Лекція 13. Прогнозування, помилки та перехресна перевірка

У цій лекції ви зосередитеся на дослідженні прогнозування, відносній важливості кроків прогнозування, помилках та перехресній перевірці.

Мета заняття:

- визначити прогнозування та його значення в процесах прийняття рішень;
- зрозуміти відносну важливість різних етапів процесу прогнозування;
- розрізняти помилки у вибірці та поза її межами та їх наслідки для прогнозного моделювання.

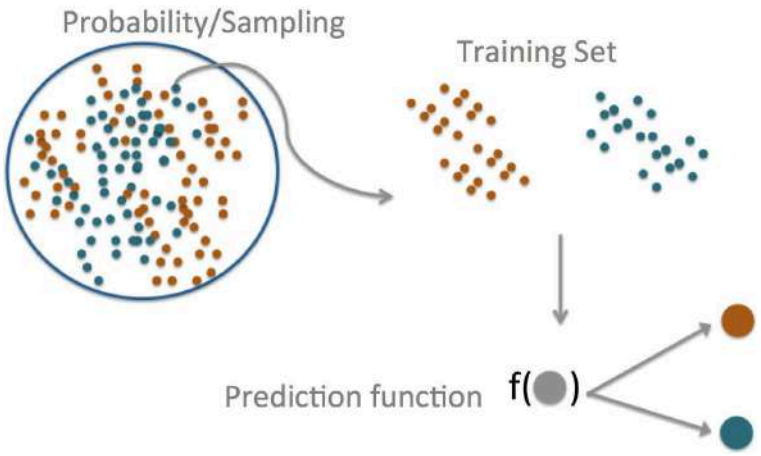
Зміст:

1. Вступ до Розділу 5

Одна з найпоширеніших задач, які виконують науковці та аналітики даних, – це прогнозування та машинне навчання. Цей розділ охоплюватиме основні компоненти побудови та застосування функцій прогнозування з акцентом на практичних застосуваннях. Розділ надасть базові знання щодо таких понять, як навчальні та тестові набори даних, перепідгонка та частота помилок. Розділ також познайомить із низкою моделей і алгоритмічних методів машинного навчання, включаючи регресію, класифікаційні дерева, наївний Байєсовий класифікатор і випадкові ліси. Розділ охоплює повний процес створення функцій прогнозування, включаючи збір даних, створення функцій, алгоритми та оцінку.

2. Що таке прогнозування?

Багато дій у машинному навчанні зосереджено на тому, які алгоритми є найкращими алгоритмами для вилучення інформації та її використання для прогнозування. Але важливо відійти назад і поглянути на всю проблему прогнозування.



Це маленька діаграма, яка була створена, щоб проілюструвати деякі з ключових питань у створенні предиктора. Припустимо, ви хочете прогнозувати для цих точок, червоні вони чи сині. Те, що ви можете зробити, – це мати велику групу точок, щодо яких ви хочете зробити прогнозування, а потім використовувати ймовірність і вибірку, щоб вибрати навчальний набір. Навчальний набір складатиметься з кількох червоних і синіх точок, і ви вимірюватимете цілий ряд характеристик цих точок. Потім ви будете використовувати ці характеристики, щоб побудувати те, що називається функцією прогнозування, і функція прогнозування візьме нову точку, колір якої ви не знаєте, але використовуючи ті характеристики, які ви виміряли, ви зможете прогнозувати, червона вона чи синя. Тоді ви можете піти і спробувати оцінити, чи ця функція прогнозування працює добре чи ні. Одне з дуже важливих і часто недооцінених аспектів побудови алгоритму машинного навчання – це розгляд кроку ймовірності та створення вибірки при створенні навчальних і тестових наборів даних. Це завжди обов'язковий компонент побудови кожного алгоритму машинного навчання – це вирішення, які вибірки ви збираєтеся використовувати для створення цього алгоритму. Але іноді це недооцінюється, тому що всі дії, про які ви чуєте щодо машинного навчання, відбуваються тут, коли ви створюєте саму функцію машинного навчання.

Одним із дуже гучних прикладів того, як це може спричинити проблеми, є обговорення Google Flu Trends. Google Flu Trends намагається використовувати терміни, які люди вводили в Google, такі як "у мене кашель", щоб прогнозувати, як часто люди хворітимуть на грип. Іншими словами, яким був рівень захворюваності на грип у певній частині Сполучених Штатів у певний час. Вони порівняли свій алгоритм із підходом, прийнятим урядом Сполучених Штатів, і фактично виміряли, скільки людей захворіло на грип у різних місцях США. У своїй оригінальній статті вони

виявили, що алгоритм Google Flu Trends зміг дуже точно представити кількість випадків грипу, які з'являться в різних місцях США в будь-який момент часу. Це було трохи швидше і трохи дешевше для вимірювання за допомогою пошукових термінів у Google. Проблема, яку вони тоді не усвідомлювали, полягала в тому, що пошукові терміни, які використовували люди, з часом змінювалися. Вони можуть використовувати різні терміни під час пошуку, і це вплине на продуктивність алгоритму. Крім того, спосіб, яким ці терміни фактично використовувалися в алгоритмі, був не дуже добре зрозумілий. Коли функція певного терміна пошуку змінюється в їхньому алгоритмі, це може спричинити проблеми. Це призводить до дуже неточних результатів алгоритму Google Flu Trends із часом, коли змінюється використання Інтернету людьми. Це дає вам уявлення про те, що вибір правильного набору даних і знання конкретного питання знову мають першочергове значення.

Ось компоненти предиктора. Вам потрібно починати, як завжди, будь-яку проблему з наукою про дані з дуже конкретного та чітко визначеного питання. Що ви намагаєтеся прогнозувати і за допомогою чого ви намагаєтеся це прогнозувати? Потім ви виходите і збираєте найкращі вхідні дані, які можете прогнозувати. І з цих даних ви можете або використовувати виміряні характеристики, які у вас є, або ви можете використовувати обчислення для створення властивості, які, на вашу думку, можуть бути корисними для прогнозування результату, який вас цікавить. На цьому етапі ви фактично можете почати використовувати алгоритми машинного навчання, про які ви, можливо, читали, наприклад, випадковий ліс або дерева рішень. Потім ви можете оцінити параметри цих алгоритмів і використати ці параметри, щоб застосувати алгоритм до нового набору даних, а потім, нарешті, оцінити цей алгоритм на цих нових даних.

Давайте розглянемо приклад. Це, очевидно, тривіальна версія того, що трапилося б у справжньому алгоритмі машинного навчання, але воно дає вам уявлення про те, що відбувається. Ви починаєте з того, що запитуєте щось про питання. Зазвичай люди починають із досить загальних питань. Отже, ось: "Чи можу я автоматично виявляти електронні листи, які є спамом, від тих, які такими не є?" Спам – це електронні листи, які надходять від компаній, які одночасно надсилаються тисячам людей і які можуть вас не зацікавити. Ви можете сформулювати своє запитання трохи конкретніше. Це часто потрібно під час машинного навчання. Запитання може бути таким: "Чи можу я використовувати кількісні характеристики цих електронних листів, щоб класифікувати їх як спам?"

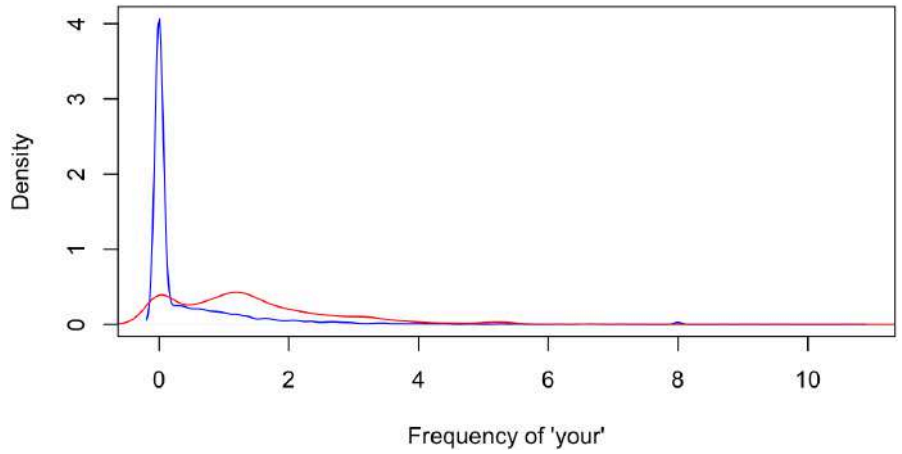
Якщо у вас є питання, вам потрібно знайти вхідні дані. У цьому випадку фактично є купа даних, які доступні та вже попередньо оброблені для вас у R. Тож насправді вони знаходяться в пакеті kernlab і це набір даних spam. Ви фактично можете завантажити цей набір даних безпосередньо в R, і він містить деяку інформацію, яка була зібрана про спам-повідомлення, які вже доступні вам. Тепер ви можете мати на увазі, що це не обов'язково ідеальні дані, насправді у вас немає всіх електронних листів, які були зібрані протягом певного часу, або у вас немає всіх електронних листів, які надсилаються вам особисто. Отже, ви повинні знати про потенційні обмеження цих даних, коли ви використовуєте їх для побудови алгоритму прогнозування.

Тоді ви хочете обчислити щось щодо властивостей. Отже, уявіть, що у вас є купа електронних листів. Ось приклад електронного листа, надісланого вам. "Dear [Name], can you send me the address, so I can send you the invitation. Thanks, Ben.". Якщо ви хочете побудувати алгоритм прогнозування, вам потрібно обчислити деякі характеристики цих електронних листів, які можна використовувати, щоб мати можливість побудувати алгоритм прогнозування. Одним із прикладів може бути те, що ви можете обчислити частоту, з якою з'являється певне слово. Тут ви шукаєте частоту появи слова "you". У цьому випадку воно з'являється двічі в цьому електронному листі, тому 2 із 17 слів або близько 11% слів у цьому електронному листі – це "you". Ви можете розрахувати той самий відсоток для кожного окремого електронного листа, який у вас є, і тепер у вас є якісна характеристика, яку ви можете спробувати використати для прогнозування.

```
library(kernlab)
data(spam)
head(spam)
#   make address  all num3d  our over remove internet order mail receive will
# 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00 0.64
# 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21 0.79
# 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38 0.45
# 4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31 0.31
# 5 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31 0.31
# 6 0.00    0.00 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00 0.00
```

Дані в пакеті kernlab, які тут показано, насправді є просто такою інформацією: у кожному електронному листі ви вказуєте частоту, з якою з'являються певні слова. Наприклад, якщо в електронному листі дуже часто з'являється "credit" або "money", ви можете уявити, що цей електронний лист може бути спамом.

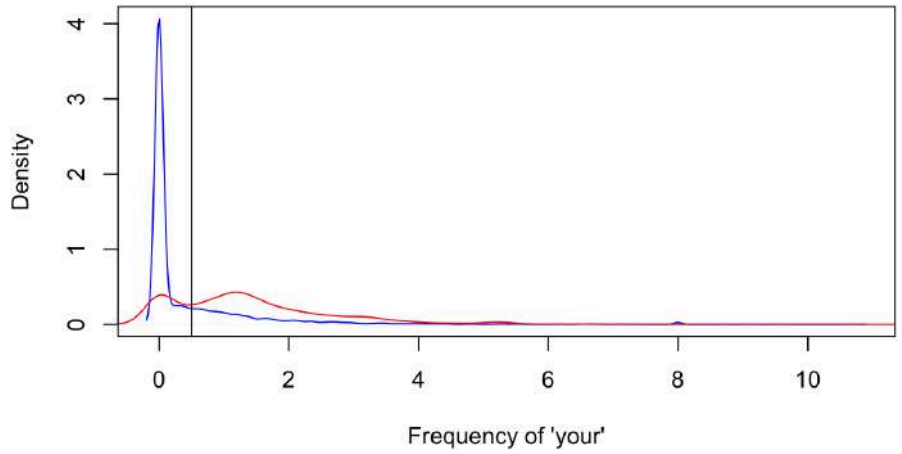
```
plot(density(spam$your[spam$type=="nonspam"]), col="blue", main="",
xlab="Frequency of 'your'")
lines(density(spam$your[spam$type=="spam"]), col="red")
```



Як один із прикладів цього ви подивилися на частоту використання слова “your” і на те, як часто воно з’являється в електронному листі. У вас є графік, це графік щільності даних. На осі x – частота, з якою “your” з’являлося в електронному листі. На осі y відкладено щільність або кількість разів, коли така частота з’являється серед електронних листів. Ви можете помітити, що більшість електронних листів, які є спамом, це ті, які позначено червоним кольором, ви можете побачити, що в них частіше зустрічається слово “your”. У той час як усі електронні листи, які ви насправді хочете отримувати, мають набагато вищий пік біля 0.

Ви можете побудувати алгоритм, у цьому випадку давайте створимо дуже дуже простий алгоритм. Ви можете оцінити алгоритм, де потрібно просто знайти поріг константи C, де якщо частота “your” вище C, ви прогнозуєте спам, а в іншому випадку ви прогнозуєте, що це не спам.

```
plot(density(spam$your[spam$type=="nonspam"]), col="blue", main="",
xlab="Frequency of 'your'")
lines(density(spam$your[spam$type=="spam"]), col="red")
abline(v=0.5, col="black")
```



Повертаючись до даних, ви можете спробувати з’ясувати, яким є найкращий поріг. Ось приклад порогу, який ви можете вибрати, тож виберіть тут поріг, який, якщо

він вище 0,5, означає, що це спам, а якщо він нижче 0,5, ви можете сказати, що це не спам. Ви думаєте, що це може спрацювати, оскільки ви бачите, що великий сплеск синіх повідомлень, які не є спамом, знаходиться нижче межі. У той час як один із великих сплесків спам-повідомлень перевищив цю межу. Ви можете собі уявити, що це вловить досить багато цього спаму. Тоді ви це оцінюєте.

```
prediction <- ifelse(spam$your > 0.5, "spam", "nonspam")
table(prediction, spam$type)/length(spam$type)
# prediction   nonspam      spam
#    nonspam 0.4590306 0.1017170
#    spam    0.1469246 0.2923278
```

Щоб ви б зробили, наприклад, розрахували прогнози для кожного з різних електронних листів. Ви берете прогноз, у якому сказано, що якщо частота “your” вище 0,5, то це спам. Потім ви складаєте таблицю цих прогнозів і ділите її на довжину всіх спостережень, які у вас є. Ви можете сказати, що якщо це не спам у 46% випадків, ви визначили правильно. Якщо приблизно 29% випадків це спам, ви визначили правильно. Отже, загальна сума, яку ви отримуєте, приблизно 46% плюс 29% – це приблизно 75%. Алгоритм прогнозування в цьому конкретному випадку точний приблизно на 75%. Ось як би ви оцінили алгоритм. Звичайно, це будь-який той самий набір даних, де ви фактично обчислили функцію прогнозування і, як ви побачите пізніше, це буде оптимістична оцінка загального рівня помилок.

3. Відносна важливість кроків

Згадайте, що говорилося про різні компоненти створення алгоритму машинного навчання. Ви почали із запитання. Це запитання, на яке ви намагаєтеся відповісти, і ви намагаєтеся зробити його якомога конкретнішим. Потім ви збираєте деякі дані, які ви збираєтеся використовувати для створення алгоритму машинного навчання та застосовувати цей алгоритм з ними пізніше. Ці дані ви стискаєте у набір властивостей, які використовуватимете для прогнозування. Потім ви вводите ці властивості в алгоритм, який потім використовуватимете для прогнозування.

Запитання є найважливішим компонентом побудови хорошого алгоритму машинного навчання, отримання дуже конкретного запитання, за яким можна збирати дані. Наступним найважливішим кроком є можливість фактично збирати дані. Часто на запитання, на яке ви справді хочете відповісти, дані будуть недоступні або вони будуть доступні лише у дотичній формі. Тоді створення властивостей є важливим компонентом, адже якщо ви не стиснете дані належним чином, ви можете втратити всю відповідну та цінну інформацію. Нарешті, алгоритм часто є найменш важливою

частиною створення алгоритму машинного навчання. Це може бути дуже важливим залежно від точної модальності типу даних, які ви використовуєте. Наприклад, для даних зображень і голосових даних можуть знадобитися певні типи алгоритмів прогнозування, які не обов'язково можуть бути такими важливими для різних типів прогнозування.

Важливий момент, на який варто звернути увагу, фактично вперше процитував Джон Тьюкі, який полягає в тому, що поєднання деяких даних і болючого бажання отримати відповідь не гарантує, що обґрунтовану відповідь можна буде витягти з даного масиву даних. Іншими словами, важливий компонент знання того, як робити прогнози, полягає в тому, щоб знати, коли здаватися, коли даних, які у вас є, просто недостатньо, щоб відповісти на запитання, на яке ви намагаєтеся відповісти.

Основний момент, який вам потрібно запам'ятати, коли ви приймаєте це рішення, – сміття на вході, сміття на виході. Іншими словами, якщо у вас є погані дані, які ви зібрали, або дані, які не є дуже корисними для прогнозування, незалежно від того, наскільки добрий ваш алгоритм машинного навчання, ви часто отримаєте дуже погані результати. Загалом найлегше прогнозувати, коли у вас є дані про те саме, що ви намагаєтеся прогнозувати. Іншими словами, у Netflix Prize вони намагалися прогнозувати нові оцінки фільмів, як люди оцінюватимуть фільми, і вони намагалися зробити це прогнозування на основі старих оцінок фільмів. Це дуже прямий зв'язок; ви можете уявити, як старі оцінки фільмів будуть контролюватися подібним процесом, як нові оцінки фільмів. Одне з того, що ви можете зробити, – це збирати молекулярні виміри, такі як експресія генів, це показник того, що відбувається в вашому тілі. Ви можете використовувати це, щоб спробувати прогнозувати, як люди реагують на хвороби або на інші різні медичні стани. Ви можете уявити шлях від даних про експресію генів, які ви зібрали, до молекулярної інформації, яку ви зібрали, до значення вашого стану захворювання. Загалом це залежить від того, що робить хороше прогнозування, які властивості ви збираєте, щоб мати можливість прогнозувати. Але в цілому, чим ближче ви можете дістатися до подібних типів даних, тим ближче ви можете дістатися до високої точності прогнозування.

Часто отримання набагато більше даних може перемогти отримання набагато кращих статистичних моделей або моделей машинного навчання у майже кожному випадку. Найважливішим кроком у розробці алгоритму машинного навчання після визначення запитання є збір правильних даних, які є відповідними для запитання, на яке ви намагаєтеся відповісти. Властивості також важливі. Так, важливі якості хороших

властивостей – це те, що вони стискають дані таким чином, що дозволяють обчислити ваш прогноз або ваш алгоритм машинного навчання дуже простим способом і, можливо, є більш зрозумілими, ніж дані, які ви зібрали, які є сирими даними, що можуть бути складними. Хороші властивості зберігають всю необхідну інформацію, коли дані стискаються, і зберігають баланс, що може бути складним. Вони зазвичай створюються з експертним знанням застосування, і є дискусія в спільноті щодо того, чи краще створювати властивості автоматично, чи краще використовувати експертне знання у сфері. Загалом здається, що експертне знання у сфері може допомогти досить значно в багатьох, багатьох застосуваннях і тому має бути використане при створенні властивостей для алгоритму машинного навчання. Деякі спільні помилки включають спробу автоматизувати вибір властивостей таким чином, що не дозволяє зрозуміти, як ці властивості фактично застосовуються для створення хороших прогнозів. Прогнози "чорних скриньок" можуть бути дуже корисними і точними, але вони також можуть раптово змінюватися, якщо ви не приділяєте уваги тому, як ці властивості фактично прогнозують результат. Неуважність до конкретних особливостей даних – це ще одна проблема, яка може виникнути у деяких випадках. Властивість певного набору даних може полягати в тому, що є викиди, якщо виявляються дивні поведінки певних ознак, і нерозуміння цього може викликати проблеми. І, очевидно, викидати інформацію без потреби – не гарна ідея.

Іноді, з обережністю, ви можете автоматизувати вибір властивостей. Фактично, існує цілий напрямок досліджень, присвячений цьому, який відомий як напів контрольоване навчання або глибоке навчання. Ідея в цій статті полягала в тому, щоб спробувати виявити властивості відеороликів YouTube, які пізніше можна було б використати в алгоритмах прогнозування. Наприклад, їм вдалося створити дуже точний прогнозатор того, чи є кіт у відео, на основі набору властивостей, зібраних способом неконтрольованого навчання. Іншими словами, вони відфільтрували дані, щоб виявити властивості, які можуть бути корисними для майбутніх алгоритмів прогнозування. Проте, навіть коли вони це робили, вони повертались до цих властивостей і намагалися зрозуміти, чому вони можуть прогнозувати. Наприклад, ця конкретна властивість демонструє, чому вона може бути хорошим прогнозувальником кота, оскільки ви можете побачити зображення кота у відео – властивість, яку вони зібрали.

Алгоритми мають менше значення, ніж ви можете подумати, і це може бути джерелом здивування та роздратування для деяких людей. Ось таблиця, де насправді намагалися прогнозувати різноманітні задачі прогнозування, наприклад, сегментаційна задача, прогнозування голосів в Палаті представників США та прогнозування хвильових форм, а також кілька інших різних задач прогнозування. І так вони зробили це двома різними способами: спочатку вони використовували щось, що називається лінійним дискримінантним аналізом, який є досить базовим і раннім прогнозувальником, який ви можете вивчити. А потім вони також намагалися для кожного набору даних знайти абсолютно найкращий алгоритм прогнозування, який вони могли мати, і потім ця таблиця показує помилку прогнозування цих двох різних підходів. І ви можете побачити, що найкраща помилка прогнозування завжди трохи краща за помилку лінійного дискримінантного аналізу. Але, фактично, вона не настільки віддалена. Так, наприклад, тут, у цьому прогнозі Pima, була помилка 0,197 з кращим методом. І 0,22 для лінійного дискримінантного аналізу. Здається, що повторне використання того самого методу знову і знову робить помилку прогнозування гірше, але не надто драматично. Цей сценарій поширений у багатьох додатках, де застосування розумного підходу часто дає значний прогрес у вирішенні проблеми. Хоча отримання абсолютно найкращого методу може призвести до покращення, запас покращення часто обмежений порівняно з більшістю хороших, розумних методів.

Performance of linear discriminant analysis and the best result we found on ten randomly selected data sets

Data set	Best method e.r.	Lindisc e.r.	Default rule	Prop linear
Segmentation	0.0140	0.083	0.760	0.907
Pima	0.1979	0.221	0.350	0.848
House-votes16	0.0270	0.046	0.386	0.948
Vehicle	0.1450	0.216	0.750	0.883
Satimage	0.0850	0.160	0.758	0.889
Heart Cleveland	0.1410	0.141	0.560	1.000
Splice	0.0330	0.057	0.475	0.945
Waveform21	0.0035	0.004	0.667	0.999
Led7	0.2650	0.265	0.900	1.000
Breast Wisconsin	0.0260	0.038	0.345	0.963

При побудові алгоритму машинного навчання слід враховувати кілька факторів, оскільки в ньому беруть участь різні компоненти. По-перше, можливо, його потрібно буде інтерпретувати, особливо якщо він буде представлений професіоналам, наприклад лікарям, або генеральним директорам веб-компанії. Наявність інтерпретованого предиктора дозволяє їм зрозуміти його функціонування. Інтерпретабельність часто передбачає простоту; те, що легко пояснити, краще, ніж бути надто складним і важким для розуміння. Однак досягнення інтерпретації та простоти

може включати компроміси з точністю. Виникає питання: наскільки допустиме зниження точності, щоб зберегти інтерпретабельність і простоту? Крім того, алгоритм повинен бути масштабованим і швидким. Під швидким мається на увазі, що дуже легко створити модель, дуже легко перевірити модель на невеликих вибірках. Масштабований означає, що його легко застосувати до великого набору даних. Чи то тому, що він дуже швидкий, чи то тому, що його можна розпаралелювати, наприклад, для кількох вибірок.

Загалом прогнозування стосується компромісів із точністю. Ідея полягає в тому, що іноді ви будете торгувати великою точністю або невеликою частиною точності для інтерпретації. Іноді ви торгуватиметеся швидкістю, простотою чи масштабованістю. Але загалом ви намагаєтеся знайти правильний баланс між тими іншими властивостями алгоритму прогнозування, які можуть бути для вас важливими, і тим, наскільки точним він повинен бути. Інтерпретабельність часто має значення, наприклад, у медичних дослідженнях.

if total cholesterol $\geq$160 **and** smoke **then** *10 year CHD risk* $\geq$ *5%*
else if smoke **and** systolic blood pressure$\geq$140 **then** *10 year CHD risk* $\geq$ *5%*
else *10 year CHD risk* $<$ *5%*

У цій статті досліджується, як деяким лікарям подобаються кращі правила прогнозування, які виглядають ось так, вони трохи схожі на дерева рішень. Іншими словами, ви починаєте і говорите, що якщо загальний холестерин перевищує певне значення, і ви курите, то ви отримуєте певний результат. Деяким людям можуть бути дуже зрозумілі твердження такого типу, якщо/тоді, і це приблизно причина, чому людям подобаються такі речі, як дерева рішень. Це може мати більше чи менше значення залежно від мети вашого алгоритму. Але можливість інтерпретації запобігає таким проблемам, як проблема Google Flu Trends, коли люди насправді не розуміли, як властивості безпосередньо пов'язані з рівнем грипу. Якщо ви не розумієте, як працюють властивості, вас можуть спіймати, коли в алгоритмі почнуть виникати проблеми або збій.

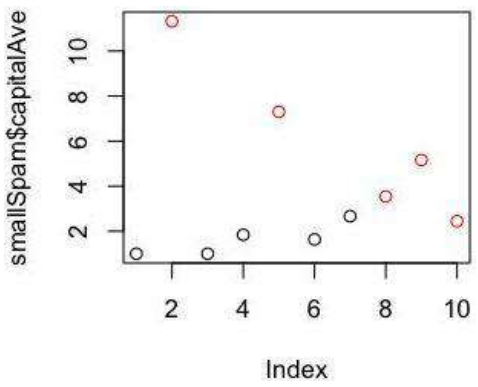
Масштабованість також має значення. Дуже цікавим компонентом цього виклику Netflix є те, що Netflix Prize був завданням у мільйон доларів, там змагалося багато команд, і кілька команд розробили аналітичні алгоритми, які були набагато кращими в прогнозуванні, ніж оригінальний алгоритм Netflix. Але виявилося, що вони насправді ніколи не реалізували остаточний алгоритм на своєму виробничому

обладнанні, і причина була в цьому, що алгоритм не дуже добре масштабується. Це було поєднання багатьох різних алгоритмів машинного навчання, і на виконання потрібно було багато часу, особливо на величезних наборах даних, які збирав Netflix. Тож вони фактично пішли з чимось менш точним, але трохи більш масштабованим. Це свідчить про те, що точність не завжди найкраща. Під час створення алгоритмів прогнозування важливі лише рішення. Хоча часто саме на точність звертають увагу найбільше.

4. Помилки у вибірці та поза її межами

Помилки у вибірці та поза її межами є одними з найбільш фундаментальних концепцій машинного навчання та прогнозування. Розуміння цієї концепції є важливим, і простий приклад може пояснити її. Помилки у вибірці стосуються помилок, які спостерігаються в тих самих даних, які використовуються для навчання предиктора. У літературі з машинного навчання це іноді називають помилкою повторної підстановки. Помилка у вибірці має тенденцію бути дещо оптимістичною, оскільки алгоритм прогнозування може дещо підлаштовуватися під шум, присутній у цьому конкретному наборі даних. При застосуванні до нових даних точність зазвичай знижується через різні характеристики шуму. Помилка поза вибіркою, також відома як помилка узагальнення в машинному навчанні, – це частота помилок, яка спостерігається під час тестування моделі на новому наборі даних, зібраному незалежно. Це допомагає оцінити реалістичне очікування того, наскільки добре алгоритм працюватиме на нових даних. У більшості випадків першорядне занепокоєння викликають помилки поза вибіркою. Якщо показники помилок наводяться лише на основі даних навчання, це надто оптимістично та може не відображати реальну продуктивність. Помилка у вибірці завжди нижча, ніж помилка поза вибіркою через перепідгонку. Перепідгонка відбувається, коли алгоритм надто точно підходить до конкретного набору даних, жертвуючи можливістю узагальнення. Іноді жертвуючи точністю в навчальній вибірці, ви може підвищити точність нових наборів даних, забезпечуючи стійкість алгоритму до різних рівнів шуму.

```
library(kernlab)
data(spam)
set.seed(3)
smallSpam <- spam[sample(dim(spam)[1], size=10),]
spamLabel <- (smallSpam$type == "spam") * 1 + 1
plot(smallSpam$capitalAve, col=spamLabel)
```



Давайте розглянемо дуже простий приклад. Тут ви використовуєте пакет kernlab і переглядаєте набір даних spam. Що ви робите, так це ви фактично берете дуже маленьку вибірку цього набору даних spam і дивитеся, чи багато ви бачите великих літер. Отже, ви в основному дивитеся на середню кількість великих літер, які ви спостерігаєте в конкретному електронному листі. Тут ви зображаєте на графіку перші десять прикладів у порівнянні з їхнім індексом. Червоним кольором позначені всі спам-повідомлення, чорним – усі не спам-повідомлення. Ви можете побачити, наприклад, що деякі спам-повідомлення мають набагато більше великих літер, ніж ті, які не є спамом. Це має сенс інтуїтивно. Тож ви можете створити прогноз на основі середньої кількості великих літер щодо того, чи є у вас спам-повідомлення чи ні. Одне, що ви можете зробити, це побудувати предиктор, який скаже, що якщо у вас багато великих літер, у вас є спам, а якщо ні, то у вас немає спаму. Ось як це правило може виглядати: ви можете сказати, що якщо у вас середнє значення вище за 2,7 великих літер, ви називатимете це спамом, якщо у вас менше 2,4, ви класифікуєте як не спам. Тоді ви можете спробувати дуже добре навчити цей алгоритм для ідеального прогнозування на цьому наборі даних. Якщо ви повернетесь до графіка різних значень, ви побачите, що в нижньому правому куті є одне спам-повідомлення, яке трохи нижче за найвище значення, яке не є спамом, з точки зору середнього значення великої літери. Ви можете створити алгоритм прогнозування, який також фіксуватиме значення спаму. Що б ви зробили тоді, ви б створили правило, яке просто вибирає одне значення з навчального набору. Там сказано, що якщо ваш рівень між 2,4 і 2,45, ви також називаєте це спамом. Це розроблено для того, щоб зробити точність навчального набору ідеальною.

```
rule1 <- function(x){
  prediction <- rep(NA, length(x))
  prediction[x > 2.7] <- "spam"
  prediction[x < 2.40] <- "nonspam"
  prediction[(x >= 2.40 & x <= 2.45)] <- "spam"
```

```
  prediction[(x > 2.45 & x <= 2.70)] <- "nonspam"
  return(prediction)
}
table(rule1(smallSpam$capitalAve), smallSpam$type)
#           nonspam spam
#   nonspam       5    0
#   spam          0    5
```

Ви можете побачити, якщо застосувати це правило до навчального набору, ви дійсно отримаєте ідеальну точність. Якщо у вас немає спаму, ви цілком класифікуєте це як не спам, а якщо у вас є спам, ви ідеально класифікуєте його як спам.

```
rule2 <- function(x){
  prediction <- rep(NA, length(x))
  prediction[x > 2.4] <- "spam"
  prediction[x <= 2.4] <- "nonspam"
  return(prediction)
}
table(rule2(smallSpam$capitalAve), smallSpam$type)
#           nonspam spam
#   nonspam       5    1
#   spam          0    4
```

Альтернативне правило не буде настільки щільно прив’язуватися до навчального набору, але все одно використовуватиме основний принцип: якщо у вас велика кількість великих літер, у вас є спам-повідомлення, тому це правило може виглядати приблизно так. Якщо ваш показник вище 2,40, це спам, якщо менше або дорівнює 2,40, це не спам. Отже, це правило в навчальному наборі пропустить це одне значення. Іншими словами, ви могли б мати прогноз, що це не спам, для того одного спам-повідомлення, яке було лише трохи нижче в навчальному наборі. Отже, загалом виглядає так, що в цьому навчальному наборі точність цього правила трохи нижча, і воно трохи простіше.

```
table(rule1(spam$capitalAve), spam$type)
#           nonspam spam
#   nonspam    2141  588
#   spam        647 1225
table(rule2(spam$capitalAve), spam$type)
#           nonspam spam
#   nonspam    2224  642
#   spam        564 1171
```

Потім ви можете застосувати його до всього набору даних spam. Іншими словами, застосуйте його до всіх значень, а не лише до значень, які ви мали в невеликому навчальному наборі, і це результати, які ви отримаєте. Прогнози в цій таблиці містяться в рядках, а в стовпцях – це фактичні значення. Ви можете побачити кількість помилок, які ви робите, це помилки, які знаходяться на недіагональних

елементах цієї маленької матриці, яку ви створили. Це кількість помилок, які ви зробили.

```
sum(rule1(spam$capitalAve)==spam$type)
# [1] 3366
sum(rule2(spam$capitalAve)==spam$type)
# [1] 3395
```

Те, на що ви можете подивитися, це середня кількість разів, які були правильними, використовуючи більш складні правила. Це лише сума випадків, коли прогноз дорівнює фактичному значенню в наборі даних spam. Це трапляється 3366 разів у цьому наборі даних. Тоді ви також можете переглянути більш спрощене правило, правило, де ви щойно використовували порогове значення, а також подивитись на кількість разів, коли це дорівнює справжньому типу спаму. Ви можете побачити ще приблизно 30 разів, ви фактично отримаєте правильну відповідь, коли використовуєте це більш спрощене правило.

У чому причина того, що спрощене правило насправді працює краще, ніж складніше правило? Причина – перепідгонка. У кожному наборі даних ви маєте дві частини, у вас є компонент сигналу, це та частина, яку ви намагаєтесь використати для прогнозування. І тоді у вас є шум, тож це просто випадкова варіація в наборі даних, який ви отримуєте, оскільки дані вимірюються з шумом. Мета предиктора – знайти сигнал і ігнорувати шум. У будь-якому невеликому наборі даних ви завжди можете створити ідеальний предиктор у вибірці, як це було з набором даних spam. Ви завжди можете розділити простір для прогнозів у цьому невеликому наборі даних, щоб охопити кожну окрему особливість цього набору даних. Але коли ви це робите, ви вловлюєте і сигнал, і шум. Наприклад, у цьому навчальному наборі було одне значення спаму, яке має дещо нижче середнє значення великих літер, ніж деякі зі значень, які не є спамом. Але це було лише тому, що ви випадково вибрали набір даних, де це було правдою, де це значення було низьким. Таким чином, цей предиктор не обов’язково працюватиме так само добре на нових вибірках, оскільки ви надто щільно налаштували його на досліджуваний навчальний набір.

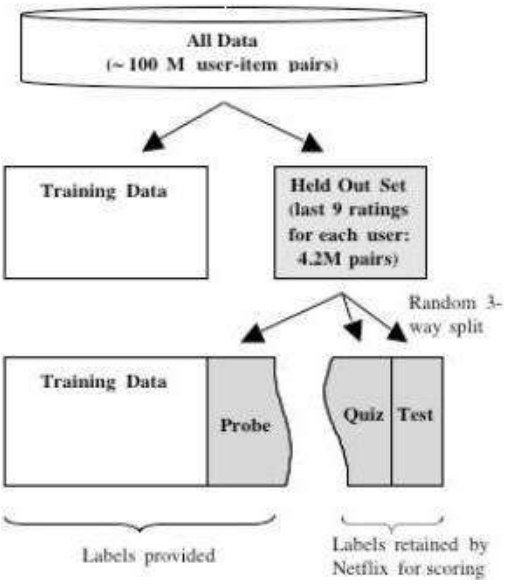
5. Проектування дослідження прогнозування

Давайте розберемося в проектуванні дослідження прогнозування і вивчимо те, як пом’якшити проблеми, пов’язані з помилками у вибірці та помилками поза її межами. У проектуванні дослідження прогнозування початковий крок включає визначення рівня помилок. Хоча спочатку може бути достатньо загальної частоти помилок, у подальших обговореннях досліджуватимуться різні потенційні частоти

помилок. Після цього дані, призначені для побудови та перевірки моделі, повинні бути розділені на два основні компоненти з необов'язковим третім компонентом. Навчальний набір життєво важливий для побудови моделі, а тестовий набір служить для оцінки її продуктивності. За бажанням можна також використати набір перевірки для подальшої перевірки моделі. На етапі навчання властивості вибираються за допомогою таких методів, як перехресна перевірка. Мета полягає в тому, щоб визначити найважливіші характеристики моделі. Згодом визначається функція прогнозування, і всі відповідні параметри оцінюються за допомогою тієї ж методики. Коли модель побудована, якщо перевірочний набір відсутній, найкраща модель застосовується до тестового набору лише один раз. Це одноразове застосування має вирішальне значення. Використання кількох моделей у наборі для тестування та вибір найкращої з них може спотворити результати, оскільки це, по суті, передбачає використання тестового набору для точного налаштування моделі. Таким чином, застосування моделі прогнозування до тестового набору лише один раз забезпечує точну оцінку її ефективності на абсолютно нових даних. Якщо доступні як набір для перевірки, так і набір для тестування, ви можете застосувати свої найкращі моделі прогнозування до набору для тестування та відповідно їх налаштувати. Під час прогнозування за межами вибірки ви можете виявити, що деякі властивості неефективні, що спонукає коригувати вашу модель. Однак важливо визнати, що помилка тестового набору може бути дещо оптимістичною порівняно з фактичною помилкою поза вибіркою. Щоб вирішити цю проблему, ви застосовуєте свою модель точно один раз до набору перевірки, використовуючи лише найефективнішу модель для створення прогнозів. Концепція полягає в тому, щоб із самого початку зарезервувати один набір даних, застосувавши до нього лише одну модель без подальшого навчання, налаштування чи тестування. Цей підхід забезпечує надійну оцінку частоти помилок поза межами вибірки.

Це проектування дослідження використовувався в конкурсі Netflix Prize. Вони почали зі 100 мільйонів пар користувач-фільм, які відображали вподобання користувачів. Цей набір даних був розділений на навчальний набір, який використовувався учасниками для побудови прогнозних моделей, і набір утримуваних оцінок, які були недоступні розробникам моделей. Поряд з навчальним набором учасникам надали набір даних зондування. Моделі були навчені за допомогою навчального набору, а потім застосовані до набору даних зондування для оцінки помилки поза вибіркою перед подачею в Netflix. Netflix використовував прогнозування,

подані учасниками, виключно на основі набору даних вікторини, який не розголошувався та не був доступний для створення моделі. Учасники можуть надіслати кілька записів до набору даних для вікторини, потенційно покращуючи свої моделі для кращої продуктивності. Для остаточної оцінки моделі всіх команд застосовувалися лише один раз до тестового набору, який притримувався до кінця змагань. Це забезпечило неупереджену оцінку того, наскільки добре моделі працюватимуть на абсолютно нових даних, яких не бачили учасники конкурсу. Ця ідея насправді була дуже важливою для проектування їхнього дослідження, і насправді виявилося, що деякі команди, які показали кращі результати для набору даних вікторини, насправді не так добре впоралися з набором даних для тестування, і це було тому, що вони налаштовували свої моделі або перепідганяли свої моделі до набору вікторини. Отже, це важливе повідомлення для вас у тому сенсі, що якщо ви будуєте моделі прогнозування, вам завжди потрібно мати один набір даних і залишати його повністю осторонь, поки ви будуєте свої моделі.



Тепер цим користуються професіонали, тому Kaggle – це компанія, яка насправді проводить багато співробітництв, конкурсів на прогнозування для цілої групи різних наборів даних, які надають компанії. І вони завжди використовують подібну модель у тому сенсі, що у них завжди є таблиця лідерів, яка складається з прогнозувань, які люди подали для перевірки набору даних, який вони мають. Але це не обов'язково набір даних, який вони використовуватимуть для перевірки своїх методів у самому кінці. Цей набір даних зберігається до самого кінця, і кожна особа може застосувати свій алгоритм до цього набору даних лише один раз.

Розбиваючи свої набори даних на набори для навчання, тестування та перевірки, важливо враховувати розмір вибірки. Важливо уникати малих розмірів вибірки, особливо в наборі для тестування. Наприклад, давайте розглянемо прогнозування бінарного результату, як-от хворі проти здорових людей або чи натиснуть користувачі на оголошення. Спрощений класифікатор може включати підкидання монети: орел для хворих, решка для здорових або натискання на рекламу проти не натискання. Ймовірність ідеальної класифікації за допомогою цього

основного методу наполовину збільшується до розміру вибірки. Іншими словами, у вас є шанс 50/50 зробити правильний прогноз просто випадково, подібно до підкидання монети. З тестовим набором, що містить лише один зразок, є 50% шанс отримати його правильно, навіть якщо підкинути монету. Навіть якщо ви досягнете 100% точності на тестовому наборі, є ймовірність 50% того, що це станеться через випадкову ймовірність. Зі збільшенням розміру тестового набору ймовірність випадкового досягнення 100% точності зменшується. Наприклад, якщо n = 2, є 25% шанс ідеальної точності; при n = 10 вона падає приблизно до 0,1%. Тому вкрай важливо переконатися, що розміри тестового набору є відносно великими, щоб гарантувати, що висока точність прогнозування не є просто випадковістю.

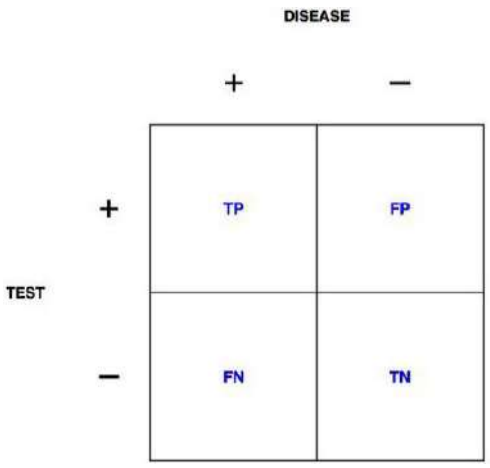
Ось кілька емпіричних правил для розділення наборів даних, хоча вони не є суворими вказівками, багато хто вважає їх практичними. Для великого набору даних зазвичай виділяють 60% на навчання, 20% на тестування та 20% на перевірку. Однак дуже важливо переконатися, що набори для тестування і перевірки не надто малі. Для набору даних середнього розміру більш доцільним може бути розподіл 60/40 між навчанням і тестуванням. Хоча це налаштування не дозволяє вдосконалювати моделі на окремому наборі для перевірки, воно забезпечує достатній розмір тестового набору. У випадку дуже маленького набору даних важливо переглянути, чи достатньо вибірок для ефективної розробки алгоритму прогнозування.

Деякі ключові принципи включають важливість відкласти тестовий або перевірочний набір і ніколи не використовувати його під час процесу побудови моделі. Це означає наявність одного набору даних, до якого застосована лише одна модель, лише один раз, і цей набір даних має бути повністю незалежним від будь-яких даних, які використовуються для побудови моделі прогнозування. Загалом доцільно робити випадкову вибірку навчальних і тестових наборів, причому випадковість залежить від типу необхідної вибірки. Наприклад, якщо ви маєте справу з даними часових рядів, ви можете сегментувати свій навчальний набір на випадкові часові інтервали. Ваш набір даних має відображати структуру проблеми. Наприклад, якщо ваші дані виявляють часові або просторові залежності, вам слід зробити їх відповідну вибірку. Цей процес, який часто називають ретроспективним тестуванням у фінансах, передбачає використання фрагментів даних, які складаються зі спостережень за певний час. Усі підмножини даних мають включати якомога більше різноманітності. Довільне призначення часто досягає цієї мети, але ви також можете розглянути можливість збалансування підмножин за функціями, хоча іноді це може бути складно.

6. Види помилок

Спершу варто звернути увагу на типи помилок, які можуть виникнути у бінарній проблемі прогнозування, де намагаються класифікувати об’єкти у одну з двох груп. Загалом це обговорюється в термінах істинно позитивних, істинно негативних, хибно позитивних та хибно негативних результатів. Основна концепція, яку варто запам’ятати, полягає в тому, що коли мова йде про істинне проти хибного, це стосується того, що алгоритм визначає, чи вважає він об’єкт належним до класу чи ні. Істинність і хибність стосуються фактичного стану світу: істинність означає, що об’єкт дійсно належить до ідентифікованого класу, тоді як хибність означає, що він не належить до цього класу. Наприклад, істинно позитивний результат означає, що ідентифікація була правильною, що свідчить про те, що є щось, що потрібно ідентифікувати, і це було вірно ідентифіковано. Навпаки, хибно позитивний результат показує неправильну ідентифікацію, коли щось помилково класифікується як належне певному класу, коли це не так.

Щоб проілюструвати цей концепт, розгляньте медичний сценарій тестування, де мета полягає в ідентифікації хворих за допомогою скринінгового тесту, наприклад, мамографія для виявлення раку молочної залози. Тут істинний аспект позначає фактичний стан здоров’я людини. Істинно позитивний результат представляє собою точну ідентифікацію хвороби, коли людина дійсно хвора і правильно ідентифікована як така. З іншого боку, хибно позитивний результат виникає, коли здорова людина помилково класифікується як хвора, незважаючи на те, що вона не хвора. Так само істинно негативний результат позначає правильну ідентифікацію здоров’я, коли здорову людину точно ідентифікують як таку. Нарешті, хибно негативний результат свідчить про помилкову ідентифікацію здоров’я для людини, яка фактично хвора, і негативна частина полягає в тому, що її ідентифікували як здорову.



Цю інформацію також можна подати у вигляді таблиці 2 на 2. Ця таблиця має два рядки і дві колонки. Колонки відображають стан хвороби. У цьому прикладі позитивний результат означає наявність хвороби, тоді як негативний результат вказує на її відсутність, що відображає фактичну правду про стан хвороби. Тест слугує прогнозом, що робить алгоритм машинного навчання. Позитивний результат вказує на прогнозовану наявність хвороби, тоді як негативний результат вказує на прогнозовану відсутність хвороби. Основні метрики, про які часто говорять, включають чутливість та специфічність. Чутливість представляє ймовірність правильного прогнозування хвороби, коли особа насправді хвора. Специфічність, з іншого боку, представляє ймовірність правильного прогнозування здорового стану, коли особа насправді здорова. Прогностична значущість позитивного результату – це ймовірність захворювання за умови позитивного прогнозу, яка відрізняється від чутливості, оскільки враховує точність позитивних прогнозів серед усіх випадків, прогнозованих як позитивні. Подібним чином, прогностична значущість негативного результату враховує точність негативних прогнозів серед усіх випадків, прогнозованих як негативні. Точність просто відображає ймовірність правильної класифікації результату. У цій таблиці точність визначається термінами на діагоналі, які представляють собою істинно позитивні та істинно негативні результати, додані разом. Ці метрики можна виразити у вигляді дробів. Наприклад, чутливість розраховується шляхом ділення істинно позитивних результатів на суму істинно позитивних та хибно негативних результатів. Той же підхід застосовується до специфічності, прогностичної значущості позитивного та негативного результатів та інших метрик. При розрахунку прогностичної значущості позитивного результату звертають увагу на істинно позитивні результати, поділені на суму істинно позитивних та хибно позитивних результатів, оскільки оцінюється точність позитивних прогнозів серед усіх позитивних результатів тесту.

У багатьох задачах прогнозування один із класів буде більш рідкісним, ніж інший. Наприклад, у медичних дослідженнях зазвичай хворіє лише невеликий відсоток людей. Розглянемо хворобу, на яку хворіє лише 0,1% генеральної сукупності. Припустимо, що є високоякісний алгоритм машинного навчання з набором для тестування, який на 99% чутливий і 99% специфічний. Іншими словами, якщо людина хвора, є 99% шансів, що тест буде правильним, а якщо людина здорова, є 99% шансів, що тест буде правильним.

Тепер давайте розглянемо два сценарії: один у генеральній сукупності, де хвороба є рідкісною, а інший, де 10% генеральної сукупності має хворобу, що робить її

більш поширеною. У першому сценарії лише близько 1% генеральної сукупності має захворювання. Так, серед хворих 99 із 100 правильно визначені як такі. Однак серед тих, хто здоровий, 99% правильно визначені як здорові. Тому, якщо людина отримує позитивний результат тесту, ймовірність справді мати недугу низька, приблизно 9%. Це пояснюється тим, що невелика кількість хворих осіб (99 із 100) затьмарюється набагато більшою кількістю здорових осіб (999 із 1000). Навпаки, якщо хворіють 10% генеральної сукупності, то хворих людей більше. У цьому випадку прогностична значущість позитивного результату значно зростає приблизно до 92%. Це означає, що якщо ви отримаєте позитивний результат тесту за цим сценарієм, існує висока ймовірність (92%), що у вас справді є захворювання. Це ілюструє важливість врахування поширеності події, яку ви прогнозуєте. Під час створення моделі прогнозування важливо розуміти генеральну сукупність, з якої ви берете вибірку.

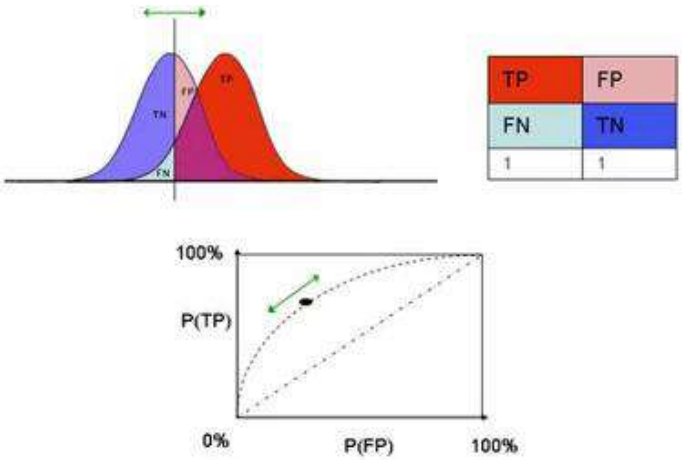
Для безперервних даних не існує такого простого сценарію, коли можливі лише два випадки та два типи помилок. Мета тут – побачити, наскільки людина близька до істини. Один із поширених способів зробити це називається середньоквадратичною помилкою. Ідея полягає в тому, що у вас є прогнозування від моделі або алгоритму машинного навчання для кожної окремої вибірку, яку ви намагаєтесь прогнозувати. Ви також можете знати правду щодо цих вибірок, скажімо, у тестовому наборі. Отже, обчислюється різниця між прогнозуванням і правдою, зводиться в квадрат, щоб усі числа були позитивними, а потім усереднюється загальна відстань між прогнозуванням і правдою. Одна річ, яка трохи складна в інтерпретації цього числа, полягає в тому, що воно зведене в квадрат, що робить його трохи складним інтерпретувати в тій самій шкалі, що й прогнозування чи правду. Отже, те, що люди часто роблять, це беруть квадратний корінь із цієї кількості, даючи середню квадратичну помилку. Це, мабуть, найпоширеніша міра помилки, яка використовується для безперервних даних.

Для безперервних даних люди часто використовують середньоквадратичну помилку. Але це часто не працює, коли є багато викидів або значення змінних можуть мати дуже різні масштаби, оскільки вона буде чутливою до цих викидів. Замість цього можна використати середнє абсолютне відхилення, яке бере медіану відстані між спостережуваним значенням і прогнозованим значенням і використовує абсолютне значення замість його зведення у квадрат, що робить його більш стійким до розміру цих помилок. Чутливість і специфічність дуже часто використовуються, коли говорять про певні медичні тести, але вони також особливо широко використовуються, якщо

хтось піклується про один тип помилки більше, ніж про інший тип помилки. Точність, яка порівнює хибно позитивні та хибно негативні результати, є важливим моментом, якщо є дуже велика розбіжність у кількості позитивних чи негативних результатів. Для багатокласових випадків можна мати щось на зразок узгодженості, і існує цілий великий клас мір відстані, кожна з яких має різні властивості, які можна використовувати при роботі з багатокласовими даними. Це деякі з поширених вимірювань помилок, які використовуються під час впровадження алгоритмів прогнозування.

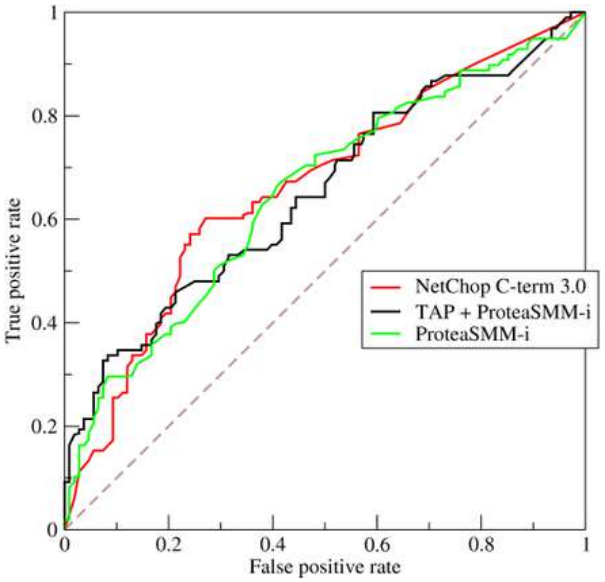
7. ROC-криві

ROC-криві – це методи, які зазвичай використовуються для вимірювання якості або ефективності алгоритму прогнозування. У бінарній класифікації зазвичай прогнозується одна з двох категорій. Наприклад, люди можуть бути прогнозовані як живі чи мертві, або хворі чи здорові. Прогнози часто дають кількісні результати; більшість алгоритмів моделювання надають ймовірності, а не призначають осіб безпосередньо до класу. Вибране граничне значення впливає на результат. Наприклад, якщо ймовірність бути живим прогнозується як 0,7, а особи зі значенням більше за 0,5 призначаються як живі, це представляє один алгоритм прогнозування. Крім того, встановлення порогу на рівні 0,28 призведе до іншого прогнозу для тієї самої особи. Різні граничні значення мають різні властивості; вони можуть краще ідентифікувати людей, які живі, або краще ідентифікувати людей, які мертві.



Використовуючи ROC-криву, люди зазвичай відкладають одиницю мінус специфічність на осі x, що представляє ймовірність хибно позитивного результату. На осі y вони відкладають чутливість або ймовірність істинно позитивного результату. Мета полягає в тому, щоб створити криву, де кожна точка відповідає певному граничному значенню. Для кожного граничного значення існує відповідна ймовірність

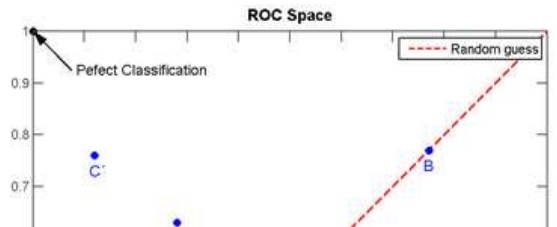
хибно позитивного результату або специфічності, позначена одиницею мінус специфічність на осі x і значення чутливості на осі y. ROC-криві дозволяють оцінити продуктивність алгоритму, побудувавши різні точки, що відповідають різним граничним значенням, а потім з'єднавши ці точки, щоб сформувати криву. Це ілюструє, як різні алгоритми прогнозують результати.



Це ілюстрація кількох алгоритмів, які використовуються для прогнозування, надає приклади автентичних ROC-кривих. Розглянемо, наприклад, алгоритм NetChop. Якщо дослідити, де 1 мінус специфічність дорівнює 0,2, що вказує на високу специфічність 0,8, відповідна точка на кривій буде приблизно тут, зображена червоною кривою. Ця точка відображає чутливість приблизно 0,4, що означає високу специфічність, але відносно низьку чутливість. Просуваючись далі по осі x праворуч, специфічність зменшується, а чутливість зростає, що ілюструє компроміси. Хоча ROC-криві пропонують зрозуміти ці компроміси, можна спробувати визначити перевагу алгоритмів прогнозування. Одним із поширених методів порівняння кривих є обчислення площі під кривою. Чим більша площа, тим ефективніший предиктор.

Далі наведено деякі стандартні значення, на які має сенс звернути увагу. Якщо площа під кривою або AUC дорівнює 0,5, це еквівалентно алгоритму прогнозування, який знаходиться лише на лінії 45 градусів, яка, по суті, випадковим чином вгадує, чи ви отримуєте істинно позитивний чи хибно позитивний результат. 0,5 насправді дуже погано, все, що менше 0,5, насправді гірше, ніж випадкове вгадування, наприклад підкидання монети. AUC, що дорівнює 1, представляє ідеальний класифікатор, що означає ідеальну чутливість і специфічність для певного значення алгоритму прогнозування, таким чином становлячи ідеальний класифікатор. Загалом це залежить від галузі та проблеми, що розглядається; люди часто вважають AUC вище 0,8 показником хорошого AUC для певного алгоритму прогнозування.

Загалом, важливо розглянути, як можна проаналізувати ROC-криву, щоб визначити її якість. Пам'ятайте, що лінія під кутом 45 градусів відповідає випадковому вгадуванню, де чутливість і специфічність



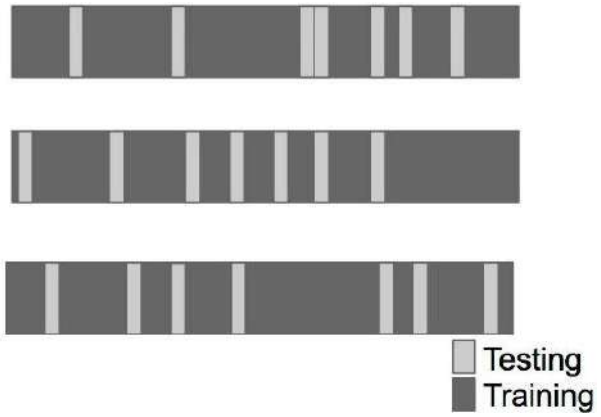
відповідають одна одній. Ідеальний класифікатор узгоджується з одиницею мінус специфічність; коли специфічність дорівнює нулю, це вказує на ідеальну специфічність, перехід безпосередньо до ідеальної чутливості. Така крива являє собою ідеальний класифікатор, коли вона досягає верхнього лівого кута. Чим ближче крива до верхнього лівого кута графіка, тим кращий ROC. І навпаки, ближче до нижнього правого кута вказує на гіршу ROC. Криві вздовж або нижче лінії 45 градусів зазвичай вважаються поганими. В ідеалі крива повинна круто підніматися до верхнього лівого кута й охоплювати якомога більшу площу. Ось як інтерпретується ROC-крива, яка корисна для вибору конкретних значень предикторів для двійкових виведень, які генерують кількісні числа.

8. Перехресна перевірка

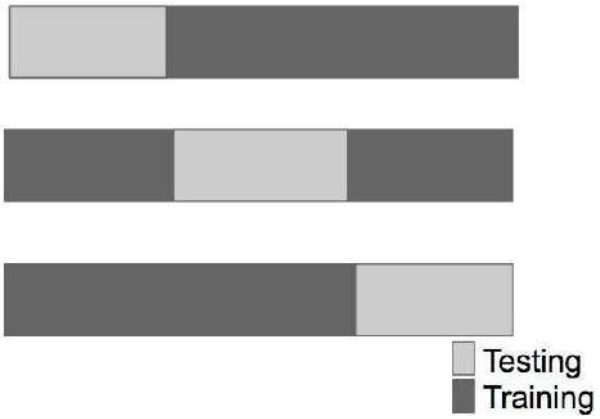
Перехресна перевірка є одним із найпоширеніших інструментів для виявлення відповідних властивостей і побудови моделей, а також оцінки їхніх параметрів у машинному навчанні. Проводячи експерименти з машинним навчанням, дослідники часто використовують певне проектування дослідження. Наприклад, у дослідженні Netflix Prize використовується набір даних, що містить 100 000 000 пар “користувач-елемент”, де фільми оцінюються конкретними користувачами. Цей набір даних поділяється на навчальний набір даних і тестовий набір даних. Побудова та оцінка моделі відбувається виключно на основі навчального набору даних, а остаточне тестування зарезервовано для тестового набору даних після завершення змагань. Однак виникає поширена проблема, коли точність на навчальному наборі, відома як точність повторної заміни, має тенденцію бути надто оптимістичною. Це пов’язано з тим, що обрана модель може бути точно підігнана відповідно до особливостей навчальних даних, що потенційно може призвести до неточного представлення точності прогнозу на нових зразках. Щоб отримати кращу оцінку, слід використовувати незалежний набір даних. Тим не менш, постійне використання тестового набору для цілей оцінювання ефективно включає його в навчальний набір, підриваючи потребу у зовнішньому вимірюванні помилки тестового набору. Щоб вирішити цю проблему, дослідники звертаються до перехресної перевірки, щоб оцінити точність набору для тестування.

Концепція перехресної перевірки передбачає взяття навчального набору, зокрема навчальних зразків, і поділ його на менші підмножини: навчальну підмножину та тестову підмножину. Потім модель будується на навчальній підмножині та оцінюється на тестовій підмножині. Цей процес повторюється кілька разів, причому

кожна ітерація забезпечує оцінену похибку. Усереднюючи ці оцінені похибки, дослідники отримують уявлення про ефективність моделі при представленні нових даних тестування. По суті, навчальний набір неодноразово ділиться на навчальні та тестові підмножини, що дозволяє перебудовувати моделі та вибирати найефективнішу на основі продуктивності на тестових наборах. Цей метод виявляється корисним для вибору змінних для включення в модель. Крім того, дослідники можуть експериментувати з різними алгоритмами прогнозування або коригувати параметри в межах функції прогнозування, зрештою вибираючи конфігурацію, яка найкраще працює на перехресно перевірених наборах тестування. Важливо, що протягом цього процесу вихідний тестовий набір залишається недоторканим і ніколи не використовується. Таким чином, коли остаточний алгоритм прогнозування застосовується до тестового набору, він забезпечує неупереджене вимірювання точності за межами вибірки.



Одним із підходів є використання випадкової підвибірки. Кожне спостереження, призначене для прогнозування, розташоване вздовж осі, кольором позначене включення до навчального або тестового набору. У цьому сценарії розглядаються лише навчальні вибірки. Підмножина цих вибірок вибирається випадковим чином і позначається як вибірка для тестування, як показано світло-сірими смугами. Потім предиктор будується з використанням решти темно-сірих вибірок. Цей процес повторюється для кількох випадкових підвибірок, причому кожна ітерація представляє іншу випадкову вибірку. Після кількох повторень помилки усереднюються для отримання загальної оцінки.



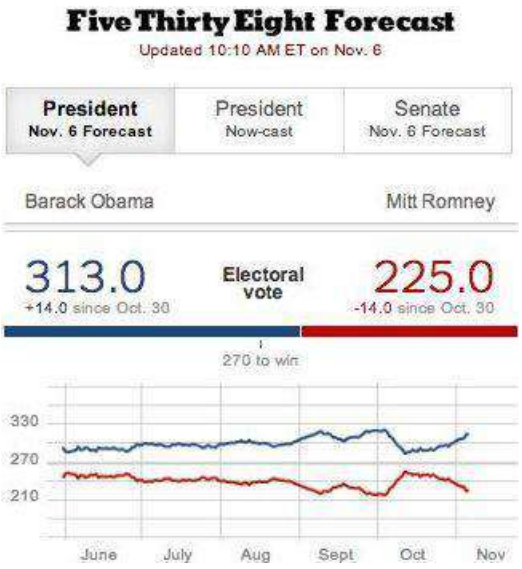
Інший підхід, який широко використовується, відомий як K-кратна перехресна перевірка. Концепція передбачає поділ набору даних на k підмножин однакового розміру. Наприклад, у трикратній перехресній перевірці набір даних розділено на три частини, як показано. На кожному кроці модель прогнозування будується з використанням підмножини навчальних даних (представленої темно-сірими компонентами), а потім застосовується до відповідної підмножини тестових даних (представленої світло-сірими компонентами). Цей процес повторюється для кожного кроку, при цьому помилки усереднюються за всіма експериментами, щоб оцінити середній рівень помилок у процедурі оцінки поза вибіркою. Уся побудова та оцінка моделі відбувається в межах навчального набору, який підрозділяється на підмножини для навчання та тестування для оцінювання моделей.

Інший дуже поширений підхід називається перехресною перевіркою за винятком одного. У цьому методі кожен зразок пропускається рівно один раз, а функція прогнозування будується з використанням всієї решти вибірок. Потім прогнозується значення для зразка, який був пропущений. Цей процес повторюється для кожного зразка в наборі даних, доки кожен зразок не буде пропущено та прогнозовано. Цей метод забезпечує ще один спосіб оцінки рівня точності поза вибіркою.

Деякі міркування виникають під час використання методів перехресної перевірки. По-перше, для даних часових рядів випадкова підвибірка генеральної сукупності недостатня; натомість потрібно використовувати суміжні фрагменти часу. Це пояснюється тим, що кожна точка часу може залежати від попередніх точок часу, і ігнорування цієї часової структури шляхом випадкової вибірки може призвести до значної втрати інформації. У k-кратній перехресній перевірці вибір k впливає на зміщення та дисперсію. Більші значення k призводять до меншого зміщення, але більшої дисперсії, тоді як менші значення k призводять до більшого зміщення, але меншої дисперсії. Наприклад, з великим k (наприклад, десятикратна або двадцятикратна перехресна перевірка) можна отримати точнішу оцінку зміщення, але з підвищеною мінливістю через залежність від випадкового вибору підмножини. І навпаки, менші значення k дають менш точні оцінки частоти помилок поза вибіркою, оскільки менше вибірок пропускається під час навчання, що призводить до зменшення дисперсії. Випадкова вибірка під час перехресної перевірки повинна проводитися без заміни, щоб забезпечити цілісність набору даних. В іншому випадку може статися вибірка із заміною, відома як бутстрепінг. Методи Bootstrap мають тенденцію недооцінювати рівень помилок, оскільки повторювані вибірки спотворюють процес оцінки. Одним із методів вирішення цієї проблеми є 0.632 Bootstrap, хоча він передбачає складні налаштування для виправлення недооцінки. При застосуванні перехресної перевірки під час вибору моделі за допомогою таких пакетів, як caret, дуже важливо оцінити помилки незалежного набору даних, щоб отримати справжні значення поза вибіркою. Перехресно-перевірені частоти помилок, хоча і вказують на найкращу модель, можуть не точно відображати реальні частоти помилок поза вибіркою. Таким чином, застосування функції прогнозування до незалежного тестового набору забезпечує більш надійну оцінку.

9. Які дані слід використовувати?

Після визначення питання, що цікавить, наступним важливим кроком є ідентифікація набору даних. Це дозволяє людям створювати оптимальні прогнози за допомогою алгоритмів машинного навчання.



Яскравим прикладом дуже успішного предиктора в Сполучених Штатах є FiveThirtyEight. Зародившись як блог, він мав на меті створити модель прогнозування виборів для прогнозування результатів президентських та інших виборів у країні. Щоб досягти цього, статистик FiveThirtyEight, відомий статистик Нейт Сільвер, використав дані опитувань із різноманітних джерел, усереднюючи їх, щоб отримати найточніший прогноз результатів виборів як на рівні штату, так і на національному рівні. Цей підхід виявився дуже успішним у точному прогнозуванні результатів виборів як 2008, так і 2012 років.

Це є прикладом використання подібних даних для прогнозування результатів. Зокрема, були використані дані опитувань таких організацій, як соціологічні агентства Gallup у Сполучених Штатах. Ці дані включали прямі запити осіб про їхні ймовірні вподобання при голосуванні на майбутніх виборах, віддзеркалюючи запитання, поставлені їм біля виборчої урни. Примітно, що аналітик застосував розумну стратегію для уточнення прогнозів, враховуючи притаманні упередження в даних. Деякі

опитування показали упередженість, яка схиляла респондентів до одного кандидата, а не до іншого, потенційно спотворюючи результати. Щоб пом’якшити це, аналітик зважив опитування на основі їхнього рівня неупередженості чи точності. Цей випадок підкреслює важливість визначення відповідного набору даних і поєднання його з тонким розумінням базових наукових принципів для максимізації ефективності прогнозування.

6.06.2010

Pollster Ratings v4.0: Methodology

by Nate Silver

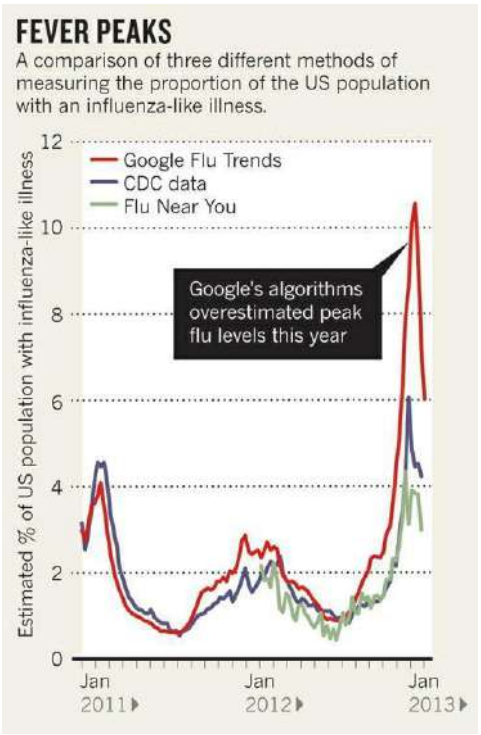
Rating pollsters is at the core of FiveThirtyEight's mission, and forms the backbone of our forecasting models. But, it has been two years since we last revised our ratings. Here, at last, is an update. We have both substantially increased the amount of data that we are evaluating, and significantly refined our methodology.

Головною концепцією тут є використання даних, тісно пов’язаних із цільовою змінною, під час прогнозування. Численні приклади ілюструють цей принцип. Наприклад, Moneyball, який пізніше був екранізований у фільмі з Бредом Піттом, спочатку залучав аналітиків, які намагалися спрогнозувати ефективність гравців у бейсбольних іграх. Вони досягли цього, використовуючи інформацію про минулу продуктивність гравця разом із даними схожих гравців, щоб прогнозувати їхній майбутній успіх у виграшних іграх своєї команди. Цей підхід “подобається прогнозувати подібне” також проявився в конкурсі Netflix Prize, де учасники мали на меті прогнозувати уподобання фільмів. Вони досягли цього, проаналізувавши минулі уподобання людей у фільмах і використовуючи цю інформацію для прогнозування їхніх майбутніх уподобань у фільмах.

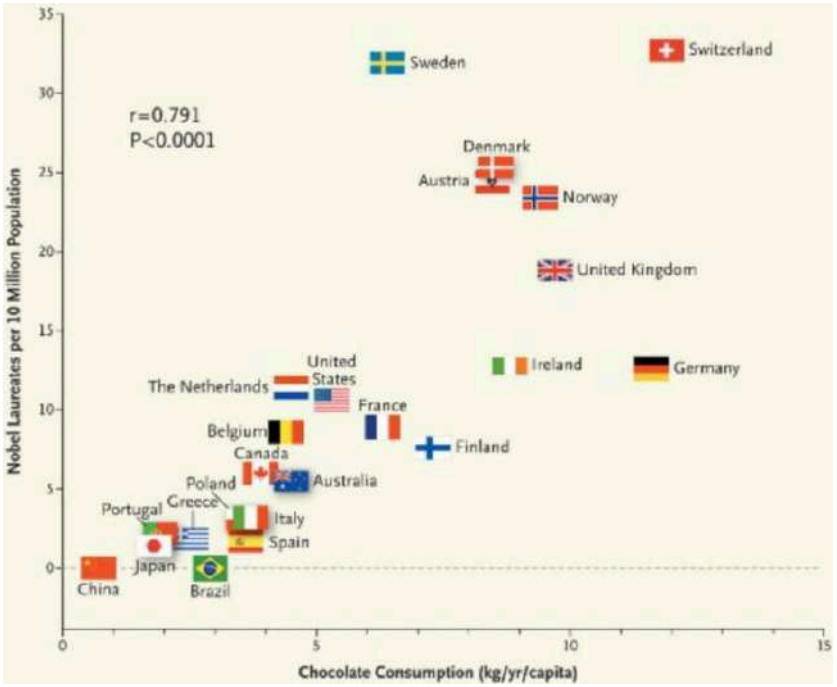
Премія Heritage Health також слугує ілюстрацією, де метою було прогнозувати госпіталізацію, використовуючи дані попередніх госпіталізацій. Знову ж таки, це передбачало використання даних, безпосередньо пов’язаних із процесом, який прогнозується в майбутньому. Таким чином, чим ближче можна підібратися до фактичних даних щодо процесу, який цікавить, тим більша ймовірність отримання точних прогнозів. Хоча це не є абсолютним правилом, є випадки, як-от використання пошуку Google для прогнозування спалахів грипу. Однак цей метод піддається критиці через значні недоліки. Зміни в моделях пошуку людей або зв’язки між пошуковими запитами та спалахами грипу можуть призвести до неточних прогнозів. Отже, деякі

стверджують, що Google Flu Trends не може бути надійним методом оцінки поширеності грипу.

Чим вільніше прогнозування, тим важче воно може бути. Наприклад, Oncotype DX – це алгоритм прогнозування, заснований на експресії генів, який оцінює молекули в організмі, зокрема підгрупу молекул. Він використовує цю інформацію для прогнозування тривалості виживання або результатів лікування хворих на рак молочної залози. Хоча зв'язки можуть бути дещо непрямими, вони все одно помітні.



Властивості даних мають значення. Наприклад, якщо виміряти людей із високою поширеністю грипоподібних симптомів за допомогою даних CDC або Flu Near You, такі алгоритми, як Google Flu Trends, можуть переоцінити кількість випадків грипу, якщо непов'язані фактори спонукають шукати схожі терміни. Розуміння того, як дані співвідносяться з передбачуваним прогнозом, є вирішальним у таких випадках.



Насправді це найпоширеніша помилка в машинному навчанні. Машинне навчання часто концептуалізують як процедуру чорного ящика, створену комп'ютерними вченими, де вхідні дані подаються в один кінець, а прогнози виходять в інший. Приклад дещо абсурдного прогнозування міститься в статті в журналі New England Journal of Medicine. У цьому дослідженні споживання шоколаду в кілограмах на рік на душу населення було відкладено по осі x, а кількість Нобелівських премій на 10 мільйонів населення було відкладено по осі y. Незважаючи на те, що дослідження показало значний зв'язок, який вказують R-квадрат та p-значення, багато інших змінних потенційно можуть пов'язувати ці два фактори. Наприклад, більше споживання шоколаду може бути пов'язане з походженням з Європи, і оскільки більшість націй у наборі даних є європейцями, це може пояснити очевидний зв'язок із Нобелівськими преміями, які в основному присуджуються європейськими організаціями. Ця кореляція не обов'язково означає прямий причинно-наслідковий зв'язок між споживанням шоколаду та Нобелівською премією. Це служить прикладом застереження про те, що використання непов'язаних даних для прогнозування може дати оманливі результати. Це підкреслює важливість використання подібних даних для прогнозування подібних результатів і обережності під час інтерпретації ефективності алгоритмів прогнозування на основі непов'язаних даних.

10. Висновок

У цій лекції ви дізналися про прогнозування, відносну важливість кроків прогнозування, помилки та перехресну перевірку.