Александр Фоменко

Предсказываем тренды

С Rattle и R в мир моделей классификации

Александр Фоменко

Предсказываем тренды. С Rattle и R в мир моделей классификации

«Издательские решения»

Фоменко А.

Предсказываем тренды. С Rattle и R в мир моделей классификации / А. Фоменко — «Издательские решения»,

ISBN 978-5-44-966305-4

Книга является практическим руководством по обучению моделей предсказаниям трендов на рынке Форекс. Берем исторические значения исходных данных — котировок, индикаторов, макроэкономических данных, и на них учим модель предсказывать «лонги-шорты».Данная книга является практическим применением пакета Rattle к рынку Форекс и терминалу MT4 с комментариями идеологии моделей классификации и их оценки.Книга доступна новичкам, а также полезна опытным трейдерам в терминале MT4.

ISBN 978-5-44-966305-4

Содержание

Предисловие 6
Организация материала 7
Текущее состояние 8
Часть 1. Введение в предсказательное моделирование 9
1. Введение 10
1.1. Анализ, прогноз, предсказание 10
1.2. Процесс предсказательного моделирования 10
1.3. Терминология 12
1.4. Используемые наборы данных 14
2. Предварительная обработка данных 15
2.1. Преобразование отдельных предикторов 16
2.2. Преобразование групп предикторов 16
2.3. Обработка пропущенных значений 18
2.4. Удаление предикторов 19
2.5. Добавление предикторов 20
2.6. Группировка предикторов 20
2.6. Функции R 20
3. Переобучение и настройка модели 22
3.1. Проблема переобучения 22
3.2. Настройка модели 22
3.3. Разделение данных 23
3.4. Методы ресемплирования 24
3.5. Функции R 26
4. Регрессионные модели 28
4.1. Результативность регрессионных моделей 28
4.2. Линейные регрессионные модели 28
4.3. Нелинейные регрессионные модели 29
Конец ознакомительного фрагмента. 31

Предсказываем тренды
С Rattle и R в мир моделей классификации

Александр Фоменко

ISBN 978-5-4496-6305-4
Создано в интеллектуальной издательской системе Ridero

Предисловие
О чем?

При построении торговых систем вообще, и с использованием терминала MT4/5 в частности, приходится решать целый комплекс взаимосвязанных задач.

Изначально, целью построения торговой системы является *предсказание* поведения некоторого рыночного инструмента, например, валютной пары. Цели предсказания могут быть разными, мы же ограничимся предсказанием *трендов*, а точнее предсказанием *роста (лонгов)* или *падения («шортов»)* значений котировки валютной пары. Кроме этого будем предсказывать боковики – нахождение *вне* рынка.

Для кого?

Книга доступна для многих читателей и не обязательно только тем, кто силен в информатике или статистике. С помощью ***Rattle*** практически любой желающий сможет построить основную часть торговой системы – предсказание котировки, а затем, при отсутствии необходимого опыта и знаний, сделать заказ реальной торговой системы, изложив свои мысли в виде готового кода на ***R.***

Для искушенного в ***R*** пользователя ***Rattle*** будет также полезен: позволяет быстро апробировать идеи в исходных данных, целевых переменных, используемых моделях, а затем перейти к соответствующим пакетам ***R,*** имеющим значительно больший по сравнению с ***Rattle*** функционал.

Данная книга является руководством по использованию пакета ***Rattle*** (Простое обучение аналитическим инструментам ***R)***, который оформлен в виде GUI – графического пользовательского интерфейса, позволяющего значительно упростить использование могучих средств ***R*** и необходимых для поставленной задачи пакетов.

Почему *Rattle*?

В качестве инструмента для предсказания поведения валютных пар выберем систему ***R,*** которая идеально подходит для задач предсказания на финансовых рынках и, в частности, предсказания поведения валютных пар. Вместе с тем ***R*** остается, прежде всего, языком программирования для высоко квалифицированного статистика и для многих остается вне досягаемости. Сложность самой ***R*** усугубляется тем обстоятельством, что инструменты для предсказания являются многочисленными и рассредоточены по многим *пакетам*, которые и составляют основную функциональность ***R.***

Rattle объединяет множество пакетов ***R,*** которые важны для построения торговых систем, но часто не легки для использования новичком. Понимание ***R*** не требуется, чтобы начать с ***Rattle***. Но результатом работы с ***Rattle*** будет код на ***R,*** который может быть использован при построении реальной торговой системы. И на этом этапе потребуется знание ***R.***

В любом случае ***Rattle*** является незаменимым инструментом на этапе проектирования торговой системы, позволяет даже новичкам быстро посмотреть результат тех или иных идей и получить их оценку.

Пакет ***Rattle*** (Уильямс, 2009) является бесплатным программным обеспечением с открытым исходным кодом, созданным в рамках статистического пакета программного обеспечения ***R*** (R Рабочая группа Разработки, 2011). Как бесплатное программное обеспечение исходный код ***Rattle*** и ***R*** доступен всем без ограничения. Исходный код ***Rattle*** написан на ***C*** и всем разрешено, и действительно поощряется, рассмотрение исходного кода для обучения, его понимания, проверки и расширения.

Организация материала

Книга состоит из следующих частей.

Часть 1. Введение в предсказательное моделирование изложено в главах 1—11. В этой части книги сжато, но достаточно подробно на описательном уровне рассматриваются основные понятия предсказательного моделирования. Необходимость этой части проистекает из того, что авторы ***Rattle*** не объясняют смысл и взаимодействие различных частей своей системы. Изучение первой части позволит осмысленно подобрать в ***Rattle*** инструменты для решения конкретно Вашей задачи.

Часть 2. Краткое описание ***Rattle*** изложено в главе 12. Эта часть полезна как на этапе первоначального знакомства с ***Rattle***, так и на этапе постоянного использования в качестве краткого справочника.

Часть 3. Полное описание ***Ratlle*** 13—29. Эта часть книги представляет собой перевод руководства по ***Rattle***. К авторскому руководству добавлены примеры для рынка Форекс, а также приведены переводы синтаксиса команд ***R,*** которые использует ***Rattle***.

Текст книги содержит большое количество программного кода на языке ***R*** и *MQL4* торгового терминала MT4 разработки MetaQuotes Software Corp. Это программный код по праву можно считать еще одной частью книги. При желании изложенный в книге код, а также необходимые для него данные, читатель может использовать для повторения, подражания или модификации. Программный код и данные доступны читателю на ЯндексДиск по ссылке https://yadi.sk/d/_pRbllwlHpxnMQ.

Текущее состояние

Новые версии ***R*** публикуются два раза в год – в апреле и октябре. ***R*** имеет несколько миллионов пользователей, что гарантирует очень малое количество ошибок в коде. Система статистики и графики ***R*** очень популярна, быстро расширяется за счет пакетов, имеет обширную информационную поддержку в виде публикаций, учебников и монографий.

Примеры, включенные в эту книгу, используют версию 3.1.1 ***R*** и версию 3.0.2 r169 ***Rattle***. ***Rattle*** – развивающийся пакет и, хотя понятия остаются, Подробности меняются. Поэтому не следует удивляться, если скриншоты, приведенные в этой книге, будут отличаться от Ваших скриншотов.

Часть 1. Введение в предсказательное моделирование

Первая часть книги является введением в идеологию предсказательного моделирования. В этой части книги сжато, но достаточно подробно на описательном уровне рассматриваются основные понятия предсказательного моделирования.

Первая часть книги дополняет техническую документацию по ***Rattle,*** так как авторы ***Rattle*** предполагают, что пользователь их продукта знаком с терминологией, проблемами и инструментами, существующими в области предсказательного моделирования.

Первая часть книги будет полезна всем без исключения проектировщиками предсказательных моделей вообще, а не только пользователям ***Rattle.*** Излагаемые в первой части материалы охватывают более широкий круг вопросов, чем необходимо для работы с ***Rattle,*** готовя читателя к использованию других, аналогичных, но более развитых инструментов для построения предсказательных моделей.

1. Введение

1.1. Анализ, прогноз, предсказание

На финансовых рынках будем различать следующий набор действий: анализ, прогноз и предсказание.

Анализ позволяет ответить нам на вопрос: почему так произошло? Например, можно поставить вопрос: почему произошло падение курса доллара по отношению к евро? Без анализа прошлого, без анализа исторических данных невозможен переход к последующим этапам – *прогнозу* или *предсказани*ю.

Прогноз. Значение слова «прогноз» буем понимать так, как это понимается в ***R*** под словом «*forecast*» – для прогноза следующего значения используется предыдущее значение, полученное в результате предыдущего шага прогноза. Пакет *forecast* является примером такого понимания значения слова «прогноз».

Предсказание. Значение слова *«предсказание»* будем понимать в смысле универсальной функции *predict* — предсказание будущего на любое число шагов вперед с использованием имеющихся данных.

В данной книге исторические данные используются для **обучения** моделей, которые в последующем используются для **предсказания** будущего.

1.2. Процесс предсказательного моделирования

Существует несколько аспектов в процессе построения модели, которые следует обсудить далее, особенно новичкам в предсказательном моделировании.

Технология предсказательного моделирования выглядит следующим образом:

– на основе некоторого набора исходных данных производится обучение *модели*;

– в последующем обученная модель используется для предсказания целевой переменной на новом наборе данных;

– в зависимости от того, чему мы учили модель: предсказывать целевую переменную на сегодня, на завтра или на ***n*** шагов вперед, мы и получим соответствующее предсказание.

1.2.1. Разделение данных

То, как выделяются данные определенным этапам (например, обучение модели, оценка результативности), является важным аспектом моделирования. Наш главный интерес, к примеру, состоит в предсказании тренда на новых данных, которые отсутствуют в момент обучения модели. Это означает, что до некоторой степени необходимо проверить, как хорошо модель *экстраполирует* на новых котировках. Если бы мы интересовались предсказанием на тех же самых данных (то есть, *интерполяцией*), то можно было бы взять простую случайную выборку данных. То, как определены набор данных для обучения и наборы данных для тестирования и проверки, должно отразить применение модели.

Сколько данных должно быть выделено обучающему набору и тестовому? Это обычно зависит от ситуации. Если пул данных небольшой, решения разделения данных могут быть критическими. Небольшой (десятки наблюдений) тестовый набор исходных данных ограничил бы суждения о результативности модели. В этом случае, уверенность в результатах могли бы предоставить методы *ресемплирования,* которые могли бы решить проблему.

На финансовых рынках достаточно часто доступны большие наборы данных. Обычно доступны наборы данных с несколькими тысячами наблюдений, например, цен валютных пар. Казалось бы, что на финансовых рынках отсутствуют проблемы небольших по объему исход-

ных данных. Однако это не так. К примеру, мы обучили модель на растущем рынке, а тестируем на падающем рынке, и выясняется, что модель дает убыток на этом тестовом наборе данных. На этом примере мы видим, что проблема в обязательности разделения котировок для обучения и тестирования остается и для больших наборов данных, так как при разделении необходимо обеспечить примерно равное количество «лонгов» и «шортов».

1.2.2. Целевая переменная

Несмотря на кажущуюся простоту, выбор цели предсказания и данных, которые материализуют эту цель в виде набора цифр, носит краеугольный характер.

Обратимся к идее предсказывать тренд. Эта идея опирается на желание торговать тренды.

Из определения тренда: «тренд растущий, если следующая цена больше предыдущей цены» и, наоборот, для падающего тренда. Из определения следует, что необходимо предсказывать цену валютной пары. Было 1.3500 для *eurusd*, предсказали 1.3550 – растущий тренд, покупаем.

Но базовыми приказами являются приказы «купить» и «продать», а мы предсказали *величину* цены. А величина цены используется, к примеру, в торговых системах на пробой уровня. Чтобы реализовать план по торговле трендов, надо будет произвести дополнительное действие по сравнению цен. При этом очевидно, что мы предсказываем не то, что собирались торговать!

Поэтому, если мы собрались делать трендовую торговую систему, то модель должна сразу предсказывать тренд. Учить модель надо трендам, целевая переменная должна принимать только два значения «купить» и «продать» или в закодированном (категориальном) виде «1» и «-1».

Можно уточнить целевую переменную, имеющую два значения за счет того, что неплохо бы предсказывать и боковики, т.е. иметь целевую переменную, принимающую значения *«купить», «продать»* и «*вне рынка»*. Или закодировав эти значения как: «1», «-1» и «0».

Различие моделей, которые используют совокупность исходных данных для вычисления будущей *величины* цены финансового актива, и моделей, которые относят совокупность исходных данных к некоторому классу, является принципиальным. Первый тип моделей относится к *регрессионным моделям*, а второй тип моделей относится к *классификационным моделям*.

По предсказательным моделям регрессионного типа вычисляется значение некоторой величины в будущем, и когда это будущее наступит, то у нас будет фактическое значение этой предсказанной величины.

По предсказательным моделям классификационного типа вычисляется класс, к которому будет отнесена совокупность поступивших на момент предсказания исходных данных.

Рассмотренные варианты не исчерпывают всего разнообразия целевых переменных, возможных на финансовых рынках. Но вывод из данного раздела: целевая переменная должно точно соответствовать целям торговой системы.

1.2.3. Независимые переменные

Независимые переменные, в дальнейшем *предикторы*, независимы в том смысле, что поступают в модель извне, являются внешними, измеряемыми переменными, или переменными, вычисленными на основе этих внешних переменных. Например, любые экономические, финансовые данные, включая котировки валютных пар, являются независимыми переменными, так как их значения образуются в результате деятельности субъектов на рынке. К этой же категории переменных относятся и индикаторы из технического анализа, которые вычисляются на основе котировок.

Выбор независимых переменных не менее важен, чем выбор целевой переменной. Более того, именно выбор независимых переменных определяет успешность моделирования. Основное время, затраченное на разработку модели, уходит как раз на анализ и подбор набора независимых переменных.

Этот вопрос рассмотрим в отдельных разделах.

1.2.4. Оценка результативности модели

Тип модели предполагает разные типы оценок.

Для регрессионных моделей – это ошибка предсказания, полученная как разность между предсказанной и фактической величиной (к примеру, RMSE).

Для классификационных моделей – это рассогласование, полученное как совпадение/несовпадение фактических и предсказанных классов.

1.2.5. Выбор модели

Наличие оценки результативности модели позволяет выбрать лучшую модель. Это можно сделать, если «лучшая» модель сильно отличается от своих конкурентов. Если это не так, то, отбросив «худшую» модель при ее наличии, можно сделать предсказание, используя имеющиеся предсказания моделей в качестве предикторов для окончательного предсказания.

1.2.6. Итоги

В первом приближении создание модели кажется ясным: выбираем метод моделирования, учим модель на наборе данных обучения – все готово, можно предсказывать.

Вам очень повезет, если столь просто удастся создать надежную, устойчивую модель, работающую на новых наблюдениях.

Чтобы получить модель, имеющую примерно одинаковые оценки результативности вне набора обучения следует сначала понять данные и цель моделирования. После понимания данных и целей, можно предварительно обработать и разделить данные. После этих шагов можно начинать создание, оценку и выбор моделей. Только после того, как эти шаги сделаны, мы, наконец, начнем создание, оценку и выбор моделей.

Существует целый ряд общих причин неудачности предсказательных моделей:

– не адекватная предварительная обработка данных;

– не адекватная проверка модели;

– неоправданная экстраполяция (применение к данным, которые имеют слабое отношение к обучающему набору);

– **наиболее важное**: *переобучение* модели на обучающем наборе данных.

1.3. Терминология

Предсказательное моделирование является одним из многих наименований, которые относятся к процессу выявления отношений внутри данных для предсказания желаемого результата. Машинное обучение, искусственный интеллект, распознавание образов, интеллектуальный анализ данных, предсказательная аналитика – много научных областей сделало вклад, что привело к синонимии разных понятий.

*Предсказательное моделир*ование – это процесс, с помощью которого модель создает, выбирает или пытается сделать лучшее предсказание вероятности результата.

Набор данных – это общий и расплывчатый термин.

Набор данных на внешнем носителе – это файл данных по тексту книги. По расширению файла можно судить о кодировке и, частично, о структуре файла. В пакете ***Rattle*** допустимы разные файлы. Наибольший интерес для нас будут представлять файлы со следующими расширениями:

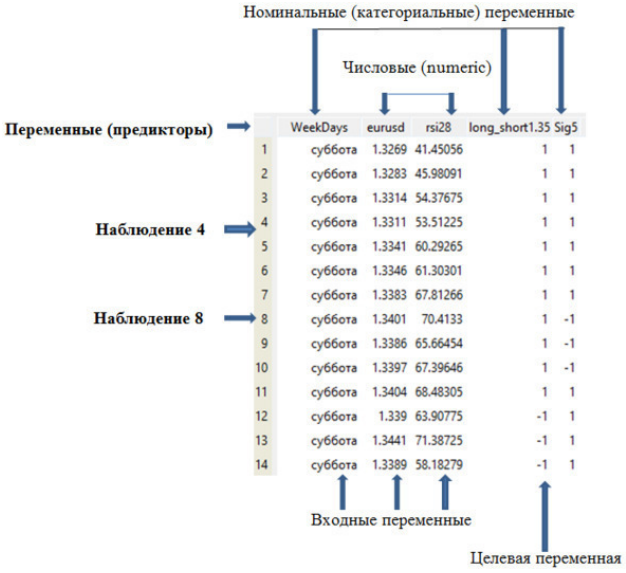
– *.txt* – обычный текстовый файл;

– *.csv* – текстовый файл Excel;

– *.RData* – файл ***R,*** в котором хранится рабочая область.

Набор данных в памяти – это некоторая совокупность данных, имеющая структуру. В терминах ***R*** – это вектор, матрица, фрейм данных или совокупность этих данных.

Матрица (редко) и фрейм данных в ***Rattle*** представлены таблицей, имеющей следующий вид:



	WeekDays	eurusd	rsi28	long_short1.35	Sig5
1	суббота	1.3269	41.45056	1	1
2	суббота	1.3283	45.98091	1	1
3	суббота	1.3314	54.37675	1	1
4	суббота	1.3311	53.51225	1	1
5	суббота	1.3341	60.29265	1	1
6	суббота	1.3346	61.30301	1	1
7	суббота	1.3383	67.81266	1	1
8	суббота	1.3401	70.4133	1	-1
9	суббота	1.3386	65.66454	1	-1
10	суббота	1.3397	67.39646	1	-1
11	суббота	1.3404	68.48305	1	1
12	суббота	1.339	63.90775	-1	1
13	суббота	1.3441	71.38725	-1	1
14	суббота	1.3389	58.18279	-1	1

Рис.1.1. Фрейм данных, представленный в ***Rattle***

Термины *выборка (sample), наблюдение (observation), пример, экземпляр (instance)* относится к отдельной строке данных. Термин *sample* также может относить к подмножеству наблюдений, которые объединены, например целью последующего использования – *обучающая* выборка или *обучающий* набор данных. Значение термина выборка будет понятно из контекста употребления термина.

Обучающий набор содержит данные, которые использовались для обучения модели, в то время как *тестовый* и *проверочный* наборы используются исключительно для оценки результативности модели.

Предикторы, *независимые переменные, атрибуты или дескрипторы* являются данными, которые используются в качестве входных переменных в уравнении предсказания. На рис.1.1 показаны три предиктора, которые играют роль в модели «входных переменных».

Результат, зависимая переменная, целевая переменная, класс, отклик (response) относится к результирующему событию или количеству, которое предсказывается.

У *числовой переменной* есть значение, которое является целым числом или вещественным числом, такими как цена валютной пары, объем торгов, процентная ставка. Числовые переменные также известны как *количественные переменные*. Числовые переменные могут быть дискретными (целыми числами) или непрерывными (действительными). Например, котировка валютной пары. У числовой переменной обычно имеется числовой масштаб. Для валютной пары *eurusd* числовой масштаб – это диапазон от 0.5 до 2.0, в который укладываются все имевшие место значения цен на эту валютную пару. Совершенно другой масштаб у валютной пары *usdjpy* – величины цен на эту валютную пару почти на два порядка больше, чем на *eurusd.*

Категориальные (*categorical*) данные, известные также как номинальные атрибуты, качественные данные, факторы имеют значения, которые не имеют масштаба. «Лонг/шорт», день недели являются примерами таких данных. «Лонг» не больше и не меньше «шорта». Категориальная переменная, которая имеет два значения, как у нас – (лонг, шорт) называют *бинарной* (двоичной) переменной. Категориальная переменная «день недели» имеет семь значений.

Категориальные переменные могут быть упорядочены, как в нашем примере *Weekdays* (дни недели). Понедельник не больше и не меньше вторника, но может быть важным для модели, чтобы ей было известно, что вторник всегда следует после понедельника.

Построение модели, обучение модели, тренировка модели или оценка параметров – все это относится к процессу определению параметров в уравнении модели.

1.4. Используемые наборы данных

Далее по тексту будут использоваться следующие наборы данных:

audit набор данных, поставляемый в составе дистрибутива ***Rattle***.

weather набор данных, поставляемый в составе дистрибутива ***Rattle***.

kot60_110101_131231_UE. txt

На основе регрессионной модели попытаемся сделать «типичный мультивалютник»:

– целевая переменная – *EURUSD;*

– предикторы – *GBPUSD, USDCHF, USDJPY, EURGBP, USDCAD.*

zz_1_5.RData

Для классификационной модели создан искусственный разнообразный набор предикторов, которые должны продемонстрировать возможности моделей по предсказанию трендов:

– целевая переменная (три варианта) – тренд;

– предикторы – день недели, час дня, *EURUSD*, приращение *EURUSD*, *macd, macd (13), macd (26), macd (39*), приращение *macd (13), macd (26), macd (39)), RSI (14), RSI (21), RSI (28),* стеллажирование на 8 уровней *(RSI (14), RSI (21), RSI (28)), MA (13), MA (26), MA (52)*, приращение (*MA (13), MA (26), MA (52)).*

Описание каждого набора данных приведено в Приложении В. Для *zz_1_5.RData* приведен скрипт на ***R,*** который формирует этот набор из набора *kot60_110101_131231_UE. txt*

2. Предварительная обработка данных

Методы предварительной обработки данных обычно состоят в добавлении, удалении или преобразовании данных обучения. Хотя мы интересуемся методами моделирования, подготовка данных может оказаться решающей для предсказательной возможности модели. У различных моделей есть разная чувствительность к типу предикторов в модели; *как* предикторы входят в модель также важно. Преобразования данных для уменьшения воздействия асимметрии данных или выбросов, могут привести к зн**а**чимым улучшениям результативности. Выделение предикторов является одним из эмпирических методов для создания фиктивных переменных, которые являются комбинациями многих предикторов. Также могут быть эффективными дополнительные, более простые стратегии, такие как удаление предикторов.

Потребность в предварительной обработке данных определяется типом используемой модели. Некоторые алгоритмы, такие как основанные на моделях деревьев, нечувствительны к характеристикам данных предиктора. Другие, как линейная регрессия, не являются таковыми. В этой главе обсужден целый ряд *возможных* методологий.

Эта глава обрисовывает в общих чертах подходы к обработке данных *без учителя*: целевая переменная не рассматривается методами предварительной обработки. В других главах обсуждаются методы *с учителем*, в которых используется целевая переменная для предварительной обработки данных. Например, модели частных наименьших квадратов (PLS) – по существу является версией с учителем анализа главных компонентов (PCA). Мы также описываем стратегии удаления предикторов, не рассматривая, как переменные могли бы быть связаны с целевой переменной.

То, как предикторы закодированы, может оказать значительное влияние на результативность модели. Например, использование комбинаций предикторов может иногда быть более эффективным, чем использование отдельного значения. Отношение двух предикторов может быть более эффективным, чем использование двух независимых предикторов. Часто больше всего эффективное кодирование данных возникает из понимания разработчиком моделируемой проблемы, и таким образом, не получено из какого-либо математического метода.

Обычно есть несколько различных методов для кодирования конкретного предиктора. В качестве примера приведем представление даты, которая может быть представлена многими способами:

– число дней, начиная со ссылочной даты;

– отдельно месяц, год и день недели как отдельные предикторы;

– номер дня в году;

– была ли дата в пределах торговой сессии (в противоположность праздничным дням или новогодним каникулам).

В нашем примере принято следующее решение по дате:

– день недели взят вместо календарной даты, так как интенсивность торгов разная в разные дни недели;

– номер часа взят вместо часа (совпадает со временем), так как интенсивность торгов разная в разное время суток.

«Корректная» разработка предиктора зависит от нескольких факторов. Во-первых, некоторые кодировки могут быть оптимальными для некоторых моделей и плохими для других. Например, основанные на дереве модели разделят данные на два или больше стеллажей. Как будет показано позднее, некоторые модели содержат встроенный *выбор предикторов*, означающий, что модель будет включать только предикторы, которые помогут максимизировать точность. В этих случаях может привередничать модель, какое представление данных является лучшим.

Отношение между предиктором и целевой переменной – следующий фактор. Существует, к примеру, логистическая модель, которая дает оценку вклада каждого предиктора в вычисление класса (модели классификации). Тем не менее, остаются крайне важным содержательное понимание связи между предикторами и целевой переменной.

Как со многими вопросами статистики, ответ на вопрос «какие методы разработки предикторов являются лучшими?» выглядит как: *это зависит*. Определенно, это зависит от используемой модели и истинного отношения с целевой переменно.

2.1. Преобразование отдельных предикторов

Преобразования предикторов могут быть необходимы по нескольким причинам. У некоторых методов моделирования могут быть строгие требования, такие как необходимость общего масштаба предикторов. В других случаях создание хорошей модели может быть затруднено определенными характеристиками данных, например, выбросами. В книге обсуждается центрирование, масштабирование и преобразования асимметрии.

2.1.1. Центрирование и масштабирование

Центрирование и масштабирование предикторов является наиболее понятным преобразованием данных. Для центрирования предиктора среднее значение предиктора вычитается из всех значений. В результате центрирования у предиктора средняя равна нулю. Точно так же, для совместимости масштабов данных, каждое значение предиктора делится на его стандартное отклонение. Масштабирование данных приводит к значениям с отклонениями в размере одного стандартного отклонения. Эти манипуляции обычно используются для улучшения числовой устойчивости некоторых вычислений. Некоторые модели, к примеру PLS, извлекают выгоду из предикторов, имеющих общий масштаб. Единственным минусом этих преобразований является потеря интерпретируемости отдельного значения, так как данные больше не находятся в исходных масштабах.

2.1.2. Преобразования для исключения асимметрии

Другая общая причина преобразований состоит в удалении исходной асимметрии – скоса. Распределение без скоса – это то, что примерно симметрично. Это означает, что уменьшение вероятности по обе стороны от среднего распределения примерно равно. У распределений с правым скосом есть большое количество точек на левой стороне распределения (меньшее значение), чем на правой стороне (большее значение).

Общее правило большого пальца в рассмотрении скошенных данных состоит в том, что если максимальное значение превосходит минимальное значение более 20 раз, то имеется значимая асимметрия. Кроме того, статистика асимметрии может использоваться в качестве диагностики. Если распределение предиктора будет примерно симметрично, то значение асимметрии будет близко к нулю. Поскольку распределение становится более отклоненным справа, то статистика асимметрии становится больше. Точно так же, поскольку распределение становится более отклоненным влево, то значение становится отрицательным.

Логарифмирование может помочь удалить скос.

Вне рамок ***Rattle***, но из инструментов ***R,*** имеется преобразование Box-Cox (1964), которые предлагают *семейство* адаптивных преобразований. Эту процедуру можно применить вне ***Rattle*** к каждому предиктору, имеющими значения, больше нуля.

2.2. Преобразование групп предикторов

Эти преобразования действуют на группы предикторов, обычно все рассматриваемого множества. Наиболее значимые методы направлены на решение проблем выбросов и уменьшения размерность данных.

2.2.1. Преобразования, решающие проблему выбросов

Мы обычно определим выбросы как наблюдения, которые исключительно далеки от основных данных. При определенных предположениях есть формальные статистические определения выброса. Даже с полным пониманием данных бывает сложно определить выбросы. Однако можно выявить необычное значение, глядя на рисунок. Если одно или более значений предиктора попадает под подозрение, сначала нужно подумать о допустимости этих значений. Необходимо соблюдать особую осторожность и не торопиться удалять или изменять значение, особенно при небольшом объеме выборки.

Есть несколько предсказательных моделей, которые являются устойчивыми к выбросам. Модели классификации на основе дерева создают разделения учебных данных, и уравнение предсказания – ряд логических операторов таких как, «если предиктор *A* больше чем *X*, то предсказываем класс *Y*», таким образом, выброс обычно не имеет исключительного влияния на модель. Машины опорных векторов для классификации обычно игнорируют часть наблюдений набора данных обучения, создавая уравнение предсказания. Исключенные наблюдения могут быть далеко от границы решения и за пределами основных данных.

Если используемая модель чувствительна к выбросам, то существует преобразование данных, которое может минимизировать задачу – это *пространственный знак*.

2.2.2. Снижение объема данных и выделение предикторов (PCA)

Методы снижения объема данных – другой класс преобразований предикторов. Эти методы сокращают данные, генерируя меньшее множество предикторов, которые стремятся получить большую часть информации из исходных переменных. Таким образом, можно использовать меньше переменных, которые обеспечивают разумную точность для исходных данных. Для большинства методов снижения объема данных новые предикторы – функции исходных предикторов; поэтому, все исходные предикторы все еще необходимы, чтобы создать суррогатные переменные. Этот класс методов часто вызывают *экстракцией сигнала* или методами *выделения предикторов*.

Алгоритм PCA – обычно используемый метод снижения объема данных. Этот метод стремится найти линейные комбинации предикторов, называемых *главными компонентами* (PC), которые содержат наибольшую возможную дисперсию. Первая PC определена как линейная комбинация предикторов, которая получает большую часть изменчивости всех возможных линейных комбинаций. Затем, последующие PC получены так, что эти линейные комбинации получают остающуюся изменчивость, также будучи некоррелированным со всеми предыдущими PC.

Основное преимущество PCA и причина, что он сохранило свою популярность как метод снижения объема данных, состоит в том, что он создает компоненты, которые не коррелированы. Как отмечалось ранее, некоторые предсказывающие модели предпочитают, чтобы предикторы были не коррелированы (или, по крайней мере, с низкой корреляцией) для улучшения устойчивости модели. Используя PCA предварительная обработка создает новые предикторы с требуемыми характеристическими.

Хотя PCA поставляет новые предикторы с требуемыми характеристиками, он должен использоваться с пониманием и вниманием. Особенно практики должны понять, что PCA ищет установленное в предиктор изменение без отношения к дальнейшему пониманию предикторов (то есть, измерительные весы или распределения) или к знанию целей моделирования (то есть, целевой переменной). Следовательно, без надлежащего руководства, PCA может генерировать компоненты, которые суммируют характеристики данных, которые не важны глубинной структуре данных и также к окончательной цели моделирования.

Поскольку PCA ищет линейные комбинации предикторов, которые максимизируют изменчивость, он будет естественно сначала брать предикторы, у которых есть больше изменения. Если исходные предикторы находятся в исходных масштабах, которые отличаются

по порядкам величины (например, котировки EURUSD и USDJPY), то японская йена будет довлеть над парой EURUSD. Это означает, что веса PC будут больше для йены. Кроме того, это означает, что PCA будет фокусировать свои усилия на идентификации структуры данных, основанной на исходных масштабах, а не основанной на важных отношениях среди данных для решаемой задачи.

Для большинства наборов данных предикторы имеют разные масштабы. Кроме того, предикторы, возможно, имеют скошенные распределения. Следовательно, для исключения в PCA избегать суммирования исходных различий и информации о масштабе предикторов лучше сначала преобразовывать предикторы, центрировать и масштабировать предикторы до выполнения PCA. Центрирование и масштабирование позволяют PCA найти базовые отношения в данных, игнорируя влияние исходных измеренных величин.

Вторая отрицательная черта PCA состоит в том, что он не рассматривает цель моделирования или переменную отклика при суммировании изменчивости. Поскольку PCA слепой к отклику, это – *неконтролируемый метод* . Если предсказательное отношение между предикторами и откликом не будет соединено с изменчивостью предикторов, то полученные PC не будут предоставлять подходящему отношению отклик. В этом случае, *контролируемый метод* такой, как PLS, создаст компоненты, одновременно учитывая соответствующий отклик.

Аналогично PCA, PLS находит линейные комбинации предикторов. Эти линейные комбинации обычно называют *компонентами* или скрытыми переменными. В то время как линейные комбинации PCA выбираются с целью максимально суммировать изменчивость пространства предикторов, линейные комбинации предикторов в PLS выбираются с целью, чтобы максимально суммировать ковариацию с откликом (целевой переменной). Это означает, что PLS находит компоненты, которые максимально суммируют изменение предикторов, одновременно требуя, чтобы эти компоненты имели максимальную корреляцию с целевой переменной. Поэтому PLS получает компромисс между целью уменьшения размерности пространства предикторов и предсказательного отношения с целевой переменной. Другими словами PLS относится к *контролируемой* процедуре уменьшения размерности.

Как только выбрано соответствующие преобразования предикторов, то можно применить PCA. Для моделей со многими предикторами следует принять решение о количестве главных компонент, подлежащих использованию. Этот вопрос решается просто при использовании средств ***R***: результат вычислений сопровождается вспомогательной информацией в виде накопленной изменчивости. Обычно берется величина 95% и выбирается такое количество главных компонент, которые совместно накопили такую изменчивость исходных данных.

При разложении исходного набора предикторов на главные компоненты указывается вес каждого предиктора в конкретной главной компоненте. Этот вес называется *нагрузкой*. Нагрузка близкая к нулю указывает, что этот конкретный предиктор не очень-то важен этому компоненту. Если среди всех отобранных главных компонент окажется предиктор с небольшой нагрузкой, то этот предиктор является кандидатом на его исключение из модели.

2.3. Обработка пропущенных значений

При включении в мультивалютные модели валютных пар с разной ликвидностью, особенно на младших тайм фреймах, может возникнуть ситуация отсутствия значений одной из валютных пар при наличии значений в других валютных парах.

Могут быть и другие причины. Например, ведение торгов в разное время по разным валютным парам. И это не единственные причины возникновения пропущенных значений на финансовых рынках.

Важно понять, *причину* пропуска значения. Прежде всего, важно знать, как связано пропущенное значение с целевой переменной. В нашем примере трендовой торговой системы можно рассмотреть две ситуации:

– отсутствуют котировки внутри торговой сессии. Можно предположить, что отсутствие значений не влияет на тренды, имеющиеся в данный момент на рынке.

– отсутствуют значения вне торговой сессии, например в выходные дни. Известно, что в выходные дни могут происходить события, которые в случае торгов повлияли бы на котировки.

Заполнение пропущенных значений было интенсивно изучено в статистической литературе, но в контексте проверки гипотез процедурами тестирования при наличии пропущенных данных. Это – отдельная проблема. Для предсказательных моделей мы обеспокоены точностью предсказаний вместо того, чтобы делать допустимые выводы.

Заполнение пропущенных значений – это только другой уровень моделирования, где мы пытаемся оценить значение предикторов, основанных на других значениях предиктора. Соответствующая схема заполнения состоит в использовании набора данных обучения для создания модели заполнения для каждого предиктора в наборе данных. До обучения самой предсказательной модели или предсказания целевой переменной заполняются отсутствующие значения предикторов. Заметим, что этот дополнительный уровень моделей увеличивает неопределенность.

Если число предикторов, на которые влияют отсутствующие значения, небольшое, анализ отношений между предикторами – хорошая идея. Например, могут использоваться такие методы как визуализация или PCA, чтобы определить, есть ли прочные отношения между предикторами. Если переменная с отсутствующими значениями чрезвычайно коррелирована с другим предиктором, у которого есть немного отсутствующих значений, используемая модель может часто быть эффективной для заполнения.

Одним из популярных методов заполнения является модель *K-ближайших* соседей. Эта модель по значения ближайших соседей может оценить значение отсутствующих значений предиктора.

2.4. Удаление предикторов

Есть потенциальные преимущества для удаления предикторов до моделирования. Во-первых, меньшее количество предикторов означает уменьшение вычислительной сложности и времени вычислений. Во-вторых, если два предиктора чрезвычайно коррелированы, это подразумевает, что они измеряют ту же самую базовую информацию. Удаление одного из них не должно ставить под угрозу результативность модели и могло бы привести к более экономной и поддающейся толкованию модели. В-третьих, некоторым моделям могут нанести вред предикторы с вырожденными распределениями. В этих случаях может быть значимое уточнение в результативности модели и/или устойчивости без проблематичных переменных.

2.4.1. Корреляции между предикторами

Коллинеарность – технический термин для ситуации, где у пары предикторов есть существенная корреляция друг с другом. Также возможно одновременно иметь отношения между многими предикторами (называется *мультиколлинеарность*).

Если набор данных состоит из слишком большого числа предикторов для визуального исследования, то можно использовать такие методы как PCA для установления характеристик проблемы. Например, если первый основной компонент учитывает большой процент дисперсии, то возникают подозрения в существовании единственной переменной для модели.

Вообще, есть серьезные основания исключить чрезвычайно коррелированные предикторы. Во-первых, избыточные предикторы часто более усложняют модели, чем добавляют

информации к ней. Использование чрезвычайно коррелированных предикторов в таких моделях, как линейная регрессия, может привести к очень нестабильным моделям, числовым ошибкам, и ухудшить предсказательную результативность.

У классического регрессионного анализа есть несколько инструментов для диагностики мульти коллинеарности для линейной регрессии. Так как коллинеарные предикторы могут воздействовать на оценку дисперсии параметра в этой модели, то может использоваться статистика, называемая фактором инфляции дисперсии (VIF), для выявления предикторов с коллинеарностью. Вне линейной регрессии этот метод может оказаться не применимым по нескольким причинам: он разрабатывался для линейных моделей и, в то время как он действительно идентифицирует коллинеарные предикторы, он не определяет предиктор, подлежащий удалению для решения проблемы.

Далее будет более подробно рассмотрена значимость предикторов и их выбор.

2.5. Добавление предикторов

Если предиктор категориальный, такой как день недели или время суток, то обычно разделяют предиктор в ряд более определенных переменных. Например, день недели имеет 7 категорий (или 5 категорий, соответствующих рабочим дням).

Обычно вместо одного предиктора вводят 7 «фиктивных» предикторов, каждый из которых соответствует одному дню недели. Обычно этот подход улучшает интерпретируемость модели. Кроме этого некоторые модели лучше работают с бинарными предикторами.

2.6. Группировка предикторов

Будем различать два варианта понятия «группировки предикторов»:

– группировка значений отдельного предиктора;

– группировка нескольких предикторов в один.

В первом случае любой числовой предиктор можно упростить путем разбивки его на несколько категорий или стеллажей. Например, возьмем индикатор RSI, который обычно используется для идентификации разворотов трендов. Разделим значения этого индикатора на 4 части, и вместо числовых значений индикатора будем использовать числа 1,2,3 и 4, где числа 1 и 4 будут соответствовать разворотам тренда. Такой вид укладывания в стеллаж соответствует основной идеи нашей торговой системы – трендовой торговли.

Во втором случае все множество предикторов, которое используется в модели скомпонуем в меньшее число предикторов так, чтобы это меньшее число объясняло большую часть изменчивости всех предикторов. Данный подход известен как «анализ главных компонент» и был рассмотрен выше.

Компоненты, получаемые по алгоритмам PCA (PLS) позволяет использовать существенно меньшее количество новых предикторов. Каждая дополнительная главная компонента объясняет все меньшее количество изменчивости. Если просуммировать изменчивость всех новых предикторов, то сумма будет равна единице, а где-то в середине будет некоторое количество предикторов, которое будет объяснять, например, 95% изменчивости. Обычно для рынка Форекс можно уменьшить количество предикторов примерно в три раза.

2.6. Функции R

Приведем некоторые функции, которые могут быть использованы при работе над данным разделом.

Приведено название функции, а в скобках название пакета, в котором функция расположена. Для использования функция необходима загрузка пакета, а если его еще нет, то и установка.

Если названия пакета не приведено – это означает, что функция имеется в базовом пакете, и не требуется предварительная загрузка пакета.

skewness (e1071)

асимметрия (скос)

boxcox (MASS)

преобразование Box-Cox. Оценивает λ, но преобразование не выполняет.

BoxCoxTrans (caret)

преобразование Box-Cox с преобразованием данных

prcomp

вычисляет главные компоненты (PCA)

preProcess (caret)

предварительная обработка

cor

корреляция

findCorrelation

возвращает список переменных, рекомендованных для удаления из-за сильной корреляции

dummyVars (caret)

создает фиктивные переменные

3. Переобучение и настройка модели

Многие современные классификационные и регрессионные модели высоко адаптируемы; они способны к моделированию комплексных отношений. Однако они могут очень легко отобразить некие случайности в экономическом процессе. Как говорят – отобразить шум. Без методологического подхода к оценке моделей разработчик модели может узнать о проблеме слишком поздно.

Переобучение (сверх подгонка) – широко известная проблема предсказательных моделей вообще и в области финансов в частности. Фактически переобучение отображает базовую проблему моделирования: модель должна отображать некие основные моменты моделируемого процесса, модель должна быть не слишком груба, но и не слишком точна, чтобы она могла находить основные моменты на новых данных, а не давать ложные сигналы, принимая шум за образцы данных.

К сожалению, отсутствуют формальные критерии переобучения. Поэтому приходится руководствоваться некими эмпирическими критериями, которые дадут практическую ценность модели. Эти эмпирические критерии состоят в том, чтобы дать разработчику предсказательной модели уверенность, что поведение модели на обучающем наборе данных и на данных вне этого обучающего набора, будет примерно одинаковым.

Без этого доверия предсказания модели бесполезны.

3.1. Проблема переобучения

Существует много методов, которые могут изучить структуру ряда данных так хорошо, что при применении модели к данным, на которых была создана модель, она правильно предсказывает каждое значение. В дополнение к изучению общих образцов в данных модель также изучила характеристики отдельного шума каждой выборки. Эта модель, как говорят, переобучена, и с плохой точностью предскажет целевую переменную на новой выборке.

Изначально, мы учим модель на наборе данных обучения и по результатам обучения получаем некую величину ошибки для регрессионных моделей, или рассогласование для классификационных моделей.

Уже на этом этапе возможно переобучение модели: оценка слишком оптимистична, например, ошибка подгонки менее 5%. Да и ошибка подгонки в 10% должна насторожить!

В этих ситуациях очень важно иметь инструмент для определения переобученности модели на учебных данных.

3.2. Настройка модели

У многих моделей есть важные параметры, которые не могут быть непосредственно оценены на данных. Например, в модели классификации *K-ближайшие соседи* предсказание основано на *K* самых близких точек данных в наборе данных обучения.

Очевиден вопрос: сколько соседей должно использоваться. Выбор слишком большого числа соседей может переобучить модель к отдельным точкам набора данных обучения, в то время как слишком малое число соседей может быть не достаточно чувствительными для получения разумной результативности. Этот тип параметра модели называется *настраиваемым параметром*, так как отсутствует аналитическая формула, доступная для вычисления соответствующего значения.

Практически у всех предсказательных моделей есть, по крайней мере, один настраиваемый параметр. Так как многие из этих параметров управляют сложностью модели, плохие варианты для значения могут привести к переобучению.

Есть разные подходы к поиску лучших параметров. Общий подход, который можно применить к почти любой модель, должен определить ряд значений кандидата, генерировать надежные оценки модели через значение кандидатов, а затем выбрать оптимальную модель.

Как только множество кандидатов значений параметра было выбрано, то следует получить правдоподобные оценки результативности модели. Результативность *вне-выборки* суммируется в профиль результативности, который затем используется для определения заключительных настраиваемых параметров. Затем создаем заключительную модель со всеми учебными данными, используя выбранные настраивающие параметры.

При построении моделей доступны подходы, такие как генетические алгоритмы или симплексные методы поиска, которые могут найти оптимальные настраиваемые параметры. Эти процедуры алгоритмически определяют соответствующее значение для настройки параметров и выполняют итерации, пока они не достигают установок параметров с оптимальной результативностью. Эти методы имеют тенденцию оценивать большое количество моделей кандидата и могут превосходить определенное множество настраиваемых параметров, если результативность модели может быть эффективно вычислена.

Как ранее обсуждалось, очевидный коэффициент ошибок может произвести чрезвычайно оптимистические оценки результативности. Лучшим является подход, который проверяет модель на выборках, не использованных для обучения.

Оценивая модель на тестовом наборе, размер набора тестов, возможно, должен быть большим.

Альтернативный подход к оценке модели на единственном тестовом наборе состоит в *ресемплирования* набора данных обучения. Этот процесс использует несколько измененных версий набора данных обучения, чтобы создать многоуровневые модели и затем использует статистические методы, чтобы обеспечить честные оценки результативности модели (то есть, не чрезмерно оптимистичные).

3.3. Разделение данных

Теперь, когда мы обрисовали в общих чертах процедуру для поиска оптимальных настраиваемых параметров, вернемся к обсуждению основы процесса: разделение данных.

Несколько общих шагов в создании модели:

– предварительная обработка данных предиктора;

– оценка параметров модели;

– выбор предикторов для модели;

– оценка результативности модели;

– правила предсказания класса точной настройки (через кривые ROC, и т.д.).

Одно из первых решений при моделировании, которое следует принять, какие наборы данных или их части будут использоваться для оценки результативности. Идеально, модель должна быть оценена на выборках, которые не использовались при создании или построении модели, так, чтобы они обеспечили несмещенную оценку эффективности модели. «*Учебный*» набор данных – общий термин для наблюдений, используемых для создания модели, в то время как набор данных «*теста*» или «*проверки*» используется для определения результативности.

Из литературы известно, что проверка, использующая единственный набор, может дать плохое решение. Могут использоваться методы ресемплирования, такие как кросс-проверка, для соответствующей оценки результативности модели, используя набор данных обучения. Хотя методы ресемплирования могут быть неправильно употреблены, они часто оценивают

результативность точнее единственного набора, потому что они оценивают много вариантов данных. Если тестовый набор считается необходимым, то есть несколько методов для разделения выборки.

В большинстве случаев желательно сделать наборы данных обучения и набор данных тестирования настолько гомогенными насколько возможно. Можно использовать методы случайных выборок для создания подобных наборов данных.

Самый простой способ разделить данные на набор данных обучения и тестовый набор состоит в том, чтобы взять простую случайную выборку. Это можно делать, если известно, что отношения классов примерно равны в обучающей и тестовой выборке. Когда у одного класса есть непропорционально большая частота по сравнению с другим, есть шанс, что распределение результатов может существенно отличаться между наборами данных обучения и тестовым набором.

Чтобы учесть результат при разделении данных, стратифицированная случайная выборка применяет случайную выборку в пределах подгрупп (таких как классы). Таким образом, получим более правдоподобную выборку, которая будет соответствовать распределению результата. Когда результат – число, подобная стратегия может использоваться; числовые значения разделены в подобные группы (например, минимум, среднее и максимум), и рандомизация выполняется в пределах этих групп.

Альтернативно, данные могут быть разделены на основе значения предиктора. Например, на *максимальной выборке несходства*. Несходство между двумя выборками может быть измерено многими способами. Самый простой метод использует расстояние между значением предиктора для двух наблюдений. Если расстояние небольшое, точки находятся в непосредственной близости. Большие расстояния между точками указывают на несходство. Чтобы использовать несходство в качестве инструмента для разделения данных, предположим, что тестовый набор создан из единственной выборки. Можно вычислить несходство между этой начальной выборкой и освобожденными выборками. Освобожденная выборка, которая является самой несходной, затем была бы добавлена к тестовому набору. Чтобы создать больше наборов тестовых наблюдений, необходим метод для определения несходства между *группами* точек (то есть, два в наборе тестов и освобожденных точках).

3.4. Методы ресемплирования

Методы ресемплирования для оценки результативности модели работают так: подмножество наблюдений используется для подгонки модели, и остающиеся выборки используются, чтобы оценить эффективность модели. Этот процесс повторен многократно, и результаты суммируются и выводятся итогом. Разности в методах обычно центрируются вокруг метода, по которому сделаны выборки из набора данных. Рассмотрим главные виды ресемплирования в следующих немногих подразделах.

3.4.1. k-кратная кросс-проверка

Выборки в произвольном порядке разделены в k множеств примерно равного размера. Производится подгонка модели на всех выборках кроме первого подмножества (названного первой *сверткой*). Вне-выборки выполняются предсказания этой моделью и используются для оценки критерии качества результата. Первое подмножество возвращается в набор данных обучения, и процедура повторяется со вторым подмножеством вне-выборки и так далее. В итоге оценивается K-передискретизованная результативность (обычно со средней и стандартной ошибкой), а используются выяснения отношений между настраиваемыми параметрами и формулой модели.

Небольшая разновидность этого метода выбирает *k*-разделов способом, который делает свертки сбалансированными относительно результата. Стратифицированная случайная выборка, обсужденная ранее, создает баланс относительно результата.

Другая версия, перекрестная проверка «пропуск одного» (LOOCV), является частным случаем, где *k* является числом наблюдений. В этом случае, так как только одна вне-выборка берется за один раз, заключительная результативность вычислена от *k* предсказаний от вне-выборок. Дополнительно, повторная *k*-кратная перекрестная проверки тиражирует процедуру многократно. Например, если бы 10-кратная перекрестная проверка была повторена пять раз, 50 различных вне-выборок использовались бы для оценки эффективности модели.

Выбор *k* обычно равняется 5 или 10, но нет никакого формального правила. Поскольку *k* становится больше, разница в размерах между набором данных обучения и подмножествами ресемплирования становится меньшей. Когда эта разность уменьшается, *смещение* метода становится меньшим (то есть, смещение меньше для $k = 10$, чем для $k = 5$). В этом контексте смещение – разность между оцененными и истинными значениями результативности.

Другой важный аспект метода ресемплирования – это неопределенность (то есть, дисперсия или шум). Несмещенный метод может оценивать корректное значение (например, истинная теоретическая результативность), но может привести к высокой неопределенности. Это означает, что повторение процедуры ресемплирования может произвести совсем другое значение (но сделанная достаточно много раз, она оценит истинное значение). *k*-кратная перекрестная проверка обычно имеет высокую дисперсию по сравнению с другими методами и, по этой причине, не может быть привлекательной. Нужно сказать, что для больших наборов данных обучения, потенциальные проблемы с дисперсией и смещением становятся незначительными.

С практической точки зрения большее значение *k* в вычислительном отношении обременительно. В экстремуме LOOCV больше всего в вычислительном отношении накладно, потому что требуется много подгонок модели как точки данных, и каждая подгонка модели использует подмножество, которое почти равно размеру набора данных обучения.

3.4.2. Повторные разделения для обучения/тестирования

Повторные разделения набора для обучения/тестирования также известны как «перекрестная проверка, «пропускают группу» или «перекрестная проверка Монте-Карло». Этот метод просто создает много разделений данных в моделировании и много предсказаний. Соотношением данных, входящих в каждое подмножество, управляют числом повторений.

Число повторений важно. Увеличение числа подмножеств имеет эффект уменьшения неопределенности в оценках результативности. Например, для получения грубой оценки результативности модели будет достаточно 25 повторений, если пользователь будет готов принять некоторую нестабильность в получающемся значении. Однако чтобы получить устойчивые оценки результативности необходимо выбрать большее число повторений (скажем 50—200). Это – также функция соотношения наблюдений, в произвольном порядке выделяемых множеству предсказаний; чем больше процент, тем больше повторений необходимо для уменьшения неопределенности в оценках результативности.

3.4.3. Бутстрэпинг

Выборка по бутстрэпингу – случайная выборка данных, взятых *с заменой.* Это означает, что, после того, как элемент данных выбран для подмножества, он все еще доступен для дальнейшего выбора. Выборка по бутстрэпингу равна исходному набору данных. В результате некоторые элементы будут представлены многократно в выборке бутстрэпинга, в то время как другие не будут выбраны вообще. Не выбранные элементы формируют выборку под названием «вне стеллажа». Для данной итерации ресемплирования в виде бутстрэпинга модель основана на сформированных выборках и используется для предсказания выборки вне стеллажа.

3.5. Функции R

Приведем некоторые функции, которые могут быть использованы при работе над данным разделом.

Приведено название функции, а в скобках название пакета, в котором функция расположена. Для использования функция необходима загрузка пакета, а если его еще нет, то и установка.

Если названия пакета не приведено – это означает, что функция имеется в базовом пакете и не требуется предварительная загрузка пакета.

Разделение

sample

создает простое случайное разделение

createDataPartition (caret)

создает случайную выборку с разделением на классы

maxdissim (caret)

генерирует набор для тестирования, используя максимальную выборку несходства.

createDataPartition (caret)

создает случайную выборку с разделением на классы

Ресемплирование

createDataPartition (caret)

создает случайную выборку с разделением на классы с дополнительным параметром times

createResamples (caret)

для бутстрэпинга

createFolds (caret)

для k-свертки перекрестной проверки

createMultiFolds (caret)

для многократной перекрестной проверки

4. Регрессионные модели

4.1. Результативность регрессионных моделей

Для моделей, предсказывающих числовой результат, используется некоторая мера точности для оценки эффективности модели. Однако есть различные способы измерить точность, каждый с его собственным нюансом. Понять силу и слабость определенной модели, полагаясь исключительно на единственную метрику проблематично. Визуализация подгонки модели, особенно графики остатков, является чрезвычайно важным по отношению к пониманию пригодности модели к цели.

Когда результат – число, наиболее распространенный метод для оценки предсказательных возможностей модели – это среднеквадратичная ошибка (MSE). Эта метрика – функция остатков модели, которые являются наблюдаемыми величинами минус предсказания модели. Среднеквадратичная ошибка (MSE) вычисляется путем возведения остатков в квадрат и их суммирования. RMSE – это квадратный корень из MSE. Значение обычно интерпретируется или как далеко (в среднем) остатки от нуля, или как среднее расстояние между наблюдаемыми величинами и предсказаниями модели.

Другая общая метрика – коэффициент детерминации, обычно обозначаемый как R^2. Это значение может быть интерпретировано как величина объясненной моделью информации в данных. Таким образом, значение R^2, равное 0.75, подразумевает, что модель может объяснить три четверти изменения в результате. Есть много формул для вычисления этого показателя, хотя самая простая версия считает коэффициент корреляции между наблюдаемыми и ожидаемыми значениями с возведением его в квадрат.

Также важно понять, что R^2 зависит от изменения в результате. Используя интерпретацию, что эта статистика измеряет соотношение дисперсии, объясненной моделью, нужно помнить, что знаменатель этого отношения вычисляется с использованием дисперсии выборки результата. Например, предположим, что у результата набора тестов есть дисперсия 4.2. Если бы RMSE предсказательной модели равнялись 1, то R^2 составил бы примерно 76%. Если бы у нас был другой набор тестов с точно тем же самым RMSE, но результатами теста было меньше переменной, то результаты выглядели бы хуже. Например, если бы дисперсия набора тестов равнялась 3, то R^2 составил бы 67%.

В некоторых случаях цель модели просто состоит в упорядочении новых наблюдений. В этом случае определятся возможность модели, а не ее предсказательная точность. Для этого определяется *порядковая корреляция* между наблюдаемыми и ожидаемыми значениями, и оценка производится с помощью более соответствующей метрики. Порядковая корреляция берет ранги наблюдаемого значения результата (в противоположность их фактическим значениям) и оценивает, как близко это к рангам предсказаний модели. Для вычисления этого значения получают ранги наблюдаемых и предсказанных результатов, и вычисляют коэффициент корреляции между этими рангами. Эта метрика обычно известна как порядковая корреляция Спирмена.

4.2. Линейные регрессионные модели

Когда мы говорим о линейных моделях, то имеется в виду, что модели являются *линейными в параметрах*.

При оценке моделей оцениваются их параметры так, чтобы сумма квадратов ошибок или функция суммы квадратов ошибок были минимизированы. Среднеквадратичная ошибка (MSE) может быть разделена на компоненты не уменьшаемого изменения, смещения модели и дисперсии модели.

Явное преимущество линейных моделей состоит в легкости их толкования.

Другое преимущество этих видов моделей состоит в том, что их математический характер позволяет вычислить стандартные ошибки коэффициентов при условии, что делаются определенные предположения о распределениях остатков модели. Затем эти стандартные ошибки могут использоваться для оценки статистической зн**а**чимости каждого предиктора в модели.

В то время как линейные модели типа регрессии легко поддаются толкованию, их использование может быть ограничено. Во-первых, эти модели состоятельны, если отношение между предикторами и откликом движется вдоль гиперплоскости. Например, при одном предикторе модель будет состоятельной, если отношение между предиктором и откликом двигалось вдоль прямой линии. С большим количеством предикторов отношение должно двигаться близко к плоской гиперплоскости. Если есть криволинейное отношение между предикторами и откликом (например, такое как квадратное, кубическое взаимодействия среди предикторов), то линейные регрессионные модели могут быть расширены с дополнительными предикторами, которые являются функциями исходных предикторов в попытке получить эти отношения. Однако нелинейные отношения между предикторами и откликом не могут быть соответственно получены этими моделями.

4.3. Нелинейные регрессионные модели

Многие из линейных моделей могут быть адаптированы к нелинейным трендам в данных, вручную прибавляя параметры модели (например, квадраты параметров). Однако для этого необходимо знать специфический характер нелинейности в данных.

Есть многочисленные регрессионные модели, которые по своей сути не линейны. При использовании этих моделей точная форма нелинейности не должна быть известна явно или специфицироваться до обучения модели. Рассмотрим несколько таких моделей: нейронные сети, машины опорных векторов (SVM) и *K-ближайшие* соседи *(KNN)*. Основанные на дереве модели также не линейны. Из-за их популярности рассмотрим отдельно.

4.3.1. Нейронные сети

Нейронные сети – это мощные нелинейные методы регрессии, вдохновленные теориями о работе интеллекта. Как частные наименьшие квадраты (PLS), результат моделируется посредством многих не наблюдаемых переменных (названными *скрытыми переменными* или скрытыми модулями здесь). Эти скрытые модули – линейные комбинации исходных предикторов.

При обработке этой модели как нелинейной регрессионной модели обычно оптимизируются параметры для минимизации суммы квадратов остатков. Это может вызвать вычислительную проблему, связанную с оптимизацией (вспомним, что нет никаких ограничений на параметры этой комплексной нелинейной модели). Параметры обычно инициируются случайным значением, а затем используются специализированные алгоритмы для решения уравнения.

Кроме того, у нейронных сетей есть тенденция к переобучению отношений между предикторами и целевой переменной из-за большого количества коэффициентов регрессии. Для преодоления этой проблемы предлагается несколько разных подходов.

Один из подходов к решению проблемы переобучения состоит в использовании *сходимости* весов. В этом случае прибавляется штраф за большие коэффициенты регрессии так, чтобы

любое крупное значение имело значимое влияние на ошибки модели. Формально, произведенная оптимизация попыталась бы минимизировать альтернативную версию суммы квадратных ошибок.

Учитывая проблему оценки большого количества параметров, подогнанная модель находит оценки параметра, которые локально оптимальны; то есть, алгоритм сходится, но получающиеся оценки параметра вряд ли будут глобально оптимальными оценками. Очень часто различные локально оптимальные решения могут произвести модели, которые очень отличаются, но имеют почти эквивалентную результативность. Эта нестабильность модели может иногда ограничивать применение этой модели. Как альтернатива, создаются несколько моделей, используя различные начальные значения с последующим использованием средних результатов с целью получения более стабильного предсказания. Такая *усредненная модель*

Конец ознакомительного фрагмента.