Сравнительный анализ двух методик поиска наиболее значимых предикторов регрессионной модели

В. Г. Мосин

Аннотация

В статье сравниваются два подхода к определению значимых признаков регрессионной задачи: 1) основанный на попарных одномерных регрессиях, и 2) основанный на коэффициентах множественной регрессии. Приводятся алгоритмы поиска значимых признаков для обоих методов, которые применяются к данным о потреблении контента пользователями одного из ведущих хостингов. Результаты анализируются, указываются сильные и слабые стороны каждой из методик.

Содержание

1 Введение

В области машинного обучения и анализа данных, определение значимых признаков является ключевым этапом в решении задач. Признаки, или переменные, представляют собой информацию, которая используется для предсказания или классификации. Однако не все признаки равнозначны и оказывают одинаковое влияние на результаты модели.

Определение значимых признаков имеет несколько важных причин. Во-первых, это помогает улучшить эффективность модели. Учитывая, что в большинстве задач машинного обучения доступно обилие данных и признаков, модель может быть склонна к переобучению. Переобучение происходит, когда модель слишком сильно адаптируется к обучающим данным, что может привести к плохим результатам при работе с новыми данными. Определение значимых признаков позволяет сократить количество признаков, используемых в модели, устраняя шумовые переменные, что ведет к снижению риска переобучения.

Во-вторых, определение значимых признаков может помочь улучшить интерпретируемость модели. В современном мире все более актуальной становится проблема объяснимости и понимания решений, принимаемых моделями машинного обучения. Определение значимых признаков позволяет выявить наиболее важные факторы, влияющие на результаты модели, что делает ее выводы более понятными для людей. Более простая и интерпретируемая модель может быть полезной не только для исследователей и разработчиков, но и для бизнеса, позволяя принимать взвешенные решения на основе результатов модели.

Наконец, определение значимых признаков позволяет улучшить скорость работы модели. Чем меньше признаков используется при обработке данных, тем быстрее модель сможет выдавать результаты. Ускорение работы модели полезно, особенно в сферах, где требуется обработка больших объемов данных в режиме реального времени.

1.1 Теоретическая часть

В настоящем исследовании мы будем рассматривать значимость признаков с точки зрения решения регрессионной задачи.

Допустим, имеется целевая функция y и некоторый набор признаков $x_1, \dots x_n$. Нам нужно спрогнозировать значений целевой функции при помощи линейной регрессии относительно признаков, но при этом мы хотим использовать не все признаки, а ограничиться их минимальным набором, который позволит нам достичь приемлемого уровня качества нашей модели. Другими словами, нам нужно выделить из всего набора признаков поднабор, который гарантировал бы нам построение хорошей модели. Признаки такого поднабора мы будем считать значимыми признаками.

Мы будем применять два подхода к определению значимых признаков: 1) на основании попарных регрессий, 2) на основании коэффициентов множественной регрессии.

1.1.1 Попарные одномерные регрессии

Идея этого метода состоит в том, чтобы выяснить, какой из признаков лучше всего аппроксимирует целевую функцию в виде одномерной линейной регрессии, в которой участвует только этот признак, а все остальные не участвуют. Для этого строится серия из n одномерных моделей

$$y = a_i x_i + b_i,$$

для каждой из которых вычисляется коэффициент детерминации R2i, выступающий в качестве метрики ее качества. Чем выше метрика, тем более значимым является соответствующий признак, и таким образом,

возникает упорядоченный список всех признаков по степени их значимости.

1.1.2 Коэффициенты множественной регрессии

В этом подходе все признаки используются в совокупности. Строится модель множественной регрессии относительно всех признаков:

$$y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b,$$

после чего более значимыми считаются те признаки, коэффициенты при которых больше по абсолютной величине (так как коэффициенты модели могут быть как положительными, так и отрицательными). После этого, так же как и в пункте 1.1.1, возникает возможность упорядочить признаки по степени их значимости.

1.1.3 Замечание о мультиколлинеарности

Иногда в поведении линейной регрессии наблюдается парадоксальный эффект (см. [5]), который позволяет предикторам обладать огромными коэффициентами, но при этом не влиять на точность модели после их удаления.

Одним из возможных объяснений этого феномена является мультиколлинеарность — в ситуации, когда группа предикторов является почти линейно зависимой, модель может обнаружить, что они обладают огромными коэффициентами, причем их величина никак не связана с их значимостью в смысле п. 1.1.2. Поэтому при использовании такого подхода требуется дополнительное исследование данных на предмет мультиколлинеарности.

1.2 Постановки задачи

1.2.1 Предмет исследования

Предметом нашего исследования являются две методики определения значимых предикторов регрессионной задачи: 1) методика, основанная на попарных одномерных регрессиях, и 2) методика, основанная на коэффициентах множественной регрессии.

1.2.2 Методика исследования

Для сравнения этих двух подходов к определению значимых признаков, мы реализуем оба похода в виде двух алгоритмов, в результате чего получаем два упорядоченных списка признаков (порядок следования признаков в них, разумеется, различен). Далее, добавляя в регрессионную

задачу признаки в порядке убывания их значимостей, мы наблюдаем, с какой скоростью растет качество модели, которое мы оцениваем при помощи коэффициента детерминации.

1.2.3 Цель исследования

Выяснить, какая из методик определения значимых признаков лучше справляется со своей задачей, а также установить специфические ограничения условий для их применения.

1.3 Библиотеки

Для выполнения вычислений и анализа данных мы пользуемся средой `Jupyter Notebook`, которая предоставляет удобные средства для работы с языком программирования Python и его главными библиотеками: `NumPy`, `Pandas`, `sklearn` и `matplotlib`. Благодаря этим инструментам, мы можем эффективно работать с данными, выполнять исследования и визуализировать результаты (см. [1], [2]).

2 Описание и предварительная обработка данных

В настоящей работе мы используем данные о потреблении контента пользователями одного из ведущих хостингов. Данные содержат 500 записей, индексированных датами с 2021-08-20 по 2023-01-01 и описанных признаками 'Дата', 'Просмотры', 'Время просмотра (часы)' и т. д. Применяем метод `read_csv` библиотеки `pandas`, результат записываем в переменную `df`. Полученный дата-фрейм имеет следующий вид:

	Дата	Просмотры	Поделились	...	Лайки
0	2023-01-01	475.0	9.0	...	16.0
1	2022-12-31	174.0	1.0	...	4.0
2	2022-12-30	490.0	3.0	...	3.0
...	...	...	...	...	...
498	2021-08-21	222.0	0.0	...	4.0
499	2021-08-20	209.0	0.0	...	1.0

Один признак относится к строковому типу, остальные являются числовыми с плавающей запятой, причем, признак 'Дата' нам не понадобится. В теле алгоритмов мы будем работать исключительно с числовыми

показателями. Поэтому, при помощи метода `drop` библиотеки `pandas`, мы исключаем строковый признак 'Дата' из датафрейма. После этого применяем к датафрейму `df` метод `describe` библиотеки `pandas` и получаем его статистики:

	min	mean	max	std
Просмотры	159.00	936.51	2200.00	418.144
Время просмотра (часы)	5.48	37.17	96.72	16.64
Поделились	0.00	6.96	71.00	6.25
Постоянные зрители	30.00	163.34	463.00	78.90
Новые комментарии	0.00	0.53	6.00	0.83
Отказались от подписки	0.00	2.77	29.00	2.55
Новые подписчики	0.00	6.49	19.00	3.51
Новые зрители	60.00	366.83	735.00	174.01
Среднее число просмотров одним пользователем	1.31	1.79	2.85	0.21
Уникальные зрители	96.00	530.18	1103.00	239.22
CTR для значков видео (%)	1.25	5.54	8.52	1.11
Показы	1938.00	8093.78	39479.00	3816.08
Подписчики	0.00	3.72	15.00	4.02
Средний процент просмотра (%)	18.68	26.72	41.29	3.41
Процент лайков	0.00	92.02	100.00	10.31
Средняя продолжительность просмотра	96.07	144.33	211.02	15.66
Дизлайки	0.00	1.28	10.00	1.34
Лайки	0.00	15.80	70.00	9.13

Значения признаков сильно разбалансированы. Например, признаки 'Показы' и 'Лайки' отличаются друг от друга на несколько порядков. Для устранения дисбаланса мы проводим нормализацию данных:

```
df = (df - df.mean())/df.std()
```

после чего все признаки центрируются и описываются величинами примерно одного порядка.

	min	mean	max	std
Просмотры	−1.85	0.00	3.02	1.00
Время просмотра (часы)	−1.90	0.00	3.57	1.00
Поделились	−1.11	0.00	10.23	1.00
Постоянные зрители	−1.69	0.00	3.79	1.00
Новые комментарии	−0.64	0.00	6.57	1.00
Отказались от подписки	−1.08	0.00	10.27	1.00
Новые подписчики	−1.84	0.00	3.55	1.00
Новые зрители	−1.76	0.00	2.11	1.00
Среднее число просмотров одним пользователем	−2.23	0.00	4.92	1.00
Уникальные зрители	−1.81	0.00	2.39	1.00
CTR для значков видео (%)	−3.84	0.00	2.67	1.00
Показы	−1.61	0.00	8.22	1.00
Подписчики	−6.63	0.00	2.80	1.00
Средний процент просмотра (%)	−2.35	0.00	4.26	1.00
Процент лайков	−8.92	0.00	0.77	1.00
Средняя продолжительность просмотра	−3.08	0.00	4.25	1.00
Дизлайки	−0.95	0.00	6.49	1.00
Лайки	−2.38	0.00	5.93	1.00

Данные подготовлены для их использования в наших алгоритмах.

3 Алгоритм на попарных одномерных регрессиях

Алгоритм будет состоять из двух частей. В первой части мы упорядочим признаки датафрейма возрастанию их значимости для целевой функции. А затем будем вычислять накопительный эффект от добавления в многомерную регрессионную модель признаков, начиная с наиболее значимых.

3.1 Сортировка по значимости

Сначала мы будем строить одномерные регрессионные модели относительно целевой функции 'Просмотры', используя поочередно в качестве предиктора один из признаков датафрейма `df`, а коэффициенты детерминации этих моделей заносить в отдельный список `list_of_score`. Для этого проведем цикл по всем столбцам датафрейма `df`, кроме 'Просмотры', в качестве счетчика цикла будем использовать имя столбца.

3.1.1 Стартовые значения переменных цикла

Определим массив `cols` как набор имен столбцов датафрейма `df` (исключая имя целевой функции 'Просмотры'):

```
cols = df.drop(columns = ['Просмотры']).columns
```

Заведем датафрейм `d`, индексированный списком `cols`:

```
d = pd.DataFrame(index = cols)
```

До исполнения алгоритма это пустой датафрейм, в котором присутствует только индексный столбец. После исполнения цикла мы запишем в него значения соответствующих коэффициентов детерминации.

Наконец, создадим список `list_of_cols`, в который будем записывать получающиеся в цикле коэффициенты детерминации:

```
list_of_cols = []
```

На старте цикла это пустой список.

3.1.2 Первая итерация цикла

Под именем `col` выбираем первый элемент из списка имен `cols`, выделяем из датафрема `df` столбец `df[col]`, методом `to_numpy` из библиотеки `pandas` переводим его в одномерный массив и преобразуем этот массив в столбец при помощи метода `reshape`:

```
X = df[col].to_numpy().reshape(-1, 1)
```

Результат присваиваем переменной `X`, как традиционно обозначают левую часть регрессионной модели. В качестве правой части `y` выбираем столбец `df['Просмотры']`:

```
y = df['Просмотры']
```

При помощи метода `LinearRegression` из модуля `linear_model` библиотеки `sklearn` создаем объект `model` и при помощи метода `fit` обучаем его на паре `(X, y)`:

```
model = skl.LinearRegression()
model.fit(X, y)
```

При помощи метода `score` вычисляем коэффициент детерминации модели `model` и при помощи метода `append` заносим его в список `list_of_score`:

```
list_of_score.append(model.score(X, y))
```

В списке коэффициентов детерминации появляется первый элемент.

3.1.3 Цикл по именам столбцов

Присваиваем переменной `col` следующее имя из списка имен `cols` и повторяем до исчерпания списка `cols`. На выходе из цикла получаем заполненный список, содержащий коэффициенты детерминации одномерных регрессий, которые связывают тот или иной признак датафрейма `df` с целевой функцией 'Просмотры'.

3.1.4 Сортировка признаков

Заносим полученный список коэффициентов детерминации в датафрем `d` и сортируем его по возрастанию при помощи метода `sort_values` библиотеки `pandas`:

```
d['score'] = list_of_score
d.sort_values('score')
```

Результат представлен в таблице ниже. Левая часть содержит имя столбца датафрейма `df`, который использовался в качестве единственного предиктора одномерной регрессионной модели, правая часть — соответствующий коэффициент детерминации.

	score
Процент лайков	0.000026
Отказались от подписки	0.001135
Среднее число просмотров одним пользователем	0.005225
Средняя продолжительность просмотра	0.030673
Средний процент просмотра (%)	0.070686
Новые комментарии	0.089768
Подписчики	0.098139
Дизлайки	0.195296

	score
Поделились	0.254338
Новые подписчики	0.302298
CTR для значков видео (%)	0.399495
Показы	0.405764
Лайки	0.428562
Постоянные зрители	0.841800
Новые зрители	0.848462
Уникальные зрители	0.939054
Время просмотра (часы)	0.941681

Мы видим, что наименее значимым в методе попарных одномерных регрессий является признак 'Процент лайков', а наиболее значимым — признак 'Время просмотра (часы)'.

3.2 Накопительный эффект

Одномерная модель, в которой присутствует единственный предиктор 'Время просмотра (часы)' обладает коэффициентом детерминации 0.94. Если добавить в модель следующий по значимости предиктор 'Уникальные зрители', коэффициент детерминации увеличится.

Идея метода попарных регрессий состоит в том, что список предикторов не нужно использовать полностью: достаточно будет нескольких наиболее значимых признаков. В этом блоке алгоритма мы последовательно в цикле будем добавлять предикторы в модель согласно порядку их значимостей и вычислять коэффициент детерминации получающейся многомерной модели.

3.2.1 Упорядоченный список столбцов

Пользуясь методом `index` библиотеки `pandas`, мы выделяем индексный столбец отсортированного датафрейма `d.sort_values('score')`, после чего переводим его в список:

```
ordered_list_of_cols = list(d.sort_values('score').index)
```

3.2.2 Срез последних элементов упорядоченного списка

Пользуясь приемом среза списка, мы можем получить список последних `i` элементов списка `ordered_list_of_cols`:

```
ordered_list_of_cols[len(ordered_list_of_cols) - i:]
```

Для i=1 это будет последний элемент списка (то есть, имя наиболее значимого признака), для i=2 — два последних элемента (то есть, имена двух наиболее значимых признаков) и. т. д. вплоть до исчерпания всего списка.

3.2.3 Список коэффициентов детерминации

Нам снова понадобится список коэффициентов детерминации, для того чтобы заносить в него результаты. До исполнения цикла мы объявляем этот список пустым:

```
list_of_score = []
```

3.2.4 Стартовая итерация цикла

Присваиваем стартовое значение счетчику i=1 и выделяем из датафрема df подфрейм, содержащий только i наиболее значимых столбцов:

```
X = df[ordered_list_of_cols[len(ordered_list_of_cols) - i:]]
```

Далее следует учесть специфику интерпретации левой части регрессионной задачи в библиотеке sklearn: если левая часть X представляет собой одномерный массив, то он интерпретируется как строка, и его нужно перевести в столбец (заметим, что одномерная ситуация возникает только в стартовой итерации цикла; во второй итерации массив X будет уже двумерен, в третьей — трехмерен и т. д.), если же он многомерен, то этого делать не нужно, потому что он интерпретируется как несколько столбцов.

```
if len(X.columns) == 1:
    X = X.to_numpy().reshape(-1, 1)
else:
    X = X.to_numpy()
```

Здесь, пользуясь методом to_numpy библиотеки pandas, мы перевели датафрейм X в массив numpy, причем для одномерной ситуации преобразовали его в столбец методом reshape. В качестве правой части регрессионной задачи используем признак 'Просмотры':

```
y = df['Просмотры'].to_numpy()
```

Теперь, когда левая и правая части регрессионной задачи сформированы, мы, пользуясь методом LinearRegression из модуля linear_model библиотеки sklearn, создаем объект model и, пользуясь методом fit, обучаем его на паре (X, y):

```
model = skl.LinearRegression()
model.fit(X, y)
```

В конце итерации при помощи метода `score` мы вычисляем коэффициент детерминации обученной модели, после чего заносим результат в список коэффициентов детерминации, применив для этого метод `append`:

```
list_of_score.append(model.score(X, y))
```

3.2.5 Цикл по убыванию значимости

Увеличиваем значение счетчика `i` на 1 и повторяем до полного исчерпания списка `ordered_list_of_cols`. После исполнения цикла список `list_of_score` представляет собой возрастающую последовательность коэффициентов детерминации, которые получаются при увеличении размерности регрессионной задачи. Результат представлен в разделе ниже (см. рис. 1).

4 Алгоритм на коэффициентах множественной регрессии

Другой подход в выделении значимых признаков состоит в анализе регрессионных коэффициентов множественной линейной регрессии, когда для прогнозирования целевой функции используются сразу все признаки в качестве предикторов.

4.1 Исключение мультиколлинеарности

Сначала мы просто формируем регрессионную задачу на всех признаках и анализируем регрессионные коэффициенты.

4.1.1 Регрессионная задача на полных данных

Как обычно, пользуясь методом `drop` библиотеки `pandas`, исключаем целевую функцию и переводим датафрейм в `numpy`, для того чтобы получить левую часть регрессионной задачи `X`. В качестве правой части выбираем столбец целевой функции и тоже переводим его в `numpy`:

```
X = df.drop(columns = ['Просмотры']).to_numpy()
y = df['Просмотры'].to_numpy()
```

Затем создаем объект `model` и обучаем его:

```
model = skl.LinearRegression()
model.fit(X, y)
```

4.1.2 Список регрессионных коэффициентов

Далее нас будут интересовать коэффициенты при предикторах модели. Мы заводим пустой датафрейм `d`, индексированный именами столбцов датафрейма `df` (без столбца целевой функции):

```
d = pd.DataFrame(index = cols)
```

Методом `coef_` библиотеки `sklearn` получаем список коэффициентов регрессионной задачи и заносим его в датафрейм `d` в столбец с именем `'coef'`. Затем добавляем столбец из абсолютных значений коэффициентов с именем `'abs_coef'` и, наконец, пользуясь методом `sort_values`, упорядочиваем датафрейм `d` по возрастанию абсолютных величин регрессионных коэффициентов:

```
d['coef'] = model.coef_
d['abs_coef'] = abs(d['coef'])
d.sort_values('abs_coef')
```

Результат представлен в таблице ниже:

	coef	abs_oef
Новые комментарии	−2.053491e−03	2.053491e−03
Лайки	3.557555e−03	3.557555e−03
Поделились	3.737851e−03	3.737851e−03
Средний процент просмотра (%)	9.400817e−03	9.400817e−03
CTR для значков видео (%)	1.474809e−02	1.474809e−02
Процент лайков (%)	−1.546911e−02	1.546911e−02
Дизлайки	−2.568136e−02	2.568136e−02
Показы	5.321350e−02	5.321350e−02
Среднее число просмотров одним пользователем	8.737729e−02	8.737729e−02
Средняя продолжительность просмотра	−1.101624e−01	1.101624e−01
Время просмотра (часы)	4.952706e−01	4.952706e−01
Постоянные зрители	2.315735e+00	2.315735e+00
Новые зрители	4.898142e+00	4.898142e+00
Уникальные зрители	−6.366331e+00	6.366331e+00
Новые подписчики	−7.301521e+11	7.301521e+11
Отказались от подписки	1.007574e+12	1.007574e+12

	coef	abs_oef
Подписчики	1.173130e+12	1.173130e+12

Мы видим, что величина коэффициентов при последних трех признаках аномально высока: 10^{12} при признаках 'Подписчики' и 'Отказались от подписки' и 10^{11} при признаке 'Новые подписчики'.

4.1.3 Семантический анализ мультиколлинеарности

Причина такой аномалии в мультиколлинеарности этих трех признаков (см. [5], [6]). Если учесть их семантическое содержание, получим следующее:

1. 'Новые подписчики' — число людей, подписавшихся на канал в течение конкретной даты;
2. 'Отказались от подписки' — число людей отписавшихся от канала в течение той же даты;
3. 'Подписчики' — общее количество подписок, которое приобрел канал в течение этой даты.

Следовательно, налицо очевидная связь:

```
'Подписчики' = 'Новые подписчики' - 'Отказались от подписки'
```

Еще интереснее следующее наблюдение, которое не является столь очевидным. Давайте рассмотрим следующую тройку в этом списке: 'Уникальные зрители', 'Новые зрители', 'Постоянные зрители'. Значения их регрессионных коэффициентов не аномальны. Но обратим внимание на их семантику:

1. 'Новые зрители' — число людей, впервые попавших на канал;
2. 'Постоянные зрители' — число людей, уже посещавших канал;
3. 'Уникальные зрители' — общее количество зрителей в течение конкретной даты.

И значит, есть еще одна связь:

```
'Уникальные зрители' = 'Постоянные зрители' + 'Новые зрители'
```

Дальнейший семантический анализ не приводит к новым мультиколлинеарностям.

4.1.4 Удаление мультиколлинеарных признаков

Чтобы исключить эффект мультиколлинеарности, достаточно из блока мультиколлинеарных признаков удалить какой-нибудь один (любой)

признак. Мы удалим из первого блока признак 'Подписчики', а из второго — признак 'Новые зрители'.

4.1.4. Регрессионная задача на очищенных данных.

Возвращаемся к началу процесса и в качестве левой части регрессионной задачи принимаем:

```
X = df.drop(columns = ['Просмотры', 'Подписчики',
                       'Новые зрители']).to_numpy()
```

Затем полностью воспроизводим все шаги.

4.1.5 Очищенный список коэффициентов

Результаты в таблице ниже:

	coef	abs_oef
Новые подписчики	0.003072	0.003072
Новые комментарии	−0.003207	0.003207
Поделились	0.003887	0.003887
Лайки	0.004111	0.004111
Средний процент просмотра (%)	0.008487	0.008487
CTR для значков видео (%)	0.014451	0.014451
Процент лайков (%)	−0.015613	0.015613
Дизлайки	−0.025699	0.025699
Отказались от подписки	−0.025721	0.025721
Показы	0.052395	0.052395
Постоянные зрители	0.075550	0.075550
Среднее число просмотров одним пользователем	0.087846	0.087846
Средняя продолжительность просмотра	−0.109265	0.109265
Уникальные зрители	0.410871	0.410871
Время просмотра (часы)	0.495632	0.495632

4.2 Накопительный эффект

Сейчас мы находимся в стадии шага 3.2 алгоритма попарных одномерных регрессий: мы точно так же имеем упорядоченный список признаков, правда, он теперь упорядочен по другому принципу, и в нем другое

число элементов (два признака мы удалили для исключения эффекта мультиколлинеарности).

Возвращаемся к шагу 3.2.1, точно так же создаем список имен столбцов, упорядоченных по возрастанию их значимости:

```
ordered_list_of_cols = list(d.sort_values('abs_score').index)
```

после чего буквально воспроизводим все действия шагов 3.2.1–3.2.5. Получаем список, в котором последовательно записаны коэффициенты детерминации, возникающие при добавлении в регрессионную задачу новых признаков, начиная с конца списка `ordered_list_of_cols`.

5 Результаты

Наша задача состояла в том, чтобы сравнить скорость двух накопительных эффектов в качестве регрессионной модели:

1. по алгоритму попарных одномерных регрессий,
2. по алгоритму коэффициентов множественной регрессии.

Последовательности коэффициентов детерминации, которые получаются в результате применения модели после добавления следующего по значимости предиктора, мы вывели в два списка. Закодируем их именами `a1` (алгоритм первый, построенный на попарных одномерных регрессиях) и, соответственно, `a2` (алгоритм второй, построенный на коэффициентах множественной регрессии). Длины списков различны, поскольку в ходе работы со вторым алгоритмом мы удалили два признака, чтобы избавиться от мультколлинеарности.

Два алгоритма возвращают значимые признаки в разной последовательности. Но первый, самый значимый признак у них один и тот же: это 'Время просмотра (часы)'. Таким образом, при добавлении в регрессионную модель первого по значимости признака, получаются равные коэффициенты детерминации. Удивительно, но и второй по значимости признак одинаков для обоих алгоритмов: это 'Уникальные зрители'. Поэтому при добавлении первых двух по значимости признаков тоже получаются равные оценки качества модели. Но вот третьи позиции в списках значимых признаков различны. Для алгоритма `a1`, построенного на попарных одномерных регрессиях, третий по значимости признак — это 'Новые зрители'. Для алгоритма `a2`, построенного на коэффициентах множественной регрессии — это 'Средняя продолжительность просмотра'. И именно на третьей позиции кривые накопленной оценки качества модели начинают расходиться (см. рис. 1).

Алгоритм `a1` приближается к значениям алгоритма `a2`, которые получаются после добавления первых трех по значимости признаков, лишь после добавления 14 признаков.

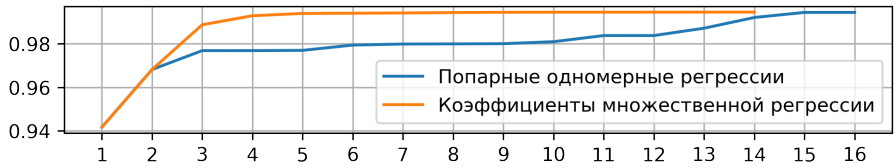

Рис. 1: Кривые оценок для двух алгоритмов

6 Выводы и замечания

Разумеется, алгоритм `a2`, построенный на коэффициентах множественной регрессии, эффективнее алгоритма `a1`, построенного на попарных одномерных регрессиях. Однако у обоих алгоритмов есть свои плюсы, и есть свои минусы.

6.1 Попарные одномерные регрессии

Очевидный минус — он не столь эффективно выделяет действительно значимые признаки по сравнению со своим конкурентом. Но есть и плюс: его можно использовать «вслепую», не апеллируя к семантике признаков и знанию предметной области.

6.2 Коэффициенты множественной регрессии

Очевидный плюс — он возвращает меньшее (причем, значительно меньшее по сравнению со своим конкурентом) число значимых признаков, за счет которых добивается максимально возможного результата. Но есть и минус: если в данных зашита мультиколлинеарность, он не справляется со своей задачей и требует вмешательства аналитика, представляющего семантику признаков и знающего предметную область.

В целом можно рекомендовать к использованию оба метода. Каждый из них может найти свое применение в зависимости от конкретной ситуации и специфики данных.

7 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. Вьюгин, В. В. Математические основы теории машинного обучения и прогнозирования / В. В. Вьюгин; — М.: МЦИМО. — 2013. — 387 с.

4. Бринк Х. Машинное обучение / Х. Бринк, Дж. Ричардс, М. Феверолф — СПб.: Питер, 2017. — 336 с.
5. Кузнецова И.С., Белозерских А.В. Проблемы мультиколлинеарности в регрессионных моделях / Актуальные направления научных исследований XXI века: теория и практика. 2015. Т. 3. № 8—3 (19—3). С. 308—312.
6. Орлова И.В. Подход к решению проблемы мультиколлинеарности с помощью преобразования переменных / Фундаментальные исследования. 2019. № 5. С. 78–84.