

**А. С. Лебедев**

студент Белорусского государственного экономического университета

## **АНАЛИЗ И ПРЕДСКАЗАНИЕ ЭФФЕКТИВНОСТИ ПРОМОРАССЫЛОК В КОМПАНИИ «ДОДО ПИЦЦА»**

### **Введение**

Достижение эффективности проморассылок является важной задачей для большинства компаний, стремящихся увеличить свою клиентскую базу и повысить лояльность существующих клиентов. В компании Dodo Brands, которой принадлежит сеть пиццерий «Додо Пицца» (1322 заведения в 24 странах мира), одной из ключевых задач является создание персонализированных предложений для каждого клиента, а также анализ их эффективности. Промоакции могут включать разнообразные предложения, такие как скидки на определенную сумму, бесплатные товары или другие бонусы, которые становятся доступны при применении промокода. Предсказание вероятности использования промокода представляет собой классическую задачу бинарной классификации. Для решения данной задачи компания Dodo Brands предоставила набор данных, включающий информацию о заказах клиентов, событиях в мобильном приложении и данные о промоакциях, выданных клиентам.

Актуальность темы обусловлена потребностью бизнеса в повышении эффективности маркетинговых кампаний и персонализации предложений на основе поведенческих данных клиентов. Объект исследования: поведение клиентов при получении ими промокодов. Предмет исследования: прогнозирование использования промокодов на основе исторических данных о действиях клиентов.

Целью исследования является разработка модели машинного обучения для предсказания вероятности использования промокода каждым клиентом, что позволит компании иметь априорное представление об эффективности промокампаний, создавать более персонализированные и эффективные промопредложения, подталкивающие клиентов заказывать чаще и на большие суммы, тем самым повысив доходность компании. Задачи работы включают рассмотрение данных о заказах, мобильных событиях и выданных промоакциях клиентов; выбрать подходящие метрики; провести исследовательский анализ данных и создать новые признаки; обучить и сравнить различные модели бинарной классификации; выбрать наилучшую модель и интерпретировать результаты; сделать прогноз на новых данных.

### **Информационная база и метод исследования**

#### **Информационная база исследования**

Информационной базой исследования послужили предоставленные компанией Dodo Brands данные о заказах клиентов, промоакциях и предложенных клиентам событиях в мобильном приложении. Данные представлены в виде нескольких дата-сетов (файлов) в формате csv (comma-separated value). Ниже дано описание каждого набора данных и той информации, которую они содержат.

Файл 1. `orders.csv` содержит подробную информацию о фактических заказах клиентов с 15 марта по 31 октября 2023 г. с указанием, был ли применен промокод. В нем содержится следующая информация:

- *OrderUUID* — уникальный идентификатор заказа
- *addressId* — идентификатор адреса доставки
- *deliverySectorId* — идентификатор сектора доставки
- *ClientUUID* — уникальный идентификатор клиента
- *Date* — день заказа
- *SaleDate* — время заказа
- *UnitUUID* — уникальный идентификатор пиццерии
- *NewClient* — значение «1», если это первый заказ клиента; «0» — иначе
- *ClientOrderNumber* — какой по счету заказ у клиента
- *ProductUUID* — уникальный идентификатор товарной позиции
- *CategoryId* — идентификатор категории товарной позиции
- *ProductTotalPrice* — цена продукта с учетом примененных скидок
- *MenuPrice* — цена в меню (без скидки)
- *OrderState* — статус заказа
- *OrderPaymentType* — тип платежа
- *OrderTotalPrice* — общая сумма заказа со скидкой
- *OrderType* — тип заказа: «1» — доставка, «2» или «3» — ресторан
- *apply\_promo* — значение «1», если использовался промокод; «0» — иначе.

Файл 2. `mobile_events.csv` содержит информацию о мобильных событиях, происходивших с 7 октября 2022 г. по 1 ноября 2023 г., однако большая часть событий относится к октябрю 2023 г. (99,6% или 3 900 477 из 3 916 652 событий). В файле содержится следующая информация:

- *ClientUUID* — уникальный идентификатор клиента
- *VisitToken* — токен сессии
- *EventName* — название события
- *Platform* — платформа, на которой произошло событие (iOS, android)
- *Timestamp* — время события.

Файл 3. `clients_promo_october.csv` содержит данные о промоакциях, выданных клиентам в октябре 2023 г.:

- *ClientUUID* — уникальный идентификатор клиента
- *Id* — идентификатор механики промоакции
- *LocalBeginDate* — дата начала промокампании
- *LocalEndDate* — дата окончания промокампании
- *OrderType* — тип заказа
- *OrderPrice* — порог срабатывания промоакции
- *Discount* — вознаграждение клиенту (скидка).

Файл 4. `train_target.csv` содержит часть данных (28 тысяч записей) о выданных промоакциях со 2 по 5 ноября 2023 г. с указанием, был ли применен промокод. Именно на основе этого файла будут обучаться модели машинного обучения. В файле содержатся следующие сведения:

- *ClientUUID* — уникальный идентификатор клиента
- *Id* — идентификатор механики промоакции
- *OrderType* — тип промокода: «1» — доставка, «2» или «3» — ресторан
- *LocalBeginDate* — дата начала кампании
- *LocalEndDate* — дата окончания кампании
- *OrderPrice* — порог срабатывания промоакции
- *Discount* — вознаграждение клиенту (скидка)
- *apply\_promo* — значение «1», если использовался промокод; «0» — иначе.

Файл 5. test.csv содержит другую часть (7 тыс. записей) о выданных промоакциях со 2 по 5 ноября 2023 г. без указания, был ли применен промокод. Он будет использован для финальной проверки модели. В нем присутствуют следующие данные:

- *ClientUUid* — уникальный идентификатор клиента
- *Id* — идентификатор механики промоакции
- *OrderType* — тип промокода: «1» — доставка, «2» или «3» — ресторан
- *LocalBeginDate* — дата начала кампании
- *LocalEndDate* — дата окончания кампании
- *OrderPrice* — порог срабатывания промоакции
- *Discount* — вознаграждение клиенту (скидка).

Файл 6. submit.csv содержит одну колонку *apply\_promo*, в которую были записаны предсказанные вероятности использования промокода, полученные лучшей моделью на основе файла test.csv. Порядок строк в файле submit.csv должен соответствовать порядку строк в test.csv, так как далее файл submit.csv отправляется на сервер компании-партнера Dodo Brands, где рассчитывается значение метрики ROC-AUC.

Целевой переменной для моделей машинного обучения будет являться бинарная переменная *apply\_promo*. В тренировочном наборе данных ее среднее значение 0,028 при стандартном отклонении равном 0,17, т. е. всего 3% пользователей применили промокод.

### Методы обучения и оценки моделей бинарной классификации при дисбалансе классов

Обсудим задачу бинарной классификации и некоторые метрики оценки эффективности моделей машинного обучения при решении задачи бинарной классификации, а точнее метрики ROC-AUC, *F1*, *CC* (коэффициент корреляции Мэтью) и точность (ассигасу).

*Задача классификации* представляет собой форму контролируемого самообучения, в котором в начале мы тренируем модель на данных, где исход известен, а затем применяем эту модель к данным, где исход неизвестен. *Задача бинарной классификации* — это разновидность задачи классификации, в которой модель должна предсказать один из двух возможных классов для каждой записи. Эти классы часто обозначаются как «0» и «1».

В пакете *scikit-learn* языка программирования Python большинство классификационных методов, предоставляет два метода предсказания: *predict* (который возвращает класс) и *predict\_proba* (который возвращает вероятности для каждого класса). Вместо того чтобы модель просто относила объект к одной из двух категорий, многие алгоритмы могут возвращать вероятностные оценки принадлежности объекта к каждому из классов. Затем для получения окончательного решения используется *порог отсечения* (*threshold*), который позволяет на основе вероятностных оценок определить, к какому классу отнести данный объект. Общая последовательность действий такова:

1. Установить пороговую вероятность для интересующего класса, выше которой мы рассматриваем запись как принадлежащую этому классу.
2. Оценить (любой моделью) вероятность того, что запись принадлежит интересующему классу.
3. Если эта вероятность выше пороговой вероятности, то назначить новую запись интересующему классу.

Чем выше порог отсечения, тем меньше записей, предсказанных как «1», т. е. принадлежащих интересующему классу, чем ниже порог отсечения, тем больше записей, предсказанных как «1» (Bruce et al., 2020).

В предсказательном моделировании общепринято тренировать большое число разных моделей на одних данных, а далее тестировать (валидировать) их на других данных, чтобы оценить, какая из моделей дает наилучшие результаты в условиях, приближенных к реальным. Этот процесс позволяет избежать переобучения (overfitting), при котором модель достаточно точно подгоняет свои предсказания под обучающие данные, но плохо работает на новых данных. Тестирование (валидация) предполагает расчет выбранных метрик эффективности модели:  $R$ -квадрат, средняя процентная ошибка аппроксимации (Marginal Average Percentage Error, MAPE), accuracy,  $F1$ , MCC и др.

В данной работе для разбиения данных на наборы для обучения и для валидации были использованы два алгоритма перекрестной проверки: K-Fold Cross-Validation и Stratified K-Fold Cross-Validation. Оба этих метода представлены в пакете scikit-learn для Python.

Перекрестная проверка (кросс-валидация) — это метод оценки модели, при котором данные делятся на несколько наборов, и модель обучается и тестируется на различных комбинациях этих наборов. Основная цель метода — получить более точную и надежную оценку модели путем учета возможных вариаций в данных и уменьшения зависимости от конкретного набора данных.

K-Fold Cross-Validation — это самый популярный метод перекрестной валидации, который заключается в следующем:

1. Разделение данных на  $k$  частей: данные делятся на  $k$  равных подмножеств или «фолдов».
2. Обучение и тестирование: модель обучается на  $k-1$  фолдах и тестируется на оставшемся фолде. Этот процесс повторяется  $k$  раз, всегда с новым фолдом для тестирования.
3. Расчет и использование средних значений результатов: все  $k$  результатов тестирования усредняются, чтобы получить окончательную оценку эффективности модели.

Stratified K-Fold Cross-Validation — это модификация метода K-Fold, использующий стратификацию при разбиении данных на фолды: каждый фолд содержит примерно такое же соотношение классов, как и все исходное множество. Такой подход может потребоваться в случае, например, крайне несбалансированного соотношения классов, когда при обычном случайном расщеплении данных (random split) некоторые фолды могут либо вообще не содержать записей каких-то классов, либо содержать их слишком мало.

В нашем случае, в тренировочном наборе данных (train\_target.csv) всего 3% пользователей применили промокод, что говорит о крайней несбалансированности классов и необходимости использовать именно метод Stratified K-Fold Cross-Validation.

После применения перекрестной валидации для одной модели у нас остается  $k$  вариантов этой модели, обученных на  $k$  разных подмножествах. Возникает вопрос, какую модель стоит использовать в конечном счете? Возможные варианты:

- делать предсказание с помощью усреднения предсказаний на основе  $k$  моделей;
- из  $k$  моделей выбрать тот вариант, который набрал лучший score на своем тестовом фолде, и использовать его;

- заново обучить модель на всех  $k$  фолдах и делать предсказания на основе этой модели.

В данной работе мы будем использовать третий вариант. В библиотеке PyCaret языка программирования Python данная процедура реализуется через функцию `finalize_model()`.

Перед переходом к метрикам оценки эффективности моделей при решении задачи классификации необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — `confusion matrix` (матрица ошибок). В задачах бинарной классификации класс, который представляет интерес для исследования, называется «положительным» и обозначается единицей (промокод применен), другой класс называется «отрицательным» и обозначается нулем (промокод не применен).

Для каждого объекта в выборке возможны четыре ситуации:

- предсказали положительную метку и угадали. Такие объекты относятся к `true positive (TP)` группе (true — предсказали правильно, positive — предсказали положительную метку);
- предсказали положительную метку, но ошиблись в предсказании — `false positive (FP)` (false — предсказание было неправильным);
- предсказали отрицательную метку и угадали — `true negative (TN)`;
- предсказали отрицательную метку, но ошиблись — `false negative (FN)`.

Для удобства все эти четыре ситуации обычно изображают в виде таблицы, которую называют матрицей ошибок (`confusion matrix`) (см. рис. 1).

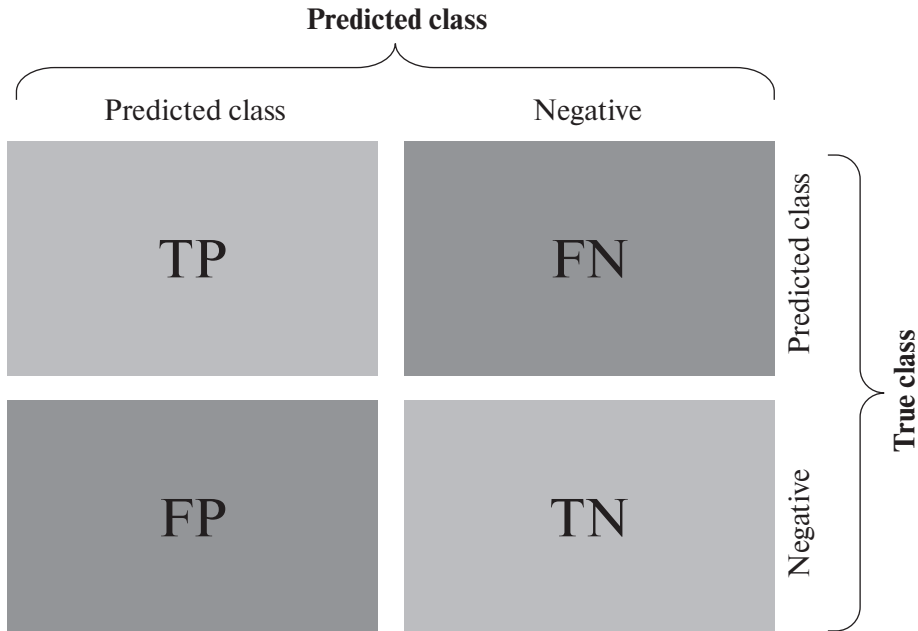


Рис. 1. Матрица ошибок для задачи бинарной классификации

Источник: Метрики классификации..., 2025.

На рис. 1 светлым цветом обозначены правильные предсказания классов, темным — неправильные предсказания классов.

Очевидной, но редко используемой метрикой качества моделей является ассигасу — доля правильных ответов алгоритма. Ассигасу принимает значения от 0 до 1 и рассчитывается по формуле:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

где  $TP$  — число наблюдений в группе true positive;  $TN$  — число наблюдений в группе true negatives;  $FP$  — число наблюдений в группе false positives;  $FN$  — число наблюдений в группе false negatives.

Ассурасу можно использовать в качестве ключевой метрики качества бинарной классификации только тогда, когда классы для нас имеют одинаковое значение, т. е. встречаются в генеральной совокупности с равной вероятностью. В нашей задаче ассурасу является лишь дополнительной метрикой при оценке моделей, так как любая модель, которая будет всегда говорить, что целевая переменная (*apply\_promo*) равна нулю, будет иметь ассурасу, равный 97,02%. Следовательно, необходимо использовать метрики, которые не учитывают  $TN$  и больше ориентируются на  $TP$ , это позволит точнее предсказать миноритарный класс, то есть тех, кто применит промокод.

Если рассмотреть долю правильно предсказанных положительных объектов среди всех объектов, предсказанных как положительные, то получим метрику, которая называется точностью (precision) и рассчитывается по формуле:

$$precision = \frac{TP}{TP + FP}.$$

Если рассмотреть долю правильно найденных положительных объектов среди всех объектов положительного класса, то получим метрику, которая называется полнотой (recall) либо чувствительностью (sensitivity) и рассчитывается по формуле:

$$recall = \frac{TP}{TP + FN}.$$

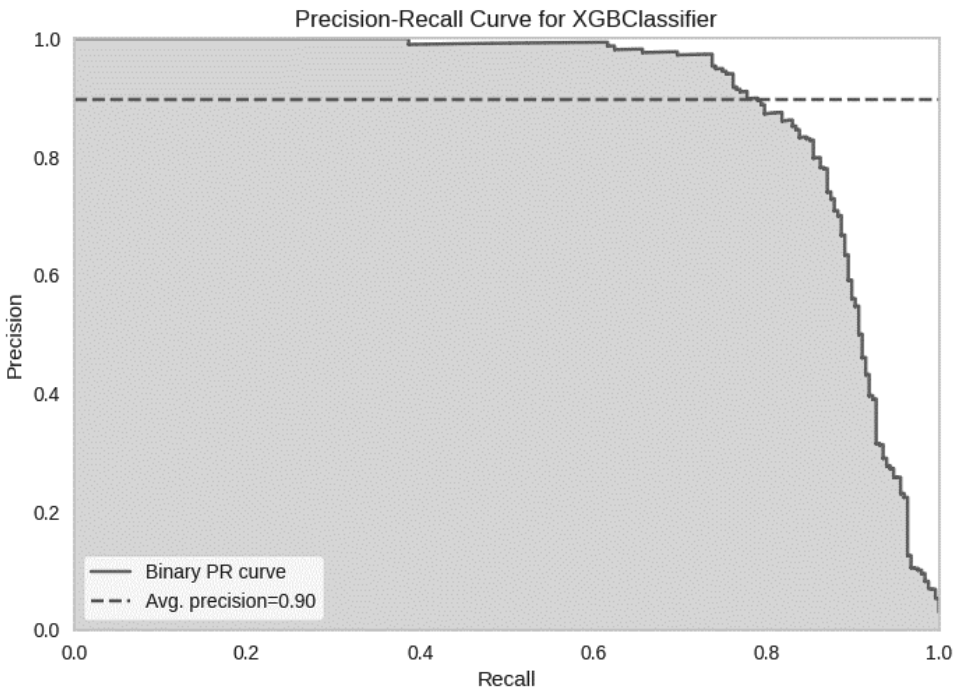


Рис. 2. Кривая Precision-Recall в пакете scikit-learn

Именно желание иметь точность (accuracy) на высоком уровне не позволяет записывать все объекты в позитивный класс и повышает точность того, что предсказанные алгоритмом единицы действительно являются единицами. В то время как мы хотим иметь высокую полноту (recall)-показатель, говорящий о высокой доле распознанных алгоритмом единиц. По сути, мы должны обеспечивать компромисс между precision и recall, который зависит от установленного порога отсечения. На рис. 2 показана кривая Precision-Recall, построенная в пакете scikit-learn, которая позволяет определить, какое значение примут метрики precision и recall при различных порогах отсечения. В идеале она стремится к точке (1;1). Average Precision — метрика, которая суммирует PR-кривую и интерпретируется как площадь под PR-кривой.

$F$ -мера (в общем случае  $F_\beta$ ) — среднее гармоническое взвешенное precision и recall, которая рассчитывается по формуле:

$$F = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}},$$

где  $\beta$  — вес precision в метрике  $F_\beta$ .

$F$ -мера достигает максимума при точности и полноте, равным единице, и близка к нулю, если один из аргументов близок к нулю.

При  $\beta$  равном 1 уравнение приобретает вид средней гармонической с множителем 2 и называется метрикой  $F1$ , имеет вид:

$$F1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}.$$

$F1$  также равняется единице, когда *precision* и *recall* равны единице и используется, когда precision и recall для нас одинаково важны. В scikit-learn есть удобная функция `_metrics.classificationreport`, возвращающая recall, precision и  $F$ -меру для каждого из классов, а также количество экземпляров каждого класса (support) (Метрики в задачах машинного..., 2017).

Последней дополнительной метрикой является МСС (Matthew correlation coefficient, коэффициент корреляции Мэтью), который принимает значения от  $-1$  до  $1$ . Оценка  $1$  указывает на полное соответствие предсказаний и реальных значений тестового набора данных;  $0$  — классификатор работает на уровне случайного предсказания, то есть его эффективность равна эффективности случайного угадывания;  $-1$  — классификатор полностью неправильно предсказывает классы, то есть все положительные предсказания на самом деле отрицательные и наоборот. Коэффициент корреляции Мэтью рассчитывается по формуле:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

В статье (Chicco, Jurman, 2020) утверждается, что МСС является более предпочтительной метрикой оценки качества моделей бинарной классификации по сравнению с  $F1$ , так как имеет два преимущества:

- $F1$  меняется при смене класса, а МСС остается неизменным, если положительный класс переименовывается в отрицательный, и наоборот.
- $F1$  не зависит от количества образцов, правильно классифицированных как отрицательные.

В `scikit-learn` имеется функция `_metrics.matthews_corrcoef`, возвращающая значение коэффициента корреляции Мэтью.

При конвертации вероятностной оценки алгоритма в бинарную метку, необходимо выбрать порог отсеечения (`threshold`), при котором 0 становится 1. Логичным кажется порог отсеечения, равный 0,5, однако он не всегда оказывается оптимальным, например при вышеупомянутом дисбалансе классов. При уменьшении порога отсеечения мы будем правильно предсказывать все большее число положительных объектов, но при этом и чаще неправильно предсказывать положительную метку на всем большем числе отрицательных объектов. Поэтому было решено ввести две метрики, которые позволят оценить вышеописанные эффекты от уменьшения порога отсеечения:

TPR (True Positive Rate) — это полнота (`recall`) — доля положительных объектов, правильно предсказанных положительными, которая рассчитывается по формуле:

$$TPR = recall = \frac{TP}{TP + FN}.$$

FPR (False Positive Rate) — это новая величина — доля отрицательных объектов, неправильно предсказанных положительными, которая рассчитывается по формуле:

$$FPR = \frac{FP}{FP + TN}.$$

ROC-кривая (Receiver Operating Characteristic curve) иллюстрирует значения TPR и FPR при различных значениях порога отсеечения. На оси  $X$  откладывается FPR, а на оси  $Y$  — TPR. Каждая точка на графике соответствует конкретному порогу отсеечения. Идеальная ROC-кривая стремится к точке (0; 1), указывая, что алгоритм правильно классифицирует положительные объекты (TPR высокое) и минимизирует количество ошибок (FPR низкое). На рис. 3 в качестве примера изображена ROC-кривая, построенная в пакете `scikit-learn`.

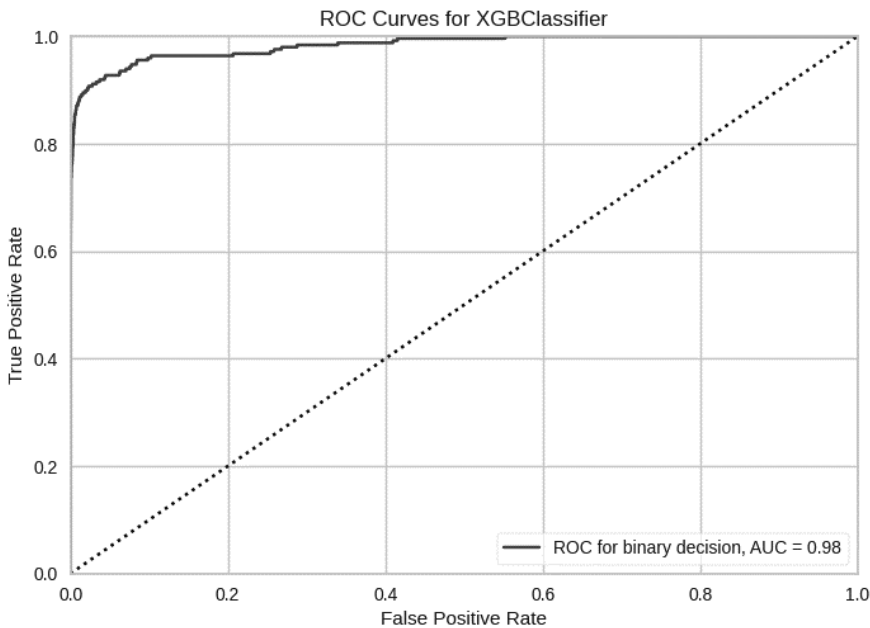


Рис. 3. ROC-кривая в пакете `scikit-learn`

ROC-AUC (Area Under Curve) — площадь под ROC-кривой, которая измеряет, насколько хорошо классификатор разделяет два класса. Когда AUC равна 1, классификатор идеально разделяет классы, при AUC близком к 0,5 классификатор предсказывает метки на уровне подбрасывания монеты. AUC является удобной метрикой, так как не зависит от конкретного порога отсечения и позволяет оценить модель в целом, в то время как ROC-кривая помогает подсказать, какой порог отсечения стоит использовать в конкретной задаче для «отлова» максимального числа позитивных объектов (TP), не начав ошибочно относить слишком много негативных объектов к позитивному классу (FP).

Как отмечалось ранее, любая предсказательная модель, которая будет определять, что целевая переменная соответствует мажоритарному классу, при высокой несбалансированности классов будет иметь ассурасу, равный доле мажоритарного класса. При этом она будет практически бесполезна. Оценить, действительно ли полученные модели изучают полезные закономерности в данных или просто угадывают значение целевой переменной, позволяет сравнение полученных моделей с базовым классификатором.

Базовый классификатор (Dummy Classifier) — это простейшая модель машинного обучения, которая делает предсказания, используя основные правила, не обучаясь на входных данных. Она служит эталоном при определении производительности более сложных моделей. Базовый классификатор помогает понять, действительно ли сложные модели изучают полезные закономерности или просто угадывают их, потратив на свое обучение большое число вычислительных ресурсов.

Базовый классификатор оперирует простыми стратегиями прогнозирования, которые не предполагают никакого реального обучения на основе данных, а только выполняют анализ распределения целевой переменной, используя следующие подходы:

- наиболее часто встречающийся (most frequent) — всегда выбирается наиболее часто встречающийся класс;
- равномерный подход (uniform) — случайно выбирается любой класс;
- стратифицированный анализ (stratified) — генерирует предсказания на основе соотношения классов в обучающем наборе.

Базовый классификатор требует минимальных вычислительных ресурсов и обеспечивает уровень производительности, являющийся эталоном для других моделей, помогая определить, когда сложные модели переобучаются или когда имеющиеся предикторы никак не помогают сложным моделям предсказать значение целевой переменной.

В данной работе предсказывается, воспользуется ли промокодом или нет клиент сети пиццерий «Додо Пицца». Для оценки качества моделей бинарной классификации в качестве целевой метрики выбрана ROC-AUC, в качестве дополнительных метрик — recall, precision, MCC, F1, accuracy и базовый классификатор (Dummy Classifier).

## Результаты исследования

Анализ данных был произведен в Google Colab, с использованием языка программирования Python и библиотек pandas, scipy, Pycaret, sklearn, matplotlib, seaborn, tqdm, numpy, gdown, warnings. Из имеющихся файлов с данными необходимо было создать новые переменные для построения модели с наилучшими метриками. На одном файле производилось обучение модели, а на основе другого — предсказание вероятности классов, на основе использования лучшей модели машинного обучения.

## Подготовка данных

### Набор данных *train* и *test*

Наборы данных *train* и *test* содержат ноябрьские данные о выданных промоакциях. При этом клиенту мог быть выдан промокод, но заказ он не сделал. Оба набора аналогичны по представленным в них признакам, но в данных *train* есть еще *apply\_promo*. Пропусков данных нет. Подчеркнем еще раз, что наборы данных крайне не сбалансированы: записи с целевым классом (с примененным промокодом) составляют только 3% от общего числа записей. Уникальность данных в колонке характеризует кардинальность (*cardinality*). Высокая *cardinality* — уникальные данные для более 25% наблюдений, низкая *cardinality* — часто повторяющиеся данные, уникальные для менее 5% наблюдений, в противном случае имеется нормальная *cardinality*. Высокая кардинальность вкупе с большим числом уникальных значений приводит либо к созданию слишком большого числа новых признаков, либо к утечке данных (*Data Preprocessing*, 2024).

У клиентов *ClientUuid* в *test* и *train* кардинальность высокая (67%), однако не все 100% *id* клиентов встречаются в *train* и *test* несколько раз, т. е. некоторым клиентам предлагалось несколько промокодов. В обучающем наборе данных *train* нет клиентов, которые входят в набор *test*. Для получения информации из других наборов данных необходимо создать новые переменные, используя *ClientUuid* как внешний ключ, который не будет использоваться при обучении модели. Анализ даты событий выявил, что в *train* и *test* у есть лишь промоакции, начавшиеся 02.11.2023 и закончившиеся 05.11.2023.

Колонки *LocalBeginDate* и *LocalEndDate* можно исключить из данных для обучения модели, однако, возможно, они могут быть полезны для получения информации из других файлов. *OrderType* в наборе данных означает, где можно применить промокод: 1, 2, 3 — акция применима где угодно, 2, 3 — применима в ресторане. Для обозначения применимости к доставке вместо *OrderType* создадим бинарную переменную *use\_for\_delivery*. В табл. 1 показано, что промокоды, которые можно применить к доставке, используются на 38% чаще, чем те, которые можно активировать только в ресторанах.

Таблица 1

Количество наблюдений, доля примененных промокодов в зависимости от применимости к доставке в файле *train*

<i>use_for_delivery</i>	<i>count</i>	<i>mean</i>
0	16 255	0,025777
1	11 486	0,035522

Источник: расчеты автора.

Посчитаем для каждого пользователя в *train* и *test*, какую долю предложенных ему промокодов составляют те, которые можно применить к доставке (*avg\_del*), и показывались ли пользователю такие промокоды (*max\_del*). В двух файлах пороги срабатывания скидок (*OrderPrice*) совпадают, кардинальность у переменной низкая (меньше 1%). На рис. 4 представлено распределение порогов срабатывания скидок в разрезе использования промокода. Возможно, клиенты предпочитают промокоды, которые можно активировать при меньшей стоимости заказа.

Приведенные графики похожи, но ожидаемо у заказов с примененным промокодом выше дисперсия, так как таких заказов меньше. Расчет коэффициента ранговой корреляции Спирмена показал незначительную отрицательную взаимосвязь (−0,18) между минимальной стоимостью заказа для срабатывания промокода и *apply\_promo*.

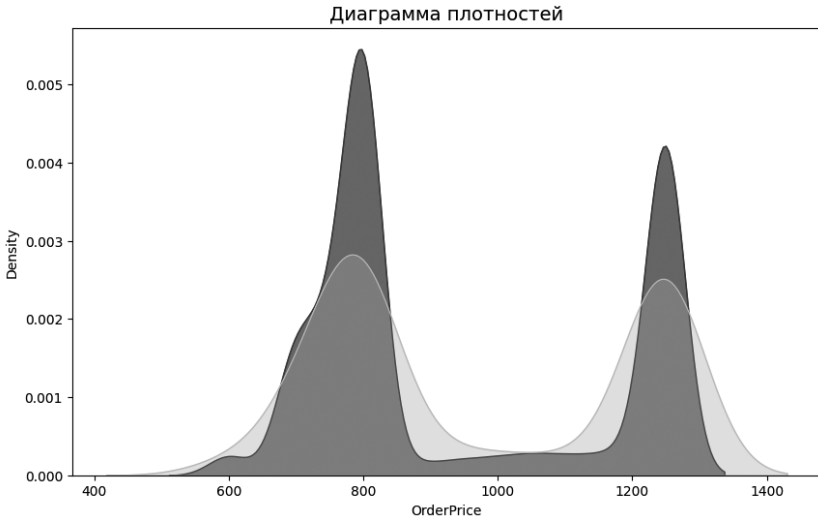


Рис. 4. Диаграммы плотностей вероятностей для OrderPrice с активированным промокодом (светлая) и с неактивированным (темная)

Источник: разработка авторов.

Рассчитаем средний и минимальный порог срабатывания у промокодов, предложенных каждому пользователю. В train и test есть только три уникальных значения *Id*, это вид скидки: в российских рублях (5), в процентах (6), в монетках сервиса (7) (1 додоеин равен 1). В монетках сервиса пользователи менее заинтересованы, чем в рублях или процентной скидке: на 26% реже применяют промокод, нежели при получении скидки в процентах, и на 19% реже, чем при получении скидки в рублях.

В train и test размеры скидок (*Discount*) одинаковы. Кардинальность *Discount* низкая. На рис. 5 показана тепловая карта доли примененных промокодов в разрезе типа скидки (*Id*) и суммы скидки (*Discount*) по промокоду. Видно, что чем больше скидка в процентах, тем скорее ей воспользуется пользователь. В случае с рублями и баллами сложно говорить о линейной связи.

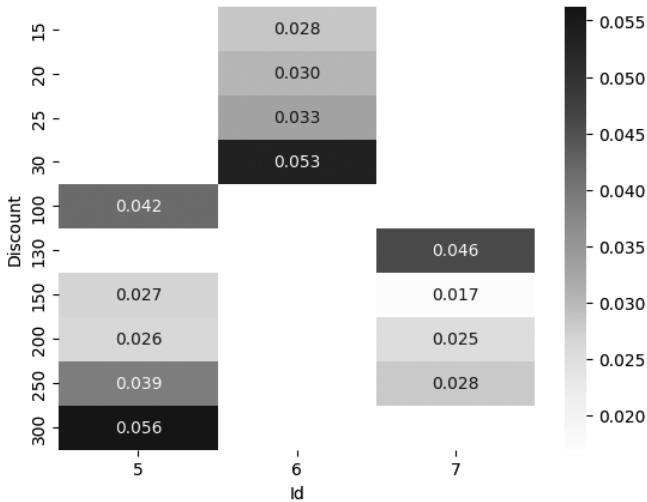


Рис. 5. Тепловая карта доли примененных промокодов в разрезе типа скидки (*Id*) и суммы скидки (*Discount*) по промокоду

Источник: разработка авторов.

Также была рассмотрена доля примененных промокодов с точки зрения порога срабатывания промокода — *OrderPrice*, однако это никак не помогло выявить взаимосвязи. Для размера скидки с учетом типа скидки (*Id*) и суммы скидки (*Discount*) была создана текстовая переменная *Id\_Discount*, а также найдено число предложенных пользователю промокодов (*n\_promos*) в файлах *train* и *test*. Далее в наборы данных *df* и *df\_test* были добавлены следующие переменные, полученные из *train* и *test*:

- *use\_for\_delivery* — можно ли применить промокод для доставки;
- *max\_del* — бинарная переменная — предлагались ли пользователю промокоды для доставки;
- *avg\_del* — переменная доля промокодов для доставки для пользователя  $[0, 1]$ ;
- *Id\_Discount* — текстовая переменная — какая скидка предлагалась клиенту;
- *n\_promos* — число показанных пользователю промокодов;
- *avg\_OrderPrice* — средняя стоимость заказа для срабатывания промокодов, предложенных каждому пользователю;
- *min\_OrderPrice* — минимальная стоимость заказа для срабатывания промокодов, предложенных каждому пользователю;

Также использовались *Id*, *OrderPrice*, *Discount*.

### Набор данных *orders*

Набор данных *orders* содержит записи только о фактических заказах клиентов, т. е. промокод мог быть выдан в *train* и *test*, но так как не был использован он не будет записан в *orders*. В целом, примерно для 54% клиентов мы смогли создать новые переменные, используя данные *orders*. Предоставленный набор данных содержит пропущенные значения в *addressID* и в *deliverySectorUUId*. Эти пропуски были заполнены нулями.

Каждое *OrderUUId* повторялось в данных в среднем четыре раза. Посчитав число сочетаний комбинации переменных: номера заказа (*OrderUUId*), номера клиента (*ClientUUId*), даты заказа (*SaleDate*) и номера товарной позиции (*ProductUUId*), можно прийти к выводу, что, скорее всего, в наборе данных *orders* представлены товарные позиции (например, пицца «Маргарита» маленькая, чай черный 500 мл и т. п.) для каждого заказа, т. е. составной первичный ключ каждой записи — комбинация *ProductUUId* и *OrderUUId*.

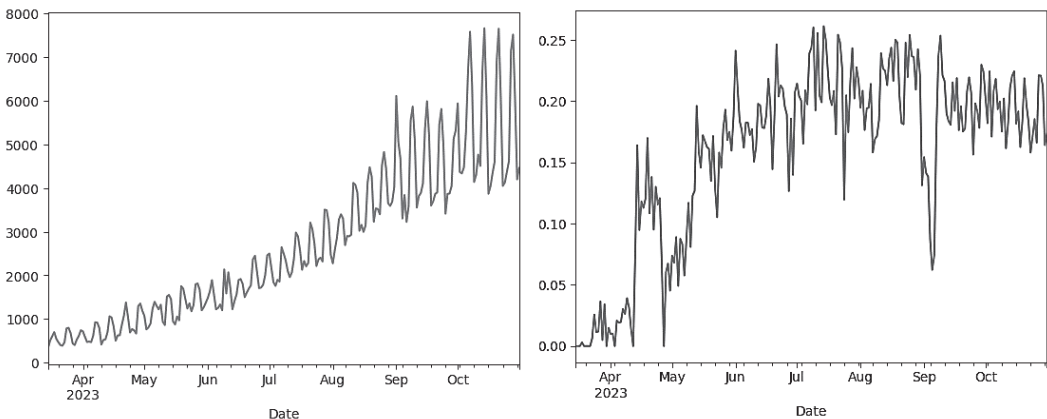


Рис. 6. Распределение во времени числа заказов (слева) и доли активированных промокодов (справа) в файле *orders*

На рис. 6 представлены распределение числа заказов и доли активированных промокодов в период с 15 марта по 31 октября 2023 г.

Виден экспоненциальный рост числа заказов, что объясняет, почему набор данных `orders` охватывает только около 54% пользователей, так как в `train` и в `test` данные только за ноябрь и оставшиеся 46% клиентов могли быть новыми клиентами, хотя в противовес этому доля пользователей с позитивной меткой `NewClient` составляет 0,07% от числа уникальных пользователей в файле `orders`. Что касается доли активированных промокодов, то с июля 2023 г. доля активированных промокодов стабилизировалась, однако в конце августа — середине сентября произошло значительное снижение данной метрики.

Для создания признаков из дат заказов для каждого пользователя напишем функцию, в результате выполнения которой получим количество заказов за последний месяц (относительно установленной даты, например 2023-11-01) — `num_orders_last_month`, средний интервал времени между заказами — `mean_days_between_orders`, число дней с момента последнего заказа — `days_since_last_order`, число дней от первого до последнего заказа — `days_between_first_and_last_order`, количество заказов в рабочие дни — `num_orders_workdays`, число дней с момента последнего использования промокода — `days_since_last_promo` (для клиентов, которые никогда не пользовались промокодом 1000 дней), число использованных промокодов за последний месяц — `promo_used_last_month`. Применим данную функцию для набора данных `orders`, сохраним результат ее исполнения в наборе данных `dates`.

Анализ новых клиентов (`NewClient`), составляющих 0,07% от числа уникальных пользователей, показал, что они статистически значимо (по критерию согласия Хи-квадрат  $p\text{-value} < 0,001$ ) на 71% реже используют промокоды.

В заказах с `OrderState`, равным 11, статистически значимо реже используются промокоды (на 8%), чем в заказах с `OrderState`, равным 4. Создадим бинарную переменную `OrderState4` и добавим ее в файлы `df` и `df_test`.

В табл. 2 показано, что доля активированных промокодов сильно отличается от метода платежа (`OrderPaymentType`). Эту переменную также стоит добавить в файлы `df` и `df_test`.

Таблица 2

**Количество наблюдений и доля примененных промокодов в зависимости применимости к доставке в файле `train`**

<code>OrderPaymentType</code>	count	mean
0	42 724	0,156727
1	18 668	0,146293
2	550 347	0,191494

Источник: разработка авторов.

Таблица 3

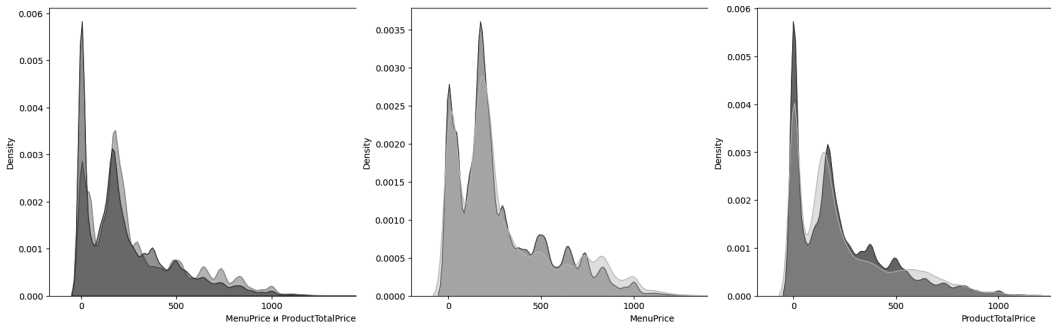
**Количество наблюдений и доля примененных промокодов в зависимости от типа заказа: «1» — доставка, «2» или «3» — ресторан**

<code>OrderType</code>	count	mean
1	488 249	0,172551
2	887	0,270575
3	122 603	0,247359

Источник: собственная разработка авторов.

В табл. 3 показано, что для заказов с доставкой (*OrderType* равный 1) доля активированных промокодов значительно меньше, чем для ресторанов. Соответственно, была создана бинарная переменная *used\_for\_delivery*, которая добавлена в файлы *df* и *df\_test*.

По категориям товарной позиции (*CategoryId*) доля активированных промокодов также различается. Всего имелось 7 товарных категорий. Рассчитывалась доля каждой категории в заказах каждого клиента, и затем она была внесена в набор данных *cats*, который был добавлен к файлу *orders*. Посмотрим на распределение цены без скидки (*MenuPrice*) и со скидкой (*ProductTotalPrice*) товарных позиций в заказах. На рис. 7 показаны диаграммы плотностей для товаров в заказах без скидки (*MenuPrice*) и со скидкой (*ProductTotalPrice*) (слева).



**Рис. 7.** Диаграммы плотностей для товаров в заказах без скидки со скидкой (слева), диаграммы плотностей вероятностей в разрезе *apply\_promo* для цены без скидки (в центре) и цены со скидкой (справа)

Источник: разработка авторов.

На рис. 7 (слева) видно, что со скидкой цена некоторых товаров становится нулевой (16% записей имеют нулевую цену). В заказах с нулевой стоимостью доля заказов с *OrderState*, равном 11 отличается от доли заказов с *OrderState*, равном 11, среди всех заказов, но не стремится к единице, что означает, что *OrderState*, равный 11, не используется для нулевых заказов.

Заказы с нулевой стоимостью могут быть связаны с действием промокода или же с рекламной акцией, не связанной с промокодами (например, с бесплатной доставкой в новой пиццерии или бесплатной пиццей, если приводишь друга). Чтобы проверить, связано ли обнуление цены с использованием промокодов, были построены диаграммы в разрезе *apply\_promo* для цены без скидки на рис. 7 (в центре) и цены со скидкой на рис. 7 (справа). Если обнуление цены связано с промокодом, то для *ProductTotalPrice* мы ожидаем смещение светлого графика влево и большую плотность распределения около нуля, относительно темного графика, а для *MenuPrice* графики должны быть почти идентичны.

Распределение цены товаров с активированным промокодом (в центре рис. 7) и с неактивированным промокодом (слева либо справа) различается в основном лишь дисперсией, так что большая доля около нулевой цены товаров скорее связана с другими рекламными акциями. Однако статистические критерии приводят к выводу, что все три распределения на рис. 7 статистически значимо различаются (Хи-квадрат и Колмогоров–Смирнов  $p$ -value < 0,001).

Чтобы не путать стоимость заказа со скидкой (*ProductTotalPrice*) и без скидки (*MenuPrice*), добавим соответствующие переменные *price\_with\_discount* и *price*, и вычислим скидку (*discount*) как разницу между этими переменными. Также рассчитаем для каждого пользователя и добавим в набор данных: среднюю и суммарную скидку (*avg\_discount*, *sum\_discount*), число использованных промокодов (*apply\_promo\_sum*),

долю использования промокодов (*apply\_promo\_mean*), долю заказов с *OrderState*, равным 4, (*OrderState\_mean*), количество платежей методами *pay\_0*, *pay\_1*, *pay\_2* (*pay\_0\_count*, *pay\_1\_count*, *pay\_2\_count*), стандартное отклонение и общую стоимость заказов со скидкой и без (*price\_with\_discont\_std*, *price\_with\_discont\_sum*, *price\_std*, *price\_sum*), а также сколько всего заказов сделал клиент (*ClientOrderNumber\_max*).

Таким образом, в наборы данных *df* и *df\_test*, кроме названных ранее переменных, были включены наборы данных *dates* и *cats*, полученные из *orders*. Все переменные рассчитаны на уровне пользователя.

### Набор данных *mobile*

Этот набор данных содержит информацию о мобильных событиях, происшедших с 7 октября 2022 г. по 1 ноября 2023 г., большая часть событий относится к октябрю 2023 г. (99,6% или 3 900 477 из 3 916 652). Для работы с этим набором данных были преобразованы названия событий и платформы (*ios*, *android*) в бинарные переменные, т. е. произведем one-hot encoding.

Далее для каждого пользователя были рассчитаны: суммарное число сессий (*count\_VisitToken*), число добавлений товаров в корзину (*\_add\_to\_cart\_sum*), число и доля открытых промо-страниц (*\_open\_bonusaction\_sum* и *\_open\_bonusaction\_mean*), число и доля принятых персональных предложений (*\_apply\_personal\_offer\_sum* и *\_apply\_personal\_offer\_mean*), число открытий и закрытий приложения (*\_open\_app\_sum* и *\_close\_app\_sum*), просмотры корзины, меню и профили (*\_screen\_cart\_sum*, *\_screen\_menu\_sum* и *\_screen\_profile\_sum*), число созданных заказов (*\_create\_order\_sum*), число открытых страниц товаров (*\_open\_product\_card\_sum*), число убранных из корзины товаров (*\_remove\_from\_cart\_sum*), число открытий приложения на *ios* и *android* (*\_ios\_sum* и *\_android\_sum*).

Набор *october* с выданными в октябре промоакциями нет смысла использовать, так как все исторические данные были получены из *orders* и *mobile*, а основа взята из *train* и *test*; они были объединены в наборы данных *df* и *df\_test*.

### Объединение данных

Чтобы не путаться в разных переменных, содержащих в названии *apply\_promo*, переименуем целевую переменную в *target*. На рис. 8 показано, какие переменные содержатся в обучающем наборе данных *df*.

```
[ 'Id', 'OrderPrice', 'Discount', 'target', 'use_for_delivery', 'avg_del',
  'max_del', 'avg_OrderPrice', 'min_OrderPrice', 'Id_Discount',
  'n_promos', 'avg_discont', 'sum_discont', 'pay_0_count', 'pay_1_count',
  'pay_2_count', 'OrderState_mean', 'ClientOrderNumber_max',
  'price_with_discont_std', 'price_with_discont_sum', 'price_std',
  'price_sum', 'apply_promo_mean', 'apply_promo_sum', 'OrderType_mean',
  'CategoryId_1', 'CategoryId_2', 'CategoryId_3', 'CategoryId_4',
  'CategoryId_5', 'CategoryId_6', 'CategoryId_7', 'MenuPrice_mean',
  'MenuPrice_std', 'count_VisitToken', '_add_to_cart_sum',
  '_apply_personal_offer_mean', '_open_bonusaction_mean',
  '_apply_personal_offer_sum', '_close_app_sum', '_create_order_sum',
  '_open_app_sum', '_open_bonusaction_sum', '_open_product_card_sum',
  '_remove_from_cart_sum', '_screen_cart_sum', '_screen_menu_sum',
  '_screen_profile_sum', '_android_sum', '_ios_sum',
  'days_between_first_and_last_order', 'num_orders_workdays',
  'promo_used_last_month' ],
```

Рис. 8. Переменные в обучающем наборе данных *df*

В результате объединения был получен набор данных `df`, состоящий из 53 переменных и 27 741 записей, а также набор данных `df_test`, состоящий из 52 переменных (в нем отсутствует `target`) и 6806 записей.

## Выбор модели машинного обучения

Для выбора наилучшей модели машинного обучения была использована библиотека `Русaret`.

### Предобработка данных в `Русaret`

После установки и импорта библиотеки `Русaret` была инициализирована среда машинного обучения с помощью функции `setup()`, в которой указывались набор данных, целевая переменная, а также то, какую предобработку необходимо провести.

На рис. 9 показаны код и параметры для эксперимента. Предобработка данных в `Русaret` начинается с автоматического определения типов переменных: числовой (`numerical`), категориальный (`categorical`), порядковый (`ordinal`), текстовый (`text`) или временной (`datetime`). Типы переменных можно также задать вручную. В `df` содержались 51 числовая переменная, 1 бинарная (целевая переменная) и 1 текстовая переменная. В данных нет пропусков, при наличии пропусков они бы заполнялись для числовых переменных средними значениями, для категориальных — модальными.

Далее данные нормализовались методом  $z$ -стандартизации (разность индивидуального значения и среднего, деленная на стандартное отклонение переменной). В результате ни одна из переменных не сможет чрезмерно влиять на модель в силу порядка ее значений.

Если для группы предикторов наблюдается корреляция Пирсона, превышающая 0,8 по модулю, то для каждой группы с мультиколлинеарностью `Русaret` оставляет только тот признак, который имеет наивысшую корреляцию с целевой переменной, удаляя остальные. Таким образом, число переменных сократилось с 53 до 43.

Тестовая переменная была указана в явном виде как `Id_Discount`. В `Русaret` доступны два метода, которые позволяют преобразовать текст в числовые признаки для обучения модели: `Bag of Words` и `tf-idf`. Метод `Bag of Words` используется для преобразования текста в числовое представление, где текст представляется как множество слов без учета их порядка и грамматики. Основные шаги включают токенизацию текста, создание словаря уникальных токенов из всего доступного для анализа текста, а затем преобразование каждого документа (в нашем случае это значение вида `Id_Discount_5_300`) в вектор, где для каждого токена указывается его частота. Векторизация приводит к созданию разреженной матрицы, в которой много нулевых значений. Однако `Bag of Words` не учитывает, насколько часто встречается токен среди всех документов. Другой метод — `tf-idf` — является продолжением `Bag of Words` и отличается тем, что фокусируется на более редких и уникальных токенах. По умолчанию в `Русaret` используется метод `tf-idf`.

Из-за дисбаланса классов для разбиения набора данных `df` был использован метод `Stratified K-Fold Cross-Validation`, чтобы каждый фолд содержал примерно такое же соотношение классов, как и все исходное множество. Несмотря на то, что данные были разбиты на 10 фолдов, внутри каждого фолда разбиение `df` на обучающие и тестовые наборы данных представлено в соотношении 7:3, т. е. в каждом фолде брались случайным образом 70% данных, а оставшиеся 30% использовались для оценки модели в каждом фолде.

```

setup(data=df,# test_data=df_test,
      target='target',
      multicollinearity_threshold=0.8,
      remove_multicollinearity = True,
      normalize_method = 'zscore',
      normalize = True,
      fold_strategy='stratifiedkfold',
      text_features=['Id_Discount'],
      ignore_features= None,
      session_id=564)

```

	Description	Value
0	Session id	564
1	Target	target
2	Target type	Binary
3	Original data shape	(27741, 53)
4	Transformed data shape	(27741, 43)
5	Transformed train set shape	(19418, 43)
6	Transformed test set shape	(8323, 43)
7	Numeric features	51
8	Text features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Text features embedding method	tf-idf
14	Remove multicollinearity	True
15	Multicollinearity threshold	0.800000
16	Normalize	True
17	Normalize method	zscore
18	Fold Generator	StratifiedKfold
19	Fold Number	10

Рис. 9. Настройка эксперимента в Rucaret

Источник: разработка автора.

Набор данных `df_test` использовался только для финальной проверки (валидации) лучшей модели. При этом расчет метрики ROC-AUC проходил на стороне сервера организаторов на основе предсказанных для `df_test` вероятностей, которые записывались в `submit`.

### Выбор лучшей модели бинарной классификации

Далее производилось обучение и сравнение всех доступные в Rucaret моделей машинного обучения для задачи бинарной классификации. Среди моделей имеются: модели регрессионного анализа (логистическая регрессия, Ridge (L2) Classifier), модели дискриминантного анализа (Linear Discriminant Analysis, Quadratic Discriminant Analysis), машины опорных векторов с использованием линейной ядерной функции (SVM — Linear Kernel), классификатор  $k$  ближайших соседей, наивный байесовский классификатор, древесные модели (Decision Tree Classifier, Random Forest Classifier, Extra Trees Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Light Gradient Boosting Machine, Extreme Gradient Boosting Classifier, Cat Boost Classifier), а также базовый классификатор (Dummy Classifier). Для обучения всех моделей согласно настройкам эксперимента достаточно вызвать функцию `compare_models()`. На рис. 10 для каждой модели представлены значения метрик, полученных по итогам усреднения метрик, рассчитанных для каждого из 10 стратифицированных фолдов.

```
best = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
<b>xgboost</b>	Extreme Gradient Boosting	0.9781	0.8216	0.2831	0.9353	0.4311	0.4235	0.5052	0.5570
<b>lightgbm</b>	Light Gradient Boosting Machine	0.9750	0.8313	0.1812	0.8806	0.2980	0.2911	0.3900	3.2080
<b>catboost</b>	CatBoost Classifier	0.9741	0.8321	0.1381	0.9664	0.2395	0.2339	0.3562	10.1220
<b>rf</b>	Random Forest Classifier	0.9740	0.8563	0.1295	1.0000	0.2276	0.2224	0.3519	2.0910
<b>et</b>	Extra Trees Classifier	0.9708	0.7986	0.0224	0.6667	0.0430	0.0417	0.1160	1.4520
<b>gbc</b>	Gradient Boosting Classifier	0.9707	0.7960	0.0362	0.6117	0.0681	0.0652	0.1424	3.6000
<b>lr</b>	Logistic Regression	0.9703	0.7707	0.0103	0.3000	0.0200	0.0191	0.0539	1.0350
<b>ridge</b>	Ridge Classifier	0.9702	0.7733	0.0000	0.0000	0.0000	0.0000	0.0000	0.3070
<b>dummy</b>	Dummy Classifier	0.9702	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3040
<b>knn</b>	K Neighbors Classifier	0.9700	0.5960	0.0052	0.1500	0.0100	0.0091	0.0254	0.6600
<b>ada</b>	Ada Boost Classifier	0.9694	0.7723	0.0035	0.2000	0.0068	0.0050	0.0220	0.9960
<b>svm</b>	SVM - Linear Kernel	0.9691	0.5238	0.0017	0.0067	0.0027	0.0008	0.0005	0.3280
<b>lda</b>	Linear Discriminant Analysis	0.9611	0.7733	0.1364	0.2400	0.1733	0.1549	0.1617	0.2960
<b>dt</b>	Decision Tree Classifier	0.9586	0.6698	0.3626	0.3257	0.3418	0.3206	0.3218	0.4840
<b>nb</b>	Naive Bayes	0.8330	0.7166	0.4474	0.0814	0.1377	0.0919	0.1343	0.2650
<b>qda</b>	Quadratic Discriminant Analysis	0.7964	0.6789	0.4334	0.0656	0.1136	0.0653	0.1034	0.3140

Рис. 10. Таблица качества классификаторов

Источник: разработка автора.

В целом в предсказании активации промокодов древесные классификаторы (случайный лес) показывают себя лучше других алгоритмов. Логистическая регрессия недалеко ушла от базового классификатора, а половина моделей слабо годится для прогнозов. Несмотря на то, что случайный лес показал наивысшее значение метрики ROC-AUC, был выделен экстремальный градиентный бустинг как наилучший классификатор из имеющихся, так как он, хоть и находится на втором месте по значению ROC-AUC, однако имеет значительно более высокие значения метрик *F1* и *MCC*. После выбора лучшей модели было произведено ее обучение на всем наборе данных через функцию `finalize_model()`.

### Экстремальный градиентный бустинг (xgboost)

Градиентный бустинг — это мощный алгоритм машинного обучения, относящийся к семейству ансамблевых методов, который применяется для задач классификации и регрессии. Его ключевая идея заключается в последовательном построении нескольких слабых классификаторов или регрессоров (обычно деревьев решений), где каждая последующая модель создается для прогнозирования ошибок или остатков предыдущих моделей; затем предсказания всех моделей последовательно объединяются для формирования окончательного предсказания. В отличие от других ансамблевых методов, таких как случайный лес, где модели строятся независимо, градиентный бустинг строит их последовательно (поэтому он и назван бустингом). Он использует градиентный спуск, определяя направление наискорейшего уменьшения функции потерь, а затем строит последующую модель (слабый классификатор), которая наилучшим образом аппроксимирует

этот антиградиент (поэтому он и градиентный). Таким образом, каждая новая модель концентрируется на тех примерах, где предыдущие модели работали плохо, постепенно улучшая общую производительность ансамбля.

Схематично модель градиентного бустинга изображена на рис. 11. В целом модель градиентного бустинга для классификации имеет много общего с логистической регрессией. Опишем алгоритм градиентного бустинга:

- 1) изначальное предсказание алгоритма — это натуральный логарифм шансов целевой переменной (в нашем случае  $\ln(0,03/0,97)$ , равный  $-3,476$ );
- 2) для каждого наблюдения из значения целевой переменной вычитается предсказанное значение, получаются остатки;
- 3) строится дерево решений, которое обучается предсказыванию остатков;
- 4) значения, предсказанные в листьях дерева решений, трансформируются с помощью специальной формулы и умножаются на темп обучения (learning rate), получается новое предсказание;
- 5) новое предсказание модели прибавляется к предыдущим предсказаниям, трансформируется в вероятность, и рассчитываются остатки предсказания;
- 6) процесс повторяется, пока не будет достигнут лимит по числу деревьев (например, 100) или пока остатки ансамбля не будут очень маленькими.

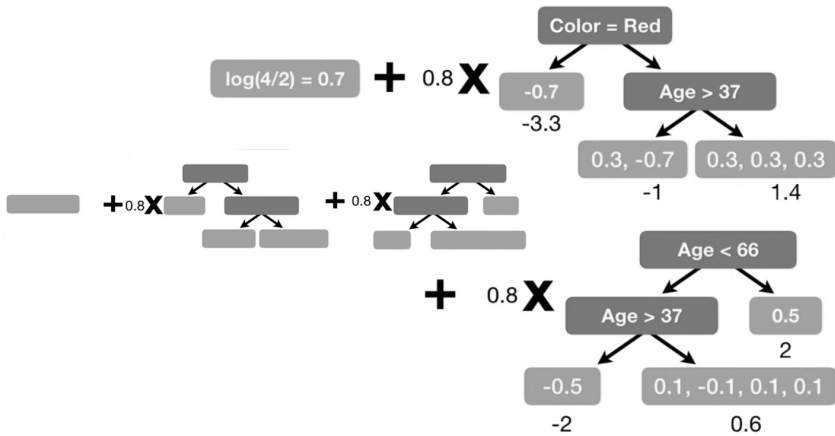


Рис. 11. Схематичная модель градиентного бустинга

**Экстремальный градиентный бустинг (xgboost)** — улучшенная версия классического градиентного бустинга, которая разработана в том числе для больших наборов данных. Он отличается следующими составляющими:

- изначальное предсказание модели — натуральный логарифм шансов от  $p = 0,5$ ;
- Regularization — добавляет штраф  $\lambda$  при трансформации предсказаний дерева перед их умножением на темп обучения (learning rate), который уменьшает чувствительность дерева к выбросам;
- Approximate Greedy Algorithm — для больших наборов данных, когда невозможно перебрать все возможные значения для расщепления (split value), используется приближенное жадное расщепление, которое выбирает переменную и split value, основываясь лишь на локальном приросте gain (на уровне узла) при переборе квантилей каждой переменной;
- Parallel Learning — поскольку перебираются квантили значений каждой переменной, вычисления распараллеливаются между потоками процессора;
- Weighted Quantile Sketch — вместо одинакового числа наблюдений в каждом квантиле, в каждом квантиле должен быть одинаковый вес наблюдений, который для каждой записи равняется cover;

- Sparsity-Aware Split Finding — служит для обработки пропущенных значений;
- Cache-Aware Access — использует кэш процессора для расчетов градиентов, а именно для производных первого и второго порядков;
- Blocks for Out-of-Core Computation — сжимает данные и использует шардинг для чтения данных, что ускоряет вычисления.

Экстремальный градиентный бустинг (xgboost) считается одним из самых лучших традиционных алгоритмов машинного обучения; до широкого распространения больших языковых моделей (LLM) в начале 2020-х гг. в задачах классификации и регрессии находился по своей эффективности на одном уровне с нейронными сетями.

В нашем случае в модели xgboost использовались 100 деревьев решений, остальные гиперпараметры модели Rusaret подобрал автоматически. В приложении 1 на рисунке показана часть расчетов вероятностной оценки принадлежности классу 1 (пользователь активирует промокод) для наблюдения 210 моделью xgboost: слева сверху — вероятность класса 1 после прохождения по каждому из 100 деревьев, справа сверху — путь наблюдения по дереву номер 2, снизу — часть дерева решений номер 2 в виде графа.

### Результаты классификации моделью xgboost

Из файла df случайным образом было отобрано 30% наблюдений или 8323 наблюдений. На рис. 12 изображена матрица ошибок для классификатора xgboost при пороге отсечения, равном 0,5.

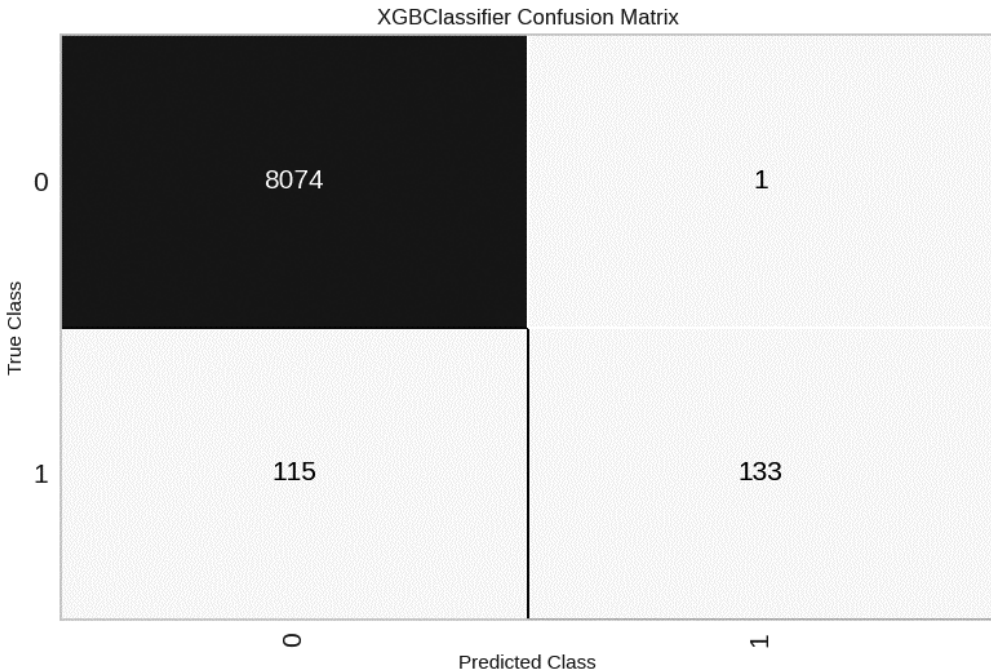


Рис. 12. Матрица ошибок для классификатора xgboost

Источник: разработка автора.

В целом, модель очень хорошо предсказывает, когда пользователь не активирует промокод (высокое количество TN), но имеет значительное количество ложно-отрицательных результатов, то есть модель пропускает пользователей, которые на самом деле активировали промокод.

В приложении 2 приведены расчеты значения метрик Accuracy, Recall, Precision, F1-меры и МСС:

- Recall (Полнота), равная 0,536, означает, что из всех пользователей, которые реально активировали промокод, модель смогла найти только 53,6%.
- Precision (Точность), равная 0,993, означает, что из всех пользователей, которых модель предсказала как активировавших промокод, 99,3% действительно его активировали.
- F1-мера, равная 0,696, означает, что это средний показатель между точностью и полнотой, указывающий на умеренный баланс между способностью модели избегать ложных срабатываний и находить все релевантные случаи.
- МСС (Коэффициент корреляции Мэтью), равный 0,730, означает хорошую общую производительность модели, учитывая баланс между всеми типами ошибок.

Рассмотрим общую производительность модели xgboost для всех возможных порогов отсечения на рис. 13. Значение ROC-AUC, равное 0,98, является очень высоким и означает, что модель очень хорошо справляется с задачей присвоения вероятностей, позволяя нам эффективно выбирать порог отсечения. Модель значительно лучше случайного угадывания (пунктирная линия по диагонали).

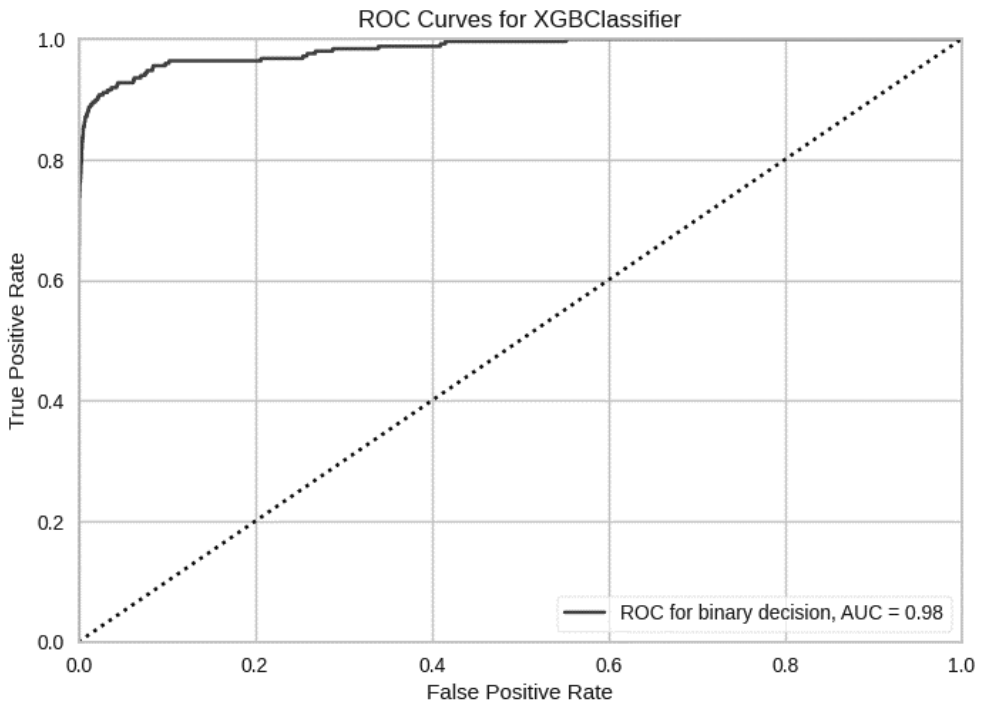


Рис. 13. ROC-кривая для классификатора xgboost

Источник: разработка автора.

При конвертации вероятностной оценки алгоритма в бинарную метку необходимо выбрать порог отсечения (threshold), при котором 0 становится 1. Порог отсечения в 0,5 в целом показывает хорошие результаты. Если для «Додо Пицца» крайне важно, чтобы каждое предсказание активации было действительно активацией, необходимо минимизировать ложноположительные результаты (то есть не отправлять промокоды тем, кто их не использует), можно выбрать более высокий порог отсечения — на рис. 13 это точки слева. Если для «Додо Пицца»

крайне важно выявить всех пользователей, которые потенциально активируют промокод, даже если это приведет к увеличению ложноположительных результатов (то есть предсказанию активации для тех, кто ее не сделает), то можно выбрать более низкий порог отсечения — на рис. 13 это точки справа. С одной стороны, «Додо Пицца» может захотеть оптимизировать свои затраты на предоставление скидок, в этом случае важнее минимизировать ложноположительные результаты и улучшить персонализацию промокодов, с другой стороны, расходы на неактивированный промокод незначительны по сравнению с расходами на закупку и размещение рекламы.

На рис. 14 показана PR-кривая для `xgboost`. Это еще один способ оценки производительности двоичного классификатора, особенно полезный для задач с несбалансированными классами, так как не учитывает TN.

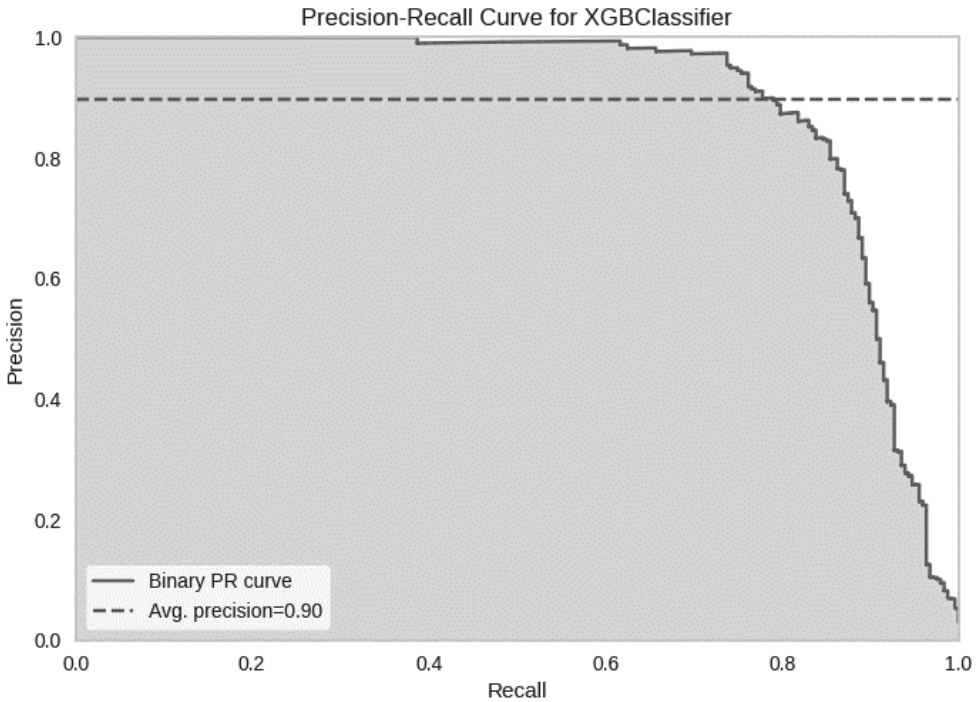


Рис. 14. PR-кривая для классификатора `xgboost`

Источник: разработка автора.

Значение Average Precision, равное 0,90, является очень высоким — это отличный показатель для модели, особенно для задач с несбалансированными классами. Высокое значение AP указывает на то, что модель способна достигать высокой точности, сохраняя при этом хорошую полноту. Это подтверждает, что классификатор `xgboost` очень эффективен в распознавании активаций промокодов, и его предсказания надежны.

Кривые ROC и PR подходят для оценки и сравнения классификаторов, но прямо не говорят, какой порог отсечения стоит использовать. Для статистического выбора порога отсечения можно воспользоваться графиком статистики Колмогорова–Смирнова (KS Statistic Plot). На оси *X* отражается порог отсечения, а на оси *Y* отражается кумулятивный процент наблюдений, для которых предсказанные вероятности ниже данного порога. Иными словами, это график кумулятивных распределений вероятностей для каждого класса (рис. 15).

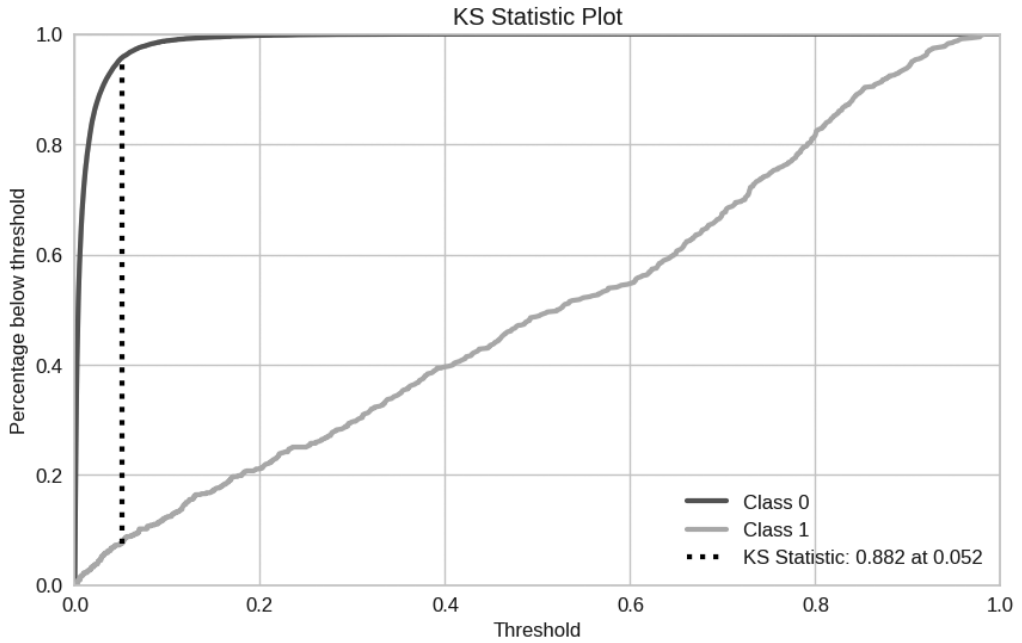


Рис. 15. KS Statistic Plot для классификатора xgboost

Источник: разработка автора.

Линия Class 0 показывает кумулятивное распределение для пользователей, не активировавших промокод. Она быстро растет, так как большинство наблюдений с классом 0 получили низкие вероятностные оценки. Линия Class 1 показывает пользователей, активировавших промокод. С увеличением порога отсечения распознается все больше пользователей, реально активировавших промокод, однако поскольку это целевой класс, то кривая растет медленнее.

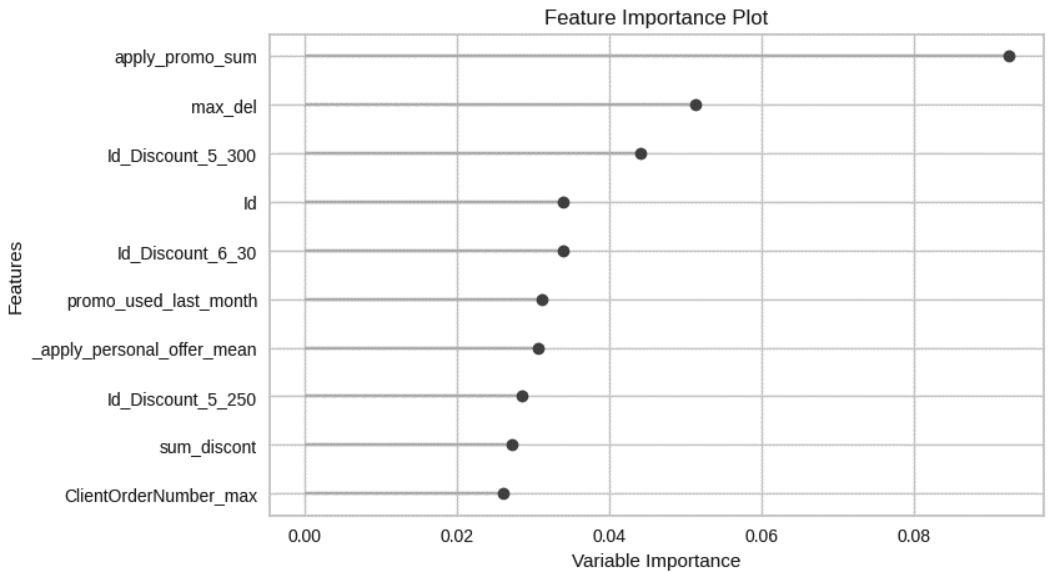


Рис. 16. Feature Importance Plot для классификатора xgboost

Источник: разработка автора.

Пунктирная линия KS Statistic указывает на порог отсечения, где расстояние между кумулятивными распределениями двух классов (черной и серой кривых) максимально. Значение 0,882 — очень высокое значение KS-статистики. Значения выше 0,4 обычно считаются хорошими, а 0,882 указывает на отличную разделительную способность модели. Полученное максимальное разделение между двумя классами достигается при пороге отсечения, равном 0,052, т. е., применив данный порог отсечения, мы слишком сильно увеличим долю ложно положительных наблюдений.

Оценим, какие переменные оказались наиболее важными для классификатора, с помощью графика Feature Importance Plot на рис. 16. Важность переменных для деревьев алгоритмов оценивается общим приростом (gain) при использовании данной переменной для расщепления (сплитования) в узлах, где больший gain указывает на более высокую важность.

Наиболее важными переменными для классификатора xgboost оказались:

- *apply\_promo\_sum* — число использованных пользователем промокодов;
- *max\_del* — предлагались ли пользователю промокоды для доставки;
- *Id\_Discount\_5\_300* — скидка в 300 российских руб.;
- *Id* — типы скидки: в 5 — рублях, 6 — процентах и 7 — монетах сервиса (1 додочкоин равен 1);
- *Id\_Discount\_6\_30* — скидка в 30%;
- *promo\_used\_last\_month* — число использованных пользователем промокодов за последний месяц;
- *\_apply\_personal\_offer\_mean* — доля принятых персональных предложений пользователем;
- *Id\_Discount\_5\_250* — скидка в 250 российских руб.;
- *sum\_discont* — суммарная скидка, полученная пользователем;

*ClientOrderNumber\_max* — сколько всего заказов сделал пользователь.

### Предсказание для набора данных *df\_test*

С помощью функции *predict\_model()*, указав модель *final\_XGboost*, набор данных *df\_test* и необходимость получить вероятностные оценки, было произведено предсказание того, воспользуется ли пользователь предложенным ему промокодом. Вероятностные оценки записывались в файл *XGboost\_submit.csv*, который потом отправлялся на сервер компании-партнера Dodo Brands, где было рассчитано значение метрики ROC-AUC. В итоге полученная метрика ROC-AUC составила 0,71.

## Заключение

В рамках данной работы была решена практическая задача предсказания использования промокодов клиентами сети пиццерий «Додо Пицца» с применением методов машинного обучения. Актуальность темы обусловлена потребностью бизнеса в повышении эффективности маркетинговых акций и персонализации предложений на основе поведенческих данных клиентов.

В работе был проведен анализ предоставленных компанией Dodo Brands данных, включающих сведения о заказах, мобильных действиях клиентов и выданных промоакциях. Особое внимание уделялось проблеме дисбаланса классов, поскольку лишь небольшая доля клиентов действительно использовала промокоды. Были выбраны подходящие метрики оценки моделей, в том числе ROC-AUC в качестве основной, а также *F1*, *MCC*, *precision*, *recall* и *accuracy*. Был выполнен подробный

исследовательский анализ данных. Сформированы новые признаки, отражающие поведение пользователей, объединены данные из различных источников, что позволило существенно расширить информационную базу для построения моделей. В результате были подготовлены обучающий и тестовый наборы данных, включающие 53 и 52 переменные соответственно, охватывающие более 34 тысяч наблюдений. Также были обучены и сопоставлены 16 моделей бинарной классификации с использованием библиотеки `Rusaret`. По итогам сравнительного анализа лучшей моделью признан классификатор на основе экстремального градиентного бустинга (`xgboost`), показавший высокие значения метрик качества при использовании алгоритма стратифицированной перекрестной проверки. Финальная модель позволила достичь значения ROC-AUC 0,71 на тестовой выборке. Таким образом, разработанная модель способна с высокой степенью надежности предсказывать вероятность использования промокодов клиентами, что позволяет Dodo Brands более эффективно распределять маркетинговые ресурсы и повышать вовлеченность клиентов. Работа продемонстрировала, что комплексный подход к подготовке данных, генерации признаков и выбору модели существенно влияет на качество предсказаний и может быть успешно применен для решения бизнес-задач в сфере персонализированных коммуникаций.

## Источники

Базовый классификатор: наглядное руководство с примерами кода для начинающих // Nuances of programming. URL: <https://nuancesprog.ru/p/23610/>.

Градиентный бустинг // Яндекс Образование. URL: <https://education.yandex.ru/handbook/ml/article/gradientnyj-busting>.

Извлечение признаков из текстовых данных с использованием TF-IDF // habr.com URL: <https://habr.com/ru/companies/otus/articles/755772/>.

Кросс-валидация // Яндекс Образование. URL: <https://education.yandex.ru/handbook/ml/article/kross-validaciya>.

Метрики в задачах машинного обучения // habr.com. URL: <https://habr.com/ru/companies/ods/articles/328372/>.

Метрики классификации и регрессии // Яндекс Образование. URL: <https://education.yandex.ru/handbook/ml/article/metriki-klassifikacii-i-regressii>.

Analyze // pycaret.gitbook.io. URL: <https://pycaret.gitbook.io/docs/get-started/functions/analyze>.  
Bag-of-words vs TF-IDF // geeksforgeeks.org. URL: <https://www.geeksforgeeks.org/nlp/bag-of-words-vs-tf-idf/>.

*Bruce P., Bruce A., Gedeck P.* Practical Statistics for Data Scientists. O'Reilly, 2020.

*Chicco D., Jurman G.* The Advantages of the Matthews Correlation Coefficient (MCC) over F1 Score and Accuracy in Binary Classification Evaluation // BMC Genomics. 2020. N 21. P. 6.

Classification // pycaret.readthedocs.io URL: <https://pycaret.readthedocs.io/en/stable/api/classification.html>.

Compute the Matthews correlation coefficient (MCC) // scikit-learn.org. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews\\_corrcoef.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html).

Data Preprocessing // pycaret.gitbook.io. URL: <https://pycaret.gitbook.io/docs/get-started/preprocessing/>

Mastering Model Validation with KS Statistics: A Practical Guide // medium.com. URL: <https://medium.com/@shridharpawar77/mastering-model-validation-with-ks-statistics-a-practical-guide-750fd7be4a4e>.

StratifiedShuffleSplit // scikit-learn.org. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.StratifiedShuffleSplit.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html).

## References

Analyze // pycaret.gitbook.io. Available at: <https://pycaret.gitbook.io/docs/get-started/functions/analyze>.

Bag-of-words vs TF-IDF // geeksforgeeks.org. Available at: <https://www.geeksforgeeks.org/nlp/bag-of-words-vs-tf-idf/>.

Bazovyy klassifikator: naglyadnoye rukovodstvo s primerami koda dlya nachinayushchikh. [Basic Classifier: A visual guide with code examples for beginners.]. Nuances of programming, 2024. Available at: <https://nuancesprog.ru/p/23610/>. (In Russian)

Bruce P., Bruce A., Gedeck P. Practical Statistics for Data Scientists. O'Reilly, 2020.

Chicco D., Jurman G. The advantages of the Matthews correlation coefficient (MCC) over *F1* score and accuracy in binary classification evaluation. *BMC Genomics*, 2020, N 21, pp. 6.

Classification // pycaret.readthedocs.io Available at: <https://pycaret.readthedocs.io/en/stable/api/classification.html>.

Compute the Matthews correlation coefficient (MCC) // scikit-learn.org. Available at: [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews\\_corrcoef.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html).

Data Preprocessing // pycaret.gitbook.io. Available at: <https://pycaret.gitbook.io/docs/get-started/preprocessing/>

Gradiyentnyy busting [Gradient Boosting]. Available at: <https://education.yandex.ru/handbook/ml/article/gradiyentnyj-busting>. (In Russian)

Izvlecheniye priznakov iz tekstovykh dannykh s ispol'zovaniyem TF-IDF [Feature Extraction from Text Data Using TF-IDF]. Available at: <https://habr.com/ru/companies/otus/articles/755772/>. (In Russian)

Kross-validatsiya [Cross-validation]. Available at: <https://education.yandex.ru/handbook/ml/article/kross-validatsiya>. (In Russian)

Mastering Model Validation with KS Statistics: A Practical Guide // medium.com. Available at: <https://medium.com/@shridharpawar77/mastering-model-validation-with-ks-statistics-a-practical-guide-750fd7be4a4e>.

Metriki klassifikatsii i regressii [Classification and regression metrics]. Available at: <https://education.yandex.ru/handbook/ml/article/metriki-klassifikatsii-i-regressii>. (In Russian)

Metriki v zadachakh mashinnogo obucheniya [Metrics in Machine Learning Tasks]. Available at: <https://habr.com/ru/companies/ods/articles/328372/>. (In Russian)

StratifiedShuffleSplit // scikit-learn.org. Available at: [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.StratifiedShuffleSplit.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html).

## Благодарность

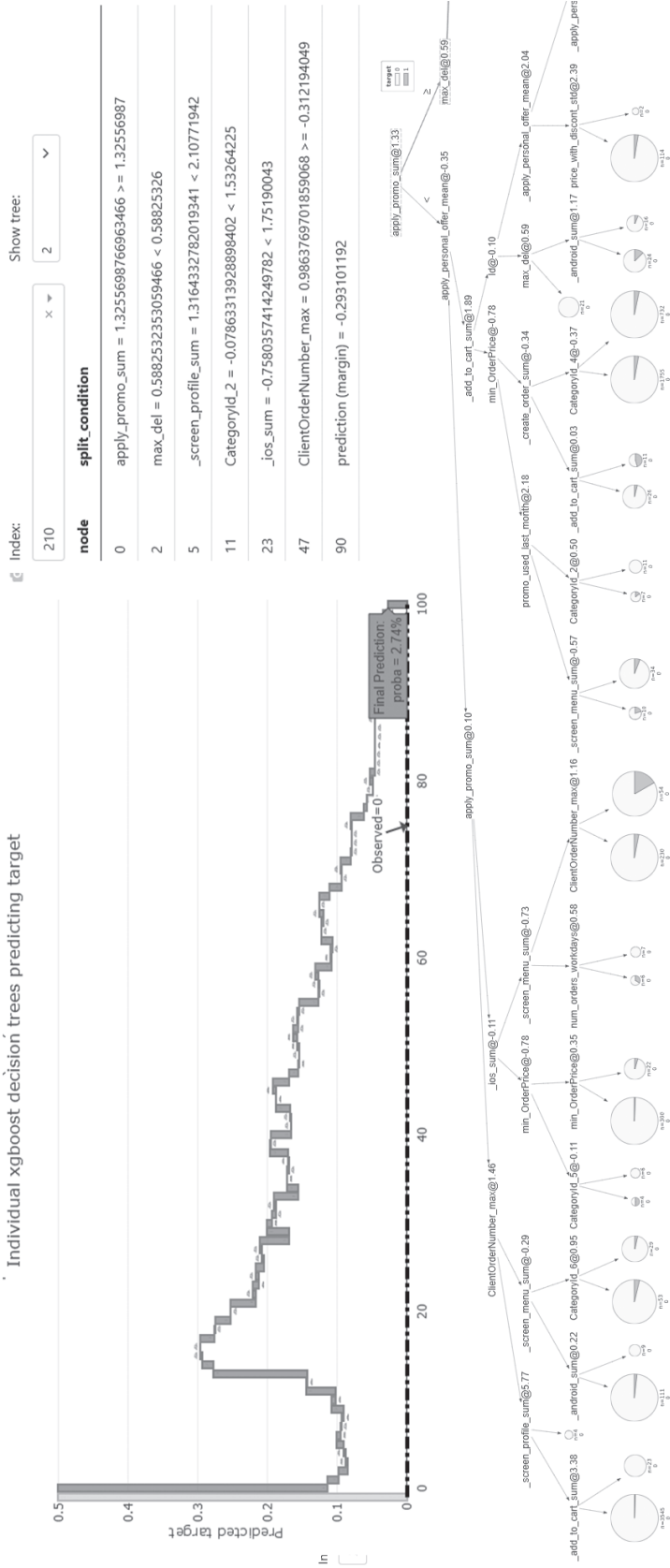
Автор благодарит Сошникову Людмилу Антоновну, д.э.н., профессора кафедры статистики Б, за научное консультирование и поддержку в проведении исследования.

# Приложение 1

## Расчеты вероятностной оценки принадлежности к классу 1 для наблюдения с id 210 моделью xgboost

- слева сверху — вероятность класса 1 после прохождения по каждому из 100 деревьев
- справа сверху — путь наблюдения по дереву номер 2
- снизу — часть дерева решений номер 2 в виде графа

Individual xgboost decision trees predicting target



## Приложение 2

## Расчеты метрик для классификатора xgboost

1. Точность (*Accuracy*):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{133 + 8074}{133 + 8074 + 1 + 115} = \frac{8207}{8323} \approx 0,986.$$

2. Полнота (*Recall*):

$$Recall = \frac{TP}{TP + FN} = \frac{133}{133 + 115} = \frac{133}{248} \approx 0,536.$$

3. Точность (*Precision*):

$$Precision = \frac{TP}{TP + FP} = \frac{133}{133 + 1} = \frac{133}{134} \approx 0,993.$$

4. F1-мера (*F1-score*):

$$F1\text{-score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0,993 \times 0,536}{0,933 + 0,536} = 2 \times \frac{0,532}{1,529} \approx 2 \times 0,345 \approx 0,696.$$

5. Коэффициент корреляции Мэттью (*MCC*):

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (TP + TN) \times (TN + FP) \times (TN + FN)}},$$

$$MCC = \frac{(133 \times 8074) - (1 \times 115)}{\sqrt{(133 + 1) \times (133 + 115) \times (8074 + 1) \times (8074 + 115)}},$$

$$MCC = \frac{1073842 - 115}{\sqrt{134 \times 248 \times 8075 \times 8189}},$$

$$MCC = \frac{1073727}{\sqrt{2164475459000}},$$

$$MCC = \frac{1073727}{1471188,4} \approx 0,730.$$

Статья поступила в редакцию 25 апреля 2025 г.

Статья рекомендована в печать 20 июня 2025 г.