Бинарная модель востребованности контента и ее вероятностный прогноз

В. Г. Мосин

Аннотация

Исследована востребованность контента с точки зрения его разделения на классы с низкой и высокой долей востребованности. Построена классифицирующая модель. Показано, что использование вероятностного прогноза принадлежности к классу позволяет повысить эффективность модели по сравнению с дефолтным бинарным прогнозом.

Содержание

1 Введение

Прогнозирование востребованности контента на основе моделей машинного обучения имеет высокую актуальность в современных медиа, и этому есть несколько важных причин. Прежде всего, это увеличение конкуренции (см. [8]). С ростом количества контента, конкуренция за привлечение внимания аудитории становится все более жесткой. Точное прогнозирование востребованности контента позволяет создавать более привлекательный и интересный контент, который может более эффективно привлекать и удерживать пользователей.

Кроме того, прогнозирование востребованности контента позволяет оптимизировать бюджет и ресурсы, распределяя их наиболее эффективно. Зная, какой контент будет популярным, можно сосредоточиться на его производстве и продвижении, тем самым снизив затраты на контент, который может не найти достаточный отклик у аудитории.

Наконец, прогнозирование позволяет анализировать текущие тренды и предсказывать будущие. Это может быть полезным для создания нового контента, который будет актуальным в будущем, и для прогнозирования развития рынка контента в целом. Все эти факторы делают задачу прогнозирования востребованности контента на основе моделей машинного обучения весьма важной для создателей контента, медиа- и развлекательных компаний, маркетологов и пользователей (см. [2]).

1.1 Теоретическая часть

Задача бинарной классификации в машинном обучении заключается в отнесении объектов к одной из двух возможных категорий (см. [9], [10]). Это означает, что модель обучается разделить данные на два класса, условно положительные и отрицательные. В медиа-индустрии широкое распространение получила задача разделения позитивного или негативного отношения к тексту, комментарию, видеоролику и т. д.

Важной частью задачи бинарной классификации является оценка качества модели, которая может выполняться с помощью метрик, таких как точность, полнота, F-мера и мера AUC. Оценка позволяет измерить, насколько точно модель предсказывает классы, и насколько хорошо она может обобщать на новые данные.

В нашей работе мы используем метрику AUC, которая является одной из самых важных и информативных метрик в задаче бинарной классификации (см. [4], [5], [7]).

1.2 Постановка задачи

Имеются данные об объектах, в качестве которых выступают видеоролики, размещенные на канале одного из ведущих хостингов. Требуется 1) построить бинарную модель классификации, прогнозирующую востребованность контента среди посетителей и зрителей канала; 2) пользуясь метрикой AUC, оценить эффективность модели; 3) повысить ее эффективность путем определения оптимального порога бинаризации.

1.3 Библиотеки

Для проведения всех расчетов и визуализации результатов мы используем среду `Jupyter Notebook`, язык программирования `Python` и его библиотеки: `pandas`, `sklearn`, `matplotlib`.

2 Описание данных

Данные содержат 500 записей о видеороликах, расположенных на канале одного из ведущих хостингов, за период с ноября 2022 г. по ноябрь 2023 г. Каждый ролик описан при помощи 18 числовых признаков, таких как 'Среднее число просмотров одним пользователем', 'Время просмотра (часы)', 'Средний процент просмотра (%)' и др. Кроме того, отдельный девятнадцатый признак является бинарным строковым признаком и описывает востребованность конкретного ролика показателем 'Доля подписок' в формате 'Высокая'/'Низкая'.

3 Алгоритм

3.1 Чтение и предварительный анализ данных

Применяя метод `read_csv` библиотеки `pandas`, формируем дата-фрейм:

	Постоянные зрители	Новые зрители	Просмотры	...	Доля подписок
0	41	11	114	...	Низкая
1	29	5	79	...	Низкая
2	54	47	84	...	Высокая
...	...	...	...	...	...
498	35	24	102	...	Низкая
499	40	29	88	...	Низкая

При помощи метода `describe` библиотеки `pandas` выводим характеристики распределений числовых признаков:

	mean	std	min	max
Постоянные зрители	3.942000	7.925419	0.0000	56.0000
Новые зрители	32.586000	94.640204	0.0000	1100.0000
Просмотры	100.530000	170.646446	5.0000	1880.0000
Клики по элементам конечной заставки	0.508000	1.048874	0.0000	9.0000
Показы элементов конечной заставки	58.786000	79.566260	0.0000	792.0000
Показы тизеров	3.942000	7.925419	0.0000	56.0000
Среднее число просмотров одним пользователем	1.241784	0.171804	1.0000	2.6000
Уникальные зрители	79.674000	133.822347	3.0000	1465.0000
Средний процент просмотра	35.109640	13.616501	9.5200	74.5700
Отказались от подписки	0.010000	0.099598	0.0000	1.0000
Новые подписчики	0.360000	1.000200	0.0000	9.0000
Новые комментарии	0.056000	0.254939	0.0000	2.0000
Поделились	0.640000	1.811343	0.0000	17.0000
Отметки 'Не нравится'	0.064000	0.322664	0.0000	4.0000
Отметки 'Нравится'	1.486000	3.079085	0.0000	40.0000
Время просмотра (часы)	4.023387	7.663491	0.3026	90.9871
Показы	621.738000	797.378602	39.0000	8889.0000
CTR для значков видео (%)	5.969620	3.318488	0.0000	21.2400

И характеристики распределения признака 'Доля подписок':

	unique	top	freq
Доля подписок	2	Низкая	400

Здесь `unique` — количество уникальных значений, `top` — значение, обладающее максимальной частотой (модальное значение), `freq` — частота модального значения.

Мы видим, что числовые признаки имеют сильно различающийся разброс (на несколько порядков), а имеющиеся два класса строкового признака 'Доля подписок' не сбалансированы (представлены в отношении 4:1). Отметим, что несбалансированность классов является одной из причин, по которой мы среди прочих выбрали именно метрику AUC: эта метрика оценивает качество разделения на классы вне зависимости от пропорций, в которых состоят условно положительный и условно отрицательный классы.

3.2 Преобразование целевого признака в числовой формат

Целевая функция алгоритма классификации, который мы собираемся обучить и настроить — это признак 'Доля подписок', значения которого относятся к строковому типу: 'Низкая доля'/'Высокая доля'. Но алгоритмы классификации, в том числе и `KNeighborsClassifier` библиотеки `sklearn`, могут работать только с числовыми данными, а любая попытка применить классификатор к строковым записям приедет к сообщению об ошибке. Поэтому, прежде чем приступать к построению классификатора, нам нужно перевести строковые данные в числовой формат.

Мы применяем метод `loc` библиотеки `pandas` и осуществляем замену по условию в столбце 'Доля подписок', а именно: каждое строковое значение 'Низкая доля' мы заменяем нулем, а каждое значение 'Высокая доля' — единицей. Теперь все данные имеют числовой формат, причем, 18 признаков относятся к типу `float`, а целевой признак 'Доля подписок' — к типу `int`, и он принимает значения 0 или 1.

3.3 Разбиение данных на обучающую и тестовую выборки

Еще один предварительный шаг моделирования состоит в разбиении данных на обучающую и тестовую выборки, что является важнейшей практикой в машинном обучении. Следует отметить, что основная цель машинного обучения заключается в создании модели, которая может

обобщать знания на новые, ранее не встречавшиеся данные. Разделение данных на обучающую и тестовую выборки позволяет оценить, насколько хорошо модель справляется с обобщением. Модель обучается на тренировочной выборке и затем тестируется на тестовой выборке. Если модель хорошо справляется с предсказанием на тестовой выборке, то это указывает на ее способность обобщать. В целом, разбиение данных на обучающую и тестовую выборки является необходимой практикой в машинном обучении для обеспечения надежной оценки модели и предотвращения переобучения.

Чтобы разбить данные на обучающую и тестовую выборки мы используем метод `train_test_split` из модуля `model_selection` библиотеки `sklearn`. Мы разбиваем данные в пропорции 8:2, при этом 80% записей приписываем к обучающей выборке, а оставшиеся 20% — к тестовой выборке.

3.4 Построение и обучение модели классификации

Формируем объект `model`, относящийся к классу `KNeighborsClassifier`, используя в качестве параметра модели 20 ближайших соседей, после чего, применяя к объекту `model` метод `fit`, обучаем модель на обучающей выборке, используя в качестве разметки значения признака 'Доля подписок'.

3.5 Прогноз на основе обученной модели

Классификатор `KNeighborsClassifier` способен возвращать два типа прогноза при помощи методов `predict` и `predict_proba`. А именно:

- метод `predict` получает на вход объекты тестовой выборки и возвращает предсказанные метки классов, то есть 0 или 1, в зависимости от того, какой класс оказался в большинстве среди k ближайших соседей каждого из объектов;
- метод `predict_proba` тоже получает на вход объекты тестовой выборки, но возвращает не метки классов, а вероятности p_0 и p_1, где p_0 означает вероятность принадлежности объекта к классу 0, а p_1 — к классу 1.

Нам требуется именно вероятностный прогноз принадлежности к классу 1. Поэтому в качестве прогнозируемых значений классификатора далее мы будем использовать массив вероятностей, и, поскольку метод `predict_proba` возвращает двумерный массив с двумя вероятностями, мы специально указываем, что в качестве рабочих вероятностей мы бу-

дем использовать тот столбец возвращаемого двумерного массива, который соответствует положительному классу.

3.6 Построение ROC-кривой

Для построения ROC-кривой мы применяем метод `roc_curve` из модуля `metrics` библиотеки `sklearn`. Этот метод возвращает три массива:

1. массив p, который содержит все возникающие в классификаторе пороги бинаризации;
2. массив FPR, который содержит соответствующие порогам бинаризации величины False Positive Rate (подробнее о величине FPR см. [2]);
3. массив TPR, который содержит соответствующие порогам бинаризации величины True Positive Rate (подробнее см. [2]).

Затем при помощи модуля `pyplot` библиотеки `matplotlib`, строим ROC-кривую, используя массивы FPR и TPR, как горизонтальную и вертикальную переменные соответственно. Кроме того, мы используем метод `roc_auc_score` из модуля `metrics` библиотеки `sklearn`, который возвращает площадь под ROC-кривой, то есть, значение метрики AUC, и получаем кривую вместе с указанием ее меры (см. рис. 1(a)).

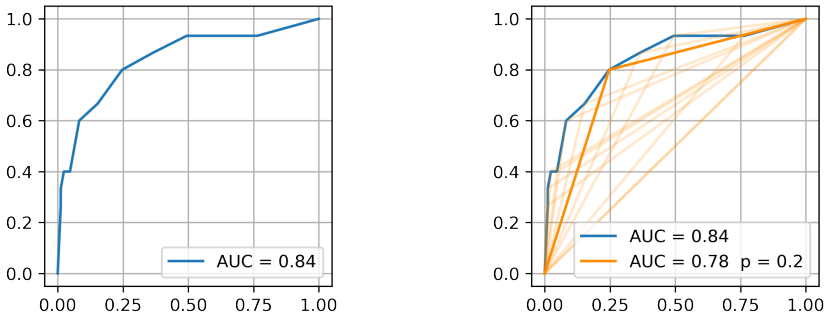

Рис. 1: ROC-кривая (a) и подбор порога бинаризации (b)

3.7 Вычисление оптимального порога бинаризации

В случае бинарного прогноза ROC-кривая представляет собой ломаную линию из двух звеньев, которая характеризуется единственной точкой внутри единичного квадрата, а метрика AUC вычисляется как площадь под этой линией. То есть для каждого порога бинаризации получается своя бинарная ROC-кривая со своим значением метрики AUC. Запуская

цикл по всем значениям из массива p, получаем серию таких характеристик и выбираем тот порог бинаризации, который отвечает максимальному значению метрики AUC (см. рис. 1(b)).

3.8 Повторный запуск

Напомним, что прежде чем начинать исследование, мы произвели случайное разбиение набора данных на обучающую и тестовую выборки. Эффект случайности неизбежно проявится, если мы заново разобьём данные: возникнут новые соотношения между представителями классов (особенно, если учесть, что классы не сбалансированы), получатся какие-то другие значения метрики AUC и порога бинаризации p.

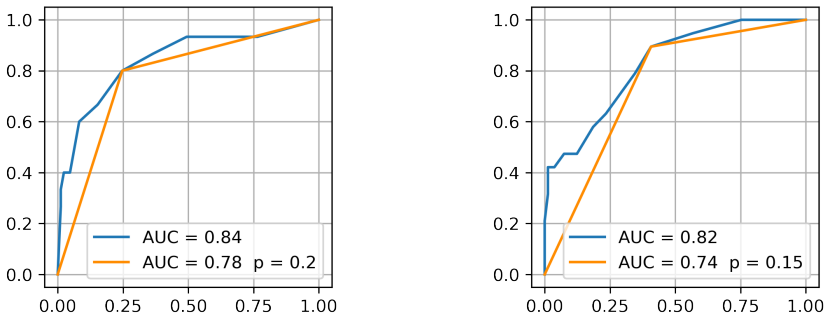

Рис. 2: Первый (a) и повторный (b) запуски

Мы производим повторный запуск алгоритма и при новом случайном разбиении действительно получаем другой результат (см. рис. 2(b)). Мы видим, что при повторном запуске порог бинаризации принимает значение 0.15, а не 0.2, как это было первоначально, и кроме того, не приводя подробных иллюстраций, просто отметим, что при следующих повторных запусках порог бинаризации варьируется случайным образом в широком диапазоне от 0.1 до 0.35.

3.9 Усреднение ROC-кривой на множестве запусков

Чтобы избежать эффекта случайности, производим 250 запусков и усредняем все возникающие при каждом запуске характеристики (см. рис. 3):

1. ROC-кривую,
2. метрику AUC для вероятностного прогноза,
3. ROC-кривую, получающуюся в результате оптимальной бинаризации,

4. метрику AUC для бинарного прогноза,
5. порог вероятности p, на котором происходит оптимальная бинаризация.

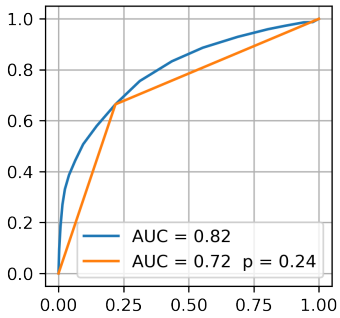

Рис. 3: Усреднение на множестве запусков

После этого в результате усреднения порог бинаризации стабилизируется примерно на уровне $p = 0.24$.

3.10 Сравнение оптимизированного прогноза с дефолтным

Теперь у нас есть возможность сравнить прогноз, который получается после оптимальной бинаризации вероятностей, с дефолтным прогнозом, который получается прямым голосованием ближайших соседей. Мы сравниваем по метрике AUC оптимизированный прогноз с дефолтным прогнозом метода `predict` классификатора `KNeighborsClassifier`, по следующей схеме:

1. запускаем случайное разбиение данных на обучающую и тестовую выборки;
2. вызываем метод `predict` и получаем дефолтный бинарный прогноз;
3. вызываем метод `predict_proba` и получаем вероятностный прогноз;
4. находим порог бинаризации вероятностного прогноза, который отвечает максимальному значению метрики AUC , так же, как это делалось на шаге 3.8;
5. выполняем бинаризацию вероятностного прогноза на найденном пороге;
6. пользуясь методом `auc_score`, находим значения метрики AUC для дефолтного и бинаризированного прогнозов;

7. строим все ROC-кривые, отмечаем все метрики AUC и выводим порог бинаризации p (см. рис. 4 (a)).

Мы видим, что метрика AUC для вероятностного прогноза (синяя линия) равна 0.81. Для бинарного прогноза, который получается после бинаризации вероятностного (желтая линия), метрика AUC равна 0.78. При этом оптимальный порог бинаризации $p = 0.15$. Для дефолтного бинарного прогноза (желтая пунктирная линия) метрика AUC равна 0.62. Затем производим повторный запуск при новом случайном разбиении данных на обучающую и тестовую выборки (см. рис. 4 (b)).

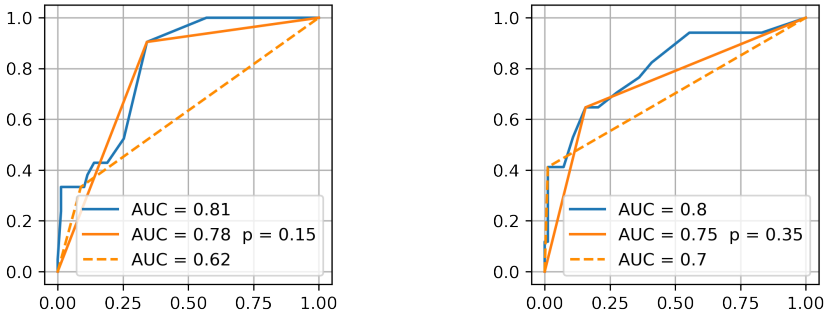

Рис. 4: Сравнение метрик при первом (a) повторном (b) запуске

Теперь для вероятностного прогноза метрика AUC равна 0.8, что дает незначительное отклонение от первого запуска. Для бинарного прогноза после бинаризации метрика AUC равна 0.75, и это тоже небольшое отклонение. Для дефолтного бинарного прогноза метрика AUC равна 0.7, то есть, отклонение опять оказывается незначительным. Вместе с тем, порог бинаризации p дает огромное отклонение: при повторном запуске его значение составляет 0.35 вместо бывшего при первом запуске значения 0.15.

3.11 Сравнение оптимизированного прогноза с дефолтным на множестве запусков

Чтобы избежать эффекта случайности при разбиении данных, мы производим 250 запусков и усредняем все возникающие при этом показатели так же, как это было сделано на шаге 3.10 (см. рис. 5). Отметим, что, строго говоря, 250 запусков — это излишнее количество. Вполне хватило бы и 100. Мы использовали столь больше число только для того, чтобы на графике получилась гладкая ROC-кривая, но если не учитывать визуальную эстетику, меньшее чосло запусков приводит к томуже оптимальному порогу бинаризации.

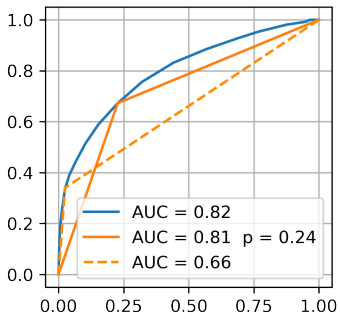

Рис. 5: Обе метрики на множестве запусков

При этом стабилизируются все показатели, а самое главное — стабилизируется разность между дефолтной и оптимизированной метриками AUC (примерно на уровне $0.81 - 0.66 = 0.15$) и порог бинаризации (примерно на уровне $p = 0.24$).

4 Результаты

Напомним, что в качестве показателя востребованности контента мы взяли бинарный признак 'Доля подписок', принимающий два возможных значения: 'Низкая', 'Высокая'. Используя это признак в качестве целевой функции, мы построили модель классификации объектов (видеороликов канала), позволяющую прогнозировать востребованность объектов, то есть, относить новый, не имеющий априорной классификации объект к одному из двух типов: a) объект с низкой долей подписок, b) объект с высокой долей подписок.

Целью нашей работы было получение бинарного прогноза из вероятностного путем выбора оптимального порога бинаризации и сравнение оптимизированного прогноза с бинарным дефолтным прогнозом. В итоге мы продемонстрировали достоверный сдвиг в сторону увеличения метрики AUC при переходе от дефолтного к оптимизированному прогнозу.

5 Выводы

При изучении востребованности контента вероятностный прогноз классификации дает более тонкий инструментарий по сравнению с дефолтным бинарным прогнозом. Он позволяет проводить более детальное разделение объектов на классы и тем самым, точнее прогнозировать востребованность контента на информационном канале.

6 Литература

1. Хейдт М. Изучаем Pandas / М. Хейдт; — Москва: ДМК Пресс, 2018. — 438 с.
2. Бурков А. Машинное обучение без лишних слов / А. Бурков; — СПб: Питер, 2020. — 192 с.
3. Николенко С. Глубокое обучение. Погружение в мир нейронных сетей / С. Николенко, А. Кадурин, Е. Архангельская; — СПб: Питер, 2018. — 481 с.
4. Лимановская, О.В. Основы машинного обучения : учебное пособие / О.В. Лимановская, Т.И. Алферьева; — Екатеринбург : Изд-во Урал. ун-та, 2020. — 88 с.
5. Шолле, Ф. Глубокое обучение на Python / Ф. Шолле; — СПб.: Питер, 2018. — 400 с.
6. Вьюгин, В. В. Математические основы теории машинного обучения и прогнозирования / В. В. Вьюгин; — М.: МЦИМО. — 2013. — 387 с.
7. Бринк Х. Машинное обучение / Х. Бринк, Дж. Ричардс, М. Феверолф — СПб.: Питер, 2017. — 336 с.
8. Дьяконов А. Г. Прогноз поведения клиентов супермаркетов с помощью весовых схем оценок вероятностей и плотностей / А. Г. Дьяконов // Бизнес-информатика. — 2014. Т. 1, № 27. — С. 68—77.
9. Михеев, А. В. Решение задач классификации методами машинного обучения / А. В. Михеев // Молодой ученый. — 2021. — № 21 (363). — С. 107–110.
10. Неделько В. М. Исследование эффективности некоторых линейных методов классификации на модельных распределениях // В. М. Неделько // Машинное обучение и анализ данных. — 2016. Т. 2, №3. — С. 305—329.