Лекція 16. Регуляризована регресія та комбінування предикторів

У цій лекції ви зосередитеся на дослідженні регуляризованої регресії та комбінованих предикторів.

Мета заняття:

- зрозуміти концепцію та застосування регуляризованої регресії в прогнозному моделюванні;
- вивчити методи комбінування предикторів для покращення продуктивності моделі;
- отримати уявлення про методи передбачення та їх практичну реалізацію;
- вивчити підходи до неконтрольованого прогнозування та їх доречність в аналізі даних.

Зміст:

1. Регуляризована регресія

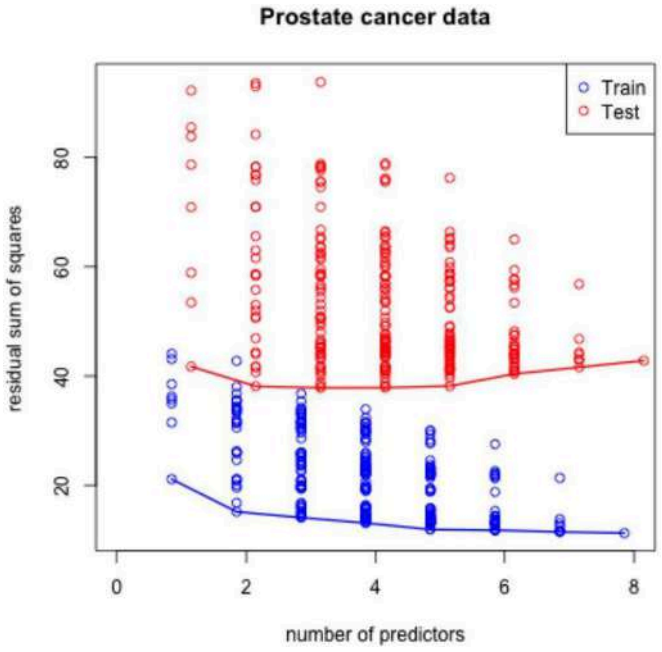
Раніше були введені лінійна регресія та узагальнена лінійна регресія. Основна ідея полягає в тому, щоб підібрати одну з цих моделей регресії, а потім зменшити великі коефіцієнти, пов'язані з певними прогностичними змінними. Цей підхід спрямований на вирішення проблеми зміщення та дисперсії, особливо коли змінні сильно корельовані. Наприклад, включення обох змінних до моделі лінійної регресії може призвести до високої дисперсії. Пропуск однієї може призвести до невеликого зміщення, але може зменшити дисперсію, таким чином покращуючи помилку прогнозування. Методи регуляризації, такі як ласо, можуть допомогти у виборі моделі, але можуть потребувати інтенсивних обчислень для великих наборів даних. Однак у практичних додатках, таких як змагання Kaggle, вони можуть не працювати так добре, як випадкові ліси або методи підсилення.

У мотиваційному прикладі розглянемо підгонку простої регресійної моделі, де результат Y прогнозується за допомогою двох коваріантів, X_1 і X_2, а також члена

перетину. Якщо X_1 і X_2 сильно корельовані, тоді модель може бути спрощена. Замість включення як X_1, так і X_2, ви можете підганяти модель, включивши лише X_1 і помноживши його коефіцієнт на коефіцієнти, призначені як для X_1, так і для X_2. Хоча це наближення може бути неточним через невеликі відмінності між X_1 і X_2, воно може забезпечити дуже точну оцінку, особливо якщо X_1 і X_2 подібні. Незважаючи на те, що цей підхід вносить деяке зміщення через пропуск одного предиктора, він може зменшити дисперсію, особливо коли дві змінні сильно корельовані.

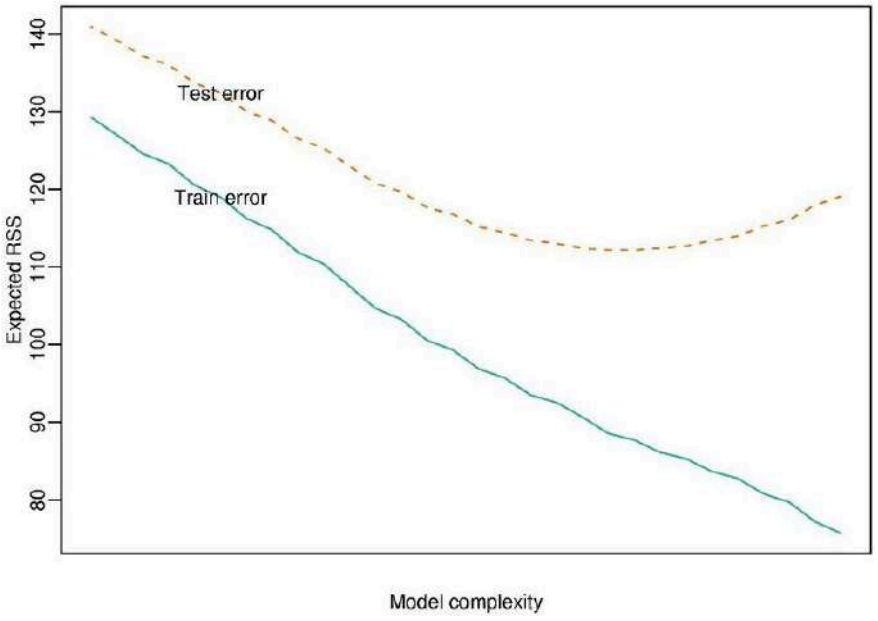
```
library(ElemStatLearn)
data(prostate)
str(prostate)
# 'data.frame': 97 obs. of  10 variables:
#  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
#  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
#  $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
#  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
#  $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
#  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
#  $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
#  $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
#  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
#  $ train  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

Ось приклад використання набору даних про рак передміхурової залози з бібліотеки ElemStatLearn. Можна завантажити дані prostate та переглянути набір даних, який містить 97 спостережень за десятьма змінними.



Можливо, ви захочете зробити прогноз щодо раку передміхурової залози на основі великої кількості предикторів у наборі даних. Цей сценарій типовий для

побудови моделей на практиці. Припустимо, що передбачення зроблено з усіма можливими комбінаціями змінних предиктора з використанням моделі лінійної регресії. Для кожного результату будується регресійна модель для кожної можливої комбінації векторів. Можна помітити, що зі збільшенням кількості предикторів зліва направо помилка навчального набору завжди зменшується. Однак із реальними даними виникає типова модель, коли помилка тестового набору спочатку зменшується зі збільшенням кількості предикторів, що вказує на покращення. Згодом помилка тестового набору досягає плато і знову починає зростати. Це відбувається тому, що модель перепідганяє навчальні дані, що свідчить про те, що включення занадто великої кількості предикторів може бути не ідеальним.



Це неймовірно поширена закономірність, по суті, будь-яка міра складності моделі на осі x проти очікуваної суми залишків квадратів прогнозованої помилки. Можна спостерігати, що в навчальному наборі помилка майже завжди монотонно зменшується. Іншими словами, у міру побудови все більш складних моделей помилка навчання завжди буде зменшуватися. Однак у наборі для тестування помилка на деякий час зменшиться, зрештою досягнувши мінімуму. Потім вона знову починає збільшуватися, оскільки модель стає надто складною та перепідганяє дані.

Загалом, найкращий підхід може бути, коли є достатньо даних і часу на обчислення для розділення вибірок. Ідея полягає в тому, щоб розділити дані на набори для навчання, тестування та перевірки. Набір перевірки розглядається як тестові дані, і кожна можлива конкуруюча модель навчається на всіх можливих підмножинах навчальних даних. Вибирається найкраща модель із набору даних перевірки. Однак,

оскільки перевірочний набір використовувався для навчання, рівень помилок потрібно оцінювати на повністю незалежному наборі. Це робиться шляхом застосування прогнозування до нових даних у тестовому наборі. Іноді поділ і аналіз повторюють кілька разів, щоб отримати кращу середню оцінку частоти помилок поза вибіркою. Однак у цьому підході є дві загальні проблеми. По-перше, дані можуть бути обмеженими, тому отримати хорошу підгонку моделі буде складно, якщо дані розділені на три набори. По-друге, виникає обчислювальна складність, особливо при переборі всіх можливих підмножин моделей, особливо з багатьма змінними предикторів.

Інший підхід полягає в декомпозиції помилки прогнозування та дослідженні альтернативних методів для включення лише змінних, необхідних для моделі. Якщо припустити, що змінну Y можна передбачити як функцію X плюс деякий член помилки, очікувана помилка прогнозування являє собою очікувану різницю між результатом і прогнозом результату в квадраті. $\widehat{f}_{\lambda}$ це оцінка з навчального набору з використанням певного набору параметрів налаштування λ. При розгляді нової точки даних відстань між спостережуваним результатом і прогнозом на новій точці даних можна розкласти на незнижувану помилку (σ^2), зміщення (різниця між очікуваним прогнозом і істинністю) і дисперсію оцінки. Основна мета під час побудови моделі прогнозування – зменшити цю загальну величину, яка, по суті, представляє очікувану середньоквадратичну помилку між результатом і прогнозом. Хоча незнижувану помилку зазвичай неможливо зменшити, оскільки вона притаманна даним, існує компроміс між зміщенням і дисперсією, який розглядається за допомогою концепції регуляризованої регресії.

```
small = prostate[1:5,]
lm(lpsa ~ ., data=small)
#
# Call:
# lm(formula = lpsa ~ ., data = small)
#
# Coefficients:
# (Intercept)       lcavol      lweight          age         lbph          svi
#     9.60615      0.13901     -0.79142      0.09516           NA           NA
#         lcp      gleason        pgg45    trainTRUE
#          NA     -2.08710           NA           NA
```

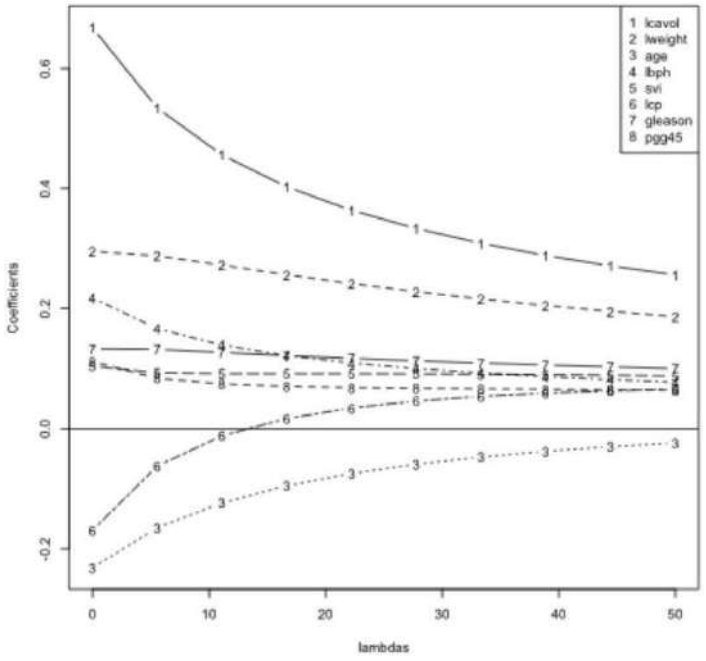
Інша проблема з високовимірними даними виникає, коли більше предикторів, ніж вибірок. Наприклад, розглянемо сценарій, у якому використовується лише невелика підмножина даних prostate, що призводить до лише п’яти спостережень у навчальному наборі при наявності більше п’яти предикторних змінних. Під час підгонки лінійної

моделі, яка пов’язує результат із усіма цими предикторами, деякі з них можуть отримати оцінки, тоді як інші будуть позначені як NA. Це відбувається тому, що існує більше предикторів, ніж зразків, що призводить до результатів, які неможливо інтерпретувати.

Одним із підходів до вирішення цієї проблеми є розгляд моделі з лінійною формою, як обговорювалося раніше. Потім обмежте, щоб тільки λ коефіцієнтів були ненульовими. Після вибору λ, якщо є, наприклад, лише три ненульові коефіцієнти, потрібно дослідити всі можливі комбінації трьох ненульових коефіцієнтів і підібрати найкращу модель. Однак цей підхід залишається вимогливим до обчислень.

Інший підхід передбачає використання регуляризованої регресії. Якщо коефіцієнти β_j у лінійній моделі не мають обмежень, вони можуть стати нестабільними, особливо якщо для прогнозування використовуються висококорельовані змінні, що призводить до високої дисперсії та менш точних прогнозів. Щоб контролювати цю дисперсію, коефіцієнти можна регуляризувати або скорочувати. Мета полягає в тому, щоб мінімізувати відстань між результатом і лінійною моделлю в квадраті, відомою як залишкова сума квадратів. Додавання “штрафного” члену допомагає зменшити занадто великі коефіцієнти, зменшуючи складність і дисперсію. При належному налаштуванні “штраф” також може зберегти структуру проблеми.

Перший підхід, який використовується в цьому типі штрафної регресії, полягає в підгонці моделі регресії. Тут мета полягає в тому, щоб зменшити відстань між результатом і регресійною моделлю. Тут також існує член, що включає λ, помножену на суму квадратів β_j. По суті, якщо квадрати β_j великі, цей член збільшиться, що призведе до поганої підгонки. Отже, деякі β_j мають бути малими. Математично це схоже на мінімізацію суми квадратів різниць із обмеженням, що сума квадратів β_j менша за s. Включення коефіцієнта λ може зробити проблему несингулярною, навіть якщо матриця X^TX не є інвертованою, наприклад, у випадках, коли існує більше предикторів, ніж спостережень.



Ось як виглядає шлях коефіцієнта. Шлях коефіцієнта означає, що для кожного іншого вибору λ в задачі штрафної регресії, коли λ збільшується, великі коефіцієнти штрафуються все більше і більше. Спочатку, коли λ дорівнює 0, β приймають певні значення, які є стандартними значеннями лінійної регресії. Однак із збільшенням λ всі коефіцієнти поступово наближаються до 0, оскільки вони штрафуються та зменшуються.

Параметр налаштування λ визначає величину контрольного коефіцієнта. Він регулює кількість застосованої регуляризації. Коли λ наближається до 0, рішення повертається до рішення за методом найменших квадратів, типового для стандартної лінійної моделі. І навпаки, коли λ наближається до нескінченності або стає дуже великою, це сильно штрафує коефіцієнти, змушуючи їх збігатися до 0. Вибір оптимального параметра налаштування може бути досягнутий за допомогою перехресної перевірки або інших методів, спрямованих на досягнення балансу між зміщенням і дисперсією.

Подібний підхід можна застосувати з невеликою зміною штрафу. Тут мета залишається мінімізувати проблему найменших квадратів, шукаючи β значення, які мінімізують відстань до результату. Однак тепер це обмежено сумою абсолютних значень β_j, яка є меншою за певне значення. Це також можна виразити як штрафну регресію цієї форми, спрямовану на мінімізацію штрафної суми квадратів. Для ортонормованої матриці існує рішення закритої форми. Це рішення передбачає взяття

абсолютного значення β_j і віднімання γ значення, зберігаючи лише позитивну частину. Якщо γ перевищує значення найменших квадратів $\widehat{\beta}_j$, результат встановлюється рівним 0. І навпаки, якщо абсолютне $\widehat{\beta}_j$ більший за значення γ, коефіцієнт зменшується на величину γ, зберігаючи знак вихідного коефіцієнта. Цей підхід, відомий як ласо, не лише зменшує коефіцієнти, але й виконує вибір моделі, встановлюючи деякі коефіцієнти рівними 0. Деякі вважають за краще цю подвійну функціональність, оскільки вона досягає одночасного зменшення коефіцієнтів і вибору моделі.

2. Комбінування предикторів

Комбінування предикторів іноді називають методами ансамблю в навчанні. Ключова ідея полягає в тому, щоб об'єднати класифікатори шляхом усереднення або голосування, навіть якщо вони принципово різні. Наприклад, посилюючий класифікатор можна поєднати з випадковим лісом і моделлю лінійної регресії. Як правило, об'єднання класифікаторів підвищує точність, але може зменшити інтерпретацію. Слід подбати про те, щоб отримана вигода переважала втрату в інтерпретації. Підсилення, беггінг та випадковий ліс є варіаціями цієї концепції, де різні класифікатори усереднюються разом. Однак ці методи передбачають усереднення одних і тих самих класифікаторів у кожному випадку.

Базову інтуїцію, що стоїть за цією концепцією, можна порівняти з голосуванням більшості. Розглянемо п'ять абсолютно незалежних класифікаторів, кожен з яких має 70% точність. У сценарії більшості голосів точність обчислюватиметься на основі кількості правильних класифікаторів. Це можуть бути три з п'яти, чотири з п'яти або всі п'ять класифікаторів. Виходить, що точність більшості голосів склала б 83,7%. Зі 101 незалежним класифікатором точність більшості голосів досягла б 99,9%.

Деякі підходи до об'єднання класифікаторів передбачають використання подібних класифікаторів і поєднання їх із такими методами, як беггінг, підсилення або випадкові ліси. Інший підхід полягає в комбінуванні різних класифікаторів за допомогою таких методів, як стекування моделей або ансамблювання.

```
library(ISLR)
data(Wage)
library(ggplot2)
library(caret)
Wage <- subset(Wage, select=-c(logwage))
# Create a building data set and validation set
```

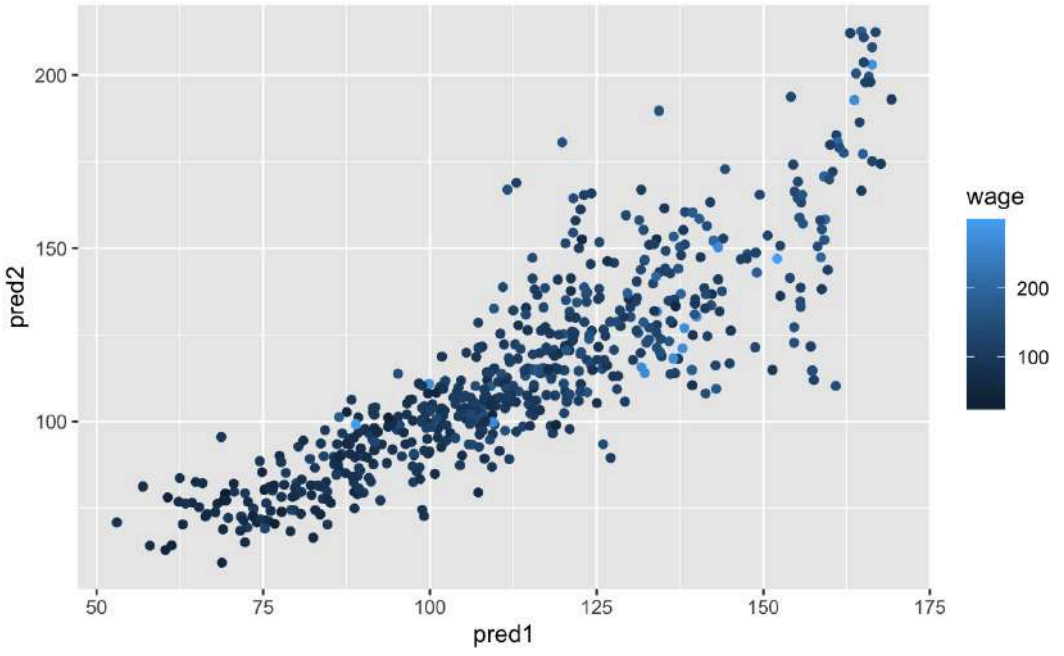
```
inBuild <- createDataPartition(y=Wage$wage, p=0.7, list=FALSE)
validation <- Wage[-inBuild,]
buildData <- Wage[inBuild,]
inTrain <- createDataPartition(y=buildData$wage, p=0.7, list=FALSE)
training <- buildData[inTrain,]
testing <- buildData[-inTrain,]
dim(training)
# [1] 1474   10
dim(testing)
# [1] 628  10
dim(validation)
# [1] 898  10
```

Ось приклад із використанням даних wage. Набір даних wage створюється без змінної logwage, оскільки мета полягає в тому, щоб передбачити wage, а logwage буде сильним предиктором. Створюються набір перевірки, навчальний набір і тестовий набір. Навчальний набір містить найбільшу частину з 1474 прикладами, потім тестовий набір із 628 прикладами та набір перевірки з 898 зразками.

```
mod1 <- train(wage ~ ., method="glm", data=training)
mod2 <- train(wage ~ ., method="rf", data=training,
trControl=trainControl(method="cv"), number=3)
```

Лінійна модель із методом glm використовується для прогнозування wage з використанням усіх інших змінних. Крім того, окрема модель будується за допомогою методу випадкового лісу. Таким чином, дві різні моделі прогнозування підганяються до одного набору даних.

```
pred1 <- predict(mod1, testing)
pred2 <- predict(mod2, testing)
qplot(pred1, pred2, colour=wage, data=testing)
```



Прогнози з лінійної моделі (pred1) і випадкового лісу (pred2) можна порівняти один з одним. Хоча вони близькі один до одного, вони не зовсім сходяться. Крім того, жоден з них не корелює повністю зі змінною wage, яка представлена кольором точок на цьому конкретному зображенні.

```
predDF <- data.frame(pred1, pred2, wage=testing$wage)
combModFit <- train(wage ~ ., method="gam", data=predDF)
combPred <- predict(combModFit, predDF)
```

Потім можна підігнати модель, яка поєднує предиктори. Створюється новий набір даних, що містить прогнози з першої та другої моделі. Після цього на основі змінної wage тестового набору даних створюється змінна wage. Потім встановлюється нова модель, яка пов’язує цю змінну wage з двома прогнозами. Замість того, щоб просто підганяти модель, яка пов’язує коваріанти з результатом, підганяються дві окремі моделі прогнозування результату. Нарешті, прогнозування можна зробити з комбінованого набору даних на нових зразках.

```
sqrt(sum((pred1 - testing$wage)^2))
# [1] 837.1452
sqrt(sum((pred2 - testing$wage)^2))
# [1] 893.8933
sqrt(sum((combPred - testing$wage)^2))
# [1] 823.4228
```

Продуктивність на тестовому наборі можна перевірити. Наприклад, перший предиктор має помилку 837,1, другий предиктор має помилку 893,8, тоді як комбінований предиктор має нижчу помилку 823,4.

```
pred1V <- predict(mod1, validation)
pred2V <- predict(mod2, validation)
predVDF <- data.frame(pred1=pred1V, pred2=pred2V)
combPredV <- predict(combModFit, predVDF)
sqrt(sum((pred1V - validation$wage)^2))
# [1] 1078.218
sqrt(sum((pred2V - validation$wage)^2))
# [1] 1104.409
sqrt(sum((combPredV - validation$wage)^2))
# [1] 1085.943
```

Набір перевірки також розглядається для оцінки, оскільки тестовий набір використовувався для змішування двох моделей. Таким чином, модель підганяється на набір перевірки, щоб уникнути упередженого оцінювання помилки поза вибіркою. Прогнози генеруються як з першої, так і з другої моделей у наборі перевірки, і з цими прогнозами створюється фрейм даних. Згодом комбінована модель використовується для прогнозування результатів на основі передбачень двох моделей у наборі даних перевірки. Тут ви можете помітити, що помилка для першої моделі, яка

використовується окремо, становитиме 1078,2, тоді як для другої моделі вона буде 1104,4. Цікаво, що комбінована модель демонструє не меншу помилку в наборі даних перевірки. Але це повинно показати те, що поєднання моделей у такий спосіб може підвищити точність шляхом об'єднання сильних сторін різних моделей.

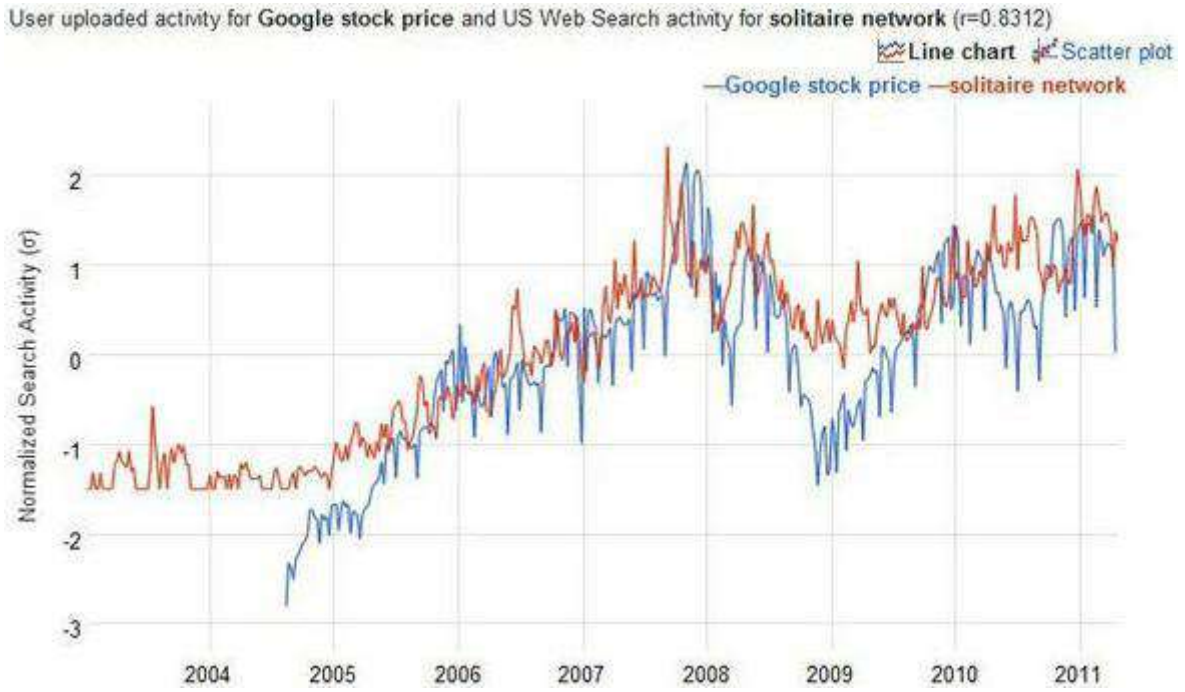
Навіть просте комбінування, як описано тут, може виявитися дуже корисним і підвищити точність. У випадку бінарних або мультикласових даних типовий підхід передбачає побудову непарної кількості моделей, прогнозування результатів для кожної моделі, а потім визначення остаточної мітки класу на основі більшості голосів. Однак варто зазначити, що існують більш складні ансамблеві техніки. Наприклад, доступний ансамблевий пакет caret, але він все ще перебуває в стадії розробки, тому під час його використання рекомендується бути обережним.

3. Передбачення

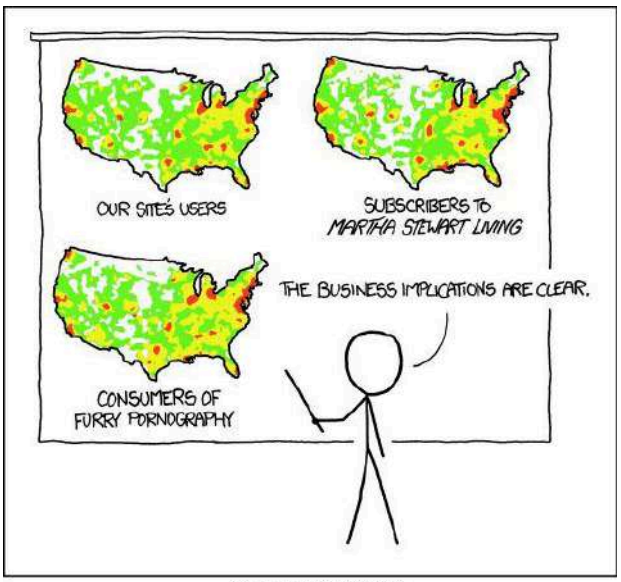
Передбачення – це спеціалізована форма прогнозування, яка часто використовується в сценаріях із використанням даних часових рядів. Наприклад, розглянемо інформацію про акції Google на біржі NASDAQ. З часом ціна акцій коливається, вводячи певні залежності та додаткові проблеми, які потрібно враховувати під час прогнозування.

Дані демонструють залежність у часі, що за своєю суттю ускладнює прогнозування порівняно зі сценаріями з незалежними прикладами. Варто звернути увагу на різні типи закономірностей, у тому числі тенденції (довготермінове підвищення або зниження), сезонні закономірності (що відбуваються протягом тижнів, місяців або років) і цикли (періодичні підйоми та спади протягом періодів, довших за

рік). Підвибірка для навчання та тестування стає складнішою, оскільки випадковий розподіл неможливий; замість цього необхідно враховувати конкретні часові інтервали вибірки та тимчасові залежності. Подібні проблеми виникають у прогнозуванні просторових даних, де спостереження поблизу демонструють залежність і може знадобитися врахувати впливи, пов'язані з місцем. Незважаючи на те, що стандартні алгоритми прогнозування все ще можуть бути застосовані, необхідна обережність при їх використанні.

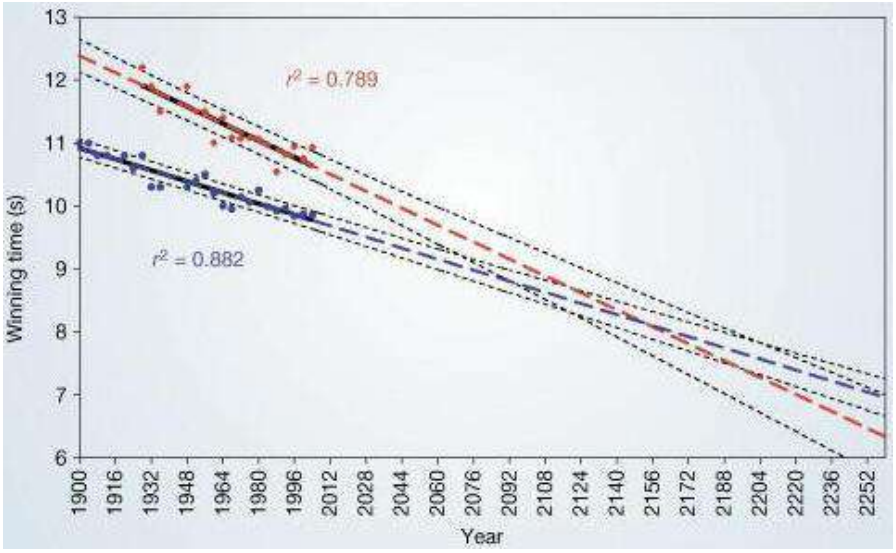


Слід проявляти обережність щодо помилкових кореляцій у даних часових рядів. Кореляції між змінними можуть існувати з причин, не пов'язаних з точністю прогнозування. Наприклад, використання таких інструментів, як Google Correlate, для аналізу частоти різних слів з часом може виявити, здавалося б, високі кореляції між непов'язаними явищами. Наприклад, може існувати кореляція між курсом акцій Google (синій) і мережевою активністю пасьянсу (червоний), незважаючи на відсутність будь-якого значущого зв'язку між ними. Покладаючись на такі кореляції для прогнозування без урахування їх причинно-наслідкового зв'язку, що лежить в їх основі, може призвести до помилкових висновків, оскільки змінні можуть суттєво відрізнятися в майбутньому через відсутність справжнього зв'язку.



PET PEEVE #208:
GEOGRAPHIC PROFILE MAPS WHICH ARE BASICALLY JUST POPULATION MAPS

У географічному аналізі переважають шаблони, що нагадують теплові карти. Мем ілюструє це явище, підкреслюючи, як теплові карти населення часто мають схожі форми через щільність населення. Наприклад, люди, які користуються певним веб-сайтом, передплачують певний журнал або споживають вміст із певного типу веб-сайту, можуть групуватися в схожих географічних місцях. Це групування відбувається тому, що найвища щільність населення в Сполучених Штатах зосереджена вздовж східного узбережжя, що призводить до схожих моделей теплових карт для різних демографічних груп та інтересів.



Ви також повинні бути обережними з екстраполяцією. Цікавий приклад ілюструє наслідки екстраполяції даних часових рядів без урахування потенційних результатів. На графіку зображено виграшний час гонок на Олімпіаді за тривалий

період. Виграшний час чоловіків представлено синім кольором, тоді як виграшний час жінок – червоним. Автори дослідження розширили дані на майбутнє та прогнозували, що до 2156 року жінки будуть бігати швидше за чоловіків у спринті. Хоча точний час такої події залишається невизначеним, ця екстраполяція підкреслює небезпеку робити прогнози занадто далеко в майбутньому. Зрештою, згідно з цією екстраполяцією, прогнозується, що і чоловіки, і жінки досягнуть негативного часу для бігу на 100 метрів. Тому дуже важливо бути обережним при екстраполяції даних за межі спостережуваного діапазону.

```
library(quantmod)
from.dat <- as.Date("01/01/08", format="&m/%d/%y")
to.dat <- as.Date("12/31/13", format="%m/%d/%y")
getSymbols("GOOG", src="google", from=from.dat, to=to.dat)
# [1] "GOOG"
head(GOOG)
#            GOOG.Open GOOG.High GOOG.Low GOOG.Close GOOG.Volume
# 2008-01-02     692.9     697.4    677.7      685.2     4306848
# 2008-01-03     685.3     686.9    676.5      685.3     3252846
# 2008-01-04     679.7     681.0    655.0      657.0     5359834
# 2008-01-07     653.9     662.3    637.4      649.2     6404945
# 2008-01-08     653.0     660.0    631.0      631.7     5341949
# 2008-01-09     630.0     653.3    622.5      653.2     6744242
```

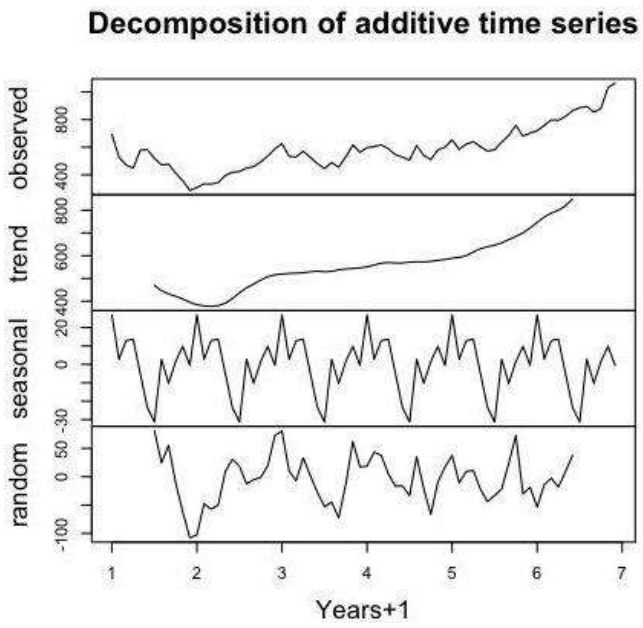
Ось короткий приклад прогнозування за допомогою пакета quantmod і даних Google. Завантаживши пакет quantmod та імпортувавши дані з біржі Google через набір фінансових даних Google, ви можете отримати доступ до різної інформації за певний період, у цьому випадку з 1 січня 2008 року до 31 грудня 2013 року.

```
mGoog <- to.monthly(GOOG)
goog0pen <- Op(mGoog)
ts1 <- ts(googOpen, frequency=12)
plot(ts1, xlab="Years+l", ylab="GOOG")
```

Процес передбачає підсумовування даних щомісяця та збереження їх у часових рядах. Це досягається за допомогою функції to.monthly для перетворення даних у місячний часовий ряд. Зокрема, інформація про відкриття витягується та використовується для створення об’єкта часового ряду за допомогою функції ts у R. Побудова цього часового ряду показує місячні ціни відкриття для Google протягом семирічного періоду.

Приклад декомпозиції часових рядів передбачає розбиття часових рядів на окремі компоненти: тренд, що представляє будь-яку послідовну закономірність у часі; сезонний характер, що вказує на періодичні коливання; і циклічні моделі, які передбачають підйоми та падіння протягом нефіксованих періодів.

```
plot(decompose(ts1), xlab="Years+l")
```



Одним із підходів до досягнення цього є використання функції decompose в R. При адитивній декомпозиції вона виявляє кілька компонентів: тенденцію до зростання курсу акцій Google, сезонну модель і випадкову циклічну модель у наборі даних.

```
ts1Train <- window(ts1, start=1, end=5)
ts1Test <- window(ts1, start=5, end=(7-0.01))
ts1Train
#     Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
# 1 692.9 528.7 471.5 447.7 578.3 582.5 519.6 472.5 476.8 412.1 357.6 286.7
# 2 308.6 334.3 333.3 343.8 395.0 418.7 424.2 448.7 459.7 493.0 537.1 588.1
# 3 627.0 534.6 529.2 571.4 526.5 480.4 445.3 489.0 455.0 530.0 615.7 563.0
# 4 596.5 604.5 617.8 588.8 545.7 528.0 506.7 611.2 540.8 509.9 580.1 600.0
# 5 652.9
```

Для навчальних і тестових наборів необхідно створити набори з послідовними точками часу. Наприклад, навчальний набір може розпочатися в момент часу 1 і закінчитися в момент часу 5, після чого слідує тестовий набір, що складається з наступних послідовних моментів. Це гарантує, що навчальні та тестові набори демонструють подібні тенденції, які спостерігаються в даних.

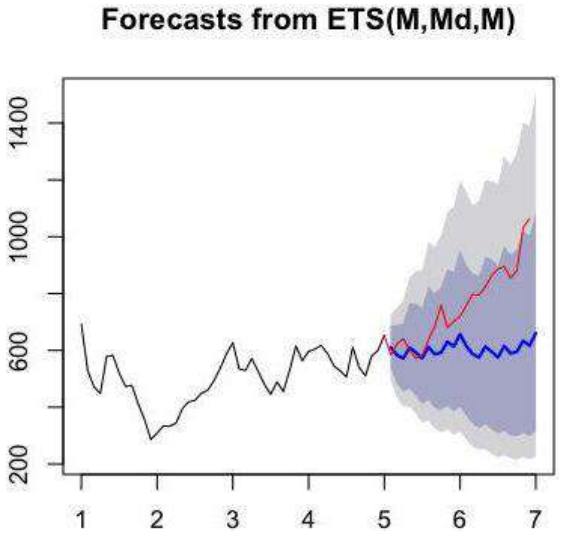
```
plot(ts1Train)
lines(ma(ts1Train, order=3), col="red")
```

Існує кілька способів прогнозування. Одним із методів є використання простого ковзного середнього, яке, по суті, обчислює середнє значення всіх значень до певного моменту часу для прогнозування.

		Seasonal Component	
Trend Component	N (None)	A (Additive)	M (Multiplicative)
N (None)	(N,N)	(N,A)	(N,M)
A (Additive)	(A,N)	(A,A)	(A,M)
A_d (Additive damped)	(A_d,N)	(A_d,A)	(A_d,M)
M (Multiplicative)	(M,N)	(M,A)	(M,M)
M_d (Multiplicative damped)	(M_d,N)	(M_d,A)	(M_d,M)

Іншим варіантом є експоненціальне згладжування. Цей метод призначає вищу вагу найближчим точкам часу, роблячи більший акцент на нещодавніх даних порівняно з даними, які віддалені в минулому. Для вибору доступні різні класи моделей згладжування.

```
ets1 <- ets(ts1Train, model="MMM")
fast <- forecast(ets1)
plot(fcast)
lines(ts1Test, col="red")
```



При експоненціальному згладжуванні можна підігнати модель із різними параметрами для різних типів трендів. Під час передбачення прогнозування та межі прогнозування можуть бути отримані з моделі передбачення, щоб вказати потенційний діапазон значень.

```
accuracy(fcast,ts1Test)
#                    ME   RMSE    MAE      MPE   MAPE   MASE    ACF1 Theil's U
# Training set   0.9464  48.78  39.35 -0.3297  7.932 0.3733 0.07298        NA
# Test set     156.1890 205.76 160.78 18.1819 18.971 1.5254 0.77025     3.745
```

Точність прогнозу можна визначити за допомогою функції accuracy. Вона надає такі показники, як середньоквадратична помилка та інші, які підходять для прогнозування на основі тестового набору.

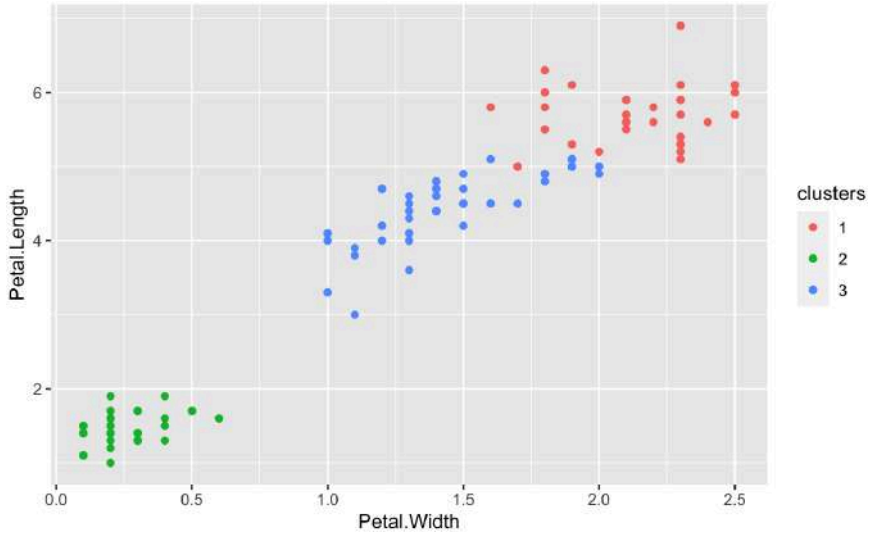
4. Неконтрольоване прогнозування

Досі розглянуті приклади часто включають ситуації, коли ви часто знаєте, якими є мітки. Іншими словами, ви, як правило, виконуєте контрольовані задачі класифікації, намагаючись прогнозувати вже відомий результат. Однак є випадки, коли мітки для прогнозування невідомі, що вимагає їх дослідження заздалегідь. Одним із підходів до вирішення цієї проблеми є створення кластерів із спостережуваних даних, призначення міток цим кластерам і подальше створення для них предиктора. Зіткнувшись з новими даними, ви повинні передбачити кластер і застосувати раніше призначену мітку. Це вводить кілька рівнів складності в проблему прогнозування. По-перше, створення кластерів не є абсолютно безшумним процесом. По-друге, точне визначення відповідних міток або ефективна інтерпретація кластерів становить значну проблему. Розробка предиктора для кластерів передусім передбачає використання алгоритмів, вивчених протягом курсу, а також створення прогнозів на основі цих кластерів.

```
data(iris)
library(ggplot2)
inTrain <- createDataPartition(y=iris$Species, p=0.7, list=FALSE)
training <- iris[inTrain,]
testing <- iris[-inTrain,]
dim(training)
# [1] 105   5
dim(testing)
# [1] 45  5
```

Швидкий приклад може проілюструвати, як працює цей процес. Завантаживши дані iris та бібліотеку ggplot2, ви можете продовжити створення навчального та тестового набору для даних iris, подібно до попередніх прикладів, не враховуючи кластери Species.

```
kMeans1 <- kmeans(subset(training, select=-c(Species)), centers=3)
training$clusters <- as.factor(kMeans1$cluster)
qplot(Petal.Width, Petal.Length, colour=clusters, data=training)
```



Один підхід може включати виконання кластеризації k-середніх. Основна ідея полягає в тому, щоб створити три окремі кластери без урахування інформації про види. Після цього кожному кластеру можна призначити інший колір для візуалізації. Отриманий графік відображає три окремі кластери, створені за допомогою процесу кластеризації k-середніх. Хоча в цьому випадку ці кластери можуть бути тісно пов'язані з мітками видів, таке узгодження не завжди є типовим, оскільки маркування кластерів часто може бути складним.

```
table(kMeans1$cluster, training$Species)
#     setosa versicolor virginica
#   1      0          1        27
#   2     35          0         0
#   3      0         34         8
```

У цьому випадку можна створити таблицю для порівняння кластерів із видами. Стає очевидним, що кластер 2 відповідає одному виду, кластер 3 – іншому виду, а кластер 1 – щее іншому виду. Однак у типових сценаріях назви конкретних видів не будуть відомі, що вимагає присвоєння імен кожному кластеру на основі їхніх характеристик.

```
modFit <- train(clusters ~ ., data=subset(training, select=-c(Species)),
method="rpart")
table(predict(modFit, training), training$Species)
#     setosa versicolor virginica
#   1      0          0        26
#   2     35          0         0
#   3      0         35         9
```

Потім можна налаштувати модель для зв'язування новоствореної змінної кластера з усіма змінними предикторів, як правило, у навчальному наборі. У цьому випадку для цієї мети використовується дерево класифікації. Згодом у навчальному наборі можна зробити прогнози, які виявлять, що хоча кластер 2 і кластер 3 точно

передбачені, у прогнозах моделі може виникнути плутанина між кластером 1 і кластером 3. Ця плутанина виникає як через притаманну варіабельність побудови прогнозів, так і через сам процес кластеризації. Таким чином, виконання неконтрольованого прогнозування таким чином виявляється складною задачею.

```
testClusterPred <- predict(modFit, testing)
table(testClusterPred, testing$Species)
# testClusterPred setosa versicolor virginica
#               1      0          0         8
#               2     15          0         0
#               3      0         15         7
```

Потім модель можна застосувати до тестового набору даних. На практиці справжні мітки можуть бути невідомі, але для ілюстрації тут показано відомі мітки видів. Прогнози робляться на основі нового набору даних і створюється таблиця, у якій порівнюються прогнозовані мітки кластерів із фактично відомими видами. Результати вказують на те, що модель працює досить добре у прогнозуванні різних видів на основі кластеризованих міток.

Загалом, це складна проблема, яка вимагає обережності при позначенні кластерів і проведенні неконтрольованого аналізу під час розуміння його операцій. Функція cl_predict у пакеті clue пропонує функціональність, подібну до описаного методу. Однак часто доцільно розробити індивідуальний підхід для неконтрольованого прогнозування, оскільки потрібен уважний розгляд при визначенні кластерів. До надмірної інтерпретації цих кластерів слід підходити з обережністю, оскільки цей метод є перш за все дослідницьким, і кластери можуть відрізнятися залежно від методів вибірки даних.

5. Висновок

У цій лекції ви дізналися про регуляризовану регресію та комбінування предикторів.