку отправлялись видеозаписи, для которых за предыдущий день набралось хотя бы 100 зафиксированных просмотров. Таким образом за два раза была размечена выборка размером в 101034 видеозаписи. Неработающие видеозаписи составили 1.36 % выборки.

Введение

Выбор темы обусловлен спецификой работы сервиса Yandex Video. В силу того, что контент данного сервиса выдается пользователю в виде плееров, интегрированных в сайт с помощью тэга `iframe`, у данного сервиса возникает необходимость вовремя детектировать неработающие видеозаписи и убирать их из выдачи. На данный момент эта задача решается с помощью периодического обхода видеозаписей с помощью специального механизма, умеющего эмулировать работу клиентского web браузера, а также умеющего нажимать на кнопку начала проигрывания видеозаписи и оценивать, началось ли воспроизведение. Данный механизм позволяет получать достаточно достоверный сигнал о недоступности видео, однако обладает рядом существенных минусов, например, невозможность проверить доступность видео в странах кроме России (так как все запросы территориально отправляются из России), то есть учесть специфику локальных блокировок сайтов и контента. Для устранения подобных недостатков было предложено использовать статистику просмотров для детектирования неработающих документов.

С этой целью по сессиям пользователей собирается анонимная статистика: когда пользователь начинает просмотр видеозаписи, включается таймер, когда же пользователь переключает видео или закрывает страницу сервиса, время просмотра записывается для дальнейшего анализа. Вместе со временем просмотра сохраняются и события плеера, если они доступны. Основной проблемой таких данных является высокая степень загрязненности. Если во время просмотра или переключения видео у пользователя пропадет интернет соединение, запись о просмотре может не прийти или содержать неверные данные. Также возможен случай, когда пользователь включает проигрывание видеозаписи, а затем отходит от компьютера. В это время таймер считает время просмотра, несмотря на то, что видео могло оказаться неработающим и так и не загрузиться.

Задачей данной работы ставится создание модели, которая смогла бы предсказывать неработающие видео с высокой точностью, чтобы данный механизм мог дополнить, а в перспективе и заменить механизм обхода видеозаписей.

В качестве метода исследования был выбран экспериментальный подход: сбор и разметка выборки, оценка точности и полноты классификации с помощью метода перекрестной проверки и анализ предсказаний полученной модели.

Предыдущие работы

Наиболее близкой задачей к поставленной является задача детектирования soft 404 страниц. В соответствии с протоколом HTTP [4], если при обращении к серверу клиент запрашивает документ, который не доступен по той или оной причине, сервер должен возвращать ошибку. Как правило это ошибка 403 (Forbidden Error) или 404 (Not Found). Тем не менее многие интернет ресурсы стремятся предоставить пользователю более дружественный интерфейс и вместо возврата ошибки имитируют нормальную работу, возвращая код возврата 200 (OK), а вместо запрашиваемого документа отображают страницу, оповещающую пользователя о причинах недоступности контента.

Подобное поведение делают задачу детектирования недоступных документов нетривиальной. Первая попытка борьбы с данной проблемой была предпринята в статье от 2004 года [1]. Авторы данной статьи предлагают спровоцировать сервер вернуть заведомо недоступный документ путем прибавления к имеющемуся названию искомого документа произвольного окончания. После этого предлагается сравнивать поведение сервера в случае обращения к исходному документу и к заведомо недоступному. Второе упоминание о проблеме soft 404 датируется 2012 годом [9]. В отличии от первой статьи, данная работа целиком посвящена проблеме детектирования soft 404 страниц. Авторы этой статьи предлагают использовать классификатор, который использует в качестве данных лексические сигнатуры, содержащиеся в заголовке или в тексте страницы.

К сожалению ни один из предложенных способов детектирования soft 404 документов не может быть применен к задаче детектирования неработающих видео. Это вызвано тем, что, в отличии от других интернет ресурсов, видеохостинги при обращении к несуществующему или уже удаленному документу чаще всего предпочитают показывать страницу, полностью идентичную странице, которая была бы отображена в случае, если видео было доступно для просмотра. Сообщение о недоступности видеозаписи как правило располагается в самом видеоплеере. Таким образом подход, основанный на детектировании переадресаций на заранее заготовленные документы, содержащие оповещение о недоступности видео не применим так как в данном случае переадресаций не происходит. Анализ лексических сигнатур, содержащихся на странице и ее заголовке также неприменим в силу того, что и тот и другой текст не зависит от доступности видеозаписи, а само сообщение о возникшей проблеме находится в плеере. Таким образом для детектирования таких сообщений требуется технически сложный механизм, который умеет эмулировать доступ к странице из пользовательского web браузера и попытку воспроизведения видеозаписи.

Подготовка данных

4.1 Исходные данные

В качестве исходных данных имеется таблица, каждая строка которой содержит некоторый идентификатор плеера, идентификатор видеозаписи, а также время просмотра и события плеера, если они доступны. Также из разметки сайта провайдера контента берется продолжительность видеозаписи. Пример исходных данных представлен в таблице 1.

Плеер	Видео	Длительность, с	Просмотр, с	Старт	Ошибка
Плеер1	Видео1	100	10	True	
Плеер2	Видео2	200	180	True	
Плеер3	Видео3	0	5		True
...

Таблица 1 — Пример исходных данных.

4.2 Выбор признаков

Так как данные сильно зашумлены, клики необходимо усреднять. При этом события плеера — достаточно надежный источник информации, тем не менее они имеют несколько существенных недостатков. К этим недостаткам относится тот факт, что события плеера доступны не для всех плееров, а также каждомуциальному плееру характерен некоторый профиль событий для различных состояний видеозаписи, однако эти профили могут существенно отличаться для двух разных плееров. Существуют два различных способа борьбы с данными недостатками: во-первых можно сделать больший упор на время просмотра и производимые из него признаки, как на более универсальный источник информации, во-вторых можно попытаться сделать отдельную модель для каждого плеера, чтобы она смогла в большей мере использовать сигнал событий плеера, если они доступны. В ходе данной работы были опробованы оба подхода.

Также в силу высокого уровня зашумленности исходных данных понятно, что их необходимо каким-то образом усреднять. В случае с данным событием видеоплеера усреднение очевидно — если событие произошло, то данной записи ставится в соответствие единица, в противном случае ноль и в качестве признака используется среднее значение, то есть доля просмотров, в которых данное событие наблюдалось.

Из времени просмотра было выбрано сгенерировано большее количество признаков исходя из некоторых предположений о природе данных. Первое предположение заключается в

том, что, если видеозапись не работает, большинство пользователей заметят это и переключать видео за довольно короткий промежуток времени, близкий к 30-40 секундам. Поэтому в качестве первой группы признаков были выбраны доли пользователей смотревших видео не менее 15, 30 и 45 секунд. Следующая идея говорит о том, что информация о том, как хорошо пользователи смотрят данное видео хранится в распределении доли просмотра контента. Для простоты и однообразности расчета распределение было введено в выборку в виде долей пользователей, досмотревших видео до 5, 10, 20, 30, 40, 50, 60, 70, 80 и 90 % длительности видеозаписи. Последнее наблюдение говорит о том, что если видеозапись попала на выдачу, то когда-то она была обнаружена обходом, а значит в какой-то момент времени она работала. Таким образом чаще всего мы занимаемся не поиском неработающего контента, а пытаемся поймать момент, в который видеозапись перестала работать. В таких случаях принято говорить о данных в форме временного ряда, то есть последовательных значениях, зависящих от времени. Однако для удобства работы с данными и расширения круга потенциальных алгоритмов классификации хотелось бы иметь данные в виде вектора фиксированной размерности. Также важно помнить о шуме в данных, а значит и необходимости усреднения данных. Для соблюдения всех ограничений было принято решение воспользоваться методом скользящего среднего. В таком случае мы получаем, например, такие признаки как среднее время просмотра за 5 последних просмотров заканчивая последним записанным просмотром и такое же среднее за 5 просмотров заканчивая предпоследним просмотром и так далее. Хотелось бы заметить, что подход со скользящим средним применим не только к времени просмотра. Скользящее среднее также считалось для событий плеера и долей смотревших видео хотя бы 15, 30, 45 секунд и так далее.

В итоге для каждого видео мы получили вектор признаков, размерность которого составила порядка 150. После того как были выбраны признаки и собрана выборка, разметка видеозаписей производилась с помощью сервиса Yandex Toloka. Таким образом был собран датасет из порядка 100000 размеченных объектов.

Обучение классификатора

5.1 Выбор модели

Заметим, что пространство признаков имеет очень большую размерность. К тому же многие из признаков сильно взаимосвязаны с другими (действительно, доля досмотревших до 90 % видео не может быть выше доли досмотревших до 50 %). Если с размерностью можно справиться, например, воспользовавшись методом главных компонент [11], то избавиться от зависимости между признаками гораздо сложнее. В данной работе было принято решение использовать модели машинного обучения, основанные на решающих деревьях [3]. Это позволяет исключить из предобработки уменьшение размерности признаков, а также перестать беспокоиться о зависимостях в признаках. Однако решающие деревья в чистом виде используются редко из-за высокой чувствительности к значениям гиперпараметров и низкой обобщающей способностью модели. Однако решающие деревья показывают очень хорошие результаты, когда они используются в виде разных ансамблей.

В качестве базовой модели был выбран случайный лес [7]. Данный выбор обусловлен тем что метод зарекомендовал себя как неприхотливый алгоритм, который тем не менее показывает довольно высокие результаты практически во всех задачах. Также было замечено, что выборка в 100000 объектов не очень велика. Также важно понимать, что среди этих объектов только около 1 % видеозаписей были размечены как неработающие. Таким образом выборка содержит довольно мало сигнала, который мы хотим детектировать. Поэтому в качестве дополнительной модели был выбран метод чрезвычайно случайных деревьев (*extremely randomized trees*) [6], который позволяет в некоторых случаях уменьшить разброс предсказаний за счет небольшого увеличения смещения предсказания. В качестве последней модели был выбран метод градиентного бустинга (*gradient boosting*) [5] на решающих деревьях, как наиболее точной хотя и наименее неприхотливый алгоритм.

Численные эксперименты производились с помощью языка программирования Python [14]. Реализации первых двух алгоритмов были взяты из библиотеки scikit-learn [12], реализация последнего была взята из библиотеки CatBoost [13].

5.2 Обучение модели

Для подбора гиперпараметров и получения оценки точности и полноты использовался сеточный поиск с оценкой точности и полноты с использованием 5-кратной перекрестной проверки. Для моделей random forest и extremely randomized trees подбиралось количество

деревьев в ансамбле. Для классификатора CatBoost кроме того подбиралась максимальная глубина дерева.

Результаты сеточного поиска представлены в таблице 2, а также на рисунках 3, 4 и 5. При этом в таблице 2 модели extremely randomized trees и классификатору из библиотеки CatBoost в таблице соответствуют по две строки каждому. Это сделано для того, чтобы показать метрики и для модели с наибольшей точностью, и для модели с наивысшей полнотой. Для модели random forest разница между точностью для этих двух классификаторов несущественна, поэтому представлены только значения метрик для модели с наивысшей полнотой.

Модель	Точность, %	Полнота, %	Число деревьев	Глубина
RandomForestClassifier	72.0	50.8	500	
ExtraTreesClassifier	74.3	47.0	1400	
ExtraTreesClassifier	73.9	47.2	1100	
CatBoostClassifier	80.3	51.4	650	3
CatBoostClassifier	77.5	53.2	500	4

Таблица 2 — Результаты сеточного поиска

Как можно видеть, модель extremely randomized trees действительно смогла немного увеличить точность классификации по сравнению с результатами базовой модели. Однако при этом модель оказалась слегка более смещенной из-за чего снизилась полнота. Лучше всех себя показал классификатор из библиотеки CatBoost. Именно с использованием данного алгоритма была достигнута наилучшая в данном эксперименте точность классификации в 80.3 %. При этом полнота классификации у данной модели оказалась выше, чем у моделей RandomForestClassifier и ExtraTreesClassifier. Наибольшей полноты классификации в данном эксперименте (51.4 %) также удалось достичь именно классификатору из библиотеки CatBoost, правда, при других значениях гиперпараметров.

5.3 Анализ модели

После численного эксперимента был произведен анализ наилучшей модели. В результате этого анализа были выявлены две существенные закономерности. Во-первых не смотря на преобладающее количество признаков, полученных из времени просмотра, события плеера, а именно усредненные события старта и ошибки, обладают высоким весом в финальной модели. Вторым важным фактом, обнаруженным во время анализа модели, является то, что выборка содержит в себе заметную долю видеозаписей, неверно размеченных как работающие. Как выяснилось в дальнейшем, это вызвано спецификой работы сервиса Yandex Toloka, который склонен верить хорошо зарекомендовавшим себя пользователям в случае, если они размечают видеозапись, как работающую (в силу того, что достоверно известно, что неработающих видеозаписей всего около 1 %).

Напомню, что изначальным желанием было избавиться или практически избавиться от влияния событий плеера на предсказание модели, так как, несмотря на аккуратность и информативность такого сигнала в целом, они доступны не для всех видеозаписей, а также

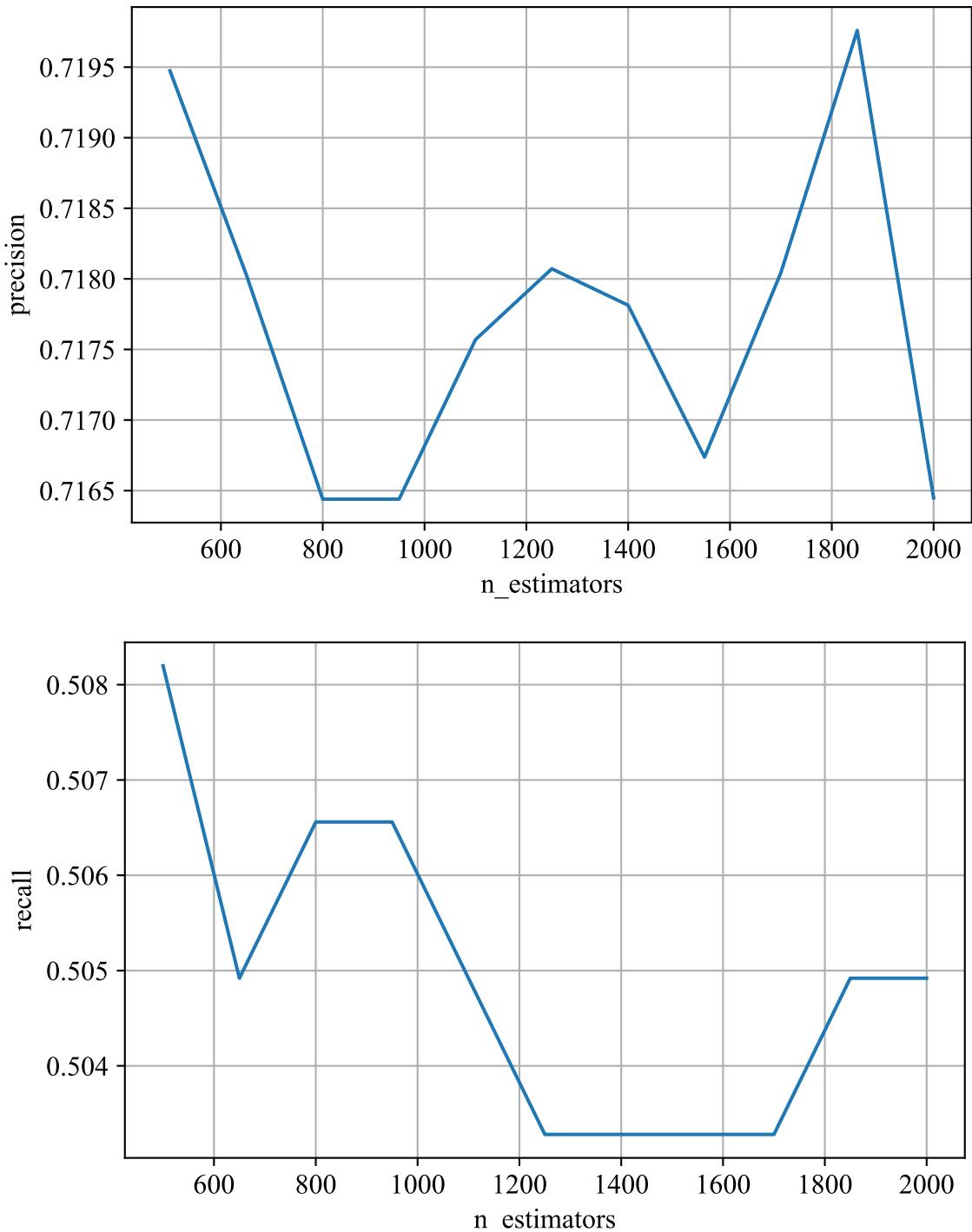


Рисунок 3 — Результаты сеточного поиска для модели RandomForestClassifier

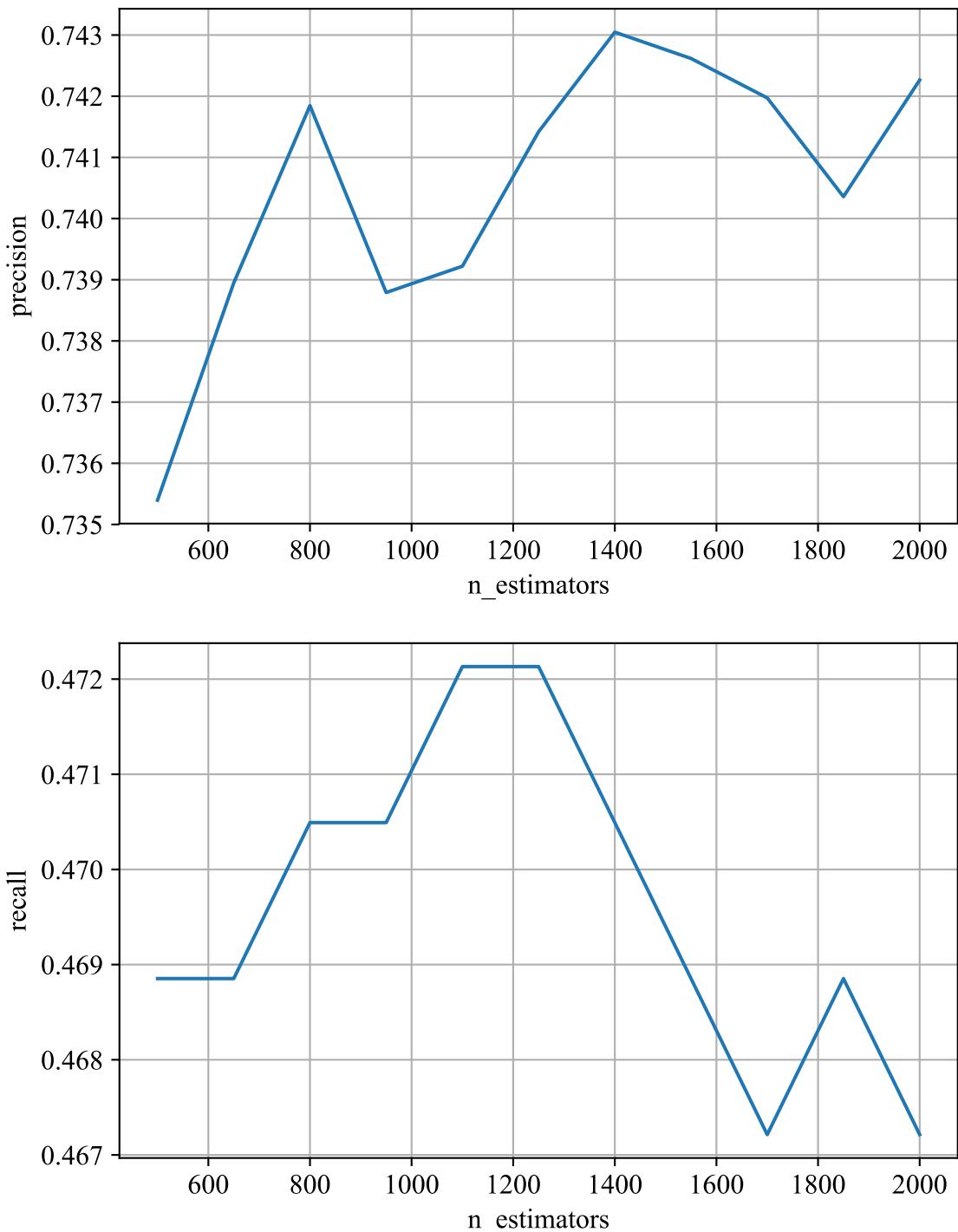


Рисунок 4 — Результаты сеточного поиска для модели ExtraTreesClassifier

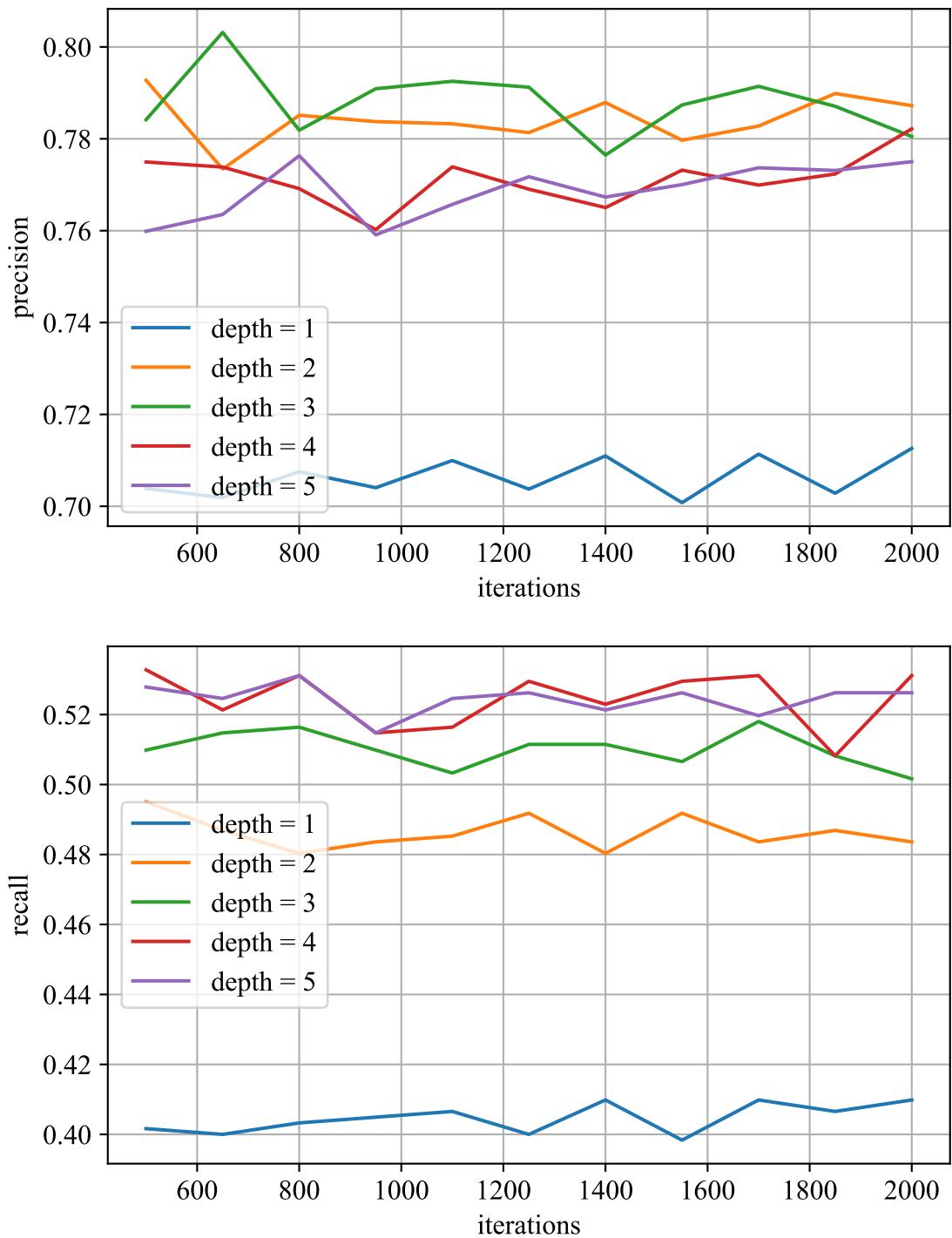


Рисунок 5 — Результаты сеточного поиска для модели CatBoostClassifier

характерный профиль событий различается от плеера к плееру. Тем не менее модель посчитала данные признаки информативными и смогла добиться достаточно высокой точности. В связи с этим было предложено добавить в данные идентификатор плеера, как категориальный признак, что позволило бы модели корректировать ожидаемый профиль событий в зависимости от их источника. Так как из всех используемых моделей с категориальными признаками умеет работать только CatBoost (для остальных моделей потребовалась бы дополнительная предобработка данных), было принято решение ограничиться им. Результаты эксперимента представлены в таблице 3, а также на рисунке 6.

Модель	Точность, %	Полнота, %	Число деревьев	Глубина
CatBoostClassifier	72.6	65.0	1250	2
CatBoostClassifier	71.3	68.1	800	5

Таблица 3 — Результаты эксперимента с категориальным признаком.

Как можно видеть, точность классификации упала на 7.7 % до 72.6 %, однако полнота возросла на 14.9 % и составила 68.1 %. Это говорит скорее всего о переобучении модели на новый категориальный признак, то есть модель просто запомнила, что видеозаписи некоторых плееров чаще всего не работают и прекратила попытки честно их классифицировать.

5.4 Одноплеерная модель

Для решения проблемы неаккуратной разметки данных, было принято решение попробовать размечать выборку с помощью данных обхода. В таком случае мы получаем очень достоверный сигнал о том, какие видеозаписи не работали. Однако возникает обратная проблема: у нас нет достоверной информации о том, какие работали. Для решения данной проблемы был выбран следующий подход: известно, что среди видеозаписей, которые представлены на выдаче поиска, не работает примерно 1 %. Исходя из этого, при составлении выборки мы считаем количество достоверно неработавших видео исходя из данных обхода и размечаем эти видеозаписи. Остальные видео сортируем в убывающем порядке по среднему времени просмотра и отбираем столько, чтобы баланс классов составлял 99 к 1. Таким образом мы получаем размеченную выборку большого размера, так как история обхода доступна за довольно длительный промежуток времени. Это позволяет ограничиться данными единственного плеера, что позволит модели в полной мере использовать данные событий плеера. Так как в результирующей выборке получилось около 1000000 объектов, время обучения модели стало занимать заметно большее время. Поэтому было принято решение ограничиться в данном эксперименте только хорошо зарекомендовавшим себя классификатором из библиотеки CatBoost. Результаты сеточного поиска представлены в таблице 4 и на рисунке 7.

Эксперимент показал прирост точности классификации в 9.1 %. Сама точность составила 89.3 %, что является хорошим результатом учитывая природу данных и их зашумленность. При этом модель показала полноту, которая не существенно отличается от случая модели, обученной на данных размеченных с помощью сервиса Yandex Toloka. Наилучшая полно-

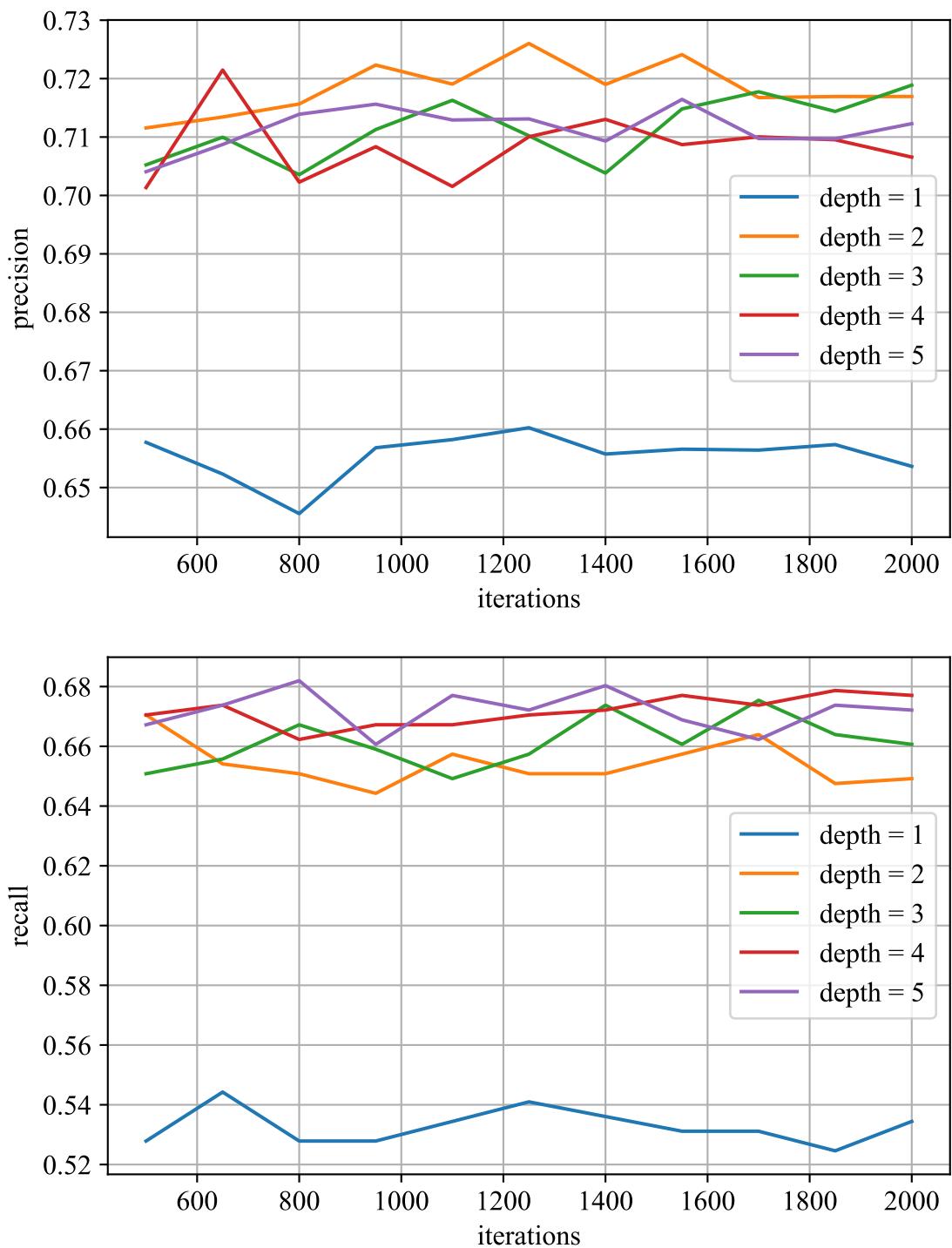


Рисунок 6 — Результаты эксперимента с категориальным признаком.

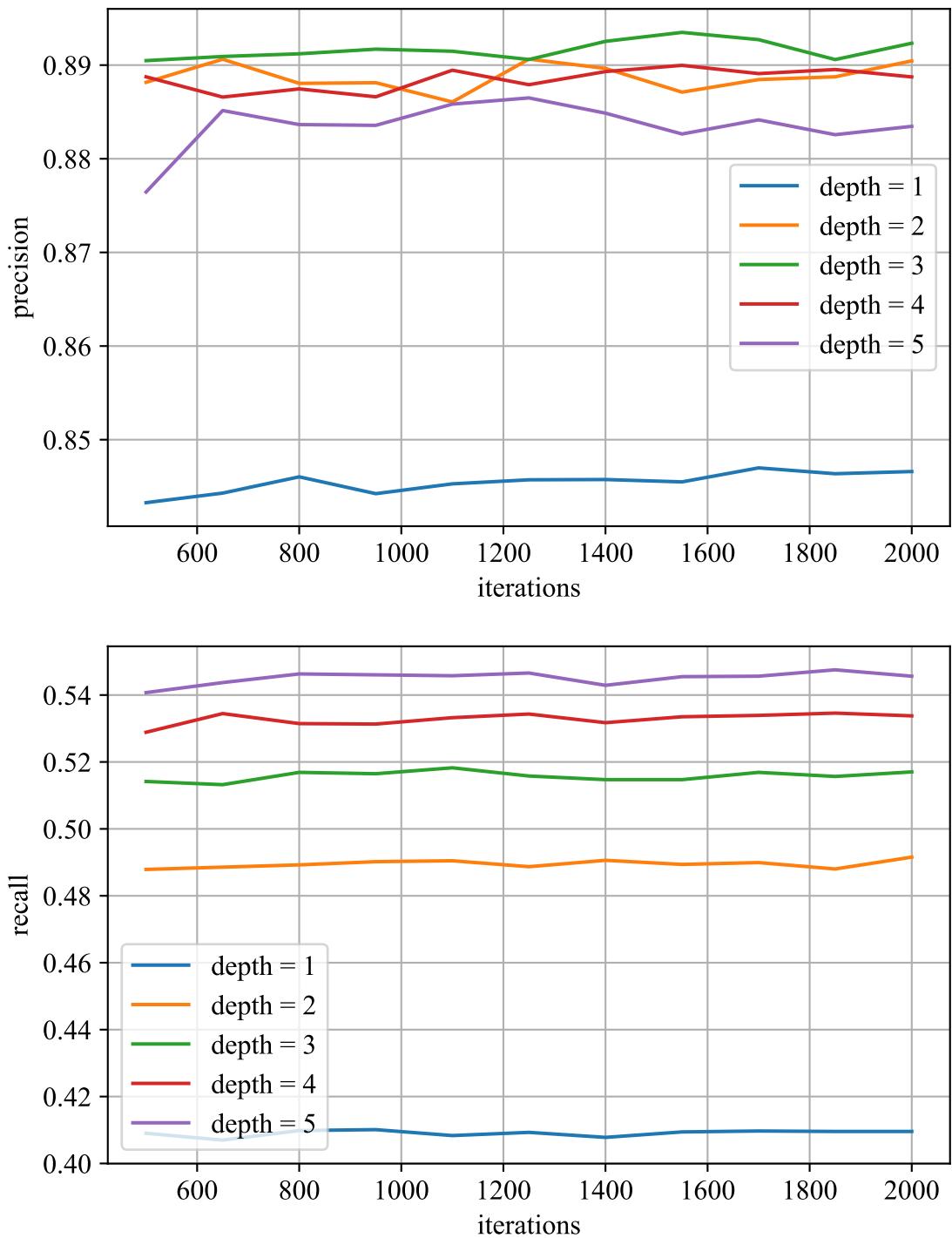


Рисунок 7 — Результаты сеточного поиска для модели обученной на данных обхода

Модель	Точность, %	Полнота, %	Число деревьев	Глубина
CatBoostClassifier	89.3	51.4	1550	3
CatBoostClassifier	88.2	54.7	1850	5

Таблица 4 — Результаты сеточного поиска для модели обученной на данных обхода.

та, которая была достигнута в данном эксперименте (54.7 %) и вовсе выше, чем в случае экспертной разметки данных.

Заключение

Данная работа была посвящена изучению частного случая проблемы soft 404, а именно детектированию неработающих видеозаписей. Так как ни один из ранее предложенных способов решения данной проблемы не подходит в данном случае из-за специфики задачи, было предложено использовать статистику просмотров для детектирования неработающих видеозаписей. Благодаря такому подходу удалось обучить классификатор из библиотеки CatBoost, который показал точность классификации 80.3 % и полноту 51.4 %. Также был предложен способ ограничиться одним плеером, что позволило увеличить точность классификации на 9 % при практически неизменной полноте. Таким образом был построен классификатор, обладающий точностью предсказания 89.3 % и полнотой 51.4 %.

Список литературы

- [1] Ziv Bar-Yossef et al. “Sic transit gloria telae”. In: *Proceedings of the 13th conference on World Wide Web - WWW '04*. ACM Press, 2004. DOI: 10.1145/988672.988716.
- [2] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.
- [3] Leo Breiman et al. *Classification And Regression Trees*. Routledge, Oct. 2017. DOI: 10.1201/9781315139470.
- [4] Roy T. Fielding and Julian Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. June 2014. DOI: 10.17487/RFC7231. URL: <https://rfc-editor.org/rfc/rfc7231.txt>.
- [5] Jerome H. Friedman. “Greedy function approximation: A gradient boosting machine.” In: *The Annals of Statistics* 29.5 (Oct. 2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451.
- [6] Pierre Geurts, Damien Ernst, and Louis Wehenkel. “Extremely randomized trees”. In: *Machine Learning* 63.1 (Mar. 2006), pp. 3–42. DOI: 10.1007/s10994-006-6226-1.
- [7] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. IEEE Comput. Soc. Press, 1995. DOI: 10.1109/icdar.1995.598994.
- [8] Petr Máša and Tomáš Kočka. “Finding Optimal Decision Trees”. In: *Advances in Soft Computing*. Springer Berlin Heidelberg, pp. 173–181. DOI: 10.1007/3-540-33521-8_17.
- [9] Luis Meneses, Richard Furuta, and Frank Shipman. “Identifying “Soft 404” Error Pages: Analyzing the Lexical Signatures of Documents in Distributed Collections”. In: *Theory and Practice of Digital Libraries*. Springer Berlin Heidelberg, 2012, pp. 197–208. DOI: 10.1007/978-3-642-33290-6_22.
- [10] Scott O’Hara et al. *HTML 5.3*. W3C Working Draft. <https://www.w3.org/TR/2018/WD-html53-20181018/>. W3C, Oct. 2018.
- [11] Karl Pearson. “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (Nov. 1901), pp. 559–572. DOI: 10.1080/14786440109462720.
- [12] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

- [13] Liudmila Prokhorenkova et al. “CatBoost: unbiased boosting with categorical features”. In: (June 28, 2017). arXiv: <http://arxiv.org/abs/1706.09516v5> [cs.LG].
- [14] *Python Programming Language*. 2019. URL: <https://www.python.org>.
- [15] *Yandex Toloka*. 2019. URL: <https://toloka.yandex.com>.
- [16] *Yandex Video*. 2019. URL: <http://yandex.ru/video/>.
- [17] *YouTube Player API Reference for iframe Embeds*. May 2018. URL: https://developers.google.com/youtube/iframe_api_reference.