Datorlaboration 3

Statistik och dataanalys 2 (ST1201), VT2023

Introduktion

Alla datorlaborationer ska genomföras som *Quarto-notebooks*. En stor fördel med notebook-formatet är att det låter er skapa ert egna kursmaterial genom att kombinera kod med text som beskriver vad koden gör. Att skriva sitt egna kursmaterial, alltså att med egna ord tvingas förklara hur saker fungerar, är ett av dom bästa sätten att lära sig.

Dom första tre laborationerna är utformade så att dom matchar mot dom tre delarna i inlämningsuppgiften. När du är klar med dagens laboration är du alltså redo att börja med del 3 på inlämningsuppgiften.

- Det är helt OK att ni sammarbetar under labben, men skriv din egna labbrapport! Det är viktigt att faktiskt skriva koden själv.
- Om du fastnar, testa att se om du kan hitta en lösning i dokumentationen. För att se dokumentationen för en viss funktion skriver du helt enkelt ett frågetecken följt av funktionens namn i konsolen, exempelvis `?lm`. Funkar inte det, testa google, och funkar inte det, fråga labbansvarig. Vi finns där för att svar på dina frågor, med det är viktigt att du tränar på att lösa problem själv.

Innehåll

I den här laboration kommer du lära dig att

- skatta en logistisk regressionsmodell,
- göra klassifikation med en logistisk regressionsmodell,
- utvärdera modeller baserat på hur bra dom är på klassifikation, och
- utvärdera modeller baserat på deras logaritmerade prediktionsvärde.

Innan du går vidare, skapa ett tomt quarto-dokument på samma sätt som du gjorde under första labben. Alltså, skapa ett nytt quarto-dokument, radera allt utom preamble, ändra `title` till något passande, och lägg sedan till en kodchunk (med rätt chunkinställningar) som du kan ha alla dina `library()`-anrop i och placera den precis efter preamble.

Del 1 - Skatta en logistisk regressionsmodell

1a - Testdata och träningsdata

Ditt jobb den här laborationen är att ta fram en så bra modell som möjligt för att kunna klassificera (predicera) om en person överlevde Titanic eller inte. Eftersom att det är svårt att få fram ny data så behöver du dela upp datasetet i ett *träningsset* (kalla det `titanic_train`) som du ska använda till att skatta dina modeller, och ett *testset* (kalla det `titanic_test`) som du ska använda till att utvärdera dina modeller. Under labben kommer du jobba med datasetet `titanic` som finns i paketet `sda123`.

- ☐ Separera datasetet `titanic` i två: ett testset på 89 observationer och ett träningsset på 798 observationer. Börja med att slumpa fram vilka 89 observationer som ska utgöra ditt testset. Detta kan du göra med `sample()`-funktionen som beskrivs nedan. Gör sedan ett dataset som endast innehåller observationerna som är i testsetet, och ett dataset som endast innehåller observationerna som *inte* är i testsetet. För att välja ut specifika observationer ur ett dataset kan du sen antingen använda base R syntax eller `tidyverse`-funktionen `slice()`!

```
test_data <- sample(
  1:nrow(titanic),
  size = 89,
  replace = FALSE
)
```

- Första argumentet anger vad du vill sampla *från*. I detta fall är ditt mål att sampla ett antal observationer från datasetet `titanic`. Anledningen till att vi då anger `1:nrow(titanic)` är att vi vill får fram ett urval av *radindex*, alltså vilka rader i datasetet som ska utgöra vårat test- respektive träningsset. Självklart vill vi att dessa värden ska ligga mellan 1 och antal rader i data.
- Argumentet `size` anger hur stort stickprov du samplar.
- `replace = FALSE` anger att du samplar *utan återläggning*. Om du tänker efter lite kan du nog komma fram till varför vi gör på det sättet!

1b - Logistisk regression

Du ska nu skatta två logistiska regressionsmodeller med hjälp av funktionen `glm()`. För att använda `glm()` behöver du ange följande *argument*:

- En *formel* som anger vilken modell det är du vill skatta. Detta funkar precis som det gör med `lm()`. Du skriver helt enkelt utfallsvariabeln följt av ett tilde och sen dom

förklarande variabler du vill inkludera, separerade med `+`. Precis som för linjär regression kan du inkludera interaktioner mellan förklarande variabler med `:`.
- Det andra argumentet är namnet på datasetet du vill använda. I den här uppgiften kommer det vara `titanic_train`.
- Det tredje argumentet är `family = "binomial"`. `glm()` står för *generalized linear model* och kan användas för att skatta en massa olika funktioner, inte bara logistisk regression, så vi måste säga till R vilken typ av generaliserad linjär modell vi är ute efter. `"binomial"` ger oss logistisk regression.

```
mod_mini <- glm(
  ...,
  data = ,
  family = "binomial"
)
```

Du ska först skatta en modell med ett fåtal förklarande variabler, och sen en mer komplicerad modell.

- ☐ Skatta en logistisk regressionsmodell med `sex`, `age`, och `firstclass` som förklarande variabel. Spara modellen som ett objekt med namnet `mod_mini`. Du kommer att komma tillbaks till modellen för att utvärdera hur bra den är på att klassificera senare.
- ☐ Skatta en logistisk regression som innehåller `sex`, `age`, `fare`, `firstclass`, samt alla interaktioner mellan dessa (det ska totalt bli sex interaktioner!) som förklarande variabler. Spara modellen som ett objekt med namnet `mod_maxi`.
- ☐ Använd funktionen `logisticreg_summary()` från paketet `sda123` för att undersöka parameterestimaten för dom två modellerna. Här skulle det vara läge att tolka dom olika skattade parametrarna, men det får du göra efter labben, för nu har du mer kodande att göra och *klockan tickar*.
- ☐ Ta fram log-likelihooden för dom två modellerna, och genomför ett likelihoodkvottest. Använd funktionen `qchisq()` för att hitta det kritiska värdet. Använd signifikansnivå $\alpha = 0.05$.

Del 2 - Klassifikation med logistisk regression

2a - Beräkna skattade sannolikheter för testdata

Du ska nu utvärdera dom två modellerna du precis har skapat genom att se hur bra dom är på att klassificera observationerna i `titanic_test`. För att göra detta behöver du först beräkna dom skattade sannolikheterna för observationerna i `titanic_test`. Precis som för linjär regression kan du använda funktionen `predict()` till detta. `predict()` kommer att

fatta att det är en logistisk regressionsmodell du vill ha prediktioner för. Vill du lära dig mer om hur `predict()` funkar för ett objekt av typen `glm` så behöver du skriva `?predict.glm` i konsolen.

- ☐ Använd `predict()` för att skatta sannolikheterna för överlevnad för alla observationer i ditt testset genom att använda argumentet `newdata = titanic_test`. När `predict()` får en logistisk regressionsmodell som input kommer den ge tillbaks *log-odds*. För att få sannolikheter istället behöver du ange `type = "response"`.
- ☐ Lägg till dinna sannoliketer i `titanic_test` som två variabler med namnen `slh_mini` respektive `slh_maxi`. Ett sätt att göra detta är med `mutate()`. (Se exempelkod nedan, du kommer behöva fylla i `predict()`-funktionerna.)

```
titanic_test <- titanic_test %>%
  mutate(
    slh_mini = predict(...)
    slh_maxi = predict(...)
  )
```

2b - Förvirringsmatriser

- ☐ Använd `mutate()` eller dollar-syntax för att lägga till ytterligare två variabler till `titanic_test`: `klassifikation_mini` och `klassifikation_maxi`. Dessa ska vara 1 om respektive sannolikhet är större än 0.5, annars 0. (Hint: om du skriver `as.numeric(min_var)` och `min_var` är en boolsk variabel (alltså, en variabel som bara antar värdena `TRUE` respektive `FALSE`) så kommer `TRUE` och `FALSE` ändras till 1 och 0.)

I testsetet `titanic_test` ska det nu finnas en kolumn med det faktiska utfallet (`survived`) och två kolumner med klassifikationen (prediktionen) för den lilla respektive stora modellen.

- ☐ Använd funktionen `tabyl()` från paketet `janitor` för att skapa en förvirringsmatris för din mindre modell. Du behöver först lägga till `library(janitor)` till din paketchunk i början av din notebook, samt eventuellt installera `janitor()` med `install.packages("janitor")`.

```
forv_mat <- titanic_test %>%
  tabyl(survived, klassification_mini)
forv_mat
```

Tabellen du precis skapat är en `data.frame`, så du kan komma åt elementen i den så här:

- `forv_mat[1, 2]` ger antalet som inte överlevde och var korrekt klassificerade,

- `forv_mat[2, 3]` ger antalet som överlevde och var korrekt klassificerade, och så vidare.

Första siffran motsvarar rad, och andra kolumn. Första kolumnen ignorerar vi eftersom att den bara anger dom två värdena på `survived`.

- ☐ Ta fram ett uttryck för specificiteten och sensitiviteten för din mindre modell utan att räkna för hand. Alltså, du ska använda olika kombinationer av `forv_mat`.
- ☐ Om du vill/har tid, beräkna förvirringsmatrisen för den större modellen också.

Del 3 - Modellutvärdering baserat på prediktionsvärden

I del 2 tog du fram dom skattade sannolikheterna för ditt testdata. Du ska nu utvärdera vilken av dom två modellerna som har bäst logaritmerat prediktionsvärde. Du kan göra detta antingen med base R-syntax eller med `summarise()`.

- ☐ Använd antingen base R (dollar-notation) eller `summarise()` för att beräkna summan av dom logaritmerade prediktionsvärdena för dom två modellerna. Vilken modell skulle du föredra baserat på dessa resultat? Titta på föreläsningsanteckningarna till F8 om du är osäker! Det kommer bli ett lite klurigt uttryck som använder både värdet i kolumnen med den skattade sannolikheten, och värdet i kolumnen `survived`.
- ☐ Beräkna summan av dom logaritmerade prediktionsvärdena för en modell som alltid ger den skattade sannolikheten 0.5. Verifiera att din modell är bättre än denna "nonsense-modell".