

Оглавление

| | |
|---|-----------|
| Введение | 7 |
| ЧАСТЬ 4. ПЛАН ПРЕДВАРИТЕЛЬНОЙ ПОДГОТОВКИ ДАННЫХ | 8 |
| 1. Введение | 8 |
| 2. Формирование выборки | 10 |
| 2.1. Генеральная и выборочная совокупности | 10 |
| 2.2. Характеристики выборки..... | 10 |
| 2.3. Детерминированные и вероятностные выборки | 12 |
| 2.4. Виды, методы и способы вероятностного отбора | 13 |
| 2.5. Подходы к определению необходимого объема выборки | 14 |
| 3. Определение «окна выборки» и «окна созревания» | 28 |
| 4. Определение зависимой переменной | 32 |
| 5. Загрузка данных из CSV-файлов и баз данных SQL..... | 33 |
| 6. Удаление бесполезных переменных, переменных «из будущего», переменных с юридическим риском..... | 39 |
| 7. Преобразование типов переменных и знакомство со шкалами переменных | 41 |
| 7.1. Количественные (непрерывные) шкалы..... | 41 |
| 7.2. Качественные (дискретные) шкалы..... | 43 |
| 8. Нормализация строковых значений | 45 |
| 9. Обработка дублирующихся наблюдений..... | 61 |
| 10. Обработка редких категорий | 62 |
| 11. Появление новых категорий в новых данных | 69 |
| 12. Импутация пропусков..... | 70 |
| 12.1. Способы импутации количественных и бинарных переменных..... | 70 |

| | |
|---|------------|
| 12.2. Способы импутации категориальных переменных | 71 |
| 12.3. Практика | 73 |
| 13. Обработка выбросов..... | 90 |
| 14. Описательные статистики | 94 |
| 14.1. Пифагорейские средние, медиана и мода | 94 |
| 14.2. Квантиль | 95 |
| 14.3. Дисперсия и стандартное отклонение | 96 |
| 14.4. Корреляция и ковариация | 97 |
| 14.5. Получение сводки описательных статистик в библиотеке pandas | 102 |
| 15. Нормальное распределение..... | 104 |
| 15.1. Знакомство с нормальным распределением | 104 |
| 15.2. Коэффициент островершинности, коэффициент эксцесса и коэффициент асимметрии | 107 |
| 15.3. Гистограмма распределения и график квантиль–квантиль..... | 111 |
| 15.4. Вычисление коэффициента асимметрии и коэффициента эксцесса, построение гистограммы и графика квантиль–квантиль для подбора преобразований, максимизирующих нормальность | 112 |
| 15.5. Подбор преобразований, максимизирующих нормальность для правосторонней асимметрии | 116 |
| 15.6. Подбор преобразований, максимизирующих нормальность для левосторонней асимметрии..... | 128 |
| 15.7. Преобразование Бокса–Кокса | 129 |
| 16. Конструирование признаков | 135 |
| 16.1. Статическое конструирование признаков исходя из предметной области | 135 |
| 16.2. Статическое конструирование признаков исходя из алгоритма | 170 |
| 16.3. Динамическое конструирование признаков исходя из особенностей алгоритма | 290 |
| 16.4. Конструирование признаков для временных рядов | 297 |
| 17. Отбор признаков | 433 |
| 17.1. Методы-фильтры | 436 |
| 17.2. Применение метода-фильтра и встроенного метода для отбора признаков (на примере соревнования BNP Paribas Cardif Claims Management с Kaggle) | 444 |
| 17.3. Комбинирование нескольких методов для отбора признаков (на примере соревнования Porto Seguro’s Safe Driver Prediction с Kaggle) | 451 |
| 18. Стандартизация..... | 475 |
| 19. Собираем все вместе | 486 |

ЧАСТЬ 5. МЕТРИКИ ДЛЯ ОЦЕНКИ КАЧЕСТВА МОДЕЛИ.... 514**1. Бинарная классификация..... 514**

| | |
|--|-----|
| 1.1. Отрицательный и положительный классы, порог отсечения | 514 |
| 1.2. Матрица ошибок | 514 |
| 1.3. Доля правильных ответов, правильность (accuracy) | 517 |
| 1.4. Чувствительность (sensitivity)..... | 519 |
| 1.5. Специфичность (specificity) | 521 |
| 1.6. 1 – специфичность (1 – specificity) | 522 |
| 1.7. Сбалансированная правильность..... | 523 |
| 1.8. Точность (Precision)..... | 524 |
| 1.9. Сравнение точности и чувствительности (полноты) | 525 |
| 1.10. F-мера (F-score, или F-measure) | 526 |
| 1.11. Варьирование порога отсечения..... | 532 |
| 1.12. Коэффициент Мэттьюса (Matthews correlation coefficient или MCC)..... | 536 |
| 1.13. Каппа Коэна (Cohen's kappa)..... | 540 |
| 1.14. ROC-кривая (ROC curve) и площадь под ROC-кривой (AUC-ROC)..... | 542 |
| 1.15. PR-кривая (PR curve) и площадь под PR-кривой (AUC-PR) | 603 |
| 1.16. Кривая Лоренца (Lorenz curve) и коэффициент Джини (Gini coefficient)..... | 616 |
| 1.17. CAP-кривая (CAP curve)..... | 620 |
| 1.18. Статистика Колмогорова–Смирнова (Kolmogorov–Smirnov statistic) | 623 |
| 1.19. Биномиальный тест (binomial test) | 626 |
| 1.20. Логистическая функция потерь (logistic loss) | 628 |

2. Регрессия..... 634

| | |
|---|-----|
| 2.1. R^2 , коэффициент детерминации (R-square, coefficient of determination) | 634 |
| 2.2. Метрики качества, которые зависят от масштаба данных (RMSE, MSE, MAE, MdAE, RMSLE, MSLE) | 643 |
| 2.3. Метрики качества на основе процентных ошибок (MAPE, MdAPE, sMAPE, sMdAPE, WAPE, WMAPE, RMSPE, RmSPE)..... | 656 |
| 2.4. Метрики качества на основе относительных ошибок (MRAE, MdRAE, GMRAE) | 689 |
| 2.5. Относительные метрики качества (RelMAE, RelRMSE) | 697 |
| 2.6. Масштабированные ошибки (MASE, MdASE)..... | 698 |
| 2.7. Критерий Диболда–Мариано | 705 |

ЧАСТЬ 6. ДРУГИЕ ПОЛЕЗНЫЕ БИБЛИОТЕКИ И ПЛАТФОРМЫ 707**1. Библиотеки байесовской оптимизации hyperopt, scikit-optimize и optuna 707**

| | |
|--|------------|
| 1.1. Недостатки обычного поиска по сетке и случайного поиска по сетке..... | 707 |
| 1.2. Знакомство с байесовской оптимизацией..... | 708 |
| 1.3. Последовательная оптимизация по модели (Sequential model-based optimization – SMBO) | 710 |
| 1.4. Hyperopt..... | 716 |
| 1.5. Scikit-Optimize..... | 727 |
| 1.6. Optuna..... | 732 |
| 2. Docker..... | 742 |
| 2.1. Введение..... | 742 |
| 2.2. Запуск контейнера Docker..... | 743 |
| 2.3. Создание контейнера Docker с помощью Dockerfile | 744 |
| 3. Библиотека H2O..... | 749 |
| 3.1. Установка пакета h2o для Python..... | 749 |
| 3.2. Запуск кластера H2O..... | 749 |
| 3.3. Преобразование данных во фреймы H2O | 750 |
| 3.4. Знакомство с содержимым фрейма..... | 751 |
| 3.5. Определение имени зависимой переменной и списка имен признаков..... | 753 |
| 3.6. Построение модели машинного обучения..... | 753 |
| 3.7. Вывод модели | 754 |
| 3.8. Получение прогнозов | 758 |
| 3.9. Построение ROC-кривой и вычисление AUC-ROC..... | 759 |
| 3.10. Поиск оптимальных значений гиперпараметров по сетке | 760 |
| 3.11. Извлечение наилучшей модели по итогам поиска по сетке..... | 762 |
| 3.12. Класс H2OAutoML..... | 762 |
| 3.13. Применение класса H2OAutoML в библиотеке scikit-learn | 771 |
| 4. Библиотека Dask..... | 783 |
| 4.1. Общее знакомство | 783 |
| 4.2. Машинное обучение с помощью библиотеки dask-ml..... | 792 |
| 4.3. Построение конвейера в Dask | 800 |
| 5. Google Colab..... | 804 |
| 5.1. Общее знакомство | 804 |
| 5.2. Регистрация и создание папки проекта | 804 |
| 5.3. Подготовка блокнота Colab | 809 |

Введение

Настоящая книга является коллекцией избранных материалов из первого модуля Подписки – обновляемых в режиме реального времени материалов по применению классических методов машинного обучения в различных промышленных задачах, которые автор делает вместе с коллегами и учениками.

Автор благодарит Дмитрия Ларько за помощь в подготовке раздела по конструированию признаков в четвертой части книги, Уилла Керсена за предоставленные материалы к первому разделу шестой части книги.

Во втором томе мы разберем собственно процесс предварительной подготовки данных, обсудим некоторые метрики качества, рассмотрим ряд полезных библиотек и фреймворков.

План предварительной подготовки данных

1. Введение

До этого момента мы знакомились с инструментами – основными питоновскими библиотеками, классами и функциями, необходимыми для предварительной подготовки данных и построения моделей машинного обучения. Мы брали относительно простые примеры, выполняли предварительную подготовку данных и строили модели машинного обучения без глубокого понимания, зачем нужна та или иная операция предварительной подготовки и что происходит «под капотом» этой операции. В реальной практике мы так действовать не можем, нам нужен четкий план действий и глубокое понимание каждого этапа.

План предварительной подготовки данных, как правило, будет состоять из двух этапов. Первый этап – операции, которые можно выполнить до разбиения на обучающую и тестовую выборки / до цикла перекрестной проверки. Второй этап – операции, которые можно выполнить только после разбиения на обучающую и тестовую выборки / внутри цикла перекрестной проверки.

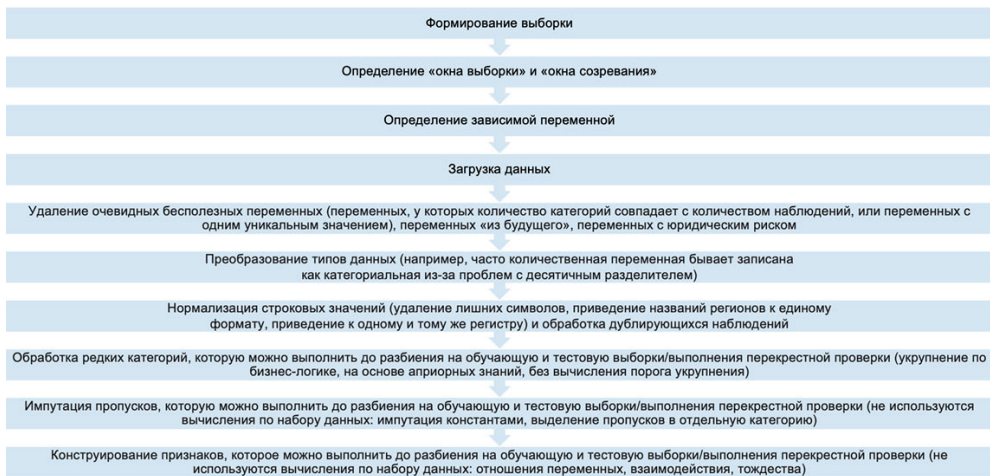
Если используются операции, использующие статистики, например укрупнение редких категорий по порогу, импутация пропусков статистиками, стандартизация, биннинг и конструирование признаков на основе статистик (frequency encoding, likelihood encoding), они должны быть осуществлены после разбиения на обучающую и тестовую выборки или внутри цикла перекрестной проверки.

Если мы используем случайное разбиение на обучающую и тестовую выборки и выполняем перечисленные операции до разбиения, получается, что для вычисления среднего и стандартного отклонения по каждому признаку для стандартизации, правил биннинга, частот и вероятностей положительного класса зависимой переменной в категориях признака использовались все наблюдения набора, часть из которых потом у нас войдет в тестовую выборку (по сути, выборку новых данных).

Если мы используем перекрестную проверку и выполняем перечисленные операции до перекрестной проверки, получается, что в каждом проходе перекрестной проверки для вычисления среднего и стандартного отклонения по каждому признаку для стандартизации, правил биннинга, частот и вероятностей положительного класса зависимой переменной в категориях признака использовались

все наблюдения набора, часть из которых у нас теперь находится в тестовом блоке (по сути, выборке новых данных). В таких случаях в Python используем классы `ColumnTransformer` и `Pipeline`. Случайное разбиение на обучающую и тестовую выборки и перекрестная проверка используются для сравнения конвейеров базовых моделей со значениями гиперпараметров по умолчанию. При подборе гиперпараметров лучшей практикой является комбинированная проверка, сочетающая случайное разбиение на обучающую и тестовую выборки и перекрестную проверку.

До разбиения на обучающую и тестовую выборки / до цикла перекрестной проверки



После разбиения на обучающую и тестовую выборки / внутри цикла перекрестной проверки

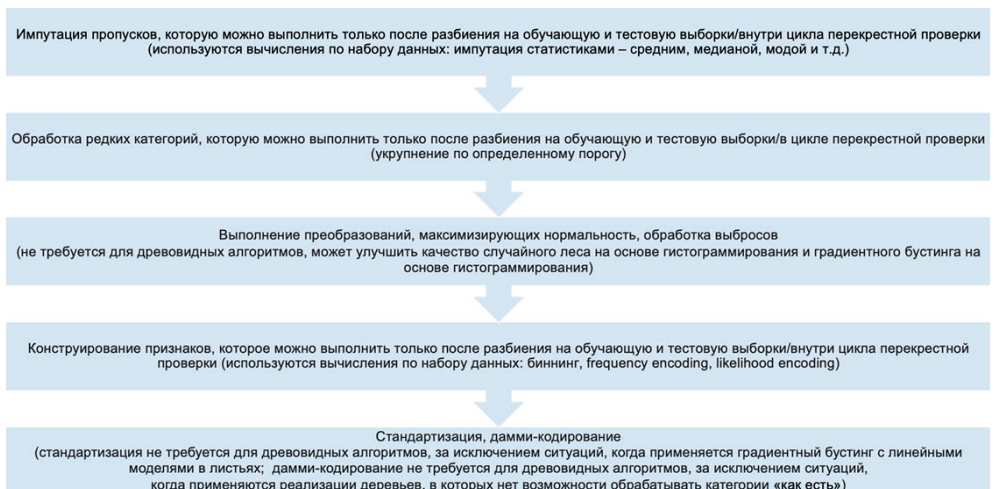


Рис. 1 План предварительной подготовки данных

2. Формирование выборки

2.1. ГЕНЕРАЛЬНАЯ И ВЫБОРОЧНАЯ СОВОКУПНОСТИ

Генеральная совокупность, или популяция (population), – совокупность всех объектов (единиц), относительно которых предполагается делать выводы при изучении конкретной задачи.

Генеральная совокупность состоит из всех объектов, которые имеют качества, свойства, интересующие исследователя. Например, в исследованиях телевизионной аудитории, проводимых компанией Mediascope, генеральной совокупностью будет население России в возрасте 4 лет и старше, проживающее в городах от 100 000 и более. А в исследованиях читательской аудитории, проводимых этой же компанией, генеральной совокупностью будет уже население России в возрасте 16 лет и старше, проживающее в городах от 100 000 и более. В исследованиях политических предпочтений в преддверии президентских выборов генеральной совокупностью будет население России в возрасте 18 лет и старше (поскольку право голосовать гражданин получает с 18 лет). В банковском скоринге генеральной совокупностью считаются все потенциально возможные заемщики банка. В таких случаях принято считать, что объем генеральной совокупности у нас неизвестен.

Выборка, или выборочная совокупность (sample), – набор объектов, выбранных с помощью определенной процедуры из генеральной совокупности для участия в исследовании.

Цель проведения выборочных обследований – на основе выборки сформировать суждение обо всей генеральной совокупности.

Допустим, нам необходимо провести исследование, цель которого – убедиться, что груши на дереве созрели. Решение заключается в том, чтобы сорвать несколько груш с дерева и попробовать их. Генеральная совокупность – все груши на дереве. Выборочная совокупность – сорванные груши с дерева. Если все сорванные груши созрели, то с большой вероятностью можно сделать вывод, что и все остальные груши на дереве тоже созрели. Если необходимо узнать, все ли груши созрели во всем саду, то это будет уже другая генеральная совокупность – груши во всем саду. Исследование будет состоять в том, чтобы сорвать и попробовать груши с разных деревьев.

2.2. ХАРАКТЕРИСТИКИ ВЫБОРКИ

Перечень всех единиц наблюдения генеральной совокупности с базовой информацией представляет **основу выборки**. **Базовая информация** – набор характеристик, известных до проведения обследования для каждого элемента основы выборки (например, фамилия, имя и отчество респондента, адрес предприятия, регион проведения интервью и другие характеристики).

Элементы отбора при формировании выборочной совокупности называются **единицами отбора**. Объект, признаки которого подлежат регистрации, называется **единицей наблюдения**. Обычно единицей наблюдения в социологических опросах является конкретный человек, который будет отвечать на вопрос анкеты. Единица наблюдения может совпадать или не совпадать с единицей

отбора. При простой случайной выборке единицы отбора и единицы наблюдения совпадают. В случае использования многоступенчатой выборки сначала отбираются регионы, потом населенные пункты, затем предприятия или адреса проживания семей (все они и будут единицами отбора), и лишь на последнем этапе будут отобраны конкретные единицы наблюдения – респонденты.

Количество элементов выборки называется **объемом (размером)** выборки.

Соответствие характеристик выборки характеристикам популяции или генеральной совокупности в целом называется **репрезентативностью**. Репрезентативность определяет, насколько возможно обобщать результаты исследования с привлечением определенной выборки на всю генеральную совокупность, из которой она была отобрана. Корректный вывод обо всей генеральной совокупности можно сделать только на основании репрезентативной выборки. Поэтому при формировании выборки должен быть такой отбор элементов, чтобы выборка была репрезентативной.

В США одним из наиболее известных исторических примеров нерепрезентативной выборки считается случай, произошедший во время президентских выборов в 1936 году.

Журнал «Литрери Дайджест», успешно прогнозировавший события нескольких предшествующих выборов, ошибся в своих предсказаниях, разослав десять миллионов пробных бюллетеней своим подписчикам, а также людям, выбранным по телефонным книгам всей страны и людям из регистрационных списков автомобилей. В 25 % вернувшихся бюллетеней (почти 2,5 миллиона) голоса были распределены следующим образом:

57 % отдавали предпочтение кандидату-республиканцу Альфу Лэндону;

40 % выбрали действующего в то время президента-демократа Франклина Рузвельта.

На действительных же выборах, как известно, победил Рузвельт, набрав более 60 % голосов. Ошибка «Литрери Дайджест» заключалась в следующем: желая увеличить репрезентативность выборки, – так как им было известно, что большинство их подписчиков считают себя республиканцами, – они расширили выборку за счёт людей, выбранных из телефонных книг и регистрационных списков. Однако они не учли современных им реалий и в действительности набрали ещё больше республиканцев: во время Великой депрессии обладать телефонами и автомобилями могли себе позволить в основном представители среднего и высшего класса (то есть большинство республиканцев, а не демократов).

В нашем игрушечном примере, когда нам нужно было убедиться, что груши на дереве созрели, примером нерепрезентативной выборки были бы груши, сорванные только с одной, южной стороны дерева. А если бы нам необходимо было узнать, все ли груши созрели во всем саду, то примером нерепрезентативной выборки были бы груши, сорванные с деревьев, которые росли поблизости (допустим, мы поленились пройти в глубь сада).

Отклонение результатов оценки значений, полученных с помощью выборки, от истинных неизвестных значений в генеральной совокупности называется **ошибкой выборки**.

В выборочных обследованиях мы будем оперировать статистиками. **Статистика** – это некоторая функция от выборочных наблюдений, например минимальное значение, среднее арифметическое, стандартное отклонение и др. Допустим, минимальный вес груши, средний вес груши.

Исследование всех объектов генеральной совокупности называется **сплошным обследованием**. Наиболее точные оценки могут быть получены при сплошном наблюдении, однако могут быть сложности. Основные проблемы, возникающие при сплошном наблюдении: ограничение по времени, ограничение финансовых ресурсов, ограничение человеческих ресурсов (здесь речь идет о физических и интеллектуальных ресурсах как опрашиваемых, так и опрошенных).

Понятно, что мы не можем для получения рейтингов кандидатов на пост Президента РФ физически опросить все население России в возрасте от 18 лет и старше. Однако даже если сплошное обследование можно организовать, оно не гарантирует получения надежных результатов. Примером, когда сплошное обследование потерпело неудачу, была сплошная перепись населения России 1897 г. Когда анализировалась численность населения по возрастам, то получалось, что максимальные численности (пики) имели возрасты, кратные 5 и в особенности кратные 10. Большая часть населения в те времена была неграмотна и свой возраст помнила только приблизительно, с точностью до пяти или до десяти лет. Чтобы все-таки узнать, каково было распределение по возрастам на самом деле, нужно было не увеличивать объем данных, а, наоборот, создать выборку из нескольких процентов населения и провести комплексное исследование, основанное на перекрестном анализе нескольких источников: документов, свидетельств и личных показаний. Это дало бы гораздо более точную картину, нежели сплошная перепись. Для решения проблем, возникающих при сплошном обследовании, как раз и используют выборочные обследования.

2.3. ДЕТЕРМИНИРОВАННЫЕ И ВЕРОЯТНОСТНЫЕ ВЫБОРКИ

По способу отбора выборки делятся на:

- **детерминированные;**
- **вероятностные.**

Детерминированный отбор – выборочный метод, в котором не применяется процедура случайного отбора единиц генеральной совокупности. Этот метод основан на индивидуальных суждениях исследователя. Примерами являются экспертный отбор, квотный отбор, отбор методом «снежного кома».

Выборка по методу «снежного кома» строится следующим образом. У каждого респондента, начиная с первого, просят контакты его друзей, коллег, знакомых, которые подходили бы под условия отбора и могли бы принять участие в исследовании. Таким образом, за исключением первого шага, выборка формируется с участием самих объектов исследования. Метод часто применяется, когда необходимо найти и опросить труднодоступные группы респондентов (например, респондентов, имеющих высокий доход, респондентов, принадлежащих к одной профессиональной группе, респондентов, имеющих какие-либо схожие хобби/увлечения и т. д.).

При квотной выборке генеральная совокупность сначала разделяется на непересекающиеся группы. Затем пропорционально из каждой группы выбираются единицы наблюдения на основании предпочтений отбирающего. Например, интервьюер может получить задание отобрать 200 женщин и 300 мужчин возрастом от 45 до 60 лет. Это значит, что внутри каждой квоты интервьюер отбирает респондентов по своим предпочтениям.

Описанный второй шаг формирования квотной выборки относит её к детерминированному типу. Отбор элементов в квотную выборку не является случайным и может быть ненадёжным. Например, интервьюеры могут в первую очередь пытаться опрашивать тех людей, которые выглядят наиболее отзывчивыми или живут поблизости. Соответственно, менее отзывчивые люди или респонденты, живущие в труднодоступных местах, криминогенных районах, в которых интервьюер побоится опрашивать, имеют меньше шансов попасть в выборку.

Квотная выборка полезна, когда время ограничено, отсутствует основа для формирования вероятностной выборки, бюджет исследования небольшой или когда точность результатов не слишком важна.

Вероятностный отбор – выборочный метод, в котором состав выборки формируется случайным образом. В вероятностном отборе каждая единица генеральной совокупности имеет определенную вероятность включения в выборку. Нас будут интересовать вероятностные выборочные методы.

2.4. Виды, методы и способы вероятностного отбора

По виду отбора различают следующие вероятностные выборки:

- выборки с индивидуальным отбором;
- выборки с групповым отбором;
- выборки с комбинированным отбором.

Выборки с индивидуальным отбором осуществляют отбор из генеральной совокупности каждой единицы наблюдения в отдельности. Например, при обследовании удовлетворенности сотрудников предприятия размером заработной платы осуществляется отбор сотрудников.

Выборки с групповым отбором осуществляют отбор групп единиц. Например, при обследовании удовлетворенности сотрудников предприятия размером заработной платы осуществляется отбор отделов предприятия.

По методу отбора различают:

- выборки без возвращения (бесповторный отбор);
- выборки с возвращением (повторный отбор).

В выборках без возвращения (бесповторный отбор) отобранный элемент не возвращается в генеральную совокупность, из которой осуществлялся отбор. В выборках с возвращением (повторный отбор) отобранный объект возвращается в генеральную совокупность и имеет шанс быть отобранным повторно. Использование повторного метода дает бóльшую ошибку выборки, чем использование бесповторного.

По способам отбора различают:

- простой случайный отбор;
- систематический отбор;
- вероятно-пропорциональный отбор;
- расслоенный случайный отбор;
- кластерный (серийный) отбор.

Более подробное обсуждение этих способов выходит за рамки книги, разберем здесь лишь процедуру простого случайного отбора.

При проведении простого случайного отбора каждая единица генеральной совокупности имеет известную и равную вероятность отбора. В простом случайном отборе каждая единица отбирается независимо от другой. Для отбора используется таблица случайных чисел или компьютерная программа.

Здесь отметим, что к повторному отбору приравнивается простой случайный отбор из генеральной совокупности, объем которой неизвестен. При вычислении необходимого объема выборки для построения моделей банковского скоринга как раз предполагают, что имеет место повторный отбор.

2.5. Подходы к определению необходимого

ОБЪЕМА ВЫБОРКИ

Необходимый объем выборки может быть известен по результатам предыдущих аналогичных исследований. Если же объем выборки неизвестен, его необходимо рассчитать.

2.5.1. Определение объема выборки согласно теории выборочных обследований

Согласно теории выборочных обследований объем необходимой выборки зависит от задаваемой точности оценки параметров, дисперсии оцениваемых параметров и способа отбора. Общее правило следующее: чем больше дисперсия оцениваемых параметров, тем больший объем выборки необходим для того, чтобы обеспечить требуемую точность. Поэтому предварительно по отобраным данным необходимо рассчитать дисперсию оцениваемых переменных. В зависимости от величины надежности выбирают значение стандартного нормального распределения.

В банковском скоринге для построения качественной модели данные о «хороших» и «плохих» клиентах максимально должны отражать поток клиентов с улицы.

Предположим, мы хотим быть уверенными на 95 %, что соотношение «хороших» и «плохих» заемщиков в обучающей выборке отражает генеральную совокупность заемщиков. В таких случаях обычно используют следующую формулу определения объема выборки для оценки генеральной доли при повторном случайном отборе (при этом предполагается, что выборка значительно меньше генеральной совокупности):

$$n = \frac{z_\gamma^2 w(1-w)}{\Delta_w^2},$$

где:

n – минимальный объем выборки;

z_γ – значение стандартного нормального распределения, определяемое в зависимости от выбранного доверительного уровня (доверительной вероятности);

w – доля «плохих» на предварительной выборке (может быть получена, исходя из опыта имеющихся априорных знаний);

Δ_w – максимально допустимая предельная ошибка оценки доли «плохих» заемщиков (предельная ошибка выборки).

Доверительный уровень (доверительная вероятность) – это вероятность того, что генеральная доля лежит в границах полученного доверительного интервала: выборочная доля (w) \pm ошибка выборки (Δ_w). Доверительный уровень устанавливает сам исследователь в соответствии со своими требованиями к надежности полученных результатов. Чаще всего применяются доверительные уровни, равные 0,95 или 0,99.

Допустим, среди 700 клиентов предварительной выборки 50 оказались «плохими». Оценка доли «плохих» клиентов, по имеющимся данным, для построения модели составила около 0,07, или 7 %. При таком значении оценки доли предположим, мы хотим ошибиться не более чем на 10 %, что будет соответствовать допустимой предельной ошибке оценки доли 0,007, т. е. $(50 / 700) \cdot 0,1$. При этом задаем 95%-ную доверительную вероятность. В этом случае z -значение стандартного нормального закона распределения составит около 1,96. Вычисляем минимальный объем выборки: $n = \frac{z_\gamma^2 w(1-w)}{\Delta_w^2} = \frac{3,84 \cdot 0,07 \cdot 0,93}{0,000049} = 5102$.

На практике чаще всего нет возможности сформировать предварительную выборку и значение w неизвестно. В таком случае w принимается за 0,5 (самый консервативный сценарий). При этом значении размер ошибки выборки будет максимален.

Допустим, оценка доли «плохих» клиентов неизвестна, принимаем ее за 0,5. При таком значении оценки доли предположим, мы хотим ошибиться не более чем на 10 %. При этом задаем 95%-ную доверительную вероятность. Получаем

$$n = \frac{z_\gamma^2 w(1-w)}{\Delta_w^2} = \frac{3,84 \cdot 0,5 \cdot 0,5}{0,000049} = 19592.$$

На собеседованиях часто просят рассчитать объем выборки с определенной предельной ошибкой. Например, рассчитайте объем выборки, предельная ошибка которой составит 4 %. При этом мы принимаем, что доверительный уровень равен 95 %, а генеральная совокупность значительно больше выборки.

Применяем знакомую формулу $n = \frac{z_\gamma^2 w(1-w)}{\Delta_w^2} = \frac{3,84 \cdot 0,5 \cdot 0,5}{0,0016} = 600$.

При этом не забывайте, вам помимо обучающей выборки еще нужно отложить наблюдения для проверки, здесь, соответственно, нужно выделить ситуацию, когда вы строите и проверяете базовую модель (вам нужен тестовый набор), и ситуацию, когда вы строите модели, настраивая гиперпараметры (вам нужен валидационный набор для настройки гиперпараметров, роль валидационного набора могут выполнять проверочные блоки перекрестной проверки и тестовый набор для итоговой оценки качества).

Когда предполагается, что выборка сопоставима с генеральной совокупностью (обычное явление при опросах организаций в B2B-исследованиях), формула определения объема выборки для оценки генеральной доли будет выглядеть несколько иначе, в ней будет фигурировать показатель объема генеральной совокупности.

$$n = \frac{\frac{z_\gamma^2 w(1-w)}{\Delta_w^2}}{1 + \frac{\frac{z_\gamma^2 w(1-w)}{\Delta_w^2} - 1}{N}}$$

2.5.2. Определение объема выборки согласно правилу NEPV

В практике банковского скоринга для ответа на вопрос об объеме выборки часто используют правило «Number of Events Per Variable» (количество событий на одну переменную, NEPV), сформулированное Фрэнком Харреллом.

Для задачи бинарной классификации оно связывает минимальный объем выборки с количеством «событий» – наблюдений в миноритарной (наименьшей по размеру) категории зависимой переменной и количеством признаков, поданным на вход модели. Согласно этому правилу, необходимо взять количество наблюдений в обучающей выборке, относящихся к миноритарной категории зависимой переменной (в кредитном скоринге это «плохие» заемщики). Это число наблюдений нужно разделить на количество заданных признаков. Для логистической регрессии на один параметр должно приходиться не менее 20 событий, при построении дерева решений CHAID на один признак должно приходиться не менее 50 событий, а для модели случайного леса, градиентного бустинга, SVM и нейронной сети на одну независимую переменную должно приходиться не менее 200 событий.

Для задачи регрессии мы просто берем количество наблюдений и делим на количество признаков, и для линейной регрессии на один параметр должно приходиться не менее 20 наблюдений, для дерева решений CHAID (в тех случаях, когда реализация алгоритма позволяет решать задачу регрессии) на один признак должно приходиться не менее 50 наблюдений, для случайного леса и других сложных моделей на один признак должно приходиться не менее 200 наблюдений. По мнению Фрэнка Харрелла, для дерева решений CART правило NEPV невозможно сформулировать из-за высокой нестабильности метода и склонности к переобучению.

Если правило выполняется для обучающей выборки, то объем выборки для обучения является достаточным. В противном случае необходимо либо увеличить объем выборки, либо сократить количество признаков, подаваемых на вход модели. Затем вся та же самая процедура применяется к тестовой выборке, если правило выполняется, объем выборки для проверки достаточен.

Мы решаем задачу классификации, и у нас есть общая выборка объемом 4424 клиента, классифицированных на два класса: класс *Остается* (2492 клиента) и класс *Уходит* (1932 клиента). Мы разбили выборку на обучающую и тестовую и получили следующее распределение классов в выборках: *Остается* (1746 клиентов) и *Уходит* (1334 клиента).

Выясняем, достаточен ли объем выборки для обучения. У нас 1746 оставшихся клиентов, 1334 ушедших клиента и 9 независимых переменных. Ми-

норитарный класс – класс *Уходит*. Проверая выполнение правила NEPV, мы получаем $1334 / 9 = 148,2$. Наша выборка обеспечивает достаточное количество событий на одну переменную, и мы можем использовать эту выборку для обучения.

Выясняем, достаточен ли объем выборки для проверки. У нас 746 оставшихся клиентов, 598 ушедших клиентов. Проверая выполнение правила NEPV, мы получаем $598 / 9 = 66,4$. Наша выборка обеспечивает достаточное количество событий на одну переменную, и мы можем использовать эту выборку для проверки.

Если выполняется перекрестная проверка, роль обучающей выборки выполняет набор обучающих блоков, а роль тестовой выборки выполняет тестовый блок.

Опять же напомним: данная схема работает для построения базовой модели без подбора гиперпараметров.

2.5.3. Определение объема выборки с помощью кривых обучения и валидации

Кроме того, необходимый объем выборки можно определить с помощью кривых обучения и валидации. Их можно построить с помощью функции `learning_curve()`. Она запускает перекрестную проверку на наборах данных разного объема. Генератор перекрестной проверки разбивает весь набор данных k раз на обучающую выборку и тестовую выборку. В итоге для набора соответствующего размера мы получаем метрику для обучающей выборки, усредненную по k проходам, и метрику для тестовой выборки, усредненную по k проходам.

```
sklearn.model_selection.learning_curve(estimator, X, y, groups=None,
                                       train_sizes=array([0.1, 0.33, 0.55, 0.78, 1.]),
                                       cv=None, scoring=None,
                                       exploit_incremental_learning=False,
                                       n_jobs=None, shuffle=False,
                                       random_state=None, return_times=False)
```

estimator, ← Модель машинного обучения, которая подвергается проверке
 X, ← Массив признаков
 y, ← Массив меток
 groups=None, ← Идентификатор групп (только для стратегии проверки GroupKFold)
 train_sizes=array([0.1, 0.33, 0.55, 0.78, 1.]), ← Относительное (если переданы числа с плавающей точкой) или абсолютное (если переданы целые числа) количество наблюдений, которое будет использоваться для формирования кривой обучения. Обратите внимание, что для классификации количество наблюдений должно быть достаточно большим для адекватного представления классов. По умолчанию используется `np.linspace(0.1, 1.0, 5)`
 cv=None, ← Стратегия перекрестной проверки
 scoring=None, ← Метрика качества
 exploit_incremental_learning=False, ← Инкрементное обучение для ускорения (только для моделей, поддерживающих его)
 n_jobs=None, ← Количество используемых ядер процессора
 shuffle=False, ← Перемешивание данных
 random_state=None, ← Стартовое значение генератора псевдослучайных чисел
 return_times=False, ← Возвращает время обучения и оценки

Рис. 2 Параметры функции `learning_curve()`

В результате функция `learning_curve()` возвращает:

- `train_sizes_abs` – количество наблюдений, использованное для построения кривой обучения;
- `train_scores` – значения метрики на обучающих выборках перекрестной проверки;
- `test_scores` – значения метрики на тестовых выборках перекрестной проверки;
- `fit_times` – время, затраченное на обучение, в секундах;
- `score_times` – время, затраченное на оценку качества, в секундах.

Давайте загрузим необходимые библиотеки, классы и функции.

```
# импортируем библиотеки, классы и функции
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
from sklearn.model_selection import learning_curve
from sklearn.model_selection import ShuffleSplit
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
```

Теперь загрузим данные. Данные записаны в файле *Response.csv*. Исходная выборка содержит записи о 30 259 клиентах, классифицированных на два класса: 0 – отклика нет (17 170 клиентов) и 1 – отклик есть (13 089 клиентов).

По каждому наблюдению (клиенту) фиксируются следующие переменные (характеристики):

- категориальный признак *Ипотечный кредит [mortgage]*;
- категориальный признак *Страхование жизни [life_ins]*;
- категориальный признак *Кредитная карта [cre_card]*;
- категориальный признак *Дебетовая карта [deb_card]*;
- категориальный признак *Мобильный банк [mob_bank]*;
- категориальный признак *Текущий счет [curr_acc]*;
- категориальный признак *Интернет-доступ к счету [internet]*;
- категориальный признак *Индивидуальный займ [perloan]*;
- категориальный признак *Наличие сбережений [savings]*;
- категориальный признак *Пользование банкоматом за последнюю неделю [atm_user]*;
- категориальный признак *Пользование услугами онлайн-маркетплейса за последний месяц [markpl]*;
- количественный признак *Возраст [age]*;
- количественный признак *Давность клиентской истории [cus_leng]*;
- категориальная зависимая переменная *Отклик на предложение новой карты [response]*.

```
# загружаем данные
data = pd.read_csv('Data/Response.csv', sep=';')
data.head(3)
```

| | mortgage | life_ins | cre_card | deb_card | mob_bank | curr_acc | internet | perloan | savings | atm_user | markpl | age | cus_leng | response |
|---|----------|----------|----------|----------|----------|----------|----------|---------|---------|----------|--------|------|-------------------|----------|
| 0 | No | No | No | No | No | No | No | No | No | No | No | 18.0 | less than 3 years | No |
| 1 | Yes | Yes | NaN | NaN | Yes | No | NaN | NaN | NaN | Yes | No | 18.0 | NaN | Yes |
| 2 | Yes | Yes | NaN | Yes | No | No | No | No | No | No | Yes | NaN | from 3 to 7 years | Yes |

Формируем массив меток и массив признаков, создаем списки переменных и трансформеры, с помощью класса *ColumnTransformer* сопоставляем транс-

формеры со списками переменных, создаем конвейер для логистической регрессии и конвейер для градиентного бустинга, каждой модели машинного обучения будет соответствовать своя последовательность моделей предварительной подготовки данных.

```
# создаем массив меток и массив признаков
y = data.pop('response')

# создаем списки категориальных
# и количественных столбцов
categorical_features = data.select_dtypes(
    include='object').columns.tolist()
numeric_features = data.select_dtypes(
    exclude='object').columns.tolist()

# создаем трансформеры
numeric_transformer_logreg = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

numeric_transformer_boost = Pipeline([
    ('imputer', SimpleImputer(strategy='median'))
])

categorical_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='constant',
                               fill_value='missing')),
    ('onehot', OneHotEncoder(sparse=False,
                              handle_unknown='ignore'))
])

# сопоставляем трансформеры спискам переменных
# для логистической регрессии
preprocessor_logreg = ColumnTransformer([
    ('num', numeric_transformer_logreg, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

# сопоставляем трансформеры спискам переменных
# для градиентного бустинга
preprocessor_boost = ColumnTransformer([
    ('num', numeric_transformer_boost, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

# формируем итоговый конвейер
pipe_logreg = Pipeline([
    ('preprocessor', preprocessor_logreg),
    ('logreg', LogisticRegression(solver='lbfgs', max_iter=400))
])

pipe_boost = Pipeline([
    ('preprocessor', preprocessor_boost),
    ('boost', GradientBoostingClassifier(
        random_state=42))]])
```

Теперь пишем функцию, которая будет строить графики на основе результатов, возвращенных функцией `learning_curve()`.

```
# пишем функцию, которая строит графики
# по результатам функции learning_curve()
def plot_learning_curve(estimator,
                        title,
                        X,
                        y,
                        axes=None,
                        ylim=None,
                        cv=None,
                        n_jobs=None,
                        train_sizes=np.linspace(.1, 1.0, 5)):
    """
    Строит 3 графика: кривые обучения и валидации, кривую зависимости между
    объемом обучающих данных и временем обучения, кривую зависимости между
    временем обучения и оценкой качества.

    Параметры
    -----
    estimator : модель машинного обучения для проверки.
    title : заголовок диаграммы.
    X : массив признаков.
    y : массив меток.
    axes : задаем область рисования (Axes) для построения кривых.
    ylim : задает минимальное и максимальное значения по оси y.
    cv : стратегия перекрестной проверки.
    n_jobs : количество используемых ядер процессора.
    train_sizes : абсолютное или относительное количество наблюдений.
    """
    if axes is None:
        _, axes = plt.subplots(1, 3, figsize=(20, 5))
    axes[0].set_title(title)
    if ylim is not None:
        axes[0].set_ylim(*ylim)
    axes[0].set_xlabel("Обучающие наблюдения")
    axes[0].set_ylabel("Оценка")
    train_sizes, train_scores, test_scores, fit_times, _ = \
        learning_curve(estimator, X, y, cv=cv,
                      scoring='roc_auc', n_jobs=n_jobs,
                      train_sizes=train_sizes,
                      return_times=True)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    fit_times_mean = np.mean(fit_times, axis=1)
    fit_times_std = np.std(fit_times, axis=1)

    # строим кривые обучения и валидации
    axes[0].grid()
    axes[0].fill_between(train_sizes, train_scores_mean - train_scores_std,
                        train_scores_mean + train_scores_std, alpha=0.1,
                        color="r")
```

```

axes[0].fill_between(train_sizes, test_scores_mean - test_scores_std,
                    test_scores_mean + test_scores_std, alpha=0.1,
                    color="g")
axes[0].plot(train_sizes, train_scores_mean, 'o-', color="r",
            label="Средняя оценка на обуч. блоках")
axes[0].plot(train_sizes, test_scores_mean, 'o-', color="g",
            label="Средняя оценка на тест. блоках")
axes[0].legend(loc="best")

# строим график зависимости между объемом
# обучающих данных и временем обучения
axes[1].grid()
axes[1].plot(train_sizes, fit_times_mean, 'o-')
axes[1].fill_between(train_sizes, fit_times_mean - fit_times_std,
                    fit_times_mean + fit_times_std, alpha=0.1)
axes[1].set_xlabel("Обучающие наблюдения")
axes[1].set_ylabel("Время обучения")
axes[1].set_title("Масштабируемость модели")
# строим график зависимости между временем
# обучения и оценкой качества
axes[2].grid()
axes[2].plot(fit_times_mean, test_scores_mean, 'o-')
axes[2].fill_between(fit_times_mean, test_scores_mean - test_scores_std,
                    test_scores_mean + test_scores_std, alpha=0.1)
axes[2].set_xlabel("Время обучения")
axes[2].set_ylabel("Оценка")
axes[2].set_title("Качество модели")
return plt

```

А сейчас будем строить графики кривых обучения и валидации.

```

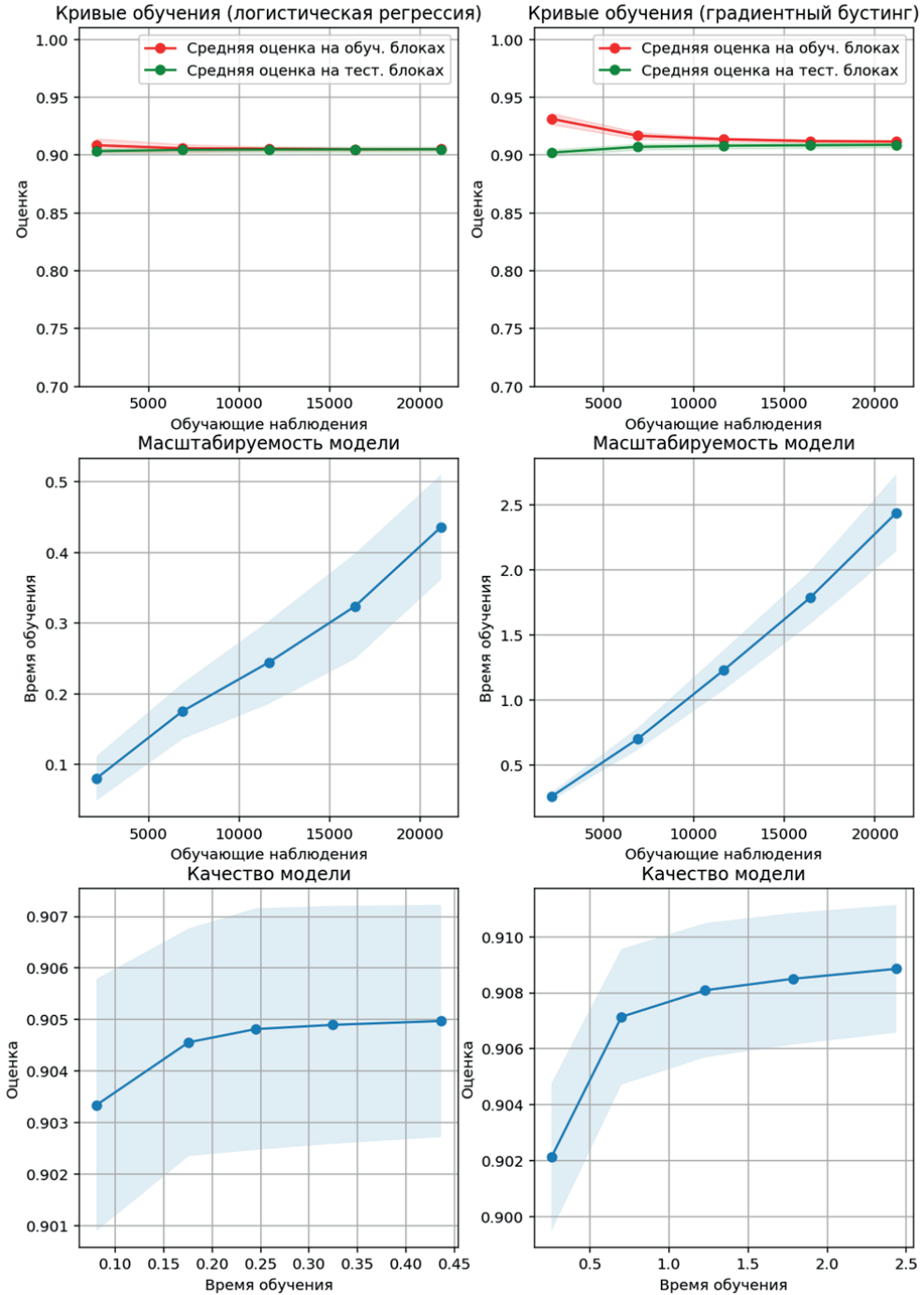
# задаем сетку и размеры графиков
fig, axes = plt.subplots(3, 2, figsize=(10, 15))
# задаем стратегию перекрестной проверки
cv = ShuffleSplit(n_splits=20, test_size=0.3, random_state=42)

# задаем заголовок
title = "Кривые обучения (логистическая регрессия)"
# строим графики для логистической регрессии
plot_learning_curve(pipe_logreg, title, data, y, axes=axes[:, 0],
                    ylim=(0.7, 1.01),
                    cv=cv, n_jobs=4)

# задаем заголовок
title = "Кривые обучения (градиентный бустинг)"
# строим графики для градиентного бустинга
plot_learning_curve(pipe_boost, title, data, y, axes=axes[:, 1],
                    ylim=(0.7, 1.01),
                    cv=cv, n_jobs=4)

# выводим графики
plt.show()

```



Для каждой модели машинного обучения мы выводим по три графика: кривые обучения и валидации, кривую зависимости между объемом обучающих

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru